ALGÈBRE ET ALGORITHMIQUE NUMÉRIQUE

Michelle Schatzman

Martín Sombra

Institut Camille Jordan (Mathématiques), Université de Lyon 1, 43 Bd. du 11 Novembre 1918, 69622 Villeurbanne Cedex, France

Universitat de Barcelona, Departament d'Àlgebra i Geometria. Gran Via 585, 08007 Barcelone, Espagne

1991 *Mathematics Subject Classification.* Primaire : xxx ; Secondaire : xxx, xxx.

Key words and phrases. xxx.

Résumé. Texte en chantier, basé sur les notes des cours de Master Recherche 2ème année donnés à l'université de Lyon 1 pendant l'automne 2004 et à l'université de Buenos Aires pendant l'automme 2005.

Table des matières

Introduction

Un nombre formidable d'applications comme par exemple la discrétisation d'une EDP, se réduit à la résolution d'un système linéaire $A\,x = b$, où A est une matrice inversible d'ordre n et $b \in \mathbb{R}^n$. En principe, ce problème pourrait se résoudre *via* l'algorithme de Gauss en $\frac{2}{3}n^3$ opérations arithmétiques, or ce coût est beaucoup trop haut quand n est grand, ce qui arrive assez souvent. Considérons par exemple la discrétisation d'une EDP sur un cube de $\mathbb{R}^3$, avec un pas de $1/100$. Ceci se traduit en un problème de taille

$$n := 10^2 \times 10^2 \times 10^2 = 10^6;$$

donc le calcul *via* l'algorithme de Gauss demanderait $\frac{2}{3}10^{18}$ opérations, soit 21 ans à une vitesse de 1Gflops.

Il en résulte l'importance d'identifier quelles sont les structures relevantes pour les applications, pour ensuite obtenir des algorithmes adaptés. Comme exemple prototypique, pour une matrice de Toeplitz

$$A := \begin{bmatrix} a_0 & a_1 & \cdots & a_{n-2} & a_{n-1} \\ a_{-1} & a_0 & \ddots & & a_{n-2} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ a_{-(n-2)} & & \ddots & a_0 & a_1 \\ a_{-(n-1)} & a_{-(n-2)} & \cdots & a_{-1} & a_0 \end{bmatrix}$$

il y a des algorithmes stables que résout l'équation $A\,x = b$ en $O(n \log^3(n))$ opérations, nettement moins que le temps $O(n^2)$ qui prendrait d'écrire la matrice complète !

Ce texte tente d'articuler les aspects algébriques et analytiques de la résolution de systèmes linéaires, avec comme but la présentation des différentes méthodes (exactes et numériques) pour la résolution des systèmes structurés. D'une façon plus générale, on vise à fournir des éléments de base de l'algèbre numérique fondamentale, ce qui comprends l'accès à un choix de méthodes algébriques modernes, la fiabilité et validation de calculs numériques, et la relation entre calcul symbolique et calcul numérique.

À discuter : dans une deuxième partie du cours, on considérera la résolution d'équations *polynomiales*, sujet dans lequel se mêlent les points de vue numérique et symbolique. Considérons le système d'équations

$$\begin{aligned} 3\,b\,c - 2\,b + c + 4 &= 0 \\ a\,c + 5\,a - c + 2 &= 0 \\ 2\,a\,b + 9\,a - b + 8 &= 0, \end{aligned}$$

provenant d'un jeu non-coopératif à trois joueurs ; les zéros positifs de ce système correspondent aux points d'*équilibre de Nash*. Grâce au célèbre résultat de Nash, on sait qu'il y a toujours des points d'équilibre. Mais combien ? Un nombre fini ou infini ? Il se pose aussi le problème du calcul numérique de ces racines.

La estimation du nombre et de la taille des racines peut se faire *via* la théorie de l'intersection (théorèmes de Bézout et de Bernstein-Koushnirenko). Pour le calcul effectif, nous présenterons l'algorithme récent d'homotopie due à M. Shub et S. Smale.

Voici la liste approximative des contenus prévus :

(1) Généralités sur le calcul en flottant et en exacte. Transformée de Fourier rapide (FFT), multiplication rapide des entiers et de polynômes.

(2) Algorithme d'élimination de Gauss : complexité en calcul flottant et en calcul exacte. Stabilité : analyse de l'erreur *a priori* et *a posteriori*.

(3) Rang de déplacement, algorithme de décomposition LU de Schur. Algorithmes "rapides" et "super-rapides" pour des structures de Toeplitz, Hankel, Vandermonde et Cauchy. Applications aux codes correcteurs de Reed-Solomon.

(4) Conditionnement d'un système linéaire. Distribution de probabilité du nombre de conditionnement (thesis de Edelman).

(5) Stabilité des systèmes triangulaires. Méthodes iteratifs ; grandes classes de méthodes, dont le gradient et le gradient conjugé.
Méthode GMRES. Gradient conjugé preconditionné, GMRES preconditionné.
Introduction au méthode multigrille géométrique.

(6) Introduction aux matrices hiérarchiques. Définition et arithmétique des matrices hiérarchiques, applications aux équations aux dérivées partielles.

(7) Systèmes d'équations polynomiales. Rudiments de géométrie algébrique. Théorèmes de Bézout et de Bernstein-Koushnirenko. Algorithme d'homothopie polyhedrale.

Chapitre 1

Valeurs singulières des matrices

La plupart des choses dans ce chapitre se trouvent aussi dans le classique [**18**] ou dans le futur classique [**10**].

1. Rappel des résultats d'algèbre linéaire

Soit $\mathbb{K}$ le corps de base ; dans ce qui suit on considérera le corps des réels $\mathbb{K} = \mathbb{R}$ ou des complexes $\mathbb{K} = \mathbb{C}$. Cependant pour les algorithmes exacts on pourra (et voudra) considérer d'autres corps (les rationnels $\mathbb{Q}$, les corps finis $\mathbb{F}_q$).

Soit $f : V \to W$ un opérateur linéaire entre des $\mathbb{K}$-espaces vectoriels V et W de dimension n et m respectivement. Soient $\mathcal{B} = (v_1, \ldots, v_n)$ et $\mathcal{C} = (w_1, \ldots, w_m)$ des bases de V et de W respectivement. La *matrice de f dans les bases $\mathcal{B}$ et $\mathcal{C}$* est la matrice

$$A = [f]_{\mathcal{B},\mathcal{C}} \in \mathbb{K}^{m\times n}$$

dont les colonnes sont l'image des vecteurs de $\mathcal{B}$ par rapport à la base $\mathcal{C}$: si $f(v_j) = \sum_{i=1}^{m} a_{i,j} w_i$ alors

$$\mathrm{col}_j(A) = \begin{bmatrix} a_{1,j} \\ \vdots \\ a_{m,j} \end{bmatrix}.$$

Typiquement $V = \mathbb{K}^n$ et $W = \mathbb{K}^m$ sont munis des bases standard. On identifie une forme linéaire $\ell : \mathbb{K}^n \to \mathbb{K}$ avec un vecteur *ligne* et un point $x \in \mathbb{R}^m$ avec un vecteur *colonne*.

1.1. Décomposition en blocs d'opérateurs et matrices. La décomposition en blocs apparaît naturellement lors des discrétisations des EDPs en plus d'une variable. Considérons les décompositions

$$V = \oplus_{j=1}^{N} V_j \quad , \quad W = \oplus_{i=1}^{M} W_i.$$

Pour $1 \le j \le N$ et $1 \le i \le M$ posons

$$J_j : V_j \hookrightarrow V \quad , \quad x_j \mapsto (0, \ldots, 0, x_j, 0, \ldots, 0),$$
$$Q_i : W \to W_i \quad , \quad y = (y_1, \ldots, y_M) \mapsto y_i$$

pour les inclusion et projection canoniques. L'opérateur $f_{i,j} = Q_i \circ f \circ J_j : V_j \to W_i$ est défini par le diagramme :

$$\begin{array}{ccc} V & \to^{f} & W \\ \uparrow J_j & & \downarrow Q_i \\ V_j & \dashrightarrow^{f_{i,j}} & W_j \end{array}$$

En d'autres termes, c'est la j-ème composante de la restriction de f au sous-espace V_j. La *décomposition en blocs* de f associée est

$$[f] = \begin{bmatrix} f_{1,1} & \cdots & f_{1,N} \\ \vdots & & \vdots \\ f_{M,1} & \cdots & f_{M,N} \end{bmatrix}$$

Soit $x = \sum_{j=1}^{N} x_j$, alors

$$f(x) = \left(\sum_{j=1}^{N} f_{1,j}(x_j), \ldots, \sum_{j=1}^{N} f_{M,j}(x_j) \right).$$

La multiplication respecte la structure en blocs : soit

$$X = \oplus_{h=1}^{L} X_h$$

et $g : W \to X$ un autre opérateur linéaire avec

$$[g] = \begin{bmatrix} g_{1,1} & \cdots & g_{1,M} \\ \vdots & & \vdots \\ g_{L,M} & \cdots & g_{L,M} \end{bmatrix}$$

la décomposition en blocs correspondante, alors

$$(g \circ f)_{h,j} = \sum_{i=1}^{M} g_{h,i} \circ f_{i,j}.$$

Par exemple

$$\begin{bmatrix} B_{1,1} & B_{1,2} \end{bmatrix} \begin{bmatrix} A_{1,1} & A_{1,2} & A_{1,3} \\ A_{2,1} & A_{2,2} & A_{2,3} \end{bmatrix} = \begin{bmatrix} B_{1,1}A_{1,1} + B_{1,2}A_{2,1} & B_{1,1}A_{1,2} + B_{1,2}A_{2,2} & B_{1,1}A_{1,3} + B_{1,2}A_{2,3} \end{bmatrix}.$$

1.2. Produits scalaires, adjonction, opérateurs hermitiens. Le *produit scalaire canonique* est

$$\langle x, y \rangle = \sum_{j=1}^{n} x_j y_j = x^T y \quad , \quad \text{pour } x, y \in \mathbb{R}^n$$

$$\langle x, y \rangle = \sum_{j=1}^{n} x_j \overline{y_j} = x^T \overline{y} \quad , \quad \text{pour } x, y \in \mathbb{C}^n.$$

Soit $f : \mathbb{K}^n \to \mathbb{K}^m$, l'*opérateur adjoint* est l'unique $f^* : \mathbb{K}^m \to \mathbb{K}^n$ tel que

$$\langle x, f^*(y) \rangle_{\mathbb{K}^n} = \langle f(x), y \rangle_{\mathbb{K}^m}$$

pour tout $x \in \mathbb{K}^n$ et $y \in \mathbb{K}^m$. Si $A = [f] \in \mathbb{K}^{m \times n}$ est la matrice de f, alors

$$[f^*] = A^* := \overline{A}^T \in \mathbb{K}^{n \times m};$$

c'est-à-dire $(A^*)_{j,i} = \overline{A_{i,j}}$.

Une matrice carrée $A \in \mathbb{K}^{n \times n}$ est *hermitienne* ou *auto-adjointe* si $A^* = A$. Le résultat fondamental est que toute matrice hermitienne est diagonalisable dans une base unitaire, et ses valeurs propres sont réels. Explicitement

$$A = U \operatorname{diag}(\lambda_1, \ldots, \lambda_n) U^*$$

avec $\lambda_j \in \mathbb{R}$ et $U \in \mathbb{K}^{n \times n}$ telle que $U^*U = \mathbf{1}_n$ (c'est-à-dire $U \in O(n)$ si $\mathbb{K} = \mathbb{R}$ et $U \in U(n)$ si $\mathbb{K} = \mathbb{C}$).

Une matrice hermitienne est *positive* (resp. *semi-positive*) si tous ses valeurs propres sont positifs (resp. positifs ou nuls).

1.3. Normes de vecteurs, opérateurs et matrices. Les normes sont utilisées pour mesurer les erreurs dans les calculs matriciels ; on a donc besoin d'apprendre à les manipuler.

Soit V un $\mathbb{K}$-espace vectoriel. Une *norme* sur V est une fonction $|\cdot| : V \to \mathbb{R}$ satisfaisant

(1) $|x| \geq 0$ pour tout $x \in V$, et $|x| = 0$ si et seulement si $x = 0$;

(2) $|\lambda \cdot x| = |\lambda| \cdot |x|$ pour $\lambda \in \mathbb{K}$ et $x \in V$;

(3) inégalité du triangle : $|x + y| \leq |x| + |y|$ pour $x, y \in V$.

Exemples typiques : soit $x = (x_1, \ldots, x_n) \in \mathbb{C}^n$; la *norme* ℓ^p est

$$|x|_p = \left(\sum_{j=1}^{n} |x_j|^p \right)^{1/p} \quad , \qquad \text{pour } 1 \leq p < \infty;$$

$$|x|_\infty = \max_{1 \leq j \leq n} |x_j| \quad , \qquad \text{pour } p = \infty.$$

Une autre famille d'exemples : norme indexée par une matrice hermitienne définie positive :

$$|x|_A := (x^T A \overline{x})^{1/2}.$$

Toutes les normes sont équivalentes sur $\mathbb{R}^n$ ou sur $\mathbb{C}^d$. Mais les constantes dépendent de la dimension ! :

$$|x|_2 \leq |x|_1 \leq \sqrt{n}|x|_2,$$
$$|x|_\infty \leq |x|_2 \leq \sqrt{n}|x|_\infty,$$
$$|x|_\infty \leq |x|_1 \leq n|x|_\infty.$$

Soit $f : V \to W$ un opérateur linéaire entre des espaces munis des normes N et M respectivement. La *norme d'opérateurs* subordonnée à N et M est

$$||f||_{N,M} = \max_{x \in V \setminus \{0\}} \frac{M(f(x))}{N(x)} = \max_{N(x)=1} M(f(x)).$$

En particulier, c'est une norme sur l'espace vectoriel $\mathrm{Hom}_{\mathbb{K}}(\mathbb{K}^n, \mathbb{K}^m)$. Par exemple, pour $A \in \mathbb{K}^{m \times n}$ on pose

$$\|A\|_{p,q} = \sup_{x \neq 0} \frac{\|Ax\|_q}{\|x\|_p},$$

$$\|A\|_p = \|A\|_{p,p}.$$

Une fonction norme $||\cdot|| : \mathbb{K}^{n \times n} \to \mathbb{R}$ est dite *matricielle* si pour tout $A, B \in \mathbb{K}^{n \times n}$ elle vérifie

$$||A \cdot B|| \leq ||A|| \cdot ||B||.$$

Toute norme d'opérateurs est matricielle, pourvu que $V = W$. Un autre exemple de norme matricielle est la *norme de Frobenius* définie par

$$\|A\|_F = (\mathrm{Tr}(A^* A))^{1/2} = \left(\sum_{i=1}^{n} \sum_{j=1}^{n} |a_{ij}|^2 \right)^{1/2}$$

pour $A \in \mathbb{K}^{n \times n}$. La norme de Frobenius n'est pas subordonnée à une norme vectorielle. Ceci se voit facilement en remarquant que $||\mathbf{1}_n||_F = \sqrt{n}$ tandis que $||\mathbf{1}_n|| = 1$ pour toute norme d'opérateurs $||\cdot||$.

La norme qui est vraiment utile est la norme 2 (c'est-à-dire $||\cdot||_2$) mais elle est difficile à calculer. Par contre on peut mieux manipuler la norme de Frobenius, et elle nous fournit des estimations pour cette norme :

$$||A||_2 \leq ||A||_F \leq \sqrt{n}||A||_2.$$

Convention : si $m > n$, $\mathbb{K}^n$ s'injecte dans $\mathbb{K}^m$, et la norme $\|x\|_*$ de $x \in \mathbb{K}^n$ est égale à la norme de $\|X\|_*$ défini par

$$X = \begin{bmatrix} x \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

La convention est symétrique si $m < n$. Dans le cas $m > n$, on pose

$$\mathbf{1}_{m\times n} = \begin{bmatrix} 1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 1 \\ 0 & \dots & 0 \\ \vdots & & \vdots \\ 0 & \dots & 0 \end{bmatrix}$$

et dans le cas $m \leq n$, on pose

$$\mathbf{1}_{m\times n} = \begin{bmatrix} 1 & \dots & 0 & 0 & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots & & \vdots \\ 0 & \dots & 1 & 0 & \dots & 0 \end{bmatrix}.$$

Alors $\|\mathbf{1}_{m\times n}\|_F = \sqrt{\min(m,n)}$ et

$$\sup_{x\neq 0} \frac{\|\mathbf{1}_{m\times n}x\|_*}{\|x\|_*} = 1,$$

puisque $\mathbf{1}x$ s'identifie à X.

Voici quelques inégalités supplémentaires : soit $A \in \mathbb{K}^{m\times n}$, alors

$$\max_{\substack{1\leq i\leq m\\ 1\leq j\leq n}} |a_{ij}| \leq \|A\|_2 \leq \sqrt{mn} \max_{\substack{1\leq i\leq m\\ 1\leq j\leq n}} |a_{ij}|, \tag{1a}$$

$$\|A\|_2 \leq \sqrt{\|A\|_1\|A\|_\infty}, \tag{1b}$$

$$\|A\|_1 = \max_{1\leq j\leq n} \sum_{i=1}^{m} |a_{ij}|, \tag{1c}$$

$$\|A\|_\infty = \max_{1\leq i\leq m} \sum_{j=1}^{n} |a_{ij}|, \tag{1d}$$

$$n^{-1/2}\|A\|_\infty \leq \|A\|_2 \leq m^{1/2}\|A\|_\infty, \tag{1e}$$

$$m^{-1/2}\|A\|_1 \leq \|A\|_2 \leq n^{1/2}\|A\|_1. \tag{1f}$$

EXERCICE 1.1. ◁ Montrer les inégalités (1a) à (1f). ▷

2. Décomposition en valeurs singulières (DVS)

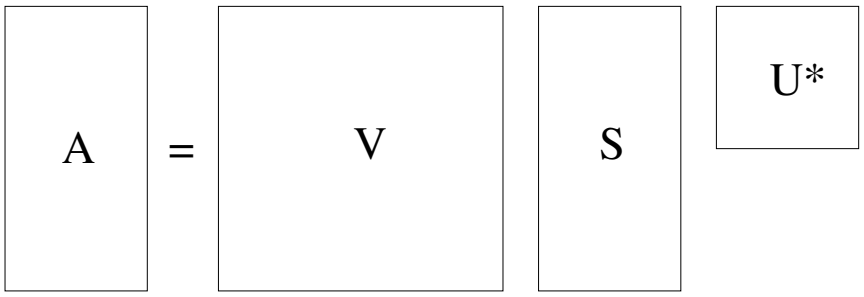
THÉORÈME 2.1 (DVS). *Soit $A \in \mathbb{K}^{m\times n}$, alors il existe U matrice unitaire dans $\mathbb{K}^{n\times n}$, V matrice unitairee dans $\mathbb{K}^{m\times m}$ et $\ell = \min(m,n)$ nombres positifs $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_\ell \geq 0$ tels que*

$$A = V\Sigma U^*.$$

avec $\Sigma = \mathrm{diag}(\sigma_1, \ldots, \sigma_\ell) \in \mathbb{K}^{m\times n}$.

REMARQUE 2.2. Attention à l'abus de notations ! La matrice $\mathrm{diag}(\sigma_1, \ldots, \sigma_\ell)$ est une matrice à m lignes et n colonnes dont le bloc supérieur gauche $\ell \times \ell$ est la matrice diagonale dont les valeurs diagonales sont les σ_i.

Graphiquement



Si on pense à la matrice A comme un opérateur $\mathbb{K}^n \to \mathbb{K}^m$, ce résultat peut s'énoncer géométriquement comme que modulo des changements unitaires des coordonnées de $\mathbb{K}^n$ (les colonnes de U) et de $\mathbb{K}^m$ (les colonnes de V), l'opérateur devient diagonal ≥ 0.

DÉMONSTRATION. On supposera s.p.d.g. $m \geq n$ et on fait récurrence sur m et n. Pour $n = 1$ et m quelconque on pose

$$V = \frac{1}{||A||_2} A \quad , \quad \Sigma = ||A||_2 \quad , \quad U = 1.$$

Maintenant soit $m \geq n \geq 2$ et supposons le résultat vrai pour $m-1$ et $n-1$. Soit $u \in \mathbb{K}^n$ un vecteur unitaire tel que $\sigma_1 := |Au|_2 = ||A||_2$.

Si $A = 0$, le résultat est évident. On suppose donc $A \neq 0$, donc $\sigma_1 > 0$. Soient

$$\widetilde{U} \in \mathbb{K}^{n\times(n-1)} \quad , \quad \widetilde{V} \in \mathbb{K}^{m\times(m-1)}$$

des matrices telles que

$$\widehat{U} = \begin{bmatrix} u & \widetilde{U} \end{bmatrix} \in U(n) \quad , \quad \widehat{V} = \begin{bmatrix} v & \widetilde{V} \end{bmatrix} \in U(m)$$

soient des matrices unitaires, et calculons $\widehat{V}^* A \widehat{U}$:

$$\widehat{V}^* A \widehat{U} = \begin{bmatrix} v^* \\ \widetilde{v} \end{bmatrix} A \begin{bmatrix} u & \widetilde{U} \end{bmatrix} = \begin{bmatrix} v^*Au & v^*A\widetilde{U} \\ \widetilde{V}^*Au & \widetilde{V}^*A\widetilde{U} \end{bmatrix} = \begin{bmatrix} v^*Au & v^*A\widetilde{U} \\ \widetilde{V}^*Au & \widetilde{A} \end{bmatrix}.$$

On a $v^*Au = |Au|_2 v^*v = \sigma_1$ et $\widetilde{V}^*Au = \widetilde{V}^*v = 0$ puisque les colonnes de $\widetilde{V}$ sont orthogonales à v par construction.

Estimons la norme de $\widehat{V}^* A \widehat{U}$, de façon à montrer que $w := v^*A\widetilde{U} = 0$: d'une part :

$$\|\widehat{V}^* A \widehat{U}\|_2 = \|A\|_2 = \sigma_1$$

car $\widehat{V}$ et $\widehat{U}$ sont unitaires ; d'autre part,

$$\|\widehat{V}^* A \widehat{U}\|_2 = \|\widehat{U} A^* \widehat{V}^*\|_2 = \sup_{x\in\mathbb{K}^n\setminus\{0\}} \frac{|\widehat{U}A^*\widehat{V}^*x|_2}{|x|_2}.$$

Choix particulier de x :

$$x = \begin{bmatrix} \sigma_1 \\ w^* \end{bmatrix};$$

alors

$$\widehat{U}A^*\widehat{V}^*x = \begin{bmatrix} \sigma_1 x_1 + ww^* \\ \widetilde{A}w^* \end{bmatrix},$$

et donc

$$|x|_2 = (\sigma_1^2 + |w|_2^2)^{1/2} \quad , \quad |\widehat{U}A^*\widehat{V}^*x|_2 = \left(\sigma_1^2 + |w|_2^2 + |\widetilde{A}w^*|_2^2\right)^{1/2},$$

et par conséquent

$$\frac{|\widehat{U}A^*\widehat{V}^*x|_2}{|x|_2} \geq \frac{\left(\sigma_1^2 + |w|_2^2 + |\widetilde{A}w^*|_2^2\right)^{1/2}}{(\sigma_1^2 + |w|_2^2)^{1/2}} \geq (\sigma_1^2 + |w|_2^2)^{1/2}.$$

L'inégalité

$$\sigma_1 \geq (\sigma_1^2 + |w|_2^2)^{1/2}$$

implique immédiatement $w = 0$.

Finalement on applique l'hypothèse de récurrence :

$$\widetilde{A} := \widetilde{V}^* A \widetilde{U} = V_1 \Sigma_1 U_1^*$$

et donc

$$A = V \begin{bmatrix} \sigma_1 & \\ & \widetilde{A} \end{bmatrix} U^* = V \begin{bmatrix} 1 & \\ & V_1 \end{bmatrix} \begin{bmatrix} \sigma_1 & \\ & \Sigma_1 \end{bmatrix} \begin{bmatrix} 1 & \\ & U_1^* \end{bmatrix} \widehat{U}^*;$$

le théorème est donc démontré. □

On remarque que

$$A^*A = (V\Sigma U^*)^*(V\Sigma U^*) = U\Sigma^2 U^*,$$

donc σ_i^2 est la i-ème valeur propre de A^*A. On peut démontrer facilement que $||A||_2 = \sigma_1$ et on en déduit

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

cette dernière égalité due à ce que $A^*A \geq 0$.

Avec la DVS on trouve une expression de A comme somme de matrices de rang 1 :

$$A = \sum_{i=1}^{n} \sigma_i u_i v_i^*.$$

Pour $0 \leq k \leq n$ posons

$$A_k = \sum_{i=1}^{k} \sigma_i u_i v_i^* \tag{2}$$

qui est donc une matrice de rang au plus k.

Soit $M_k(m,n) \subset \mathbb{C}^{m\times n}$ l'ensemble des matrices de rang au plus k. Ceci est une variété algébrique *déterminantale* (c'est-à-dire définie par l'annulation d'un nombre fini de déterminants) :

$$M_k(m,n) = Z(\det(B_{i_1,\ldots,i_{k+1};j_1,\ldots,j_{k+1}}) \\ : 0 \leq i_1 < \cdots < i_{k+1} \leq m, 0 \leq j_1 < \cdots < j_{k+1} \leq n\}).$$

THÉORÈME 2.3 (Théorème d'approximation). *La distance pour la norme* 2 *de* A *à l'ensemble des matrices* $m \times n$ *de rang au plus* k *est*

$$\operatorname{dist}_{||\cdot||_2}(A, M_k(m,n)) = ||A - A_k||_2 = \sigma_{k+1}.$$

DÉMONSTRATION. On a

$$A'_k = V \operatorname{diag}(\sigma_1, \ldots, \sigma_k, 0, \ldots, 0) U^*$$

et

$$\|A - A_k\|_2 = \|\operatorname{diag}(\underbrace{0, \ldots, 0}_{k \text{ zéros}}, \sigma_{k+1}, \ldots, \sigma_l)\|_2 = \sigma_{k+1}.$$

Réciproquement, supposons qu'il existe une matrice B de rang $\leq k$ telle que $\|A - B\|_2 < \sigma_k$. Soient $v_1, \ldots, v_{k+1}$ les $k+1$ premiers vecteurs colonne de la matrice V ; comme le noyau de B est au moins de dimension $n-k$, l'espace engendré par $v_1, \ldots, v_{k+1}$ est d'intersection non vide avec le noyau de B ; soit x un élément de norme 1 appartenant à cette intersection ; alors

$$\|A - B\|_2 \geq |Ax - Bx|_2 = |Ax|_2.$$

Si on pose

$$x = \sum_{j=1}^{k+1} x_j v_j,$$

alors

$$Ax = \sum_{j=1}^{k+1} x_j A v_j = \sum_{j=1}^{k+1} x_j \sigma_j u_j,$$

et donc

$$|Ax|_2 = \left(\sum_{j=1}^{k+1} \sigma_j^2 x_i^2 \right)^{1/2} \geq \sigma_{k+1} |x|_2 = \sigma_{k+1}.$$

On en conclut $\|A - B\|_2 \geq \sigma_{k+1}$. □

En particulier la distance pour la norme 2 de A à l'ensemble des systèmes mal conditionnés $M_{n-1}(m,n)$ est la dernière valeur singulière σ_n.

La DVS s'applique à la compression d'images. Une image est juste une matrice A dont le coefficient $a_{i,j}$ donne l'intensité du pixel (i,j). A la place des mn coefficients, on peut juste stocker A_k, qui selon le théorème d'approximation ci-dessus est la meilleure approximation de A dans $M_k(m,n)$, à partir de laquelle on peut reconstruire partiellement l'image en question. La matrice A_k est représentée par $(m+n)k$ coefficients (*voir* l'expression (2) ci-dessus) ; le taux de compression est

$$\frac{(m+n)k}{mn}.$$

Pour des exemples concrets, *voir* [**10**, § 3.2.3].

La DVS permet de calculer facilement l'*inverse de Moore-Penrose* d'une matrice rectangulaire $A \in \mathbb{K}^{m \times n}$ $(n \leq m)$ de rang maximal n :

$$A^+ := (A^* A)^{-1} A^* = U \Sigma^{-1} V^*.$$

PROPOSITION 2.4. *Soit $S^{n-1} = \{x \in \mathbb{R}^n : |x|_2 = 1\}$ la sphère unité de $\mathbb{R}^n$, alors $A(S^{n-1})$, l'image de cette sphère sous l'application A, est l'ellipsoïde de $\mathbb{R}^m$ centré en $\mathbf{0}$ d'axes principaux $\sigma_j v_j$ pour $j = 1, \ldots, m$.*

Par exemple pour

$$A = \begin{bmatrix} 1 & 2 \\ -2 & -1 \end{bmatrix} = \left(\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \right) \begin{bmatrix} 1 & 0 \\ 0 & 3 \end{bmatrix} \left(\frac{1}{\sqrt{2}} \begin{bmatrix} -1 & 1 \\ 1 & 1 \end{bmatrix} \right),$$

l'ensemble $A(S^2)$ est l'ellipse d'axes principaux $3(-1/\sqrt{2}, 1/\sqrt{2})$ et $(1/\sqrt{2}, 1/\sqrt{2})$.

DÉMONSTRATION. Puisque U est unitaire on a $U^*(S^{n-1}) = S^{n-1}$. Ensuite, $\Sigma(S^{n-1})$ est l'ellipsoïde d'équation $\sum_{i=1}^m (w_i/\sigma_i)^2 = 1$, d'axes principaux $\sigma_i e_i$, où e_i est le i-ème vecteur de la base standard de $\mathbb{R}^m$. Finalement la multiplication par V a l'effet de changer les e_is en les v_is. □

Les origines de la DVS sont expliquées dans l'article [**35**]. La DVS (théorème 2.1) fut indépendamment découverte par Eugenio Beltrami en 1873, et par Camille Jordan un an après. Le théorème 2.3 d'approximation est du à Erhard Schmid (celui du procédé d'orthogonalisation de Gram-Schmid et un étudiant de Hilbert) en 1907 et peut se considérer comme le résultat fondamental de la DVS. Ce résultat est souvent (et à tort) appelé théorème de Eckart-Young, qui l'ont redécouvert 29 ans après.

2.1. Pour ceux qui connaissent un peu d'analyse fonctionnelle. On note $\mathcal{L}(H)$ l'ensemble des opérateurs bornés d'un espace de Hilbert H dans lui-même. Les valeurs singulières d'un opérateur $A \in \mathcal{L}(H)$ sont les racines carrées des valeurs propres de A^*A, rangées par ordre décroissant :

$$\sigma_1(A) \geq \sigma_2(A) \geq \cdots$$

On rappelle que si $B \in \mathcal{L}(H)$ est un opérateur autoadjoint ≥ 0 il possède une unique racine carrée autoadjointe ≥ 0, qui sera notée $\sqrt{B}$. La norme de $x \in H$ est notée $|x|$ et la norme d'opérateur subordonnée à la norme de H est notée $\|A\| = \sup\{|Ax| : |x| \leq 1\}$.

EXERCICE 1.2. ◁ Soit $A \in \mathcal{L}(H)$. Montrer qu'on a

$$\sigma_{n+1}(A) = \min\{\|A\,|_{E^\perp}\| : \ E \text{ de dimension } n\}.$$

Montrer que ce minimum est atteint en prenant comme E l'espace engendré par les n plus grandes valeurs propres comptées avec leur multiplicité de $|A| = \sqrt{A^*A}$.

Indication : utiliser la caractérisation maximin des valeurs propres d'un opérateur autoadjoint borné B dans un espace de Hilbert, dont les valeurs propres sont notées $\lambda_1(B) \geq \lambda_2(B) \geq \cdots$ avec leur multiplicité :

$$\lambda_k(B) = \min\left\{\max\left\{\frac{|Bx|}{|x|} : x \in V\right\}, \dim V = k\right\}.$$

▷

EXERCICE 1.3. ◁ Montrer qu'on a également

$$\sigma_{n+1}(A) = \inf\{\|A - X\| : X \in R_n\}, \tag{3}$$

où R_n désigne l'ensemble des opérateurs de $\mathcal{L}(H)$ de rang au plus n. ▷

EXERCICE 1.4. ◁ Déduire de la caractérisation (3) que, pour tout n, pour tous A et B dans $\mathcal{L}(H)$:

$$|\sigma_n(A) - \sigma_n(B)| \leq \|A - B\|.$$

▷

EXERCICE 1.5. ◁ Déduire de l'inclusion $R_n + R_m \subset R_{m+n}$ l'inégalité

$$\sigma_{m+n+1}(A + B) \leq \sigma_{m+1}(A) + \sigma_{n+1}(B).$$

▷

EXERCICE 1.6. ◁ Montrer également qu'on a l'inégalité

$$\sigma_{m+n+1}(AB) \leq \sigma_{m+1}(A)\sigma_{n+1(B)},$$

et en déduire que pour tout n

$$\sigma_n(AB) \leq \sigma_n(A)\|B\|, \quad \sigma_n(AB) \leq \sigma_n(B)\|A\|.$$

Indication : utiliser $X = AX_2 + X_1B - X_1X_2$, avec $X_1 \in R_m$ et $X_2 \in R_n$. ▷

EXERCICE 1.7. ◁Montrer qu'un opérateur A est compact si et seulement si

$$\lim_{n\to\infty} \sigma_n(A) = 0.$$

▷

EXERCICE 1.8. ◁ Montrer que l'ensemble $\mathcal{K}$ des opérateurs compacts de H dans lui-même est un idéal bilatère de $\mathcal{L}(H)$, c'est à dire :

$$A \in \mathcal{K}, B \in \mathcal{L}(H) \Longrightarrow AB \in \mathcal{K}, BA \in \mathcal{K}.$$

▷

EXERCICE 1.9. ◁ Soit $\mathcal{L}^p(H)$ l'ensemble des opérateurs A dans $\mathcal{L}(H)$ tels que

$$\|A\|_p = \left(\sum_{j=1}^{\infty} \sigma_n(A)^p\right)^{1/p} < \infty.$$

soit fini.

Montrer que pour tout $p \in [1, \infty)$, $\mathcal{L}^p(H)$ est un espace vectoriel, et que de plus, c'est un idéal bilatère d'opérateurs compacts. Dans le cas $p = 1$, l'espace $\mathcal{L}^1(H)$ est l'espace des opérateurs de la classe de trace ; dans le cas $p = 2$, montrer qu'on retrouve les opérateurs de Hilbert-Schmidt. ▷

EXERCICE 1.10. ◁ On pose

$$s_N(A) = \sum_{n=1}^{N} \sigma_n(A).$$

Montrer que

$$s_N(A) = \sup\{\|AP_E\|_1 : \dim E = N\},$$

avec P_E la projection orthogonale sur E. En déduire que pour tout $N \geq 1$, s_N est une norme sur E. ▷

Chapitre 2

La transformée de Fourier rapide

1. Complexité d'algorithmes

Dans une première approximation, la complexité d'un algorithme est une formalisation de la notion de vitesse d'exécution. Un algorithme est un procédé effectif admettant comme *entrée* (en anglais : *input*) des mots finis (par exemple une suite finie de 0 et de 1) qui ou bien finit au bout d'un certain temps d'exécution donnant comme résultat une *sortie* (en anglais : *output*), ou bien ne finit pas. Ceci n'est nullement une définition précise d'algorithme, c'est seulement l'idée mais ça suffit à nos besoins.

Pour un mot a, on notera

$$\tau(a) \in \mathbb{N}$$

sa *taille* ou *longueur*. Bien entendu, cette notion présuppose un alphabet ou codification de base. La *complexité* $c_A(E)$ d'un algorithme A sur une entrée E est le nombre d'opérations réalisées par A sur cette entrée. La *complexité* de A tout court est une fonction

$$\mathcal{C}_A : \mathbb{N} \to \mathbb{N} \cup \{\infty\}$$

telle que pour $\tau \in \mathbb{N}$, $\mathcal{C}_A(\tau)$ est le nombre maximal d'opérations réalisées par A sur une entrée de taille τ.

Considérons comme exemple la somme en binaire de deux entiers $a, b \in \mathbb{N}$ suivant l'algorithme qu'on nous a appris à l'école. La taille de l'entrée est la somme de la longueur de la représentation en binaire de a et de b :

$$\tau(a, b) = \tau(a) + \tau(b) = \lfloor \log_2 a \rfloor + \lfloor \log_2 b \rfloor + 2.$$

Soit $c := a + b$ et $s := \tau(a)$, $t := \tau(b)$ et $u := \tau(c) \leq \max(s, t) + 1$ et écrivons les représentations binaires

$$a = \overline{a_s a_{s-1} \cdots a_1 a_0}, \quad b = \overline{b_t b_{t-1} \cdots b_1 b_0}, \quad c = \overline{c_u c_{u-1} \ldots c_1 c_0};$$

schématiquement l'algorithme est décrit par la figure

$$\begin{array}{cccccccc} & & r_{s+1} & r_s & r_{s-1} & \cdots & r_1 & \\ & & & a_s & a_{s-1} & \cdots & a_1 & a_0 \\ + & b_t & b_{t-1} & \cdots & \cdots & \cdots & b_1 & b_0 \\ \hline c_u & c_{u-1} & \cdots & \cdots & \cdots & \cdots & c_1 & c_0 \end{array}$$

où r_j est la j-ème *retenue*. En pseudocode :

Algorithme Σ :
Entrée : $a_s, \ldots, a_0; b_t, \ldots, b_0 \in \{0, 1\}$ **;**
Sortie : $c_u, \ldots, c_0 \in \{0, 1\}$**.**
For $k = 0$ **to** $v := \max(s, t)$ **do**
$r_{k+1} \cdot 2 + c_k \leftarrow a_k + b_k + r_k$ **od ;**
if $r_{v+1} = 1$ **then** $c_{v+1} \leftarrow 1$ **;**
end.

La taille de la sortie est $\leq \tau(a,b)$, donc il y a des chances d'avoir un algorithme linéaire : la taille de la sortie est une minoration triviale (mais souvent significative) de la complexité d'un algorithme. Pour le calcul de la complexité on a

(1) $3(v+1)+1$ lectures en mémoire ;

(2) $2(v+1)$ opérations binaires (sommes dans $\mathbb{Z}$ des 0 et des 1) ;

(3) 1 décision ($r_{v+1} =^{?} 1$) ;

(4) $2(v+1)+1$ écritures en mémoire.

Il est difficile de comparer le coût relatif de chacune de ces opérations. Comment les pondérer ? La pratique de l'analyse numérique et du calcul symbolique est de ne compter que les opérations binaires ($+,-,\cdot$ dans $\mathbb{Z}$ des 0 et des 1) c'est-à-dire le point (2) ci-dessus, dans l'hypothèse (ou l'espoir !) que ceci soit la partie la plus lourde dans l'algorithme, et par conséquent son coût domine celui des autres opérations. Avec cette convention $c_\Sigma(a,b) \leq 2(v+1) = 2\max(\tau(a),\tau(v)) \leq 2\tau(a,b)$ donc

$$\mathcal{C}_\Sigma(\tau) \leq 2\tau.$$

Cette convention nous permet de généraliser la notion d'algorithme. Soit $\mathbb{F}$ une structure algébrique quelconque (groupe, anneau, corps) et faisons comme si les opérations de $\mathbb{F}$ sont effectives, ce qui n'est pas toujours le cas (par exemple si $\mathbb{F} = \mathbb{R}$). On considérera des pseudo-algorithmes de type algébrique, sur des machines capables de manipuler des éléments de $\mathbb{F}$ (faire des opérations arithmétiques, les stocker en mémoire, etc). La *complexité*

$$\mathcal{C}_A(\mathbb{F};\tau)$$

de A relative à $\mathbb{F}$ sera définie comme le nombre maximal d'opérations arithmétiques de $\mathbb{F}$ (sommes, différences, multiplications et divisions dans le cas d'un corps) réalisées par A sur une entrée de taille τ.

Ces pseudo-algorithmes sont parfois appelés *machines BSS sur* $\mathbb{F}$, à cause du travail de L. Blum, M. Shub et S. Smale sur le sujet [**3**]. On peut vérifier que les algorithmes ou machines de Turing standard correspondent à des algorithmes sur $\mathbb{F}_2 = \{0,1\}$. Notons que si $\mathbb{F}$ es effectif (par exemple si $\mathbb{F} = \mathbb{Q}$), le pseudo-algorithme A est un véritable algorithme, et sa complexité totale doit tenir compte de la complexité $\mathcal{C}_A(\mathbb{F};\tau)$ relative à $\mathbb{F}$ *et* du coût binaire de chacune des opérations effectuées.

Définition 1.1. Un algorithme A est *linéaire/polynomial/exponentiel* s'il existe $\nu > 0$ tel que

$$\mathcal{C}_A(\tau) = O(\tau) \quad / \quad O(\tau^\nu) \quad / \quad O(\exp(\tau^\nu))$$

respectivement. La classe composée de tous les algorithmes linéaires/polynomiaux/exponentiels est notée par L/P/Exp respectivement.

On a

$$\mathrm{L} \subset \mathrm{P} \subset \mathrm{Exp}.$$

Grosso modo, cette classification est censée classifier les algorithmes en optimaux/acceptables/intractables. On a $\Sigma \in \mathrm{L}$; comme on vient de signaler ceci est le mieux à quoi on peut s'attendre. Mais en fait, on voudra toujours avoir des algorithmes linéaires (ou quasi-linéaires) : chaque fois que la puissance des ordinateurs est améliorée, la portée d'un algorithme linéaire est améliorée du même facteur. Un algorithme quadratique ou d'ordre supérieur n'est pas tellement sensible à l'amélioration des technologies (sa portée s'accroît comme la racine carrée de la puissance de l'ordinateur).

L'algorithme *de multiplication longue* est l'exemple typique d'algorithme quadratique : schématiquement

$$\begin{array}{cccccccccc}
 & & & & a_s & a_{s-1} & \cdots & a_1 & a_0 \\
\times & & b_t & b_{t-1} & \cdots & \cdots & & b_1 & b_0 \\
\hline
 & & & & b_0\cdot(a_s & a_{s-1} & \cdots & a_1 & a_0) \\
 & & & b_1\cdot(a_s & a_{s-1} & \cdots & a_1 & a_0) & \\
 & & \cdots & \cdots & \cdots & & & & \\
b_t\cdot(a_s & a_{s-1} & \cdots & a_1 & a_0) & & & & \\
\hline
d_v & d_{v-1} & \cdots & \cdots & \cdots & \cdots & \cdots & d_1 & d_0
\end{array}$$

donc on a au moins $(s+1)(t+1)$ multiplications binaires ; la complexité totale est estimée en $O(\tau^2)$.

Comme dernière illustration, considérons la factorisation des entiers, pour laquelle on ne connaît pas d'algorithme polynomial : pour $a \in \mathbb{N}$ donné, il s'agit de trouver les nombres premiers $p_1, \ldots, p_k$ et les exposants $e_1, \ldots, e_k \in \mathbb{N}$ tels que

$$a = p_1^{e_1} \cdots p_k^{e_k},$$

ou ce qui est équivalent, de trouver un diviseur premier p de a. L'algorithme naïf consiste à considérer successivement les candidats à diviseur $d \geq 2$ et de tester si $d|a$. Si oui, on déclare $p \leftarrow d$ le nombre premier en question. Si ce n'est pas le cas et si $d < \lfloor\sqrt{a}\rfloor$, on continue avec $d+1$ et ainsi de suite. S'il n'existe pas $d < \lfloor\sqrt{a}\rfloor$ tel que $d|a$ on déclare a premier. Cet algorithme a une complexité de type

$$a^{1/2},$$

or cette quantité est *exponentielle* en la taille $\lfloor \log_2 a \rfloor + 1$ de l'entrée ; on a

$$\mathcal{C}(\tau) = O(e^{\tau/2}).$$

Par contre, on peut *deviner* un nombre premier p tel que $p|a$ en temps polynomial, par un algorithme non-déterministe.

Définition 1.2. Un algorithme est *polynomial non-déterministe* si toute instance positive peut se vérifier en temps polynomial. La classe des algorithmes polynomiaux non-déterministes est notée par NP.

Évidemment P $\subset$ NP. La célèbre conjecture P $\neq$ NP est l'enjeu d'un des prix de US\$ 10^6 de la Fondation Clay aux États Unis.

Exercice 2.1. ◁ Soient $a, b \in \mathbb{N}$ deux entiers de taille $\tau(a), \tau(b) \leq \tau$.

(1) Estimer en $O(\tau^2)$ la complexité de $a \cdot b$ par l'algorithme de multiplication longue.

(2) Estimer en $O(\tau^2)$ la complexité de la division avec reste

$$a = q\,b + r \quad , \qquad 0 \leq r \leq b-1.$$

En déduire un algorithme pour le calcul du pgcd(a, b) avec complexité binaire $O(\tau^3)$.

▷

Exercice 2.2. ◁ Soit $n \geq 1$ et considérons

$$\mathbb{Z}_n := \mathbb{Z}/(n\mathbb{Z}) = \{\overline{0}, \overline{1}, \ldots, \overline{n-1}\}$$

l'anneau de congruences modulo n. Montrer que les versions "modulo n" des algorithmes classiques pour l'addition et la multiplication ont une complexité $O(\log_2(n))$ y $O(\log^2(n))$ respectivement. ▷

2. Transformée de Fourier discrète (TFD) et rapide (TFR)

La transformée de Fourier rapide est un algorithme pour le calcul de la transformée de Fourier discrète d'ordre N en temps $O(N\log(N))$. Cet algorithme remarquable fut proposé par J.W. Cooley et J.W. Tukey in 1965 [**7**] mais fut apparemment découvert avant par Runge et König en 1924 et semble-t-il aussi connu par Gauss. Les références pour cette section sont [**32**] et [**1**].

2.1. La transformée de Fourier discrète. Les coefficients de Fourier d'une fonction continue $f : \mathbb{R} \to \mathbb{R}$ de période 1 sont

$$\widehat{f}(k) = \int_0^1 f(x)e^{-2i\pi kx}\,\mathrm{d}x \quad , \quad k \in \mathbb{Z}.$$

L'inversion est donnée par la série de Fourier

$$f(x) = \sum_{k\in\mathbb{Z}} \widehat{f}(k)e^{2i\pi kx}$$

valable pour f suffisamment régulière (par exemple si $f \in \mathcal{C}^2(\mathbb{R}/\mathbb{Z})$). Soit $N \in \mathbb{N}$ et posons

$$\Omega_N = \left\{ \frac{j}{N} : j = 0, \dots, N-1 \right\} \subset [0,1];$$

la discrétisation de l'intégrale est

$$\widehat{f}_N(k) = \frac{1}{N} \sum_{x\in\Omega_N} f(x)e^{-2i\pi kx} \quad , \quad k \in \mathbb{Z}.$$

Notons que $k \mapsto U_k$ est périodique de période N car

$$\widehat{f}_N(k) = \frac{1}{N} \sum_{x\in\Omega_N} f(x)e^{-2i\pi(k+N)x} = \frac{1}{N} \sum_{x\in\Omega_N} f(x)e^{-2i\pi kx} = \widehat{f}_N(k),$$

donc cette formule produit au plus N nombres complexes différents. La *transformée de Fourier discrète (TFD)* (en anglais : *discrete Fourier transform (DFT)*) est l'opérateur $F_N : \mathbb{C}^N \to \mathbb{C}^N$ tel que pour $u = (u_1, \dots, u_N) \in \mathbb{C}^n$

$$U_k = (F_N u)_k = \sum_{j=0}^{N-1} u_j e^{-2i\pi kj/N} \quad , \quad 0 \le k \le N-1.$$

Posons $\omega := e^{-2i\pi/N}$, la matrice de la TFD est

$$F_N = [\omega_{i,j}]_{0\le i,j\le N-1} = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \cdots & \omega^{N-1} \\ 1 & \omega^2 & \omega^4 & \cdots & \omega^{2(N-1)} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & \omega^{N-1} & \omega^{2(N-1)} & \dots & \omega^{(N-1)(N-1)} \end{bmatrix}.$$

Cette matrice est symétrique (mais pas hermitienne !), et unitaire à un facteur scalaire près :

LEMME 2.1. *Pour tout $N \in \mathbb{N}$,*

$$F_N^* F_N = \overline{F}_N F_N = N\mathbf{1}_N.$$

DÉMONSTRATION.

$$(F_N)_{i,k} = \sum_{j=0}^{N-1} \omega^{-ji}\omega^{jk} = \sum_{j=0}^{N-1} \omega^{(k-i)j}.$$

Si $i = k$ alors

$$(F_N)_{i,k} = \sum_{j=0}^{N-1} 1 = N;$$

sinon $\omega^{k-i} \neq 1$ et

$$(F_N)_{i,k} = \sum_{j=0}^{N-1} \omega^{(k-i)j} = (1-\omega^{k-i})^{-1}(1-(\omega^{k-i})^N) = 0.$$

La seconde identité se suit de la première du fait que F_N est symétrique. □

La formule d'inversion est donc

$$f(x) = \sum_{k=0}^{N-1} \widehat{f}_N(k) e^{2i\pi kx} \quad , \quad x \in \Omega_N,$$

en analogie avec le cas continu.

2.2. Algorithme TFR. La TFD est une multiplication matrice-vecteur $F_N u$, donc *a priori* nécessite de N^2 multiplications complexes. La *transformée de Fourier rapide (TFR)* (en anglais : *fast Fourier transform (FFT)*) a pour but de calculer la TFD en complexité $O(N\log(N))$. C'est notre premier exemple de matrice structurée pour laquelle on sait accélérer les calculs.

Présentons l'idée pour le cas $N = 4$. Écrivons les matrices de F_2 et de F_4 explicitement

$$F_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \quad , \quad F_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & -1 & i \\ 1 & -1 & 1 & -1 \\ 1 & i & -1 & -i \end{bmatrix}.$$

Donc

$$\begin{aligned} V_0 &= v_0 + v_1, \\ V_1 &= v_0 - v_1; \end{aligned}$$

et

$$\begin{aligned} U_0 &= u_0 + u_1 + u_2 + u_3, \\ U_1 &= u_0 - iu_1 - u_2 + iu_3, \\ U_2 &= u_0 - u_1 + u_2 - u_3, \\ U_3 &= u_0 + iu_1 - u_2 - iu_3. \end{aligned}$$

Si l'on réorganise les lignes de F_4 on obtient

$$\begin{aligned} \begin{bmatrix} U_0 \\ U_2 \end{bmatrix} &= \begin{bmatrix} u_0 + u_1 + u_2 + u_3 \\ u_0 - u_1 + u_2 - u_3 \end{bmatrix} = F_2 \begin{bmatrix} u_0 + u_2 \\ u_1 + u_3 \end{bmatrix}, \\ \begin{bmatrix} U_1 \\ U_3 \end{bmatrix} &= \begin{bmatrix} u_0 - iu_1 - u_2 + iu_3 \\ u_0 + iu_1 - u_2 - iu_3 \end{bmatrix} = F_2 \begin{bmatrix} 1 & \\ & -i \end{bmatrix} \begin{bmatrix} u_0 - u_2 \\ u_1 - u_3 \end{bmatrix}. \end{aligned}$$

On a trouvé des auto-similarités nous permettant factoriser F_4 en termes de deux copies de F_2 et d'une matrice diagonale.

Présentons les détails de la TFR pour le cas général. Supposons $N = 2M$ pair, alors pour $k = 2\ell$ pair aussi on a

$$\begin{aligned} U_k &= \sum_{j=0}^{N-1} \omega^{(2\ell)j} u_j \\ &= \sum_{j=0}^{M-1} (\omega^2)^{\ell j} u_j + \omega^{2\ell M} \sum_{j=0}^{M-1} (\omega^2)^{\ell j} u_{M+j} \\ &= \sum_{j=0}^{M-1} (\omega^2)^{\ell j} (u_j + u_{M+j}). \end{aligned}$$

Pour $k = 2\ell + 1$ impair

$$\begin{aligned} U_k &= \sum_{j=0}^{N-1} \omega^{(2\ell+1)j} u_j \\ &= \sum_{j=0}^{M-1} (\omega^2)^{\ell j} (\omega^j u_j) + \omega^{(2\ell+1)M} \sum_{j=0}^{M-1} (\omega^2)^{\ell j} (\omega^j u_{M+j}) \\ &= \sum_{j=0}^{M-1} (\omega^2)^{\ell j} (\omega^j (u_j - u_{M+j})). \end{aligned}$$

Notons

$$u_I := \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{M-1} \end{bmatrix} \quad , \quad u_{II} := \begin{bmatrix} u_M \\ u_{M+1} \\ \vdots \\ u_{N-1} \end{bmatrix},$$

et

$$U_{\text{pair}} = \begin{bmatrix} U_0 \\ U_2 \\ \vdots \\ U_{N-1} \end{bmatrix} \quad , \quad U_{\text{impair}} = \begin{bmatrix} U_1 \\ U_3 \\ \vdots \\ U_{N-1} \end{bmatrix}.$$

Posons $D_M = \text{diag}(\omega^j : 0 \leq j \leq M-1)$; alors

$$\begin{bmatrix} U_{\text{pair}} \\ U_{\text{impair}} \end{bmatrix} = \begin{bmatrix} F_M & \\ & F_M \end{bmatrix} \cdot \begin{bmatrix} \mathbf{1}_M & \\ & D_M \end{bmatrix} \cdot \begin{bmatrix} \mathbf{1}_M & \mathbf{1}_M \\ \mathbf{1}_M & -\mathbf{1}_M \end{bmatrix} \cdot \begin{bmatrix} u_I \\ u_{II} \end{bmatrix}. \tag{4}$$

La permutation σ_N des lignes donnant

$$\begin{bmatrix} U_{\text{pair}} \\ U_{\text{impair}} \end{bmatrix} \mapsto \begin{bmatrix} U_I \\ U_{II} \end{bmatrix}$$

est définie par

$$\sigma_N(k) = \begin{cases} 2k & \text{si } 0 \leq k \leq M-1, \\ 2(k-M)+1 & \text{si } M \leq k \leq 2M-1; \end{cases}$$

on a donc la factorisation

$$F_N = P_{\sigma_N} \cdot \begin{bmatrix} F_M & \\ & F_M \end{bmatrix} \cdot \begin{bmatrix} \mathbf{1}_M & \\ & D_M \end{bmatrix} \cdot \begin{bmatrix} \mathbf{1}_M & \mathbf{1}_M \\ \mathbf{1}_M & -\mathbf{1}_M \end{bmatrix}.$$

Ainsi $F_N u$ peut se calculer avec deux TFD d'ordre $N/2$. Si N est une puissance de 2, on peut continuer jusqu'à se réduire au cas trivial $F_1 = 1$.

On peut représenter graphiquement la TFR par un diagramme de papillons et flèches horizontales, *voir* le cas $N = 8$ (avec $\omega = e^{-i/4}$) ci-dessous. Les papillons représentent l'application

$$\begin{bmatrix} u_I \\ u_{II} \end{bmatrix} \mapsto \begin{bmatrix} u_I + u_{II} \\ u_I - u_{II} \end{bmatrix}.$$

Les flèches horizontales représentent le transfert d'un coefficient, éventuellement multiplié par un scalaire indiqué sur la flèche. La permutation à la fin est

$$(\sigma_8 \circ (\sigma_4, \sigma_4))^{-1}.$$

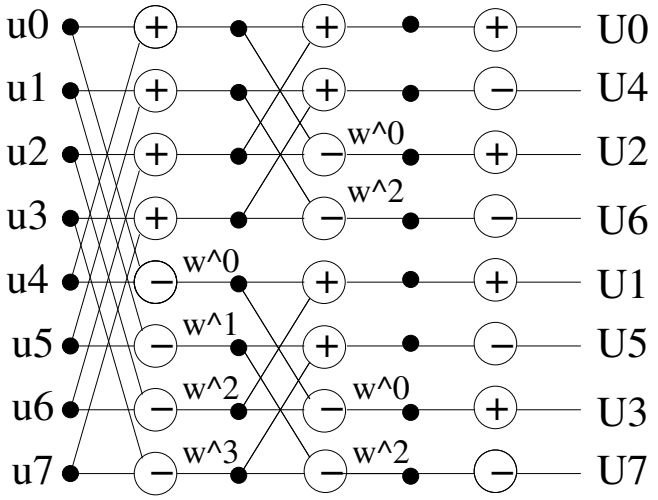


FIGURE 1. Papillons TFR

THÉORÈME 2.2. *Soit $N = 2^n$, alors la TFR évalue $F_N u$ en au plus $1,5N\log_2(N)$ opérations de $\mathbb{C}$ (additions/soustractions et multiplications par ω^j avec $0 \leq j \leq N-1$).*

DÉMONSTRATION. Notons

$$t_n := \mathcal{C}_{TFR}(\mathbb{C}; 2^n)$$

la complexité de la TFR en dimension N. Pour obtenir $u_I + u_{II}$ et $u_I - u_{II}$ on fait N additions/soustractions et pour la multiplication $D_M(u_I - u_{II})$ on fait $M = N/2$ multiplications de $\mathbb{C}$; puis pour les TFR en dimension M on fait $2t_{n-1}$ opérations de $\mathbb{C}$. On a l'inégalité

$$t_n \leq 2t_{n-1} + \frac{3}{2}2^n.$$

Par induction et le fait que $t_0 = 0$ on trouve

$$t_n \leq \frac{3}{2}n2^n = 1,5N\log_2(N).$$

□

2.3. Convolution de vecteurs. Il y a une relation étroite entre la TFD et l'évaluation et l'interpolation de polynômes. Pour $u = (u_0, \dots, u_{N-1}) \in \mathbb{C}^N$ considérons le polynôme

$$P_u = \sum_{j=0}^{N-1} u_j x^j.$$

Alors

$$F_N u = \begin{bmatrix} P_u(1) \\ P_u(\omega) \\ \vdots \\ P_u(\omega^{N-1}) \end{bmatrix}$$

est l'évaluation de P_u en les puissances successives de ω, tandis que

$$\frac{1}{N} F_N^* \begin{bmatrix} v_0 \\ v_1 \\ \vdots \\ v_{N-1} \end{bmatrix} = \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{bmatrix}$$

donne les coefficients de l'unique polynôme P de degré au plus $N-1$ tel que $P(\omega^j) = v_j$. Donc on identifie la TFD avec l'application évaluation

$$F_N : \mathbb{C}[x]_{\leq N-1} \to \mathbb{C},$$

et son inverse $\frac{1}{N} F_N^*$ avec l'interpolation.

Pour des vecteurs $u, v \in \mathbb{C}^N$, la *convolution* est définie comme

$$u * v = \left[\sum_{j=0}^{N-1} u_j v_{k-j} : 0 \leq k \leq 2N-1 \right]^T \in \mathbb{C}^{2N}.$$

On a

$$P_{u*v} = P_u \cdot P_v.$$

Une des principales propriétés de la TFD est qu'elle traduit la convolution en multiplication :

THÉORÈME 2.3. *Soient $u, v \in \mathbb{C}^N$, alors*

$$u * v = \frac{1}{2N} F_{2N}^* (F_{2N} u \cdot F_{2N} v).$$

DÉMONSTRATION. On a $P_{u*v}(\omega^j) = P_u(\omega^j) \cdot P_v(\omega^j)$ donc

$$F_{2N}(u * v) = F_{2N} u \cdot F_{2N} v;$$

le résultat s'obtient en appliquant l'inverse de F_{2N} à cette identité. □

Joint à la TFR, ceci entrane un algorithme rapide pour multiplier sur $\mathbb{C}$ deux polynômes

$$f = f_0 + \cdots + f_{N-1} x^{N-1} \quad , \quad g = g_0 + \cdots + g_{N-1} x^{N-1} \in \mathbb{C}[x].$$

COROLLAIRE 2.4. *Soient $f, g \in \mathbb{C}[x]$ deux polynômes de degré au plus $N-1$ (N étant une puissance de 2), alors $f \cdot g$ peut se calculer en $4,5N \log_2(N) + O(N)$ opérations de $\mathbb{C}$.*

DÉMONSTRATION. Suivant le théorème de convolution, h se calcule avec 3 TFD en dimension $2N$, N produits et une division, la complexité est

$$3 \times 1,5(2N) \log_2(2N) + N + 1 = \frac{9}{2} N \log_2(N) + O(N).$$

□

Notons que l'algorithme naïf

$$h = f \cdot g = \sum_{k=0}^{2N-1} \left(\sum_{j=0}^{N-1} f_j g_{k-j} \right) x^k$$

comprend $2N^2 + O(N)$ opérations de $\mathbb{C}$.

2.4. La TFR en binaire. Les constructions précédentes se généralisent à un anneau commutatif A quelconque à la place de $\mathbb{C}$, muni d'une racine principale d'ordre N de 1, qui jouera le rôle de ω.

Définition 2.5. Soit A un anneau commutatif et $N \in \mathbb{N}$. Un élément $\omega \in A$ est une racine N-ème *principale* de l'unité si

(1) $\omega^N = 1$;

(2) $\omega^M - 1$ n'est pas un diviseur de 0, pour $1 \leq k \leq N-1$.

Notons que dans le cas d'un *domaine* A, cette définition coïncide avec celle de racine N-ème *primitive* de l'unité. La condition (2) est nécessaire pour que le lemme 2.1 reste valable (et avec la même démonstration) dans ce cadre plus général.

On appliquera cette idée aux anneaux de congruences de type

$$A_N = \mathbb{Z}_{2^N+1} := \mathbb{Z}/(2^N+1)\mathbb{Z}$$

pour $N = 2^n$. Le lemme suivant nous garantit l'existence de racines principales :

Lemme 2.6. *Soit $N = 2^n$, alors $2 \in \mathbb{Z}_{2^N+1}$ est une racine $2N$-ème principale de l'unité.*

Démonstration. On a

$$2^{2N} \equiv (-1)^2 = 1 \pmod{2^N+1}$$

donc la première condition est vérifiée. Pour chaque $1 \leq M \leq 2N-1$, la condition que $2^M - 1$ ne soit pas un diviseur de 0 dans $\mathbb{Z}_{2^N+1}$ est équivalente à ce que

$$\gcd(2^M - 1, 2^N + 1) = 1. \tag{5}$$

Maintenant pour tout $a \geq 2$ et $k, \ell \geq 1$ on a

$$\gcd(a^k - 1, a^\ell - 1) = a^{\gcd(k,\ell)} - 1;$$

en appliquant cela à $a = 2$, $k = M$ et $\ell = 2N$ on obtient

$$\gcd(2^M - 1, 2^N + 1) | \gcd(2^M - 1, 2^{2N} - 1) = 2^{\gcd(M,2N)} - 1 | 2^N - 1$$

car $\gcd(M, 2N) | N$ puisque $M < 2N$ et que $2N$ est une puissance de 2. Donc

$$\gcd(2^M - 1, 2^N + 1) | 2 = (2^N + 1) - (2^N - 1)$$

et on en déduit $\gcd(2^M - 1, 2^N + 1) = 1$ car $2^M - 1$ est impair. □

Ceci nous permet de construire la TFD sur $\mathbb{Z}_{2^{N/2}+1}$ pour $\omega = 2$ racine principale d'ordre $N = 2^n$; par la suite on estime la complexité binaire de la TFR correspondante.

Théorème 2.7. *Soit $N = 2^n$, alors pour $u \in \mathbb{Z}_{2^{N/2}+1}$ la TFR évalue $F_N u$ en au plus $1,5N^2 \log_2(N) + O(N \log_2(N))$ opérations binaires.*

DÉMONSTRATION. La TFR calcule $F_N u$ en $1,5N\log(N)$ opérations de $\mathbb{Z}_{2^{N/2}+1}$ de type addition/soustraction et multiplication par 2^k pour $0 \leq k \leq N-1$.

Chaque addition/soustraction coûte $2(N/2+O(1)) = N+O(1)$ opérations binaires (exercice (2.2). En outre, tout élément $a \in \mathbb{Z}_{2^{N/2}+1}$ est représenté par au plus N bits et peut donc s'écrire comme

$$a = \sum_{j=0}^{N/2} a_j 2^j$$

avec $a_j = 0$ ou 1. Notons $r_{N/2}(j+k)$ le reste de diviser $j+k$ par $N/2$, alors

$$2^k \cdot a = \sum_{j=0}^{N/2} \pm a_j 2^{r_{N/2}(j+k)}$$

est un déplacement suivi d'un certain changement de signe. On peut donc écrire $2^k \cdot a$ comme différence de deux éléments de $\mathbb{Z}_{2^{N/2}+1}$, ce qui peut se calculer aussi en complexité $N+O(1)$. La complexité totale est

$$(1,5N\log(N))(N+O(1)) = 1,5N^2\log_2(N) + O(N\log_2(N)).$$

□

La même estimation s'applique au calcul de l'inverse $N^{-1}\overline{F}_N$. La matrice $\overline{F}_N$ est la TFD associée à la racine principale

$$2^{-1} = -2^{N/2} = 2^{N-1} \in \mathbb{Z}_{2^{N/2}+1};$$

le coût du calcul de $v = \overline{F}_N u$ est le même que celui de la TFR, en rajoutant les multiplications

$$N^{-1}v_i = 2^{(N-1)n}v_i \quad , \quad 1 \leq i \leq N$$

qui coûtent $O(N)$ chacune ; la complexité totale reste en $1,5N^2\log_2(N)+O(N\log_2(N))$ opérations binaires.

2.5. L'algorithme de Schönhage-Strassen pour la multiplication d'entiers. Maintenant on se tourne vers une application importante de la TFR : la multiplication rapide d'entiers. Soient

$$a, b \in \mathbb{N}$$

deux entiers de longueur binaire $\leq N = 2^n$. L'algorithme de multiplication longue calcule le produit

$$c = a \cdot b$$

en $O(N^2)$ opérations binaires ; on montrera que ceci peut se faire en complexité quasi-linéaire

$$O(N\log^2(N)\log(\log(N))).$$

L'algorithme qu'on présentera est une légère simplification de celui dû à Schönhage et Strassen [**33**] qui fait cette tâche en complexité $O(N\log(N)\log(\log(N)))$.

Classiquement on note $M(N)$ la complexité de multiplier deux entiers de longueur au plus N. Le théorème à démontrer est donc

THÉORÈME 2.8. $M(N) = O(N\log^2(N)\log(\log(N)))$.

Soient

$$t, \ell \in \mathbb{N}$$

des puissances de 2 telles que $t\ell = N$ et écrivons a et b en base 2^ℓ :

$$a = \sum_{j=0}^{t-1} a_j (2^\ell)^j \quad , \quad b = \sum_{j=0}^{t} b_j (2^\ell)^j$$

avec $0 \leq a_j, b_j \leq 2^\ell - 1$. Considérons les polynômes

$$P_a := \sum_{j=0}^{t-1} a_j x^j \quad , \quad P_b = \sum_{j=0}^{t} b_j x^j \in \mathbb{Z}[x].$$

Si on arrive à calculer rapidement le produit $P_a \cdot P_b$, on peut calculer le produit $a \cdot b$ comme

$$a \cdot b = (P_a \cdot P_b)(2^\ell).$$

Ceci se fait rapidement puisque l'évaluation en 2^ℓ consiste à faire une série de shifts et d'additions, ce qui prend $O(N \log(N))$ opérations binaires.

Intuitivement, on voudrait calculer la convolution $P_a \cdot P_b$ *via* la TFR, mais il y a un point délicat parce que la convolution repose elle même sur des multiplications d'entiers. Seulement par un choix convenable des paramètres t, ℓ la taille de ces entiers diminue et nous permet de monter une récurrence. Posons

$$T(N)$$

la complexité de multiplier deux entiers de longueur au plus N modulo $2^N + 1$. Trivialement

$$M(N) \leq T(2N)$$

puisque si $\tau(a), \tau(b) \leq N$ alors $0 \leq a \cdot b \leq 2^{2N}$ coïncide avec son reste modulo $2^{2N} + 1$. Soit

$$Q = P_a \cdot P_b = \sum_{j=0}^{2t-2} c_j x^j.$$

avec $c_j = \sum_{i=0}^{t-1} a_i b_{j-i}$ donc

$$0 \leq c_j \leq t 2^{2\ell}.$$

Pour calculer $c_j \pmod{2^N + 1}$ il suffit donc de le faire modulo $t(2^{2\ell} + 1)$, ou ce qui est équivalent grâce au théorème chinois, modulo t et modulo $2^{2\ell} + 1$ séparément.

Le calcul de Q modulo t se fait par l'algorithme classique, tandis que le calcul modulo $2^{2\ell} + 1$ se fait avec la TFR. Remarquons que 2 est une racine 4ℓ-ème de l'unité modulo $2^{2\ell} + 1$, donc le calcul de Q *via* la TFR en dimension $2t$ sera possible si

$$4\ell \geq \deg(Q) + 1 = 2t.$$

On prends donc le ℓ minimal satisfaisant aussi $t\ell = N$, donc

$$\ell := 2^{\lfloor n/2 \rfloor} \quad , \quad t := 2^{[n/2]}.$$

La racine $2t$-ème principale de l'unité utilisée est $\omega = 4$ pour n pair et $\omega = 2$ pour n impair. Voici l'algorithme complet :

Algorithme : multiplication d'entiers de Schönhage-Strassen.
Entrée : $a, b \in \mathbb{N}$ **entiers de longueur bit** $\leq N = 2^n$ **;**
Sortie : $a \cdot b$ **modulo** $2^N + 1$**.**

(1) $\ell \leftarrow 2^{\lfloor n/2 \rfloor}$**,** $t \leftarrow N/\ell$ **;**

(2) Écrivons

$$a = \sum_{j=0}^{t-1} a_j (2^\ell)^j \quad , \quad b = \sum_{j=0}^{t} b_j (2^\ell)^j$$

avec $0 \leq a_j, b_j \leq 2^\ell - 1$**, alors**

$$P_a \leftarrow \sum_{j=0}^{t-1} a_j x^j, P_b \leftarrow \sum_{j=0}^{t} b_j x^j;$$

(3) calcul de

$$R \leftarrow P_a \cdot P_b \pmod{t}$$

avec l'algorithme de multiplication longue ;

(4) calcul de

$$S \leftarrow P_a \cdot P_b \pmod{2^{2\ell}+1}$$

avec la TFR d'ordre $2t$, et multiplication d'entiers modulo $2^{2\ell}+1$ en utilisant cet algorithme récursivement ;

(5) $Q \leftarrow P_a \cdot P_b = (2^{2\ell}+1)(R - S \pmod{t}) + R$ **;**

(6) $a \cdot b \leftarrow Q(2^\ell) \pmod{2^N+1}$**.**

fin.

Estimons la complexité de cet algorithme. Le pas (3) est la multiplication de deux polynômes de degré $\leq t-1$ modulo t ; ceci demande $O(t^2)$ opérations modulo t, soit

$$O(t^2 \log^2(t)) = O(N \log^2(N))$$

opérations binaires. La TFR du pas (4) se fait en

$$O(\ell^2 \log(\ell)) = O(N \log^2(N))$$

opérations binaires. Ensuite il faut multiplier deux à deux modulo $2^{2\ell}+1$ les $2t$ coefficients de la TFD, ce qui demande

$$2tT(2\ell)$$

opérations binaires. La complexité du pas (5) est $O(N)$ et peut donc se négliger. Au total on obtient la récurrence

$$T(N) \leq 2tT(2\ell) + O(N \log^2(N)). \tag{6}$$

PROPOSITION 2.9. $T(N) = O(N \log^2(N) \log(\log(N)))$.

DÉMONSTRATION. On fait récurrence sur N. Posons $\widehat{T}(N) := T(N)/N$, alors la récurrence (6) se récrit en

$$\widehat{T}(N) \leq \frac{2tT(2\ell)}{N} + c_1 \log^2(N) = 4\widehat{T}(2\ell) + c_1 \log^2(N) \leq 4\widehat{T}(\sqrt{4N}) + c_1 \log^2(N)$$

pour un certain $c_1 > 0$. Soit N_0 tel que pour $N \geq N_0$ on ait $\sqrt{4N} \leq N^{2/3}$, alors l'hypothèse inductive $\widehat{T}(2\ell) \leq c_2 \log^2(2\ell) \log(\log(2\ell))$ entraîne

$$\begin{aligned}
\widehat{T}(N) &\leq 4c_2 \log^2(\sqrt{4N}) \log(\log(\sqrt{4N})) + c_1 \log^2(N) \\
&= c_2 \log^2(4N) \log(\frac{2}{3}\log(N)) + c_1 \log^2(N) \\
&\leq c_2 \log^2(N) \left(\frac{\log^2(4N)}{\log^2(N)} (\log(\log(N)) - \log(\frac{3}{2})) + \frac{c_1}{c_2} \right) \\
&\leq c_2 \log^2(N) \log(\log(N)
\end{aligned}$$

pour c_2 suffisamment grand et $N \gg 0$. □

REMARQUE 2.10. La complexité du pas (3) s'améliore en

$$O(t^{1,59} \log^2(t)) = O(N^{0,8} \log^2(N))$$

en utilisant par exemple la multiplication de polynômes de Karatsuba à la place de la multiplication longue. Dans le pas (4) on peut utiliser une "wrapped convolution" [**1**] à la place de la convolution habituelle.

Ces deux changements permettent d'aboutir à une complexité $O(N\log(N)\log(\log(N)))$ comme dans la version originale de l'algorithme, *voir* [**1**] pour les détails.

Les graphiques suivants illustrent la performance *pratique* des algorithmes de multiplication longue, de Karatsuba, et de Schönhage-Strassen. Ces expériences furent conduites par Emiliano Kargieman en utilisant la librairie GMP de précision multiple (`http//www.sox.com/gmp`).

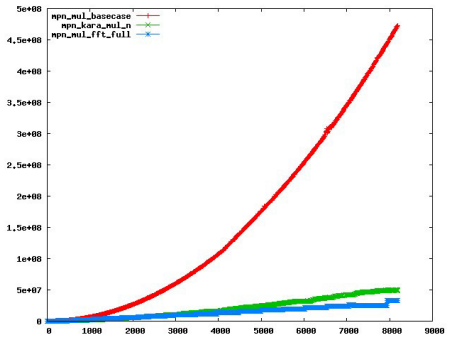


FIGURE 2. Comparaison des trois méthodes (de 8 à 8192 bits en incréments de 8 bits)

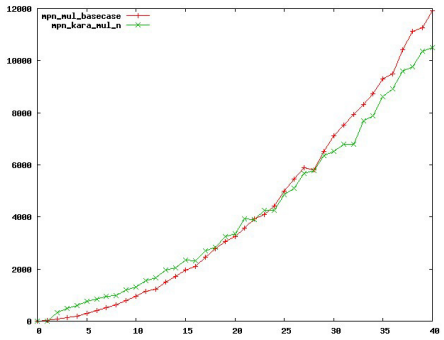


FIGURE 3. Quand Karatsuba gagne contre la multiplication longue (de 0 à 40 bits en incréments de 1 bit)

On voit que Karatsuba commence à gagner contre la multiplication longue pour des nombres de 23 bits. La TFR dépasse la multiplication longue pour des nombres d'à partir 340 bits approximativement, mais pour qu'elle gagne contre toutes les autres méthodes il faut que les nombres soient d'au moins 2300 bits ! Si l'on suppose que les complexités $2N^2$ et $cN\log_2(N)\log_2(log_2(N))$ de la multiplication longue et de l'algorithme de Schönhage-Strassen reflètent le temps d'exécution réel, on conclut que la constante est au moins

$$c \geq \frac{2\cdot 340^2}{340\log(340)\log(\log(340))} = 66,17.$$

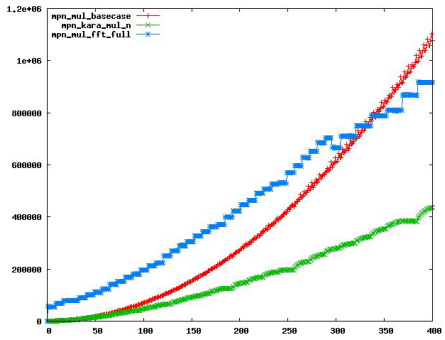


FIGURE 4. Quand la TFR gagne à la multiplication longue (de 0 à 400 bits en incréments de 1 bit)

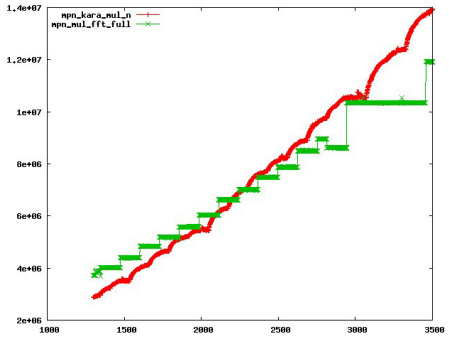


FIGURE 5. Quand la TFR gagne contre Karatsuba (de 1300 à 3500 bits en incréments de 1 bit)

3. Exercices

EXERCICE 2.3. ◁ Le but de cet exercice est de montrer que la TFR est numériquement stable. Soit $N = 2^k$, on va utiliser la factorisation de $F_N \in \mathbb{C}^{N\times N}$ comme

$$F_N = P \cdot \begin{bmatrix} F_{N/2} & \\ & F_{N/2} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{1}_{N/2} & \\ & D_{N/2} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{1}_{N/2} & \mathbf{1}_{N/2} \\ \mathbf{1}_{N/2} & -\mathbf{1}_{N/2} \end{bmatrix}$$

où P est une matrice de permutations et $D_{N/2} = \mathrm{diag}(1, \omega, \omega^2, \ldots, \omega^{N/2-1}) \in \mathbb{C}^{N/2\times N/2}$ avec $\omega = e^{-2i\pi/N}$.

Soit $\mathcal{F}$ un système de flottants avec base 2 et précission t. Pour une entrée

$$u \in (\mathcal{F} + i\mathcal{F})^N,$$

on note $TFR_{\mathcal{F}}(u)$ le résultat calculé de la TFR relatif à l'arithmétique de $\mathcal{F} + i\mathcal{F}$.

(1) Montrer que les matrices dans la factorisation de F_N sont unitaires, sauf pour un facteur scalaire ; en déduire son conditionnement.

(2) Soit $A \odot v$ le résultat calculé d'une multiplication matrice-vecteur $A \cdot v$ relatif à l'arithmétique de $\mathcal{F} + i\mathcal{F}$. Montrer que si $A \cdot v$ consiste en *une seule* opération par ligne, alors

$$A \odot v = A \cdot (v + e)$$

avec $|e|_2 \leq 2^{-t}\kappa_2(A)|v|_2$. Il faut d'abord montrer que l'erreur relatif d'une opération flottante complexe est majoré par

$$|u * v - \mathrm{fl}(u * v)| \leq 2^{-t}|u * v|$$

pour $u, v \in \mathcal{F} + i\mathcal{F}$.

(3) Analyse *a posteriori* : soit $e_N \in \mathbb{C}^N$ tel que

$$TFR_{\mathcal{F}}(u) = F_N \cdot (u + e_N)$$

et soit $\rho_N \geq 0$ tel que

$$|e_N|_2 \leq \rho_N |u_N| \quad \text{pour tout } u \in \mathbb{C}^N.$$

Montrer que $\rho_1 = 0$, et que pour $N = 2^k$ avec $k \geq 1$ on a

$$\rho_N + 1 \leq (1 + 2^{-t})^2(\rho_{N/2} + 1).$$

En conclure que

$$|e_N|_2 \leq ((1 + 2^{-t})^{2k} - 1)|u|_2.$$

(4) Démontrer l'inégalité

$$(1 + 2^{-t})^\ell \leq 1 + \ell 2^{1-t} \qquad \text{para } 0 \leq \ell \leq 2^{t-1}.$$

(*Indication :* utiliser le développement de Taylor $f(x) = f(0) + f'(0)x + f''(\xi)x^2/2$ pour $x > 0$ et un certain $0 < \xi < x$, appliqué à $f(x) = (1 + x)^\ell$.) En déduire que

$$|e_N|_2 \leq k 2^{2-t}|u|_2 \qquad \text{para } N \leq 2^{2^{t-2}}.$$

(5) Anlyse *a priori* : montrer que

$$\frac{|TFR_{\mathcal{F}}(u) - F_N \cdot u|_2}{|u|_2} \leq N^{1/2} \log_2(N) 2^{2-t}, \qquad \text{para } N \leq 2^{2^{t-2}};$$

en particulier la perte de précission est de $\log_2(N) + \log_2(\log_2(N)) + O(1)$ bits.

▷

Chapitre 3

Représentation des données

1. Représentation exacte

L'avantage évident des algorithmes travaillant avec la représentation exacte des données c'est qu'on n'a pas de perte de précision ; ou tout au moins la seule instabilité est celle induite par la précision de l'entrée et non pas par les troncages internes de l'algorithme. Le désavantage est que la taille des calculs intermédiaires peut exploser, conduisant souvent à une augmentation sensible du temps d'exécution et/ou de l'occupation de place mémoire.

La multiplication des entiers est, à une constante près, du même coût que la division des entiers et du calcul de réciproques. Le coût du gcd est $O(\log(N)$ fois le coût de la multiplication. Introduisons un peu de notation :

$M(N)$	la complexité de la multiplication de deux entiers de taille N ;
$D(N)$	la complexité de la division avec reste de deux entiers de taille N ;
$R(N)$	la complexité de N bits du réciproque d'un entier de taille N ;
$GCD(N)$	la complexité du gcd de deux entiers de taille N.

L'algorithme de Schönhage-Strassen montre que

$$M(N) = O(N \log(N) \log(\log(N))).$$

On a

Théorème 1.1. *$M(N)$, $R(N)$ et $D(N)$ sont comparables à un facteur constant près ; en particulier*

$$R(N), D(N) = O(N \log(N) \log(\log(N))).$$

Dans les exercices on fera quelques unes de ces comparaisons ; on renvoie à [**1**, Ch. 9] pour la démonstration de la division avec reste. La complexité du gcd nécessite d'une analyse plus fine, qui est développée dans la même référence :

Théorème 1.2. *$GCD(N) = O(M(N) \log(N))$.*

Soit $\xi \in \mathbb{Q}$ un nombre rationnel et soit $\xi = p/q$ sa *représentation réduite* avec $p \in \mathbb{Z}$ et $q \in \mathbb{N}^\times$ premiers entre eux. La *hauteur exponentielle* de ξ est

$$H(\xi) := \max\{|p|, q\}.$$

On considérera surtout la *hauteur (logarithmique)* de ξ définie par

$$h(\xi) := \log_2(H(\xi)) = \max\{\log_2(|p|), \log_2(q)\} \in \mathbb{R}_+.$$

Ceci est la hauteur considérée usuellement en théorie des nombres, qu'on préférera à la complexité binaire

$$\tau(\xi) := \tau(p) + \tau(q) = \lfloor \log_2(|p|) \rfloor + 1 + \lfloor \log_2(q) \rfloor + 1.$$

Ces notions sont comparables :

$$h(\xi) \leq \ell(\xi) \leq 2\, h(\xi) + 2$$

mais la hauteur a un comportement plus agréable par rapport aux opérations arithmétiques. Soient $\xi, \eta \in \mathbb{Q}^\times$, alors

(1) $h(\xi+\eta) \leq h(\xi) + h(\eta) + \log(2)$.
Si $\xi, \eta \in \mathbb{Z}$, alors $h(\xi+\eta) \leq \max\{h(\xi), h(\eta)\} + 1$.

(2) $h(\xi \cdot \eta^{\pm 1}) \leq h(\xi) + h(\eta)$.

Comme conséquence de notre étude du coût des opérations arithmétiques entre nombres entiers, on voit que la complexité binaire des opérations arithmétiques entre nombres rationnels de hauteurs bornées par h est

$$O(h \log(h) \log(\log(h)))$$

pour l'addition, la soustraction, la multiplication et la division. Si l'on veut travailler uniquement avec des expressions réduites, il faut simplifier les facteurs redondants, ce qui demande un calcul de gcd. Le coût binaire des opérations arithmétiques avec représentation réduite des rationnels est donc

$$O(h \log^2(h) \log(\log(h))).$$

La notation $f = O^*(g)$ veut dire "à des facteurs logarithmiques près", c'est-à-dire $f \leq c\, g \log^\nu(g)$ pour certains $c, \nu > 0$.

COROLLAIRE 1.3. *Soit A un algorithme sur $\mathbb{Q}$, et notons $h_A(\tau)$ la hauteur maximale des calculs intermédiaires effectués par A sur une entrée de taille τ, alors*

$$\mathcal{C}_A(\tau) \leq O^*(\mathcal{C}_A(\mathbb{Q}; \tau) \cdot h_A(\tau)).$$

Posons $\mathbb{Q}_{\leq H} := \{\xi \in \mathbb{Q} : H(\xi) \leq x\}$ la filtration de la droite rationnelle par la hauteur. Son aspect est vachement différent de la droite flottante. Pour $H = 4$ les points sont

$$\frac{0}{1}, \frac{1}{1}; \frac{1}{2}, \frac{2}{1}; \frac{1}{3}, \frac{2}{3}, \frac{3}{2}, \frac{3}{1}; \frac{1}{4}, \frac{3}{4}, \frac{4}{3}, \frac{4}{1};$$

et les négatifs correspondants. La figure suivante représente ces rationnels :

FIGURE 1. Les rationnels de hauteur exponentielle au plus 4.

EXERCICE 3.1. ◁ Un comptage naïf donne l'estimation

$$\mathrm{Card}(\mathbb{Q}_{\leq H}) \leq (2H+1)H;$$

le but de cet exercice est de démontrer l'asymptotique

$$\mathrm{Card}(\mathbb{Q}_{\leq H}) = \frac{12}{\pi^2} H^2 + O(H \log(H)) = 1,216 H^2 + O(H \log(H)). \tag{7}$$

Un couple $z = (p, q) \in \mathbb{Z}^2 \setminus \{(0,0)\}$ est *visible* si l'intérieur du segment $\overline{0\, z}$ ne contient aucun point de $\mathbb{Z}^2$, ce qui équivaut à ce que p, q soient premiers entre eux.

(1) Montrer que $\mathbb{Q}^\times$ s'identifie aux points visibles depuis le $(0,0)$ quand on "regarde" vers le nord. Dans ce modèle, la hauteur correspond à la norme ℓ^∞.

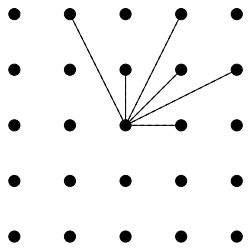


FIGURE 2. Points visibles depuis le $(0,0)$

(2) Posons

$$m(t) := \#\{(p,q) \in \mathbb{Z}^2 \setminus \{(0,0)\} \,:\, |p|,|q| \leq t\},$$

$$n(t) := \#\{(p,q) \in \mathbb{Z}^2 \setminus \{(0,0)\} \,:\, \gcd(p,q)=1,\ |p|,|q| \leq t\}.$$

Montrer que $m(t) = (2t+1)^2 - 1$ et que

$$m(t) = \sum_{d \geq 1} n(t/d).$$

(3) Soit $\mu : \mathbb{N} \to \mathbb{N}$ la *fonction de Möbius*, définie par $\mu(d) = (-1)^k$ si $d = p_1 \cdots p_k$ avec $p_1, \ldots, p_k$ nombres premiers différents deux à deux, et $\mu(d) = 0$ sinon. Prouver la formule d'inversion

$$n(t) = \sum_{d \geq 1} \mu(d)\, m(t/d).$$

(4) En déduire l'asymptotique pour $t \to \infty$

$$n(t) = 4\Big(\sum_{d \geq 1} \frac{\mu(d)}{d^2}\Big)\, t^2 + O(t \log(t)).$$

(5) Soit

$$\zeta(s) := \sum_{n \geq 1} n^{-s} = \prod_{p \text{ premier}} \frac{1}{1-p^{-s}}, \quad (Re(s) > 1)$$

la fonction zeta de Riemann. Observer que

$$\sum_{d \geq 1} \frac{\mu(d)}{d^2} = \frac{1}{\zeta(2)};$$

et que $\mathrm{Card}(\mathbb{Q}_{\leq H}) = \frac{1}{2}\, n(t) - 1$; en déduire (7), en utilisant que $\zeta(2) = \pi^2/6$.

▷

2. Nombres flottants

Références pour cette section : [**10**], [**32**]. La forme générale des nombres flottants est :

$$f = \pm .d_1 d_2 \ldots d_t \times \beta^e, \quad 0 \leq d_i < \beta, \quad d_1 \neq 0, \quad L \leq e \leq U; \tag{8}$$

$d_1 d_2 \ldots d_t$ est la *mantisse*, t la *précision*, β la *base*, L le *dépassement inférieur* (en anglais : *underflow*) et U le *dépassement supérieur* (en anglais : *overflow*). On impose $0 \leq d_i < \beta$ et $d_1 \neq 0$, ce qui assure l'unicité de l'expression.

Pour $\beta \geq 2, t \geq 1$ et $(L,U) \in \mathbb{Z}^2$ donnés, on désigne par

$$F(\beta, t, L, U)$$

l'ensemble des flottants associés plus le 0. Prenons un cas particulier très simple :

$$\beta = 2, \quad t = 3, \quad U = -1, \quad L = 1. \tag{9}$$

Par conséquent :

$$1 \leq d_1 \leq \beta - 1 \Longrightarrow d_1 = 1.$$

Les mantisses possibles sont

$$0.100, \quad 0.101, \quad 0.110, \quad 0.111,$$

soit les fractions $1/2, 5/8, 3/4, 7/8$, puis on a le droit de les multiplier par ± 2, ± 1, ou $\pm 1/2$. Voici la représentation graphique de ces flottants :

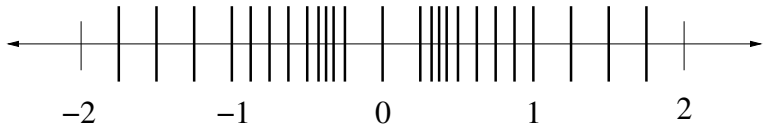


FIGURE 3. Les flottants définis par les données (9).

Cette droite est pleine de trous, on a donc besoin d'une fonction arrondi. Soit $F := F(\beta, l, L, U)$, si $f \in F \setminus \{0\}$ alors

$$m \leq |f| \leq M$$

avec

$$m = \beta^{L-1}, \quad M = \beta^U(1 - \beta^{-t}). \tag{10}$$

Autre nom de m : ε de la machine, et de M : capacité de la machine.

EXERCICE 3.2. ◁ Vérifier les formules dans (10). ▷

Valeurs typiques de $(\beta,\ t,\ L,\ U)$: IBM 370 (une antiquité) : $(16, 14, -64, 43)$; Cray 1 (pas tout jeune, mais nettement plus récent) : $(2, 48, -16384, 8191)$.

Aujourd'hui le *standard IEEE pour l'arithmétique binaire* est le plus répandu. Il est utilisé dans les workstations Sun, DEC, HP, IBM, et dans toutes les PCs. L'arithmétique IEEE inclut deux systèmes de flottants : *simple précision* (32 bits) et *double précision* (64 bits).

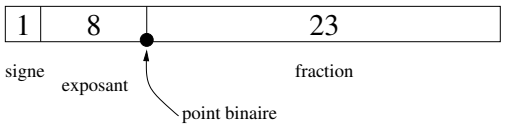


FIGURE 4. Standard IEEE simple précision.

Dans le format simple précision, le signe s est codé par 1 bit, l'exposant e par 8 bits, et la fraction q par les 23 bits restants ; le nombre codé est

$$(-1)^s(1 + q)2^{e-127}.$$

Puisque la base es 2, le premier bit de la mantisse sera toujours 1, donc on n'a pas besoin de le coder. En passant à la représentation standard des flottants (7), on a

$$t = 24, \quad \beta = 2, \quad L = -126, \quad U = 129.$$

L'erreur relative maximale est

$$2^{-24} \cong 6 \cdot 10^-8$$

et le rang va de $2^{-127} \cong 6 \cdot 10^{-38}$ jusqu'à $2^{129}(1 - 2^{-24}) \cong 7 \cdot 10^{39}$.

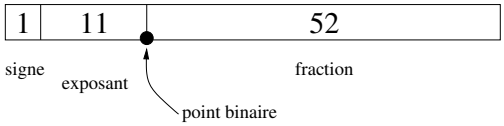


FIGURE 5. Standard IEEE double précision.

Dans le format double précision, le signe s est codé par 1 bit, l'exposant e par 11 bits, et la fraction q par les 52 bits restants ; le nombre codé est

$$(-1)^s(1+q)2^{e-1023}.$$

Dans la représentation standard des flottants on a

$$t = 53, \quad \beta = 2, \quad L = -1022, \quad U = 1026.$$

L'erreur relative maximale est

$$2^{-53} \cong 6 \cdot 10^-16$$

et le rang va de $2^{-1022} \cong 2 \cdot 10^{-308}$ jusqu'à $2^{129}(1-2^{-24}) \cong 7 \cdot 10^{309}$.

Le modèle d'arithmétique des flottants, et en particulier du standard IEEE, est assez compliqué dans les détails. Ici on présentera une version simplifiée mais suffisante à nos besoins.

Une fonction

$$\mathrm{fl} : \mathbb{R} \to F(\beta, t, L, U) \cup \{E\}$$

qui envoie les réels sur les flottants est une fonction *arrondi* si elle vérifie :

(1) $\mathrm{fl}([-M, M]) \subset F(\beta, t, L, U)$;
(2) pour $x \in \mathbb{R}$ soient $f, f' \in F(\beta, t, L, U)$ tels que $x \in [f, f']$, alors $\mathrm{fl}(x) \in [f, f']$ (en particulier, fl laisse F invariant) ;
(3) si $|x| > M$ alors $\mathrm{fl}(x) = E$, si $|x| < m$ alors $\mathrm{fl}(x) = 0$; dans les deux cas on dit que x dépasse la capacité de la machine.

Dans le cas de l'*arithmétique arrondie* (qui est celle utilisée par le standard IEEE), $\mathrm{fl}(x)$ est défini comme le nombre dans F le plus proche de x ; si x est exactement à mi-chemim entre deux flottants, on choisit celui qui est le plus proche de 0.

Soit $\square$ une des quatre opérations arithmétiques $+, -, \times, /$; le modèle du calcul revient à supposer que la version "ordinateur" de chacune de ces quatre opérations est donnée par $\mathrm{fl}(a \square b)$.

EXEMPLE 2.1. Addition dans le système flottant jouet (9) :

$$\begin{aligned} 0.100 \times 2^1 + 0.110 \times 2^{-1} &= 0.10000 \times 2^1 + 0.00110 \times 2^1 \\ &= 0.10110 \times 2^1 \\ &\to \mathrm{fl}(0.100 \times 2^1 + 0.110 \times 2^{-1}) = 0.101 \times 2^1 \end{aligned}$$

Cependant les opérations sur les flottants ne sont pas nécessairement associatives. Dans le système jouet avec $\beta = 2$, $t = 3$, $L = -1$ et $U = 2$, si on utilise l'arithmétique tronquée :

$$\mathrm{fl}(0.1 \times 2^{-1} + 0.100 \times 2^2) = \mathrm{fl}((0.000100 + 0.100000) \times 2^2) = 0.100 \times 2^2;$$

et donc

$$\mathrm{fl}(\mathrm{fl}((0.1 \times 2^{-1} + 0.100 \times 2^2) + \mathrm{fl}(-0.100 \times 2^2))) = 0.$$

En revanche,

$$\mathrm{fl}(0.100 \times 2^2 + -0.100 \times 2^2) = 0,$$

si bien que

$$\mathrm{fl}(0.100 \times 2^{-1} + \mathrm{fl}(0.100 \times 2^{2} + (-0.100 \times 2^{2}))) = 0.100 \times 2^{-1}.$$

On peut montrer que l'opérateur fl satisfait

$$\mathrm{fl}(x) = x(1+\varepsilon), \quad |\varepsilon| \leq \mu,$$

avec $\mu = \mu(F) = \beta^{1-t}/2$. Par conséquent $\mathrm{fl}(a \square b) = (a \square b)(1+\varepsilon)$, $|\varepsilon| \leq \mu$ et donc en termes d'erreur relative, chaque opération arithmétique *individuelle* en flottant a un comportement satisfaisant :

$$\frac{|\mathrm{fl}(a \square b) - (a \square b)|}{a \square b} \leq \mu, \quad \text{si } a \square b \neq 0.$$

Tous les nombres flottants occupent la même place mémoire dans la machine. Le coût d'un calcul dépend des facteurs suivants :

(1) le nombre d'opérations arithmétiques $\oplus, \ominus, \otimes, \oslash$ et de tests $x < y$;

(2) la lecture et écriture en mémoire ;

(3) la place mémoire.

Dans la pratique de l'analyse numérique, on considère surtout (1) et (3).

3. Perte de précision catastrophique

Si on additionne deux nombres flottants de signe opposé et de taille comparable, on n'a plus guère de chiffres significatifs. L'un des meilleurs exemples mettent ce phénomène en évidence est le calcul de l'exponentielle au moyen de son développement en série entière :

$$\exp(x) = \sum_{k=0}^{\infty} \frac{x^k}{k!}.$$

On utilise l'algorithme suivant, écrit en `scilab` (`http//www.scilab.org/`), qui est un clone gratuit de matlab :

```
x=input("donner la valeur de l'argument");
n=input("donner le nombre de termes apres 1");
s=1;y=1;
for j=1:n,
    y=y*x/j;s=s+y;
end
reponse=[s, exp(x),abs(s-exp(x))]
```

L'exécution de ce programme pour $x = 1$ et $n = 10$ donne

sommation	exponentielle	différence
2.7182818	2.7182818	$2.731E-08$

On refait le même essai avec $x = 10$ et $n = 100$; on trouve

sommation	exponentielle	différence
22026.466	22026.466	$7.276E-12$

Que se passe-t-il si l'on passe à des exposants négatifs ? Pour $x = -1$ et $n = 10$, on trouve

sommation	exponentielle	différence
0.3678795	0.3678794	$2.311E-08$

Jusqu'ici, tout va bien. Passons à $x = -10$ et $n = 10$:

sommation	exponentielle	différence
1342.5873	0.0000454	1342.5873

C'est mauvais ; peut-être n'a-t-on pas assez pris de termes... Pour le même $x = 10$, prenons successivement $n = 20$, $n = 30$, $n = 40$:

exponentielle	$n = 20$	différence	$n = 30$	différence
0.0000454	13.396866	13.396821	0.0009703	0.0009249

exponentielle	$n = 40$	différence
$0.2061154E-08$	0.0000454	$2.412E-09$

Prenons $x = -20$, et de la même façon, faisons les calculs pour n prenant les valeurs de 40 à 80, de 20 en 20 :

exponentielle	$n = 40$	différence	$n = 60$	différence
$0.2061154E-08$	4442.0344	4442.0344	0.0000343	0.0000343

exponentielle	$n = 80$	différence
$0.2061154E-08$	$0.5621885E-08$	$0.3560731E-08$

Rien ne nous empêche de chercher à mettre encore plus de termes. Avec 100 termes, la somme calculée est toujours de $0.5621884E-08$, et nous voyons qu'il est impossible d'obtenir ne serait-ce qu'un seul chiffre significatif de $\exp(-20)$ par sommation de la série, en utilisant `scilab`.

Chapitre 4

Résolution de systèmes linéaires généraux

L'algorithme d' élimination de Gauss c'est l'exemple classique de méthode *directe* de résolution, dont dans l'absence d'erreur d'arrondi, il fourni la solution exacte de $Ax = b$ au bout d'un temps fini. Cet algorithme fut esquissé par Gauss dans [**13**] puis décrit de façon explicite en 1823 dans [**14**, § 31]. C'est un fait remarquables que ces développements sont bien antérieurs à l'utilisation de la notation matricielle ; dans le cas de Gauss la factorisation LU fut exprimée en termes de formes quadratiques.

Sa complexité $2N^3/3$ n'est pas optimale. Strassen [**36**] a montré qu'on peut résoudre $Ax = b$ avec une méthode de type "divide and conquer" avec $O(N^\beta)$ ops, ou β est l'exposant de la multiplication de deux matrices de taille $N \times N$. À l'heure actuelle on sait que $2 \leq \beta \leq 2.37$ grâce à l'algorithme de Winograd [**5**]. Malgré quelques efforts pour rendre l'algorithme de Strassen praticable pour l'exposant $\beta = \log_2(7) = 2.78$ [**25**] ; l'algorithme d'élimination reste la méthode de choix pour la résolution de systèmes linéaires non structurés ou quand on veut un algorithme stable avec un temps d'exécution garanti.

Dans ce chapitre on fait l'étude détaillé de cet algorithme. On le décrit matriciellement, puis on compte le nombre d'opérations arithmétiques sans et avec pivotage, ce qui nous donnera son coût en arithmétique flottante. En estimant la taille des calculs intermédiaires, on obtient sa complexité en calcul exacte.

Dans une seconde étape, on introduit la notion de conditionnement d'un système linéaire et on étude l'erreur *a priori* et *a posteriori* d l'algorithme d'élimination. Finalement on donne un aperçu sans démonstrations de l'application de cet algorithme au cas des matrices bande.

Références pour cette section : [**10, 18, 32**].

1. L'algorithme d'élimination sans pivotage

Pour gagner en simplicité on se restreindra au cas d'une matrice carrée inversible $A = [a_{i,j}]_{1 \leq i,j \leq N} \in \mathbb{F}^{N \times N}$ et $b \in \mathbb{R}^N$; cependant l'algorithme marche aussi bien pour des matrices rectangulaires quelconques. Par exemple

$$\begin{aligned} x + 2y &= 1, \\ 2x - y &= 1. \end{aligned}$$

Pour le résoudre, l'algorithme d'élimination soustrait deux fois la première équation à la deuxième pour éliminer la variable x dans cette dernière. On abouti à un système triangulaire

$$\begin{aligned} x + 2y &= 1, \\ -5y &= -1 \end{aligned}$$

qu'on résout par *backward substitution* en $y = 1/5$ et $x = 1 - 2y = 3/5$.

1.1. Interprétation matricielle. Écrivons cela en général et considérons d'abord la situation *sans pivotage*. Supposons que $\pi_1 := a_{1,1} \neq 0$ et posons

$$a' := \begin{bmatrix} a_{2,1} \\ \vdots \\ a_{N,1} \end{bmatrix}, \quad p' := \frac{1}{\pi_1} a', \quad \ell' := \begin{bmatrix} a_{1,2} \cdots a_{1,N} \end{bmatrix},$$

et $A_{2,2} = [a_{i,j}]_{2 \leq i,j \leq N-1} \in \mathbb{F}^{(N-1)\times(N-1)}$ la matrice A privée des premières ligne et colonne. Soit

$$\widehat{M} := \begin{bmatrix} 1 & \\ -p' & \mathbf{1}_{N-1} \end{bmatrix},$$

alors

$$\widehat{M} \cdot A = \begin{bmatrix} 1 & \\ -p' & \mathbf{1}_{N-1} \end{bmatrix} \cdot \begin{bmatrix} \pi_1 & \ell' \\ a' & A_{2,2} \end{bmatrix} = \begin{bmatrix} \pi_1 & \ell' \\ 0 & A_{2,2} - p' \cdot \ell' \end{bmatrix}. \tag{11}$$

La sous-matrice $S_1 = A_{2,2} - p' \cdot \ell' \in \mathbb{F}^{(N-1)\times(N-1)}$ est le *complément de Schur*; on a

$$(S_1)_{i,j} = (A_{2,2} - p' \cdot \ell')_{i,j} = a_{i,j} - p'_i \ell'_j = a_{i,j} - a_{i,1} a_{1,1}^{-1} a_{1,j}, \qquad (2 \leq i,j \leq N)$$

donc (11) est bien l'algorithme d'élimination classique. Également on doit multiplier b :

$$\widehat{M} \cdot b = \begin{bmatrix} 1 & \\ -p' & \mathbf{1}_{N-1} \end{bmatrix} \cdot \begin{bmatrix} b_1 \\ b' \end{bmatrix} = \begin{bmatrix} b_1 \\ b' - p' \cdot b_1 \end{bmatrix}. \tag{12}$$

Le système originel $Ax = b$ est équivalent au système $\widehat{M} \cdot Ax = \widehat{M} \cdot b$ car $\widehat{M}$ est inversible. À partir de la solution $x' = (x_2, \ldots, x_N)$ de $S_1 x' = b' - p' \cdot b_1$ on obtient x_1 comme

$$x_1 = b_1 - \ell' \cdot x'.$$

On vérifie que

$$\widehat{L} := \widehat{M}^{-1} = \begin{bmatrix} 1 & \\ p' & \mathbf{1}_{N-1} \end{bmatrix}$$

donc

$$A = \begin{bmatrix} 1 & \\ p' & \mathbf{1}_{N-1} \end{bmatrix} \cdot \begin{bmatrix} \pi_1 & \ell' \\ & S_1 \end{bmatrix}.$$

Si le deuxième pivot $\pi_2 := s_{2,2}$ est non nul, on peut appliquer le même procédé à S_1 et remplir la deuxième colonne de $\widehat{M} \cdot A$ avec des zéros à partir de la troisième ligne. Le procédé ne change pas ni la première ni la deuxième lignes de $\widehat{M} \cdot A$. Pourvu qu'on ait la chance de ne pas croisser aucun pivot nul dans notre chemin, on obtient par récurrence un système de type

$$\begin{bmatrix} \pi_1 & * & \cdots & * \\ 0 & \pi_2 & \cdots & * \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \pi_N \end{bmatrix} \cdots \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_N \end{bmatrix}$$

équivalent au système originel $Ax = b$.

Définition 1.1. Soit $A \in \mathbb{F}^{N\times N}$ et pour $1 \leq k \leq N$ notons $A^{(k)}$ le bloc principal

$$A^{(k)} = [a_{i,j}]_{1 \leq i,j \leq k} \in \mathbb{F}^{k\times k}.$$

On dit que A est *fortement inversible* si $A^{(k)}$ est inversible pour tout $1 \leq k \leq N$.

PROPOSITION 1.2. *Soit $A \in \mathbb{F}^{N\times N}$, alors A est fortement inversible si et seulement si tous les pivots successifs au cours de l'élimination sont non nuls. Dans ce cas, il existe des uniques matrices $L \in \mathbb{F}^{N\times N}$ triangulaires inférieure avec 1s dans la diagonal, et $U \in \mathbb{F}^{N\times N}$ triangulaire supérieure telles que*

$$A = L \cdot U.$$

DÉMONSTRATION. Avec les notations précédentes

$$A = \begin{bmatrix} 1 & \\ p' & \mathbf{1}_{N-1} \end{bmatrix} \cdot \begin{bmatrix} \pi_1 & \ell' \\ & S_1 \end{bmatrix}$$

donc pour $1 \leq k \leq N-1$ on a

$$\det A^{(k+1)} = \pi_1 \cdot \det S_1^{(k)}.$$

On en déduit que$A$ est fortement inversible si et seulement si il est de même pour le complément de Schur S_1 et $\pi_1 \neq 0$, et par récurrence si et seulement si $\pi_i \neq 0$ pour $1 \leq i \leq N$.

Supposons que A est fortement inversible, et procédons par récurrence. Soit $S_1 = L_2 \cdot U_2$ la décomposition LU en dimension $N-1$, alors

$$\begin{aligned} A &= \begin{bmatrix} 1 & \\ p' & \mathbf{1}_{N-1} \end{bmatrix} \cdot \begin{bmatrix} \pi_1 & \ell' \\ & L_2 \cdot U_2 \end{bmatrix} \\ &= \begin{bmatrix} 1 & \\ p' & \mathbf{1}_{N-1} \end{bmatrix} \cdot \begin{bmatrix} 1 & \\ & L_2 \end{bmatrix} \cdot \begin{bmatrix} \pi_1 & \ell' \\ & U_2 \end{bmatrix} \\ &= \begin{bmatrix} 1 & \\ p' & L_2 \end{bmatrix} \cdot \begin{bmatrix} \pi_1 & \ell' \\ & U_2 \end{bmatrix}. \end{aligned}$$

Pour l'unicité, soient $L', U' \in \mathbb{R}^{N\times N}$ respectivement triangulaire inférieure avec 1s dans la diagonal et triangulaire supérieure telles que $A = L' \cdot U'$, alors

$$(L')^{-1} \cdot L = U' \cdot U^{-1}$$

c'est une matrice triangulaire inférieure avec 1s dans la diagonal et triangulaire supérieure à la fois. La seule possibilité est $\mathbf{1}_N$ donc $L' = L$ et $U' = U$. □

La matrice L contient beaucoup d'information :

PROPOSITION 1.3. *Le coefficient $L_{i,j}$ ($1 \leq j < i \leq N$) est la valeur par laquelle on multiplie la j-ème ligne pour la soustraire à la i-ème ligne, dans j-ème pas de l'algorithme d'élimination.*

DÉMONSTRATION. La matrice A peut s'écrire comme

$$A = \widehat{L}_1 \cdots \widehat{L}_{N-1} \cdot U$$

avec

$$\widehat{L}_j = \begin{bmatrix} \mathbf{1}_{j-1} & & \\ & 1 & \\ & p'_j & \mathbf{1}_{N-j} \end{bmatrix}$$

correspondant au j-ème pas de l'algorithme d'élimination, où $(p'_j)_i$ ($1 \leq j < i \leq N$) est la valeur par laquelle on multiplie la j-ème ligne pour la soustraire à la i-ème ligne, et on vérifie que

$$\mathrm{col}_j(L) = \mathrm{col}_j(\widehat{L}_1 \cdots \widehat{L}_{N-1}) = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ p'_j \end{bmatrix}.$$

□

EXERCICE 4.1. ◁ Soit

$$A = \begin{matrix} \begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{bmatrix} & \begin{matrix} i \\ N-i \end{matrix} \\ \begin{matrix} i & N-i \end{matrix} & \end{matrix}$$

une matrice fortement inversible. En particulier $A_{1,1}$ est inversible, et on posera

$$S_i := A_{2,2} - A_{2,1} \cdot A_{1,1}^{-1} \cdot A_{1,2} \in \mathbb{F}^{(N-i)\times(N-i)}$$

pour son *i-ème complément de Schur*. Montrer que S_i coïncide avec le bloc correspondant dans la matrice A après i pas de l'algorithme d'élimination de Gauss. ▷

1.2. L'algorithme d'élimination en pratique. A partir de la décomposition $A = L \cdot U$ la résolution d'un système $Ax = b$ se fait en deux étapes :

(1) on résout $Ly = b$;
(2) on résout $Ux = y$.

En effet $Ax = LUx = Ly = b$. Chacun de ces systèmes est facile à résoudre par substitution successive puisque triangulaires. La partie la plus lourde du point de vue des calculs est la construction de L et de U. Remarquons que si on doit résoudre plusieurs systèmes avec la matrice A, il faut garder la factorisation LU pour ne pas avoir à la recalculer à chaque fois.

Voici l' algorithme d'élimination sans pivotage en pseudo-code :

Algorithme de factorisation LU sans pivotage :
Entrée : $a_{i,j}$ **pour** $1 \leq i, j \leq N$ **;**
Sortie : $\ell_{i,j}, u_{i,j}$ **pour** $1 \leq i, j \leq N$**.**
For i **from** 1 **to** $N-1$ **do**

(1) for j **from** i **to** N **do**

$$\ell_{j,i} \leftarrow a_{j,i}/a_{i,i} \quad , \quad u_{i,j} \leftarrow a_{i,j};$$

od ;

(2) for j, k **from** $i+1$ **to** N **do**

$$a_{j,k} \leftarrow a_{j,k} - \ell_{j,i} \cdot u_{i,k};$$

od ;

od ;

$$\ell_{N,N} \leftarrow 1 \quad , \quad u_{N,N} \leftarrow a_{N,N};$$

end.

Notons qu'une fois que la i-ème colonne de A est utilisée pour calculer la i-ème colonne de L, elle n'est plus réutilisée. Similairement, la i-ème ligne de A n'est plus utilisée après le calcul de la i-ème ligne de U. Ceci permets d'écrire L et U sur A au fur et mesure qu'on les calcule, et donc on n'a pas besoin d'espace additionnel pour les stocker : L et U occupent respectivement le triangle inférieur et supérieur de A.

Cette observation simplifie l'algorithme, et on le réduit encore en utilisant la notation Matlab :

Algorithme de factorisation LU sans pivotage, réécrivant L et U sur A :
Entrée : $A \in \mathbb{F}^{N\times N}$ **;**
Sortie : Réécriture de L, U sur A.
For i **from** 1 **to** $N-1$ **do**

(1) $A(i+1:N, i) = A(i+1:N, i)/A(i,i)$ **;**

(2) $A(i{+}1 : N, i{+}1 : N) = A(i{+}1 : N, i{+}1 : N) - A(i{+}1 : N, i) \cdot A(i, i{+}1 : N)$ **;**

od ; end.

1.3. Complexité. Estimons le nombre d'opérations arithmétiques requises par cet algorithme, ce qui reviendra à estimer sa complexité en calcul flottant :

PROPOSITION 1.4. *Soit $A \in \mathbb{F}^{N\times N}$ une matrice fortement inversible, alors la factorisation LU via l'algorithme d'élimination sans pivotage se fait en $2N^3/3$ ops.*

DÉMONSTRATION. **Pas 1.** Pour chaque $1 \leq i \leq N-1$ on fait $N-i$ divisions (on ne compte pas l'assignation triviale $\ell_{i,i} \leftarrow a_{i,i}/a_{i,i} = 1$), soit au total

$$\sum_{i=1}^{N-1} N - i = \frac{N(N-1)}{2} \quad \text{ops.}$$

Pas 2. Pour chaque $1 \leq i \leq N-1$ on fait $2(N-i)^2$ opérations, soit au total

$$\sum_{i=1}^{N-1} 2(N-i)^2 = 2\sum_{i=1}^{N-1} i^2 = \frac{(N-1)N(2N-1)}{6} \quad \text{ops ;}$$

donc la factorisation LU demande

$$\frac{N(N-1)}{2} + \frac{(N-1)N(2N-1)}{6} \leq \frac{2}{3}N^3 \quad \text{ops.}$$

□

Considérons maintenant la résolution de $Ax = b$. Le système $Ly = b$ s'écrit comme

$$\begin{aligned} y_1 &= b_1 \\ \ell_{2,1}y_1 + y_2 &= b_2 \\ &\vdots \\ \ell_{N,1}y_1 + \ell_{N-1,1}y_2 + \cdots + y_N &= b_N. \end{aligned}$$

Pour résoudre la première équation on fait 0 ops, pour la deuxième on fait 2 ops, et en général pour résoudre la j-ème équation on fait $2(j-1)$ ops. Au total, la résolution de $Ly = b$ demande

$$\sum_{j=1}^{N} 2(j-1) = (N-1)N \quad \text{ops.}$$

Le système $Ux = y$ est similaire, sauf qu'on fait une opération de plus par ligne (l'inversion de $s_{j,j}$) donc sa résolution demande

$$(N-1)N + N = N^2$$

ops. On obtient :

PROPOSITION 1.5. *La résolution de $Ax = b$ à partir de la factorisation $A = L{\cdot}U$ se fait en $2N^2$ ops.*

Si on veut résoudre un système $1,000 \times 1,000$ *via* élimination gaussienne, cela nous demandera

$$2 \times 1,000^3/3 + 2 \times 1,000^2 \sim 7 \times 10^8 \quad \text{ops,}$$

ce qui prendra moins d'une seconde à 10^9 flops (*floating point operations per second*). Par contre, résoudre un système $1,000,000 \times 1,000,000$ nous demandera

$$2 \times 1,000,000^3/3 + 2 \times 1,000,000^2 \sim 7 \times 10^{17} \quad \text{ops,}$$

ce qui prendra plus de 21 ans à 10^9 flops.

1.4. Faut-il inverser des matrices ? L'inverse de $A \in \mathbb{F}^{N\times N}$ est la matrice A^{-1} dont l'image des vecteurs e_j de la base standard est la colonne v_j de A correspondante. Le calcul de cette inverse revient donc à résoudre les systèmes

$$Ax = v_j \quad , \quad 1 \leq j \leq N.$$

À l'aide de la factorisation LU, la résolution de ces systèmes se fait en deux étapes

$$Lw_j = e_j \quad , \quad Uv_j = w_j.$$

Quel est le coût de ces résolutions ? Le système triangulaire $Lw_j = e_j$ est assez spécial. On sait *a priori* que $(w_j)_k = 0$ pour $1 \leq k \leq j-1$, donc ce système se réduit à

$$\begin{array}{lcl} w_j & = & 1 \\ \ell_{j+1,j}w_j + w_{j+1} & = & 0 \\ & \vdots & \\ \ell_{N,j}w_j + \ell_{N-1,j+1}w_{j+1} + \cdots + w_N & = & 0 \end{array}$$

donc le coût de sa résolution est de $(N-j+1)^2$ ops. La complexité du calcul de $w_1, \ldots, w_N$ est donc

$$\sum_{j=1}^{N}(N-j+1)^2 = \frac{N^3}{3} + O(N^2).$$

par contre on ne peut pas faire ce genre d'économie pour le calcul des v_js, donc le calcul de A^{-1} à partir de la décomposition LU es estimé en

$$\frac{N^3}{3} + O(N^2) + N^3 = \frac{4}{3}N^3 + O(N^2).$$

PROPOSITION 1.6. *Soit $A \in \mathbb{F}^{N\times N}$ une matrice fortement inversible, alors le calcul de A^{-1} via l'algorithme d'élimination sans pivotage se fait en $2N^3 + O(N^2)$ ops.*

DÉMONSTRATION. Conséquence de l'antérieur joint au coût $2N^3/3$ de la factorisation LU. □

Donc l'inversion coûte approximativement 3 fois le prix de résoudre un système linéaire. On pourrait penser que cette perte est compensée si l'on doit résoudre plusieurs systèmes avec la même matrice A. Voyons si c'est correcte : supposons qu'on doit résoudre

$$Ax_k = b_k \quad , \quad 1 \leq k \leq K$$

pour $K \gg 0$. Si l'on garde la décomposition LU pour résoudre chacun de ces systèmes, cela nous prendra

$$2N^3/3 + 2KN^2 \quad \text{ops}.$$

Si par contre on calcule d'abord l'inverse A^{-1} puis on calcule $x_k \leftarrow A^{-1}b_k$, ceci nous prendra

$$2N^3 + 2KN^2 + O(N^2) \quad \text{ops},$$

c'est-à-dire $4N^3/3$ ops de plus par rapport au résultat précédent. En conséquent, en général on préférera la décomposition LU au calcul de A^{-1}, pour la résolution de systèmes d'équations linéaires.

2. Élimination avec pivotage

Si dans le cours de l'élimination on rencontre un pivot zéro, l'algorithme se plante. Et si on rencontre un pivot qui n'est pas zéro mais tout petit, l'algorithme ne se plante pas mais le calcul peut être atteint des erreurs d'arrondi considérables, comme on va le voir dans l'exemple suivant.

Soit $\varepsilon \neq 0$ petit, et considérons le système

$$\begin{aligned} \varepsilon x + y &= 1, \\ x + y &= 2. \end{aligned}$$

En applicant l'élimination en exacte on obtient le système équivalent

$$\begin{aligned} \varepsilon x + y &= 1, \\ \left(1 - \frac{1}{\varepsilon}\right) y &= 2 - \frac{2}{\varepsilon} \end{aligned}$$

d'où

$$y = \frac{1-2\epsilon}{1-\varepsilon} \cong 1 \quad , \quad x = \frac{1-y}{\epsilon} = \frac{1}{1-\varepsilon} \cong 1.$$

Supposons maintenant que ε est suffisamment petit par rapport à la précision de la machine ; par exemple prenons

$$\varepsilon = 10^{-4}$$

sur un système de flottants à trois décimaux. Donc

$$1 \ominus \frac{1}{\varepsilon} = -\frac{1}{\varepsilon} \quad , \quad 2 \ominus \frac{1}{\varepsilon} = -\frac{1}{\varepsilon}$$

entraînant

$$y = 1$$

et la substitution donne

$$x = (1 \ominus y) \oslash \varepsilon = 0$$

ce qui est un erreur inacceptable. Essayons en changeant l'ordre des équations :

$$\begin{aligned} x + y &= 2, \\ \varepsilon x + y &= 1. \end{aligned}$$

Le processus d'élimination nous donne cette fois le système

$$\begin{aligned} x + y &= 2, \\ (1-\varepsilon) y &= 1 - 2\varepsilon. \end{aligned}$$

En arithmétique flottante devient

$$y = 1 \quad , \quad x = 2 - y = 1,$$

maintenant l'erreur est tout à fait raisonnable. L'erreur dans le premier essai vient de diviser un nombre par un petit pivot ε, ce qui amplifie les erreurs.

2.1. Pivotage partiel et total.

Définition 2.1. Soit $\sigma \in S_N$ une permutation, la *matrice de permutacions* correspondante P_σ est l'identité avec les lignes permutées suivant σ.

Lemme 2.2. *Soient $\sigma, \tau \in S_N$ des permutations et $P_\sigma, P_\tau \in \mathbb{R}^{N\times N}$ les matrices associées, et $A \in \mathbb{F}^{N\times N}$ une matrice quelconque, alors*

(1) $P_\sigma \cdot A$ est la matrice A avec les lignes permutées suivant σ ;

(2) $P_\sigma^{-1} = P_{\sigma^{-1}} = P_\sigma^*$;

(3) $P_\sigma \cdot P_\tau = P_{\sigma\circ\tau}$.

DÉMONSTRATION. Exercice. □

Le *pivotage partiel* consiste à chaque étape i de l'élimination à choisir le coefficient $a_{k,i}$ de plus grande valeur absolue dans la première colonne du complément de Schur S_i, puis échanger les lignes i et k pour utiliser ce coefficient comme pivot. Ceci consiste à l'introduction d'un pas intermédiaire dans l'algorithme d'élimination juste avantle pas (1) :

(0.5) $a'_i \leftarrow a_{i,i}$, $k \leftarrow i$;
for ℓ **from** i **to** N **do**
if $|a_{\ell,i}| > a'_i$ **then** $a'_i \leftarrow a_{\ell,i}$, $k \leftarrow \ell$;
od ;

Notons que ceci assure que les coefficients de la matrice L sont tous de valeur absolue ≤ 1.

Le *pivotage total* consiste à choisir le coefficient $a_{k,j}$ de plus grande valeur absolue dans *tout* le complément de Schur S_i, pour ensuite ensuite échanger les lignes i et k et la colonne i et j.

THÉORÈME 2.3. *Soit $A \in \mathbb{F}^{N\times N}$ une matrice inversible, alors il existent des (non uniques) matrices $P, L, U \in \mathbb{R}^{N\times N}$, P matrice de permutations, L triangulaire inférieure avec 1s dans la diagonal telle que $|L_{i,j}| \leq 1$ pour tout i, j, et U triangulaire supérieure, telles que*

$$A = P \cdot L \cdot U.$$

DÉMONSTRATION. Soit $\sigma \in S_N$ la permutation obtenue *via* pivotage partiel, alors

$$P_\sigma \cdot A = L \cdot U$$

avec de plus $|\ell_{i,j}| \leq 1$, donc $A = P^*_\sigma \cdot L \cdot U$. □

Similairement le pivotage total corresponds à une factorisation

$$A = P \cdot L \cdot U \cdot Q$$

avec P, Q matrices de permutations.

Le pas (0.5) rajoute $N - i$ comparaisons pour chaque $1 \leq i \leq N - 1$, soit un total de

$$\sum_{i=1}^{N-1} N - i = \frac{N(N-1)}{2} \quad \text{comparaisons.}$$

Le coût total de l'*élimination gaussienne avec pivotage partiel* (EGPP) reste en $2N^3/3 + O(N^2)$. Le pivotage total demande $(N - i)^2$ comparaisons pour chaque $1 \leq i \leq N - 1$, soit

$$\sum_{i=1}^{N-1} (N-i)^2 \cong \frac{N^3}{3} \quad \text{comparaisons.}$$

Le coût total de l'*élimination gaussienne avec pivotage total* (EGPT) monte à $N^3 + O(N^2)$.

EGPP est la façon la plus habituelle d'implémenter l'élimination en pratique. À cause de son coût plus élevé, EGPT n'est presque jamais utilisée, bien qu'il y ait des rares exemples où EGPP tombe à défaut et EGPT réussi à calculer la solution correcte.

3. Complexité de l'élimination en exacte

Considérons l'exemple suivant :

$$A = \begin{bmatrix} 7 & -2 & 1 \\ 1 & 5 & 3 \\ 1 & 1 & 8 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ \frac{1}{7} & 1 & 0 \\ \frac{1}{7} & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 7 & -2 & 1 \\ 0 & 37/7 & 20/7 \\ 0 & 16/7 & 55/7 \end{bmatrix}$$
$$= \begin{bmatrix} 1 & 0 & 0 \\ 1/7 & 1 & 0 \\ 1/7 & \frac{16/7}{37/7} & 1 \end{bmatrix} \cdot \begin{bmatrix} 7 & -2 & 1 \\ 0 & 37/7 & 20/7 \\ 0 & 0 & (55/7) - (20/7)\frac{16/7}{37/7} \end{bmatrix}$$
$$= \begin{bmatrix} 1 & 0 & 0 \\ 1/7 & 1 & 0 \\ 1/7 & 16/37 & 1 \end{bmatrix} \cdot \begin{bmatrix} 7 & -2 & 1 \\ 0 & 37/7 & 20/7 \\ 0 & 0 & 265/37 \end{bmatrix}.$$

À chaque étape, le coût des opérations arithmétiques devient de plus en plus lourd. Ceci est dû à l'augmentation de la taille des calculs intermédiaires, caractéristique des calculs exacts. Une estimation naïve (*via* récurrence) montre qu'on s'attend *a priori* à des calculs intermédiaires de taille binaire 2^{N-1} fois le taille des coefficients de A, ce qui rendrait l'algorithme impraticable puisque de complexité exponentielle. Comme on le voit dans l'exemple, heureusement des simplifications se produisent. En fait la croissance de la taille des calculs intermédiaires est *linéaire* :

PROPOSITION 3.1. *Soit $A \in \mathbb{Z}^{N\times N}$ et posons $h(A) := \max_{1\leq i,j\leq N} h(a_{i,j})$ la hauteur de A, alors*

$$h(L), h(U) \leq N(h(A) + \log(N)).$$

DÉMONSTRATION. Soit $C \in \mathbb{Z}^{N\times}$, à l'aide du développement du déterminant on voit que $h(\det(C)) \leq N(h(C) + \log(N))$. La proposition est conséquence de l'exercice ci-dessous. □

EXERCICE 4.2. ◁ Soit $A \in \mathbb{F}^{N\times N}$ une matrice admettant une décomposition LU : (13)

$$A = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,N} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ a_{N,1} & a_{N,2} & \cdots & a_{N,N} \end{bmatrix} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ \ell_{2,1} & 1 & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ \ell_{N,1} & \ell_{N,2} & \cdots & 1 \end{bmatrix} \cdot \begin{bmatrix} s_{1,1} & s_{1,2} & \cdots & s_{1,N} \\ 0 & s_{2,2} & \cdots & s_{2,N} \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & s_{N,N} \end{bmatrix}.$$

Pour $1 \leq i_1 < \cdots < i_p \leq N$ et $1 \leq j_1 < \cdots < j_p \leq N$ soit

$$A(i_1, \ldots, i_p; j_1, \ldots, j_p) := [a_{i_k, j_\ell}]_{1\leq k,\ell\leq p} \ \in \mathbb{F}^{p\times p}$$

la sous-matrice de A associée aux lignes $i_1, \ldots, i_p$ et aux colonnes $j_1, \ldots, j_p$. Montrer que

$$\ell_{i,j} = \frac{\det(A(1, 2, \ldots, j-1, i; 1, 2, \ldots, j-1, j))}{\det(A(1, 2, \ldots, j-1, j; 1, 2, \ldots, j-1, j))} \qquad (i > j)$$

et

$$s_{i,j} = \frac{\det(A(1, 2, \ldots, i-1, i; 1, 2, \ldots, i-1, j))}{\det(A(1, 2, \ldots, i-2, i-1; 1, 2, \ldots, i-2, i-1))} \qquad (i \leq j).$$

Indication : Effacez de façon convenable des lignes et des colonnes dans la factorisation (13). ▷

PROPOSITION 3.2. *Soit $A \in \mathbb{Z}^{N\times N}$ une matrice inversible, alors la factorisation PLU se fait en $O^*(N^4 h(A))$ opérations binaires.*

Rappelons que la notation O^* signifie "à des facteurs logarithmiques près".

DÉMONSTRATION. Par rapport à l'algorithme EGPP, il faut juste remplacer les opérations flottants par les opérations de $\mathbb{Q}$ avec des expressions réduites. Le corollaire 1.3 du chapitre 3 entraîne

$$\mathcal{C}_{EGPP}(\tau) \leq O^*(N^3 \cdot h_{EGPP}(\tau))$$

avec $\tau := h(A)$. Le résultat est donc conséquence de la proposition 3.1. □

4. Conditionnement d'un opérateur

Les sorties des algorithmes de l'analyse numérique sont rarement exactes. Les sources d'erreurs possibles sont

(1) erreurs dans l'entrée dues aux erreurs de méditions et d'arrondi ;
(2) propagation d'erreurs d'arrondi au cours de l'exécution de l'algorithme ;
(3) seulement pour les méthodes itératives : erreur d'approximation.

Dans cette section on traitera la théorie de perturbations qui nous permettra traiter les erreurs de type (1). Soient

$$\widehat{A} = A + \delta A \in \mathbb{F}^{N\times N} \quad , \quad \widehat{b} = b + \delta b \in \mathbb{F}^N$$

des perturbations d'une matrice inversible A et d'un vecteur b. Quel est l'erreur $\delta x = \widehat{x} - x$ de la résolution *exacte* du système $\widehat{A}\widehat{x} = \widehat{b}$? On a

$$\begin{array}{rrcl} & (A+\delta A)(x+\delta x) & = & b+\delta b \\ - & Ax & = & b \\ \hline & \delta Ax + (A+\delta A)\delta x & = & \delta b. \end{array}$$

On en obtient $\delta x = A^{-1}(-\delta A\widehat{x} + \delta b)$ et donc pour une norme vectorielle $|\cdot|$ et la norme d'opérateurs subordonnée $||\cdot||$

$$|\delta x| \leq ||A^{-1}||(||\delta A|| \cdot |\widehat{x}| + ||\delta b||).$$

DÉFINITION 4.1. La quantité

$$\kappa(A) := ||A^{-1}|| \cdot ||A|| \geq 1$$

est le *conditionnement* de la matrice A relatif à la norme vectorielle $|\cdot|$. Pour A non inversible on pose $\kappa(A) := \infty$.

Avec cette définition

$$\frac{|\delta x|}{|\widehat{x}|} \leq \kappa(A)\left(\frac{||\delta A||}{||A||} + \frac{|\delta b|}{||A|| \cdot |\widehat{x}|}\right). \tag{14}$$

Le conditionnement de A mesure l'erreur relatif $|\delta x|/|\widehat{x}|$ en termes de l'erreur relatif $||\delta A||/||A||$. La majoration ci-dessus dépends de δx (*via* $\widehat{x}$) et donc paraît difficile à interpréter. Cependant elle est utile en pratique parce que on connaît la quantité calculée $\widehat{x}$ et on peut évaluer la majoration directement. Alternativement, on peut déduire une borne plus attractive du point de vue théorique :

$$\begin{aligned} \frac{|\delta x|}{|x|} &\leq \kappa(A)\left(\frac{||\delta A||}{||A||}\Big(1 + \frac{|\delta x|}{|x|}\Big) + \frac{|\delta b|}{||A|| \cdot ||x||}\right) \\ &\leq \kappa(A)\left(\frac{||\delta A||}{||A||}\Big(1 + \frac{|\delta x|}{|x|}\Big) + \frac{|\delta b|}{|b|}\right) \end{aligned}$$

donc

$$\left(1 - \kappa(A)\frac{||\delta A||}{||A||}\right)\frac{|\delta x|}{|x|} \leq \kappa(A)\left(\frac{||\delta A||}{||A||} + \frac{|\delta b|}{|b|}\right)$$

c'est-à-dire

$$\frac{|\delta x|}{|x|} \leq \frac{\kappa(A)}{1-\kappa(A)\dfrac{||\delta A||}{||A||}}\left(\frac{||\delta A||}{||A||}+\frac{|\delta b|}{|b|}\right). \tag{15}$$

On a

$$\frac{\kappa(A)}{1-\kappa(A)\dfrac{||\delta A||}{||A||}} \cong \kappa(A) \qquad \text{pour } \frac{||\delta A||}{||A||} \to 0.$$

Par exemple, pensons au cas où $\widehat{A} = \mathrm{fl}(A)$ et $\widehat{b} = \mathrm{fl}(b)$ sont produits par la troncation à t bits correctes d'une matrice et d'un vecteur réels. Soit β la base de l'arithmétique flottante en question, alors

$$\frac{|x - \mathrm{fl}(x)|}{|x|} \leq \mu := \beta^{1-t}/2$$

pour tout $x \neq 0$. Donc si l'on pose $\delta b = b - \mathrm{fl}(b)$ on a $|(\delta b)_i| \leq \mu |b_i|$ pour tout i donc

$$|\delta b|_\infty = \max_i |(\delta b)_i| \leq \mu \max_i |b_i| = \mu |b|_\infty.$$

Similairement si l'on pose $\delta A = A - \mathrm{fl}(A)$ on a

$$||\delta A||_\infty = \max_i \sum_{j=1}^N |(\delta A)_{i,j}| \leq \mu \cdot \max_i \sum_{j=1}^N |A_{i,j}| = \mu \cdot ||A||_\infty$$

soit

$$\frac{|\delta b|_\infty}{|b|_\infty}, \frac{||\delta A||_\infty}{||A||_\infty} \leq \mu. \tag{16}$$

La majoration (15) entraîne

$$\frac{|\delta x|_\infty}{|x|_\infty} \leq 2\mu \cdot \frac{\kappa_\infty(A)}{1-\kappa_\infty(A)\mu}$$

et si $\kappa_\infty(A)\mu \leq 1/2$ (ou de façon équivalente $\kappa_\infty(A) \leq \beta^{t-1}$) on a

$$\frac{|\delta x|_\infty}{|x|_\infty} \leq 4\mu\kappa_\infty(A).$$

Autrement-dît, la *perte de précision* due au troncations dans l'entrée est de (pour la base $\beta = 2$)

$$\log_2(\kappa_\infty(A)) + 2 \quad \text{bits.}$$

Une façon alternative d'estimer l'erreur est *via* le *résidu* de $\widehat{x}$:

$$r = A\widehat{x} - b \in \mathbb{F}^N.$$

On a $\delta x = A^{-1}r$ donc

$$||\delta x|| \leq ||A^{-1}||||r||.$$

On a $\kappa(A) \geq 1$ pour toute matrice A. Le conditionnement est invariant par multiplication par scalaires : $\kappa(\lambda \cdot A) = \kappa(A)$. Si $\kappa(A)$ n'est pas trop grand par rapport à la dimension, on dit que la matrice est *bien conditionnée*, autrement on dit qu'elle est *mal conditionnée*.

Soient $\sigma_1 \geq \cdots \geq \sigma_N > 0$ les valeurs singulières d'une matrice inversible A, alors $||A||_2 = \sigma_1$ et $||A^{-1}||_2 = \sigma_N^{-1}$ donc pour la norme 2

$$\kappa_2(A) = \sigma_1/\sigma_N$$

est l'élongation de l'ellipsoïde $\{Ax : |x|_2 = 1\} \subset \mathbb{F}^N$. En particulier, le conditionnement d'une matrice unitaire U est parfait : $\kappa_2(U) = 1$.

Le conditionnement admets une caractérisation géométrique comme l'inverse de la distance à l'ensemble des matrices singulières :

PROPOSITION 4.2. *Soit A une matrice inversible, alors*

$$\frac{1}{\kappa(A)} = \min\left\{ \frac{||\delta A||_2}{||A||_2} : A + \delta A \text{ est singulière} \right\}.$$

DÉMONSTRATION. C'est une reformulation du théorème d'approximation de Schmid : dans la notation du théorème 2.3 du chapitre 1

$$\mathrm{dist}_{||\cdot||_2}(A, M_{N-1,N-1}) = \sigma_N = ||A^{-1}||^{-1}.$$

□

Les conditionnements associés aux différentes métriques sont comparables, or les constantes dépendent de la dimension :

$$\begin{aligned}
&\frac{1}{N}\kappa_2(A) \leq \kappa_1(A) \leq N\kappa_2(A);\\
&\frac{1}{N}\kappa_\infty(A) \leq \kappa_2(A) \leq N\kappa_\infty(A);\\
&\frac{1}{N^2}\kappa_2(A) \leq \kappa_\infty(A) \leq N^2\kappa_1(A).
\end{aligned}$$

Vue l'importance du nombre de conditionnement, on peut se demander quel est la probabilité qu'une matrice soit bien (ou mal) conditionnée. Concrètement, soit

$$N = 10^6 = (10^2)^3$$

notre taille de matrices préférée, Quel est le volume relatif dans la sphère unité (par rapport à la métrique de Frobenius) de $\mathbb{R}^{N\times N}$ de l'ensemble des matrices A telles que

$$\kappa_2(A) \leq 10^4?$$

La réponse est ... 10^{17776} ! Donc la probabilité de tomber la-dessus est pratiquement 0. Fort heureusement, le conditionnement n'est pas trop grand pour beaucoup d'applications intéressantes : pour une matrice A issue de la discrétisation d'une EDP elliptique, typiquement on a

$$\kappa_2(A) = O(h^{-2})$$

où h est le pas de discrétisation.

5. Analyse formelle de l'erreur dans l'algorithme d'élimination

5.1. Stabilité du produit scalaire. Commençons notre étude d'erreurs d'arrondi (c'est-à-dire les erreurs de type (2) dans la liste dans la section 4) en considérant les erreurs d'arrondi de l'algorithme standard pour le produit scalaire :

Entrée : $x, y \in \mathbb{R}^N$ **;**
Sortie : $\langle x, y\rangle$**.**
$s_0 \leftarrow 0$ **;**
For $k = 1$ **to** N **do**
$s_k \leftarrow s_{k-1} + x_k \cdot y_k$ **;**
od ; end.

En essayant de quantifier les erreurs d'arrondi, on est tout de suite confronté avec un problème notationnel : de distinguer les quantités calculées des quantités exactes. Quand le contexte est clair, on utilisera la notation $\mathrm{fl}(\cdot)$ pour noter les quantités calculées. Ainsi $\mathrm{fl}(\langle x, y\rangle)$ désigne la sortie de l'algorithme ci-dessus. Aussi on notera

$$\mathrm{abs}(x) := (|x_1|, \ldots, |x_N|) \in \mathbb{R}^N_+$$

et pour $x, y \in \mathbb{R}^N$ on dit $x \leq y$ si $x_i \leq y_i$ pour $1 \leq i \leq N$. Similairement pour une matrice A on pose

$$\mathrm{abs}(A) := [|A_{i,j}|]_{1 \leq i,j \leq N} \in \mathbb{R}_+^{N \times N}.$$

Soit

$$\mu := \beta^{1-t}/2$$

la précision du système flottant choisi.

PROPOSITION 5.1.

$$|\,\mathrm{fl}(\langle x, y\rangle) - \langle x, y\rangle| \leq N\langle \mathrm{abs}(x), \mathrm{abs}(y)\rangle\mu + O(\mu^2).$$

DÉMONSTRATION. Soit s_k le k-ème calcul intermédiaire dans l'algorithme, alors

$$s_1 = x_1 \odot y_1 = x_1 \cdot y_1(1 + \delta_1)$$

avec $|\delta_1| \leq \mu$. Puis

$$\begin{aligned} s_2 &= s_1 \oplus (x_2 \odot y_2) \\ &= (s_1 + (x_2 \odot y_2))(1 + \varepsilon_2) \\ &= (x_1 \cdot y_1(1 + \delta_1) + x_2 \cdot y_2(1 + \delta_2))(1 + \varepsilon_2) \end{aligned}$$

avec $|\delta_2|, |\varepsilon_2| \leq \mu$. Similairement

$$s_N = \mathrm{fl}(\langle x, y\rangle) = \sum_{j=1}^{N} x_j \cdot y_j(1 + \gamma_j)$$

avec

$$1 + \gamma_j = (1 + \delta_j)\prod_{\ell=j}^{N}(1 + \varepsilon_\ell) \qquad \text{(convention : } \varepsilon_1 = 0)$$

avec $|\delta_j|, |\varepsilon_\ell| \leq \mu$. On vérifie

$$\gamma_j \leq N\mu + O(\mu^2)$$

et ainsi

$$|\,\mathrm{fl}(\langle x, y\rangle) - \langle x, y\rangle| \leq \sum_{j=1}^{N} |x_j| \cdot |y_j| \cdot |\gamma_j| \leq N\langle \mathrm{abs}(x), \mathrm{abs}(y)\rangle\mu + O(\mu^2).$$

□

Si $\langle x, y\rangle \ll \langle \mathrm{abs}(x), \mathrm{abs}(y)\rangle$, l'algorithme est passible de produire d'erreurs considérables. Pensez à l'exemple du calcul de l'exponentielle $\exp(x)$ pour $x < 0$ en sommant N termes de sa série de Taylor : ceci est le produit scalaire des vecteurs

$$X := (1, x, x^2, \ldots, x^{N-1}) \quad , \quad Y := \Big(\frac{1}{0!}, \frac{1}{1!}, \frac{1}{2}, \ldots, \frac{1}{(N-1)!}\Big);$$

dans ce cas

$$\langle X, Y\rangle \cong \exp(x) \ll \langle \mathrm{abs}(X), \mathrm{abs}(Y)\rangle \cong \exp(-x).$$

5.2. Analyse d'erreur *a priori* **et** *a posteriori*. L'analyse du calcul du produit scalaire est un exemple d'analyse d'erreur *a priori*, où l'on estime l'erreur à partir des données.

Le paradigme d'analyse d'erreurs de troncation de la résolution d'un système $Ax = b$ est l'analyse *a posteriori*. Ceci consiste à montrer que le résultat $\widehat{x}$ calculé inexactement par l'algorithme, est le résultat exacte d'une perturbation

$$\widehat{A}\widehat{x} = \widehat{b}$$

avec $\delta A = \widehat{A} - A$ et $\delta b = \widehat{b} - b$ petits. Ceci permets d'unifier l'analyse d'erreurs de type (1) et de type (2), et de les controler par la théorie de perturbations introduite dans la section 4.

Exemple : multiplication en flottant de deux matrices 2×2 :

$$\begin{aligned}\mathrm{fl}(A \cdot B) &= \begin{bmatrix} a_{1,1} & a_{1,2} \\ & a_{2,2} \end{bmatrix} \odot \begin{bmatrix} b_{1,1} & b_{1,2} \\ & b_{2,2} \end{bmatrix} \\ &= \begin{bmatrix} a_{1,1}b_{1,1}(1+\varepsilon_1) & \Big((a_{1,1}b_{1,2}(1+\varepsilon_2) + a_{1,2}b_{2,2}(1+\varepsilon_3)\Big)(1+\varepsilon_4) \\ & a_{2,2}b_{2,2}(1+\varepsilon_5) \end{bmatrix}\end{aligned}$$

avec $|\varepsilon_i| \leq \mu$. Écrivons

$$\widehat{A} := \begin{bmatrix} a_{1,1} & a_{1,2}(1+\varepsilon_3)(1+\varepsilon_4) \\ & a_{2,2}(1+\varepsilon_5) \end{bmatrix} \quad , \quad \widehat{B} := \begin{bmatrix} b_{1,1}(1+\varepsilon_1) & b_{1,2}(1+\varepsilon_2)(1+\varepsilon_4) \\ & b_{2,2} \end{bmatrix},$$

alors

$$\mathrm{abs}(\widehat{A} - A) \leq 2\,\mathrm{abs}(A)\mu + O(\mu^2) \quad , \quad \mathrm{abs}(\widehat{B} - B) \leq 2\,\mathrm{abs}(B)\mu + O(\mu^2).$$

et

$$A \odot B = \widehat{A} \cdot \widehat{B}.$$

5.3. Stabilité de systèmes triangulaires. Considérons un système triangulaire inversible

$$\begin{aligned} u_{1,1}x_1 + u_{1,2}x_2 + \cdots + u_{1,N}x_N &= y_1, \\ u_{2,2}x_2 + \cdots + u_{1,N}x_N &= y_2, \\ &\vdots \\ u_{N,N}x_N &= y_N. \end{aligned}$$

Sa résolution se fait par substitution successive :

For $i = N$ **to** 1 **do**
$x_i \leftarrow \frac{1}{u_{i,i}}(y_i - u_{i,i+1}y_{i+1} - \cdots - u_{i,N}y_N)$ **;**
od ; end.

Analysons la stabilité de cet algorithme :

PROPOSITION 5.2. *Soit $\widehat{x}$ la solution calculée de $Ux = y$, alors*

$$(U + G)\widehat{x} = y$$

avec $\mathrm{abs}(G) \leq N\,\mathrm{abs}(U)\mu + O(\mu^2)$.

Similairement, pour $L \in \mathbb{R}^{N\times N}$ matrice triangulaire inférieure avec 1s dans la diagonal, la solution calculée $\widehat{y}$ de $Ly = b$ vérifie

$$(L + F)\widehat{y} = b$$

pour une matrice $F \in \mathbb{R}^{N\times N}$ telle que $\mathrm{abs}(F) \leq N\,\mathrm{abs}(L)\mu + O(\mu^2)$.

DÉMONSTRATION. On a

$$\widehat{x}_N = y_N \oslash u_{N,N} = \frac{y_N}{u_{N,N}}(1+\gamma_N) = \frac{y_N}{u_{N,N}(1+\gamma_N)^{-1}}$$

avec $|\gamma_N| \leq \mu$ donc on fait

$$G_{N,N} := u_{N,N}(1-(1+\gamma_N)^{-1}) = u_{N,N}\mu + O(\mu^2).$$

Puis

$$\begin{aligned}\widehat{x}_{N-1} &= (y_{N-1} \ominus u_{N-1,N} \odot \widehat{x}_N) \oslash u_{N-1,N-1} \\ &= \frac{1}{u_{N-1,N-1}}\left(y_{N-1} - u_{N-1,N}(1+\eta_{N-1})\widehat{x}_N\right)(1+\gamma_{N-1})(1+\delta_{N-1})\end{aligned}$$

avec $|\gamma_{N-1}|, |\eta_{N-1}|, |\delta_{N-1}| \leq \mu$, donc on fait

$$\begin{aligned}G_{N-1,N-1} &:= u_{N-1,N-1}\eta_{N-1} = u_{N-1,N-1}\mu + O(\mu^2), \\ G_{N-1,N} &:= u_{N-1,N}(1-(1+\gamma_{N-1})^{-1}(1+\delta_{N-1})^{-1}) = 2u_{N-1,N}\mu + O(\mu^2).\end{aligned}$$

Les autres coefficients de G se définent et estiment similairement. □

5.4. Stabilité de la décomposition LU. L'intuition derrière l'analyse d'erreur dans la décomposition LU est que si les quantités dans le produit $\widehat{L}\cdot\widehat{U}$ des matrices triangulaires calculées sont grandes en comparaison avec A, l'information dans A sera essentiellement perdue dans la résolution. Donc on se mets à estimer l'erreur dans le calcul des facteurs triangulaires calculés :

THÉORÈME 5.3. *Soit $A \in \mathbb{R}^{N\times N}$ telle qu'aucun des pivots successivement calculés par l'algorithme d'élimination soit nul, alors les matrices calculées $\widehat{L}, \widehat{U}$ vérifient*

$$\widehat{L}\cdot\widehat{U} = A + H$$

avec $\operatorname{abs}(H) \leq 2N(\operatorname{abs}(\widehat{L})\cdot\operatorname{abs}(\widehat{U}) + \operatorname{abs}(A))\mu + O(\mu^2)$.

DÉMONSTRATION. La preuve est par récurrence sur N. Le résultat est trivial pour $N = 1$. Maintenant supposons-le vraie pour les matrices de taille $(N-1)\times(N-1)$ et écrivons

$$A = \begin{bmatrix}\pi_1 & \ell_1 \\ c_1 & A_{1,1}\end{bmatrix}.$$

Alors

$$\widehat{p} = c_1 \oslash \pi_1 \in \mathbb{R}^N \quad , \quad \widehat{S}_1 = A_{1,1} \ominus \widehat{p}\odot\ell \in \mathbb{R}^{N\times N}$$

sont calculés dans le premier pas de l'algorithme. Par récurrence $\widehat{L}_1\cdot\widehat{U}_1 = \widehat{S}_1 + H_1$ avec

$$\operatorname{abs}(H_1) \leq 2(N-1)(\operatorname{abs}(\widehat{L}_1)\cdot\operatorname{abs}(\widehat{U}_1) + \operatorname{abs}(\widehat{S}_1))\mu + O(\mu^2),$$

alors

$$\begin{aligned}H := \widehat{L}\cdot\widehat{U} - A &= \begin{bmatrix}1 & \\ \widehat{p} & \widehat{L}_1\end{bmatrix}\cdot\begin{bmatrix}\pi_1 & \ell_1 \\ & \widehat{U}_1\end{bmatrix} - A \\ &= \begin{bmatrix}\pi_1 & \ell_1 \\ \widehat{p}\cdot\pi_1 & \widehat{L}_1\cdot\widehat{U}_1 + \widehat{p}\cdot\ell_1\end{bmatrix} - \begin{bmatrix}\pi_1 & \ell_1 \\ c_1 & A_{1,1}\end{bmatrix} \\ &= \begin{bmatrix}0 & 0 \\ \widehat{p}\cdot\pi_1 - c_1 & \widehat{L}_1\cdot\widehat{U}_1 - (A_{1,1} - \widehat{p}\cdot\ell_1)\end{bmatrix}.\end{aligned}$$

On a

$$\widehat{p} = c_1/\pi_1 + f$$

avec $\operatorname{abs}(f) \leq \operatorname{abs}(c_1/\pi_1)\mu$ donc

$$\operatorname{abs}(\widehat{p}\cdot\pi_1 - c_1) \leq \operatorname{abs}(c_1)\mu.$$

On a

$$\operatorname{abs}(\widehat{S}_1 - (A_{1,1} - \widehat{p}\cdot\ell_1)) \le 2\mu(\operatorname{abs}(A_{1,1}) - \operatorname{abs}(\widehat{p})\cdot\operatorname{abs}(\ell_1)) + O(\mu^2)$$

et donc

$$\begin{aligned}\operatorname{abs}(\widehat{L}_1\cdot\widehat{U}_1 - (A_{1,1} - \widehat{p}\cdot\ell_1)) &= \operatorname{abs}(H_1 + \widehat{S}_1 - (A_{1,1} - \widehat{p}\cdot\ell_1))\\ &\le 2(N-1)(\operatorname{abs}(\widehat{L}_1)\cdot\operatorname{abs}(\widehat{U}_1) + \operatorname{abs}(\widehat{S}_1))\mu\\ &\quad + 2\mu(\operatorname{abs}(A_{1,1}) - \operatorname{abs}(\widehat{p})\cdot\operatorname{abs}(\ell_1)) + O(\mu^2)\\ &\le 2N(\operatorname{abs}(\widehat{L})\cdot\operatorname{abs}(\widehat{U}) + \operatorname{abs}(A))\mu + O(\mu^2).\end{aligned}$$

□

5.5. Stabilité de la résolution. On analyse l'effet des arrondis quand $\widehat{L}$ et $\widehat{U}$ sont utilisées pour résoudre le système $Ax = b$:

THÉORÈME 5.4. *Soient $\widehat{L}$ et $\widehat{U}$ les facteurs de A calculées par l'algorithme d'élimination, et soit $\widehat{x}$ la solution calculée de*

$$\widehat{L}y = b \quad , \quad \widehat{U}x = y,$$

alors $(A+E)\widehat{x} = b$ avec

$$\operatorname{abs}(E) \le N(4\operatorname{abs}(\widehat{L})\cdot\operatorname{abs}(\widehat{U}) + 2\operatorname{abs}(A))\mu + O(\mu^2).$$

DÉMONSTRATION. Par la proposition 5.2 on a

$$\begin{aligned}(\widehat{L}+F)\widehat{y} = b \qquad &\text{avec } \operatorname{abs}(F) \le N\operatorname{abs}(\widehat{L})\mu + O(\mu^2);\\ (\widehat{U}+G)\widehat{x} = \widehat{y} \qquad &\text{avec } \operatorname{abs}(G) \le N\operatorname{abs}(\widehat{U})\mu + O(\mu^2)\end{aligned}$$

donc

$$(\widehat{L}+F)(\widehat{U}+G)\widehat{x} = (\widehat{L}\widehat{U} + \widehat{L}G + F\widehat{U} + FG)\widehat{x} = b.$$

Par le théorème 5.3 on a $\widehat{L}\cdot\widehat{U} = A + H$ avec

$$\operatorname{abs}(H) \le 2N(\operatorname{abs}(\widehat{L})\cdot\operatorname{abs}(\widehat{U}) + \operatorname{abs}(A))\mu + O(\mu^2)$$

et alors en écrivant

$$E := H + F\widehat{U} + \widehat{L}G + FG$$

on a $(A+E)\widehat{x} = b$ et

$$\begin{aligned}\operatorname{abs}(E) &\le \operatorname{abs}(H) + \operatorname{abs}(F)\operatorname{abs}(\widehat{U}) + \operatorname{abs}(\widehat{L})\operatorname{abs}(G) + O(\mu^2)\\ &\le N(4\operatorname{abs}(\widehat{L})\cdot\operatorname{abs}(\widehat{U}) + 2\operatorname{abs}(A))\mu + O(\mu^2).\end{aligned}$$

□

Il est tout à fait possible que le terme $\operatorname{abs}(\widehat{L})\cdot\operatorname{abs}(\widehat{U})$ soit grand. Il n'y a rien qu'empêche de rencontrer l'apparition des petits pivots même si la matrice A est bien conditionnée, comme le montre l'exemple

$$A = \begin{bmatrix}\varepsilon & 1\\ 1 & \end{bmatrix}$$

Donc l'algorithme d'élimination est instable. Cette désavantage est réparée par l'application du *pivotage*. Examinons la stabilité de EGPP, l'algorithme d'élimination avec pivotage partiel par lignes. La solution calculée $\widehat{x}$ satisfait $(A+E)\widehat{x} = b$ avec

$$\operatorname{abs}(E) \le N(4P\operatorname{abs}(\widehat{L})\cdot\operatorname{abs}(\widehat{U}) + 2\operatorname{abs}(A))\mu + O(\mu^2)$$

où P est la matrice de permutations produite par le pivotage partiel. Notons

$$|A|_{\max} := \max_{i,j}|A_{i,j}|$$

la norme ∞ de A vue comme un vecteur. Le pivotage partiel entraîne alors

$$|L|_{\max} \leq 1$$

et donc

$$|E|_{\max} \leq N\mu(2|A|_{\max} + 4N|U|_{\max}) + O(\mu^2) \leq N\mu|A|_{\max}(2 + 4N\rho_{EGPP}(A))$$

où

$$\rho_{EGPP}(A) := |U|_{\max}/|A|_{\max}$$

est le *facteur de croissance des pivots* pour EGPP. En utilisant théorie des perturbations on obtient

$$\begin{aligned}\frac{\delta x}{\widehat{x}} &\leq \kappa_\infty(A)\frac{||\delta A||_\infty}{||A||_\infty} \\ &\leq \kappa_\infty(A)N\frac{|E|_{\max}}{||A||_\infty} \\ &\leq \kappa_\infty(A)N^2(2 + 4N\rho_{EGPP}(A))\mu\end{aligned}$$

La stabilité de EGPP est alors équivalente au fait que $\rho_{EGPP}(A)$ soit petit devant la dimension. En pratique $\rho_{EGPP}(A)$ est presque toujours $\leq N$; malheureusement il y des exemples où il est égal à 2^{N-1}.

PROPOSITION 5.5. *Soit $A \in \mathbb{R}^{N\times N}$, alors*

$$\rho_{EGPP}(A) \leq 2^{N-1}.$$

Cette majoration est optimale.

DÉMONSTRATION. Dans le i-ème pas de l'élimination on fait

$$\widetilde{a}_{j,k} \leftarrow a_{j,k} - \ell_{j,i}a_{i,k} \qquad \text{pour } i+1 \leq j,k \leq N.$$

On a $|\ell_{j,i}| \leq 1$ et donc

$$|S_i|_{\max} \leq 2|S_{i-1}|_{\max};$$

par récurrence

$$|U|_{\max} \leq \max_i |S_i|_{\max} \leq 2^{N-1}|A|_{\max}.$$

L'optimalité sera démontrée en exercice. □

Wilkinson [**39**] a démontré que le facteur de croissance des pivots quand on fait un pivotage *total* est majorée par

$$\rho_{EGPT}(A) \leq N^{1/2}(2\cdot 3^{1/2}\cdots N^{1/(N-1)})^{1/2} \cong N^{1/2+\log(N/4)}.$$

Cette estimation est beaucoup trop grande par rapport à ce qu'on trouve en pratique. C'était une vielle conjecture de démontrer que $\rho_{EGPT}(A) \leq N$, mais elle fut récemment réfutée [**11, 19**]. C'est toujours un problème ouvert de trouver une bonne estimation pour $\rho_{EGPT}(A)$, qu'on croît toujours de l'ordre de $O(N)$.

5.6. Matrices bande. Dans un nombre important d'applications, la matrice A est bande. Typiquement c'est le cas quand A est la discrétisation d'une équation différentielle ordinaire ; dans ce cas on peut ordonner les variables de façon à ce que chaque variable x_i n'apparaît que dans quelques peu équations au voisinage de la i-ème ligne. Comme exemple considérons l' équation de Poisson en dimension 1 avec conditions de Dirichlet aux bords nulles :

$$-u_{xx} = f \quad \text{ pour } x \in \Omega = [0,1] \text{ et } \quad u(0) = u(1) = 0.$$

On discrétise cette équation avec des différences finies. Pour $N \in \mathbb{N}$ on pose

$$h := 1/(N+1)$$

et on considère la grille

$$\Omega_h := \{jh : j = 1, \ldots, N\}$$

des nodes *à l'intérieur* de l'intervalle $[0,1]$ divisé en $N+1$ sous-intervalles, et on pose $\mathcal{G}(\Omega_h) : \Omega_h \to \mathbb{R}$ pour l'ensemble des fonctions réelles de cet ensemble. L'opérateur discret qui en résulte est

$$L_h : \mathcal{G}(\Omega_h) \to \mathcal{G}(\Omega_h) \quad , \quad L_h(u)(x) = h^2\Big(u(x-h) - 2u(x) + u(x+h)\Big)$$

avec la convention $u(0) = u(1) = 0$. Ceci est une approximation d'ordre 2 de l'opérateur laplacien

$$-u_{xx} - L_h(u) = O(h^2)$$

pour $h \to 0$ et des fonctions u suffisamment régulières (par exemple $u \in \mathcal{C}^4(\Omega)$). L'équation approchée $L_h u_h = f_h$ se traduit dans un système linéaire $N \times N$ *bande* :

$$h^2 \begin{bmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{bmatrix} \begin{bmatrix} u(h) \\ u(2h) \\ \vdots \\ u((N-1)h) \\ u(Nh) \end{bmatrix} = \begin{bmatrix} f(h) \\ f(2h) \\ \vdots \\ f((N-1)h) \\ f(Nh) \end{bmatrix}.$$

Formellement, on dit qu'une matrice $A = [a_{i,j}]_{1 \le i,j \le N}$ possède *largeur de bande supérieure* q si $a_{i,j} = 0$ pour $j > i + q$, et *largeur de bande inférieure* p si $a_{i,j} = 0$ pour $i > j + p$:

$$A = \begin{bmatrix} a_{1,1} & \cdots & a_{1,q+1} & & & \\ \vdots & & & a_{2,q+2} & & \\ a_{p+1,1} & & & & \ddots & \\ & a_{p+2,2} & & & & a_{N-q,N} \\ & & \ddots & & & \vdots \\ & & & a_{N,N-p} & \cdots & a_{N,N} \end{bmatrix}$$

La décomposition LU sans pivotage respecte la structure bande :

THÉORÈME 5.6. *Soit $A \in \mathbb{F}^{N \times N}$ une matrice fortement inversible avec largeur de bande supérieure q et inférieure p, alors $A = L \cdot U$ avec L triangulaire inférieure avec largeur de bande p et U triangulaire supérieure avec largeur de bande q.*

La version "bande" de l'algorithme d'élimination sans pivotage calcule L et U en $2pqN$ ops. En particulier, on peut résoudre la discrétisation de l'équation de Poisson en dimension 1 en $8N$ ops.

L'élimination avec pivotage partiel peut être organisée de façon à profiter de la structure bande. Cependant le pivotage change la largeur de bande et la version "bande" de EGPP n'est pas si nette que la version sans pivotage :

THÉORÈME 5.7. *Soit $A \in \mathbb{F}^{N \times N}$ une matrice inversible avec largeur de bande supérieure q et inférieure p, alors $A = L \cdot U$ avec U triangulaire supérieure avec largeur de bande $p+q$ et L est sparse, avec au plus $p+1$ coefficients $\neq 0$ par colonne.*

On renvoie le lecteur à [**18**, § 4.3] pour une preuve de ces résultats ainsi que pour plus de détail sur les matrices bande.

Chapitre 5

Structure de déplacement

Nombre d'applications dans les sciences et les ingénieries font appel à des matrices structurées telles que les matrices circulantes, résultantes, Bézout, Toeplitz, Hankel, Vandermonde, Cauchy, Loewner, et Pick. Ces applications demandent souvent la résolution de systèmes linéaires de grande taille, d'où l'intéret pour les algorithmes de résolution adaptés.

Les types de matrices mentionnées sont denses, or ses coefficients dépendent de $O(N)$ paramètres seulement. La notion de rang de déplacement permets d'unifier l'ensemble de ces structures dans une seule approche : toutes les matrices mentionnées ont un rang de déplacement $O(1)$ par rapport à des opérateurs convénables.

La notion de rang de déplacement est la clé pour l'obtention d'algorithmes efficaces pour des matrices structurées, pour des tâches comme résolution de systèmes d'équations, calcul de noyau et d'image, inversion, multiplication matrice-vecteur, etc. Pour une matrice $A \in \mathbb{K}^{N\times N}$ à rang de déplacement α, on peut résoudre $Ax = b$ avec des algorithmes "rapides" en

$$O(\alpha N^2) \quad \text{ops}$$

et avec des algorithmes "super-rapides" en

$$O(\alpha^2 N \log^2(N)) \quad \text{ou} \quad O(\alpha^2 N \log^3(N)) \quad \text{ops}\,.$$

Plus important encore, cette approche permets non seulement de résoudre les structures classiques, mais aussi de traiter des matrices "proches" au sens qu'elles ont un petit rang de déplacement par rapport aux opérateurs associés à chacune de ces structures.

Le rang de déplacement d'une matrice fut introduit dans l'article [**23**] comme une mesure de sa proximité à la classe Toeplitz, *voir* aussi [**12**] ; cependant cette idée fut beaucoup plus profonde, puissante et générale qu'imaginé dans un premier moment. Parmi les antécedents, remarquons la thèse de Morf [**27**] et la celèbre formule de Gohberg et Semencul pour l'inverse d'une matrice de Toeplitz [**17**]. L'idée de déplacement fut étendue à des autres structures dans les articles [**22, 15, 16**].

Dans ce chapitre on introduira les bases de cette théorie et illustrera les algorithmes avec una application importante, la décodification de codes correcteurs d'erreurs de Reed-Solomon. Les principales références pour ce chapitre sont [**24, 28, 30**].

1. Rang de déplacement

Définition 1.1. Soient $M, N \in \mathbb{N}$ des entier et $S \in \mathbb{K}^{M\times M}$ et $T \in \mathbb{K}^{N\times N}$ des matrices fixées, l'*opérateur de déplacement* associé est

$$\nabla_{S,T} : \mathbb{K}^{M\times N} \to \mathbb{K}^{M\times N} \quad , \quad A \mapsto S \cdot A - A \cdot T.$$

Le *rang de déplacement* ou (S,T)*-rang* ou encore ∇*-rang* est

$$\operatorname{rang}_{S,T}(A) := \operatorname{rang}(\nabla_{S,T}(A)).$$

On dira que A est (S,T)-*structurée* si

$$\operatorname{rang}_{S,T}(A) \ll \min(M,N);$$

bien entendu il s'agit d'une notion quantitative et non qualitative. Pour gagner en simplicité, on se restreindra au cas d'une matrice carrée et inversible $A \in \mathbb{K}^{N\times N}$.

Passons revue aux quattre cas les plus celèbres de matrices structurées : Pour $\lambda \in \mathbb{K}$ on déssigne

$$Z_\lambda = \begin{bmatrix} 0 & 1 & & \\ & \ddots & \ddots & \\ & & \ddots & 1 \\ \lambda & & & 0 \end{bmatrix} \in \mathbb{K}^{N\times N},$$

en particulier Z_0 est le bloc de Jordan ou opérateur de shift de taille N.

(1) Matrices de Toeplitz :

$$\mathcal{T} = [a_{i-j}]_{1\le i,j\le N} = \begin{bmatrix} a_0 & a_{-1} & \cdots & a_{-(N-1)} \\ a_1 & a_0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & a_{-1} \\ a_{(N-1)} & \cdots & a_1 & a_0 \end{bmatrix} \in \mathbb{K}^{N\times N}.$$

Les déplacement associés sont

$$S = Z_1 \quad , \quad T = Z_0.$$

Calculons le déplacement pour $N = 3$:

$$\begin{aligned} \nabla_{Z_1,Z_0}(\mathcal{T}) &= \begin{bmatrix} & 1 & \\ & & 1 \\ 1 & & \end{bmatrix} \cdot \begin{bmatrix} a_0 & a_{-1} & a_{-2} \\ a_1 & a_0 & a_{-1} \\ a_2 & a_1 & a_0 \end{bmatrix} - \begin{bmatrix} a_0 & a_{-1} & a_{-2} \\ a_1 & a_0 & a_{-1} \\ a_2 & a_1 & a_0 \end{bmatrix} \cdot \begin{bmatrix} & 1 & \\ & & 1 \\ 0 & & \end{bmatrix} \\ &= \begin{bmatrix} a_1 & a_0 & a_{-1} \\ a_2 & a_1 & a_0 \\ a_0 & a_{-1} & a_{-2} \end{bmatrix} - \begin{bmatrix} 0 & a_0 & a_{-1} \\ 0 & a_1 & a_0 \\ 0 & a_2 & a_1 \end{bmatrix} = \begin{bmatrix} a_1 & & \\ a_2 & & \\ a_0 & a_{-1}-a_2 & a_{-2}-a_1 \end{bmatrix}. \end{aligned}$$

Ce calcul est tout à fait général, pour tout N on a $\operatorname{rang}_{Z_1,Z_0}(\mathcal{T}) \le 2$.

(2) Matrices de Hankel :

$$\mathcal{H} = [a_{i+j-2}]_{1\le i,j\le N} = \begin{bmatrix} a_0 & a_1 & \cdots & a_{N-1} \\ a_1 & a_2 & \diagup & a_N \\ \vdots & \diagup & \diagup & \vdots \\ a_{N-1} & a_N & \cdots & a_{2N-2} \end{bmatrix} \in \mathbb{K}^{N\times N}.$$

Dans ce cas on prends

$$S = Z_1 \quad , \quad T = Z_0^T,$$

faisons le calcul pour $N = 3$:

$$\begin{aligned} \nabla_{Z_1,Z_0^T}(\mathcal{H}) &= \begin{bmatrix} & 1 & \\ & & 1 \\ 1 & & \end{bmatrix} \cdot \begin{bmatrix} a_0 & a_1 & a_2 \\ a_1 & a_2 & a_3 \\ a_2 & a_3 & a_4 \end{bmatrix} - \begin{bmatrix} a_0 & a_1 & a_2 \\ a_1 & a_2 & a_3 \\ a_2 & a_3 & a_4 \end{bmatrix} \cdot \begin{bmatrix} & & 0 \\ 1 & & \\ & 1 & \end{bmatrix} \\ &= \begin{bmatrix} a_1 & a_2 & a_3 \\ a_2 & a_3 & a_4 \\ a_0 & a_1 & a_2 \end{bmatrix} - \begin{bmatrix} a_1 & a_2 & 0 \\ a_2 & a_3 & 0 \\ a_3 & a_4 & 0 \end{bmatrix} = \begin{bmatrix} & & a_3 \\ & & a_4 \\ a_0-a_3 & a_1-a_4 & a_2 \end{bmatrix}. \end{aligned}$$

Ce calcul est général, pour tout N on a $\operatorname{rang}_{Z_1,Z_0^T}(\mathcal{H}) \le 2$.

(3) Matrices de Vandermonde : pour $x_1, \dots, x_N \in \mathbb{K}^\times$

$$\mathcal{V}(x) = [x_i^{j-1}]_{1\le i,j\le N} = \begin{bmatrix} 1 & x_1 & \cdots & x_1^{N-1} \\ 1 & x_2 & \cdots & x_2^{N-1} \\ \vdots & \vdots & & \vdots \\ 1 & x_N & \cdots & x_N^{N-1} \end{bmatrix} \in \mathbb{K}^{N\times N}.$$

Pour les déplacements

$$S = \operatorname{diag}(x_1^{-1}, \dots, x_n^{-1}) \quad , \quad T = Z_0$$

on a $\operatorname{rang}_{S,T}(\mathcal{V}) = 1$. Vérifions cela pour $N = 3$:

$$\begin{aligned} \nabla_{S,T}(\mathcal{V}(x)) &= \begin{bmatrix} x_1^{-1} & & \\ & x_2^{-1} & \\ & & x_3^{-1} \end{bmatrix} \cdot \begin{bmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ 1 & x_3 & x_3^2 \end{bmatrix} - \begin{bmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ 1 & x_3 & x_3^2 \end{bmatrix} \cdot \begin{bmatrix} & 1 & \\ & & 1 \\ 0 & & \end{bmatrix} \\ &= \begin{bmatrix} x_1^{-1} & 1 & x_1 \\ x_2^{-1} & 1 & x_2 \\ x_3^{-1} & 1 & x_3 \end{bmatrix} - \begin{bmatrix} 0 & 1 & x_1 \\ 0 & 1 & x_2 \\ 0 & 1 & x_3 \end{bmatrix} = \begin{bmatrix} x_1^{-1} & 0 & 0 \\ x_2^{-1} & 0 & 0 \\ x_3^{-1} & 0 & 0 \end{bmatrix}. \end{aligned}$$

(4) Matrices de Cauchy : soient $x_1, \dots, x_N, y_1, \dots, y_N \in \mathbb{K}$ tels que $x_i \neq y_j$ pour tout i, j, alors

$$\mathcal{C}(x,y) = \left[\frac{1}{x_i - y_j}\right]_{1\le i,j\le N} = \begin{bmatrix} \frac{1}{x_1-y_1} & \frac{1}{x_1-y_2} & \cdots & \frac{1}{x_1-y_n} \\ \frac{1}{x_2-y_1} & \frac{1}{x_2-y_2} & \cdots & \frac{1}{x_2-y_N} \\ \vdots & \vdots & & \vdots \\ \frac{1}{x_N-y_1} & \frac{1}{x_N-y_2} & \cdots & \frac{1}{x_N-y_N} \end{bmatrix} \in \mathbb{K}^{N\times N}.$$

Dans ce cas, les bons déplacements sont

$$S = \operatorname{diag}(x_1, \dots, x_n) \quad , \quad T = \operatorname{diag}(y_1, \dots, y_n).$$

On a $\operatorname{rang}_{S,T}(A) = 1$; vérifions cela pour $N = 3$:

$$\begin{aligned} \nabla_{S,T}(\mathcal{C}(x,y)) &= \begin{bmatrix} x_1 & & \\ & x_2 & \\ & & x_3 \end{bmatrix} \cdot \begin{bmatrix} \frac{1}{x_1-y_1} & \frac{1}{x_1-y_2} & \frac{1}{x_1-y_3} \\ \frac{1}{x_2-y_1} & \frac{1}{x_2-y_2} & \frac{1}{x_2-y_3} \\ \frac{1}{x_3-y_1} & \frac{1}{x_3-y_2} & \frac{1}{x_3-y_3} \end{bmatrix} \\ &\quad - \begin{bmatrix} \frac{1}{x_1-y_1} & \frac{1}{x_1-y_2} & \frac{1}{x_1-y_3} \\ \frac{1}{x_2-y_1} & \frac{1}{x_2-y_2} & \frac{1}{x_2-y_3} \\ \frac{1}{x_3-y_1} & \frac{1}{x_3-y_2} & \frac{1}{x_3-y_3} \end{bmatrix} \cdot \begin{bmatrix} y_1 & & \\ & y_2 & \\ & & y_3 \end{bmatrix} \\ &= \begin{bmatrix} \frac{x_1}{x_1-y_1} & \frac{x_1}{x_1-y_2} & \frac{x_1}{x_1-y_3} \\ \frac{x_2}{x_2-y_1} & \frac{x_2}{x_2-y_2} & \frac{x_2}{x_2-y_3} \\ \frac{x_3}{x_3-y_1} & \frac{x_3}{x_3-y_2} & \frac{x_3}{x_3-y_3} \end{bmatrix} - \begin{bmatrix} \frac{y_1}{x_1-y_1} & \frac{y_2}{x_1-y_2} & \frac{y_3}{x_1-y_3} \\ \frac{y_1}{x_2-y_1} & \frac{y_2}{x_2-y_2} & \frac{y_3}{x_2-y_3} \\ \frac{y_1}{x_3-y_1} & \frac{y_2}{x_3-y_2} & \frac{y_3}{x_3-y_3} \end{bmatrix} \\ &= \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}. \end{aligned}$$

On summarise :

S	T	Structure	$\text{rang}_{S,T}(A)$
Z_1	Z_0	Toeplitz	2
Z_1	Z_0^T	Hankel	2
$\text{diag}(x_i^{-1})$	Z_0	Vandermonde	1
$\text{diag}(x_i)$	$\text{diag}(y_i)$	Cauchy	1

Ces quattre structures dépendent de $O(N)$ paramètres et les opérateurs de déplacement ∇ associés sont simples. Les matrices Toeplitz et Hankel peuvent se recupérer à partir de leurs déplacements $\nabla(A)$, les matrices de Cauchy se récupérent à partir de l'opérateur ∇ lui même ; les matrices de Vandermonde peuvent se récupérer soit de l'opérateur ∇ soit du déplacement $\nabla(A)$.

Une matrice $A \in \mathbb{K}^{N\times N}$ est *type Toeplitz/Hankel/Vandermonde/Cauchy* si pour les déplacements correspondant à ces structures on a

$$\text{rang}_{S,T}(A) \ll N.$$

1.1. Générateurs. La méthode de rang de déplacement appliquée à une matrice A consiste en trois étapes :

(1) **Compression :** la matrice A est codée par son déplacement $\nabla(A)$: le rang de $\nabla(A)$ doit être petit pour que la compression soit efficace.

(2) **Opération :** les opérations sur A (multiplication matrice-vecteur, élimina-tion) peuvent se faire à niveau des déplacements avec des techniques adaptées.

(3) **Decompression :** les résultats des opérations se recupérent à partir de leurs déplacements.

LEMME 1.2. *Soit $C \in \mathbb{K}^{M\times N}$ une matrice de rang α, alors il existent $G, B \in \mathbb{K}^{M\times\alpha}$ telles que*

$$C = G \cdot B^T.$$

DÉMONSTRATION. Deux démonstrations possibles :

(1) Considérons la DVS

$$C = V \cdot \Sigma \cdot U^*$$

avec $V \in U(M)$, $U \in U(N)$ et $\Sigma \in \mathbb{K}^{M\times N}$ "diagonal" dont les coefficients sont les valeurs singulières $\sigma_1 \geq \cdots \geq \sigma_{\min(M,N)} \geq 0$. Du fait que $\text{rang}(C) = \alpha$ on a $\sigma_j = 0$ pour $j > \alpha$. Soient

$$v_1, \ldots, v_\alpha \in \mathbb{K}^M \quad , \quad u_1, \ldots, u_\alpha \in \mathbb{K}^N$$

les premières α colonnes de V et de U respectivement, alors

$$C = [v_1 \cdots v_\alpha] \cdot \text{diag}(\sigma_1, \ldots, \sigma_\alpha) \cdot [u_1 \cdots u_\alpha]^*$$

en on peut prendre

$$G := [v_1 \cdots v_\alpha] \cdot \text{diag}(\sigma_1^{1/2}, \ldots, \sigma_\alpha^{1/2}), \tag{17}$$

$$B := \cdot[\overline{u}_1 \cdots \overline{u}_\alpha] \cdot \text{diag}(\sigma_1^{1/2}, \ldots, \sigma_\alpha^{1/2}).$$

(2) Soient $c_1, \ldots, c_N \in \mathbb{K}^N$ les colonnes de C et prenons $g_1, \ldots, g_\alpha \in \mathbb{K}^M$ une base quelconque de l'espace linéaire engendré par ces vecteurs. Soient alors $b_{j,k}$ $(1 \leq j \leq N, 1 \leq k \leq \alpha)$ tels que

$$c_j = \sum_{k=1}^{\alpha} b_{k,j} g_k \quad \text{pour } 1 \leq j \leq N. \tag{18}$$

On pose alors

$$G := [g_1 \cdots g_\alpha] \quad , \quad B := [b_{j,k}]_{1\leq j\leq N, 1\leq k\leq \alpha};$$

l'équation (18) équivaut alors à ce que $C = G \cdot B^T$.

□

Une couple (G, B) comme dans le lemme ci-dessus est un système de *générateurs* de la matrice C. Le nombre de coefficients de C est MN et celui des générateurs (G, B) est $\alpha(M + N)$. Donc si α est petit devant M, N, la représentation $G \cdot B$ est nettement plus compacte. La morale à retenir est :

Une matrice à petit rang est computationnallement petite

Les générateurs d'une matrice ne sont nullement uniquement déterminés, et une choix possible est de prendre $g_1, \ldots, g_\alpha$ parmi les colonnes de C comme dans la deuxième démonstration du lemme ci-dessus. Cela fixe d'avance α colonnes de B (c'est une sous-matrice $\mathbf{1}_\alpha$) et donc le nombre de coefficients non triviaux dans B est $\alpha(N - \alpha)$. Avec cette choix, les générateurs (G, B) seront représentés par $\alpha(M + N - \alpha)$ coefficients ; observons que

$$MN \geq \alpha(M + N - \alpha).$$

Cependant, les générateurs à colonnes orthogonales qu'on obtient *via* la DSV (display (17)) sont préferables si l'on veut assurer la stabilité numérique de la représentation, *voir* [**30**, §4.6.1] pour plus d'information.

En conséquent, les matrices de petit rang seront codées et traitées *via* des générateurs et non pas comme des matrices denses. En particulier, une matrice A à petit rang de déplacement sera codée par des générateurs (G, B) pour le déplacement $\nabla(A)$.

Générateurs pour les structures classiques : on fait toujours le cas $N = 3$, pour N quelconque les formules se généralisant de façon évidente.

(1) Matrices de Toeplitz :

$$\nabla_{Z_1,Z_0}(\mathcal{T}) = \begin{bmatrix} a_1 & & \\ a_2 & & \\ a_0 & a_{-1} - a_2 & a_{-2} - a_1 \end{bmatrix} = \begin{bmatrix} a_1 & \\ a_2 & \\ a_0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & & \\ 0 & a_{-1} - a_2 & a_{-2} - a_1 \end{bmatrix}.$$

(2) Matrices de Hankel :

$$\nabla_{Z_1,Z_0^T}(\mathcal{H}) = \begin{bmatrix} & & a_3 \\ & & a_4 \\ a_0 - a_3 & a_1 - a_4 & a_2 \end{bmatrix} = \begin{bmatrix} & a_3 \\ & a_4 \\ 1 & a_2 \end{bmatrix} \cdot \begin{bmatrix} a_0 - a_3 & a_1 - a_4 & \\ & & 1 \end{bmatrix}.$$

(3) Matrices de Vandermonde :

$$\nabla_{S,T}(\mathcal{V}) = \begin{bmatrix} x_1^{-1} & 0 & 0 \\ x_2^{-1} & 0 & 0 \\ x_3^{-1} & 0 & 0 \end{bmatrix} = \begin{bmatrix} x_1^{-1} \\ x_2^{-1} \\ x_3^{-1} \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}.$$

(4) Matrices de Cauchy :

$$\nabla_{S,T}(\mathcal{C}) = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}.$$

1.2. Opérations. Passons revue aux propriétés de base, qu'on démontrera en exercice :

LEMME 1.3. *Soient* $A, B, S, T, U \in \mathbb{K}^{N\times N}$, *alors*

(1) $\nabla_{S,T}(A+B) = \nabla_{S,T}(A) + \nabla_{S,T}(B)$;

(2) $\nabla_{T^T,S^T}(A^T) = \nabla_{S,T}(A)^T$;

(3) $\nabla_{T,S}(A^{-1}) = -A^{-1}\cdot\nabla_{S,T}(A)\cdot A^{-1}$;

(4) $\nabla_{S,U}(A\cdot B) = \nabla_{S,T}(A)\cdot B + A\cdot\nabla_{T,U}(B)$.

En particulier

(1) $\mathrm{rang}_{S,T}(A+B) \le \mathrm{rang}_{S,T}(A) + \mathrm{rang}_{S,T}(B)$;

(2) $\mathrm{rang}_{T^T,S^T}(A^T) = \mathrm{rang}_{S,T}(A)$;

(3) $\mathrm{rang}_{T,S}(A^{-1}) = \mathrm{rang}_{S,T}(A)$;

(4) $\mathrm{rang}_{S,U}(A\cdot B) \le \mathrm{rang}_{S,T}(A) + \mathrm{rang}_{T,U}(B)$.

Ces propriétés se généralisent à des matrices rectagulaires dès que les dimensions sont compatibles.

Ces formules sont particulièrement sympathiques lorsque $S = T$, par exemple pour les matrices de Toeplitz, où l'on peut prendre $S = T = Z_0$. Dans ce cas, les matrices formées à partir d'autres au moyen d'opérations arithmétiques sont de rang de déplacement controlé. Par exemple, si T_1, T_2, T_3 sont Toeplitz, alors

$$T_1^{-1}T_2 - T_3$$

est de (Z_0, Z_0)-rang au plus 6.

Ainsi on peut obtenir des générateurs pour les matrices produites, en termes de générateurs des matrices données (sauf pour l'inversion, bien evidémment !). Soient

$$\nabla_{S,T}(A) = G\cdot C^T \quad , \quad \nabla_{S,T}(B) = G'\cdot (C')^T$$

des générateurs de longueur α et β respectivement, alors

$$\nabla_{S,T}(A+B) = \nabla_{S,T}(A) + \nabla_{S,T}(B) = G\cdot C^T + G'\cdot (C')^T = [G|G']\cdot[C|C']^T \tag{19}$$

où $[G|G'], [C|C'] \in \mathbb{K}^{N\times(\alpha+\beta)}$ est la *concatenation* de matrices G, G' et C, C' respectivement. Pour

$$\nabla_{S,T}(A) = G\cdot C^T \quad , \quad \nabla_{T,U}(B) = G'\cdot (C')^T,$$

on a

$$\nabla_{S,U}(A\cdot B) = \nabla_{S,T}(A)\cdot B + A\cdot\nabla_{T,U}(B) = [G|AG']\cdot[B^TC|C']^T. \tag{20}$$

Les générateurs ainsi produits ne sont pas forcémment de longueur minimale. Dans la sous-section 2.1 on donne des techniques pour rendre minimale la longueur d'un système de générateurs donné.

Considérons la décomposition en blocs

$$A = \begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{bmatrix} \quad \begin{matrix} i \\ N-i \end{matrix}$$
$$\begin{matrix} i & N-i \end{matrix}$$

et similairement pour les opérateurs S et T. Le lemme suivant fournit des formules pour le déplacement de chacun des blocs ; la vérification en est directe :

LEMME 1.4. *Pour* $1 \le i, j \le 2$

$$\nabla_{S_{i,i},T_{j,j}}(A_{i,j}) = \nabla_{S,T}(A)_{i,j} - A_{i,3-j}\cdot T_{3-j,j} + S_{i,3-i}\cdot A_{3-i,j}.$$

Soit

$$\beta_{S,T} := \max\{\operatorname{rang}(S_{1,2}), \operatorname{rang}(S_{2,1}), \operatorname{rang}(T_{1,2}), \operatorname{rang}(T_{2,1})\}$$

le maximum des dimensions des blocs de S et T en dehors de la diagonale, alors ce lemme entraîne

$$\begin{aligned}\operatorname{rang}_{S_{i,i},T_{j,j}}(A_{i,j}) &\leq \operatorname{rang}(\nabla_{S,T}(A)_{i,j}) + \operatorname{rang}(T_{3-j,j}) + \operatorname{rang}(S_{i,3-i}) \\ &\leq \operatorname{rang}(\nabla_{S,T}(A)_{i,j}) + 2\beta_{S,T}.\end{aligned}$$

Les déplacements S et T sont *quasi-diagonal par blocs* si pour tout $1 \leq i \leq N-1$ les blocs

$$S_{1,2} \quad , \quad S_{2,1} \quad , \quad T_{1,2} \quad , \quad T_{2,1}$$

sont de rang petit devant N. Pour les quattre structures classiques on a $\beta_{S,T} \leq 1$. Pour des déplacements quasi-diagonales par blocs, le rang de déplacement des matrices produites par les opérations de somme, multiplication, inversion, transposition, et projection aux blocs reste controlé.

Supposons $A_{1,1} \in \mathbb{K}^{i\times i}$ inversible et soit

$$A_{(i)} := A_{2,2} - A_{2,1}A_{1,1}^{-1}A_{1,2} \in \mathbb{K}^{(N-i)\times(N-i)}$$

le i-ème complément de Schur de A. On peut obtenir une expression pour le déplacement de $A_{(i)}$ en combinant les lemmes 1.3 et 1.4 ; cependant la proposition suivante nous en fournit une expression plus compacte :

PROPOSITION 1.5. *Soient $A, S, T \in \mathbb{K}^{N\times N}$ et soit*

$$\begin{aligned}\nabla_{S,T}(A) &= \begin{bmatrix} S_{1,1} & S_{1,2} \\ S_{2,1} & S_{2,2}\end{bmatrix} \cdot \begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2}\end{bmatrix} - \begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2}\end{bmatrix} \cdot \begin{bmatrix} T_{1,1} & T_{1,2} \\ T_{2,1} & T_{2,2}\end{bmatrix} \\ &= \begin{bmatrix} G_1 \\ G_2\end{bmatrix} \cdot \begin{bmatrix} B_1^T & B_2^T\end{bmatrix},\end{aligned}$$

la décomposition du déplacement de A dans des blocs de taille i et $N-i$. Supposons $A_{1,1}$ inversible et soit $A_{(i)}$ le complément de Schur correspondant, alors

$$\nabla_{S_{2,2},T_{2,2}}(A_{(i)}) = H \cdot C^T + A_{2,1} \cdot A_{1,1}^{-1} \cdot S_{1,2} \cdot A_{(i)} - A_{(i)} \cdot T_{2,1} \cdot A_{1,1}^{-1} \cdot A_{1,2} \tag{21}$$

avec

$$H = G_2 - A_{2,1} \cdot A_{1,1}^{-1} \cdot G_1 \quad , \quad C = B_2 - (A_{1,1}^{-1} \cdot A_{1,2})^T \cdot B_1 \quad \in \mathbb{K}^{(N-i)\times\alpha}. \tag{22}$$

DÉMONSTRATION. On a

$$A = \begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2}\end{bmatrix} = \underbrace{\begin{bmatrix} \mathbf{1}_i & \\ A_{2,1}A_{1,1}^{-1} & \mathbf{1}_{N-i}\end{bmatrix}}_{V} \cdot \begin{bmatrix} A_{1,1} & \\ & A_{(i)}\end{bmatrix} \cdot \underbrace{\begin{bmatrix} \mathbf{1}_i & A_{1,1}^{-1}A_{1,2} \\ & \mathbf{1}_{N-i}\end{bmatrix}}_{W}.$$

Les matrices V et W ci-dessus sont inversibles et on a

$$\begin{aligned}V^{-1}SV &= \begin{bmatrix} \mathbf{1}_i & \\ -A_{2,1}A_{1,1}^{-1} & \mathbf{1}_{N-i}\end{bmatrix} \cdot \begin{bmatrix} S_{1,1} & S_{1,2} \\ S_{2,1} & S_{2,2}\end{bmatrix} \cdot \begin{bmatrix} \mathbf{1}_i & \\ A_{2,1}A_{1,1}^{-1} & \mathbf{1}_{N-i}\end{bmatrix} \\ &= \begin{bmatrix} * & * \\ * & S_{2,2} - A_{2,1}A_{1,1}^{-1}S_{1,2}\end{bmatrix}\end{aligned}$$

et similairement

$$WTW^{-1} = \begin{bmatrix} * & * \\ * & T_{2,2} - T_{2,1}A_{1,1}^{-1}A_{1,2}\end{bmatrix};$$

donc

$$\begin{aligned} V^{-1}(SA-AT)W^{-1} &= (V^{-1}SV)(V^{-1}AW^{-1}) - (V^{-1}AW^{-1})(WTW^{-1}) \\ &= \begin{bmatrix} * & * \\ * & S_{2,2} - A_{2,1}A_{1,1}^{-1}S_{1,2} \end{bmatrix} \cdot \begin{bmatrix} A_{1,1} & \\ & A_{(i)} \end{bmatrix} \\ &- \begin{bmatrix} A_{1,1} & \\ & A_{(i)} \end{bmatrix} \cdot \begin{bmatrix} * & * \\ * & T_{2,2} - T_{2,1}A_{1,1}^{-1}A_{1,2} \end{bmatrix} \\ &= \begin{bmatrix} * & * \\ * & \nabla_{S_{2,2}-A_{2,1}A_{1,1}^{-1}S_{1,2},T_{2,2}-T_{2,1}A_{1,1}^{-1}A_{1,2}}(A_{(i)}) \end{bmatrix}. \end{aligned}$$

De l'autre côté

$$\begin{aligned} V^{-1}(SA-AT)W^{-1} &= \begin{bmatrix} \mathbf{1}_i & \\ -A_{2,1}A_{1,1}^{-1} & \mathbf{1}_{N-i} \end{bmatrix} \cdot \begin{bmatrix} G_1 \\ G_2 \end{bmatrix} \cdot \begin{bmatrix} B_1^T & B_2^T \end{bmatrix} \cdot \begin{bmatrix} \mathbf{1}_i & -A_{1,1}^{-1}A_{1,2} \\ & \mathbf{1}_{N-i} \end{bmatrix} \\ &= \begin{bmatrix} G_1 \\ H \end{bmatrix} \cdot \begin{bmatrix} B_1^T & C^T \end{bmatrix} \end{aligned}$$

avec H, C comme dans les formules (22) ; en identifiant les blocs $(2,2)$ on obtient

$$\begin{aligned} &\nabla_{S_{2,2}-A_{2,1}A_{1,1}^{-1}S_{1,2},T_{2,2}-T_{2,1}A_{1,1}^{-1}A_{1,2}}(A_{(i)}) \\ &= \nabla_{S_{2,2},T_{2,2}}(A_{(i)}) - A_{2,1}A_{1,1}^{-1}S_{1,2}A_{(i)} + A_{(i)}T_{2,1}A_{1,1}^{-1}A_{1,2} = H \cdot C^T. \end{aligned}$$

□

Pour des opérateurs S et T respectivement triangulaire inférieur et supérieur on obtient le résultat le plus net :

Corollaire 1.6. *Avec les notations de la proposition 1.6, suppossons S, T respectivement triangulaire inférieur et supérieur par blocs ($S_{1,2} = T_{2,1} = 0$), alors*

$$\nabla_{S_{2,2},T_{2,2}}(A_{(i)}) = H \cdot C^T$$

avec

$$H = G_2 - A_{2,1} \cdot A_{1,1}^{-1} \cdot G_1 \quad , \quad C = B_2 - (A_{1,1}^{-1} \cdot A_{1,2})^T \cdot B_1 \quad \in \mathbb{K}^{(N-i)\times\alpha}. \tag{23}$$

En particulier

$$\operatorname{rang}_{S_{2,2},T_{2,2}}(A_{(i)}) \leq \operatorname{rang}_{S,T}(A).$$

Ainsi, dans cette situation le rang de déplacement n'augmente pas par l'opération de prendre complément de Schur, et de plus grâce aux formules (23) on peut faire le passage

$$(G, B) \to (H, C)$$

sans avoir à expliciter le complément de Schur $A_{(i)}$ lui même. Ce genre de propriété est la clé de tous les algorithmes dans cette domaine. Notons en passant que ces formules d'actualisation ne font intervernir les matrices de déplacement.

Plus généralement, si les déplacements S et T sont respectivement *quasi-triangulaires par blocs* inférieure et supérieure si les blocs

$$S_{1,2} \quad , \quad T_{2,1}$$

sont de rang petit devant N. Soit

$$\gamma_{S,T} := \max\{\operatorname{rang}(S_{1,2}), \operatorname{rang}(T_{2,1})\},$$

alors les formules (24) entraînent que dans ce cas, le rang de déplacement du complément de Schur reste controlé :

$$\operatorname{rang}_{S_{2,2},T_{2,2}}(A_{(i)}) \leq \operatorname{rang}_{S,T}(A) + \operatorname{rang}(S_{1,2}) + \operatorname{rang}(T_{2,1}) \leq \operatorname{rang}_{S,T}(A) + 2\gamma_{S,T}; \tag{24}$$

Pour les quattre structures classiques :

Structure	$\beta_{S,T}$	$\gamma_{S,T}$
Toeplitz	1	1
Hankel	1	1
Vandermonde	1	0
Cauchy	0	0

1.3. Reconstruction. Dans cette section on étude l'inversion de l'opérateur de déplacement, necessaire pour l'étape de decompression dans les algorithmes de résolution pour les matrices structurées. Le problème est celui de résoudre en A l'équation

$$S \cdot A - A \cdot T = \nabla \tag{25}$$

pour $\nabla \in \mathbb{K}^{M\times N}$ donnée. Cette équation linéaire a un nom, c'est l'*équation de Sylvester continue*, et sa solvabilité admets une caractérisation sympathique (Proposition 1.8 ci-dessous).

D'abord on linéarise le problème et pour cela on a besoin d'un peu de notation. Pour une matrice $A \in \mathbb{K}^{M\times N}$ soient $a_1, \ldots, a_N \in \mathbb{K}^M$ ses colonnes et posons

$$\mathrm{Vect}(A) = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_N \end{bmatrix} \in \mathbb{K}^{MN}$$

le *vecteur colonne* fait des colonnes de A empilées les unes sur les autres. Pour $C \in \mathbb{K}^{M\times N}$ et $D \in \mathbb{K}^{P\times Q}$, le *produit tensoriel* $C \otimes D$ est

$$C \otimes D = \begin{bmatrix} c_{1,1}D & \cdots & c_{1,N}D \\ \vdots & & \vdots \\ c_{M,1}D & \cdots & c_{M,N}D \end{bmatrix} \in \mathbb{K}^{MP\times NQ}.$$

Proposition 1.7. *L'équation $S \cdot A - A \cdot T = \nabla$ équivaut à*

$$(\mathbf{1}_N \otimes S + T^T \otimes \mathbf{1}_N) \cdot \mathrm{Vect}(A) = \mathrm{Vect}(\nabla). \tag{26}$$

Démonstration. Le processus de vectorisation est une bijection linéaire, donc l'équation de Sylvester continue équivaut à

$$\mathrm{Vect}(S \cdot A) - \mathrm{Vect}(A \cdot T) = \mathrm{Vect}(\nabla).$$

Soient $a_1, \ldots, a_N \in \mathbb{K}^N$ les colonnes de A, alors

$$(\mathbf{1}_N \otimes S)\mathrm{Vect}(A) = \begin{bmatrix} S & & \\ & \ddots & \\ & & S \end{bmatrix} \begin{bmatrix} a_1 \\ \vdots \\ a_N \end{bmatrix} = \mathrm{Vect}(S \cdot A) \tag{27}$$

et pour l'autre terme,

$$(T^T \otimes \mathbf{1}_M)\mathrm{Vect}(A) = \begin{bmatrix} t_{1,1}\mathbf{1}_M & \cdots & t_{N,1}\mathbf{1}_M \\ \vdots & & \vdots \\ t_{1,N}\mathbf{1}_M & \cdots & t_{N,N}\mathbf{1}_M \end{bmatrix} \begin{bmatrix} a_1 \\ \vdots \\ a_N \end{bmatrix} = \mathrm{Vect}(A \cdot T), \tag{28}$$

d'où on conclut. □

Proposition 1.8. *Les valeurs propres de $\nabla_{S,T}$ sont de la forme $\sigma - \tau$, où σ et τ sont des valeurs propres de S et de T, respectivement. En particulier, $\nabla_{S,T}$ est inversible si et seulement si* $\mathrm{Spec}(S) \cap \mathrm{Spec}(T) = \emptyset$.

DÉMONSTRATION. Quitte à passer à $\mathbb{C}$, les matrices S et T sont conjuguées de matrices triangulaires supérieure et inférieure, respectivement. Écrivons des telles factorisations

$$S = E \cdot U \cdot E^{-1}, \quad T = F \cdot L \cdot F^{-1}$$

avec $E, U \in \mathbb{C}^{M\times M}$ et $F, L \in \mathbb{C}^{N\times N}$, E, F inversibles et U, L triangulaires supérieure et inférieure, respectivement. On a

$$\begin{aligned} E^{-1} \cdot \nabla_{S,T}(A) \cdot F^{-1} &= (E^{-1}SE)(E^{-1}AF^{-1}) - (E^{-1}AF^{-1})(FTF^{-1}) \\ &= U(E^{-1}AF^{-1}) - (E^{-1}AF^{-1})L = \nabla_{U,L}(E^{-1}AF^{-1}) \end{aligned} \tag{29}$$

donc les valeurs propres de $\nabla_{S,T}$ coïncident avec ceux de $\nabla_{U,L}$.

Les expressions (27) et (28) appliquées à U, L à la place de S, T, montrent que la matrice $\mathbf{1}_n \otimes U + L^T \otimes \mathbf{1}_m$ est triangulaire supérieure et donc ses valeurs propres sont les éléments dans la diagonale principale. Cette diagonale est

$$(\mathrm{diag}(U), \dots, \mathrm{diag}(U)) - (\ell_{1,1}, \dots, \ell_{1,1}, \ell_{2,2}, \dots, \ell_{2,2}, \dots, \ell_{n,n}, \dots, \ell_{n,n}) \in \mathbb{C}^{MN},$$

et ses éléments sont de la forme $\sigma - \tau$ avec $\sigma \in \mathrm{Spec}(U) = \mathrm{Spec}(S)$ et $\tau \in \mathrm{Spec}(L) = \mathrm{Spec}(T)$. □

On isole l'identité (29) dans la preuve ci-dessus, pour utilisation ultérieure :

LEMME 1.9. *Soient $S = E \cdot U \cdot E^{-1}$ et $T = F \cdot L \cdot F^{-1}$ avec $E, U \in \mathbb{C}^{M\times M}$ et $F, L \in \mathbb{C}^{N\times N}$, alors*

$$\nabla_{S,T}(A) = E \cdot \nabla_{U,L}(E^{-1}AF^{-1}) \cdot F.$$

Reconstruction de matrices type Cauchy :

PROPOSITION 1.10. *Soient $x, y \in \mathbb{K}^N$ tels que $x_i \neq y_j$ pour tout $1 \leq i, j \leq N$, et $A, G, B \in \mathbb{K}^{N\times N}$ tel que*

$$\mathrm{diag}(x) \cdot A - A \cdot \mathrm{diag}(y) = G \cdot B^T,$$

alors

$$A = \sum_{k=1}^{\alpha} \mathrm{diag}(g_k) \cdot \mathcal{C}(x, y) \cdot \mathrm{diag}(b_k).$$

DÉMONSTRATION. Notons C la matrice définie par l'expression de droite, alors

$$\nabla(C) = \sum_{k=1}^{\alpha} \mathrm{diag}(g_k) \cdot \nabla(\mathcal{C}(x, y)) \cdot \mathrm{diag}(b_k)$$

puisque les déplacements sont diagonaux et donc commutent avec $\mathrm{diag}(g_k)$ et $\mathrm{diag}(b_k)$. Ainsi

$$\nabla(C) = \sum_{k=1}^{\alpha} \mathrm{diag}(g_k) \cdot \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \cdot \begin{bmatrix} 1 \cdots 1 \end{bmatrix} \cdot \mathrm{diag}(b_k) = \sum_{k=1}^{\alpha} g_k \cdot b_k^T = G \cdot B^T = \nabla(A),$$

ce qui entraîne $A = C$ car $\mathrm{Spec}(\mathrm{diag}(x))$ et $\mathrm{Spec}(\mathrm{diag}(y))$ sont disjoints. □

En particulier, si S, T sont diagonalizables à spectre disjoint, on peut résoudre $SA - AT = \nabla$ par reduction au cas des type Cauchy, grâce au lemme 1.9.

Reconstruction des matrices type Vandermonde :

PROPOSITION 1.11. *Soient $x_1, \dots, x_N \in \mathbb{K}^\times$ et $A, G, B \in \mathbb{K}^{N\times N}$ such that*

$$\operatorname{diag}(x_i^{-1})_{1\le i\le N} \cdot A - A \cdot Z_0 = G \cdot B^T,$$

alors

$$A = \sum_{k=1}^{\alpha} \operatorname{diag}(x_i g_{i,k})_{1\le i\le N} \cdot \mathcal{V}(x) \cdot \begin{bmatrix} b_{1,k} & b_{2,k} & \cdots & b_{N,k} \\ & b_{1,k} & \cdots & b_{N-1,k} \\ & & \ddots & \vdots \\ & & & b_{1,k} \end{bmatrix}. \tag{30}$$

DÉMONSTRATION. On a

$$\mathcal{B}_k := \begin{bmatrix} b_{1,k} & b_{2,k} & \cdots & b_{N,k} \\ & b_{1,k} & \cdots & b_{N-1,k} \\ & & \ddots & \vdots \\ & & & b_{1,k} \end{bmatrix} = b_{1,k}\mathbf{1}_N + b_{2,k}Z_0 + \cdots + b_{N,k}Z_0^{N-1}$$

donc $\mathcal{B}_k$ commute avec Z_0. Soit C la matrice définie para l'expression de droite dans (30), alors

$$\begin{aligned} \nabla(C) &= \sum_{k=1}^{\alpha} \left(\operatorname{diag}(x_i g_{i,k})_i \cdot \operatorname{diag}(x_i^{-1})_i \cdot \mathcal{V}(x) \cdot \mathcal{B}_k - \operatorname{diag}(x_i g_{i,k})_i \cdot \mathcal{V}(x) \cdot Z_0 \cdot \mathcal{B}_k\right) \\ &= \sum_{k=1}^{\alpha} \operatorname{diag}(x_i g_{i,k})_i \cdot \begin{bmatrix} x_1^{-1} \\ x_2^{-1} \\ \vdots \\ x_N^{-1} \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & \cdots & 0 \end{bmatrix} \cdot \mathcal{B}_k \\ &= \sum_{k=1}^{\alpha} g_k \cdot b_k^T \\ &= G \cdot B^T = \nabla(A), \end{aligned}$$

ce qui entraîne $A = C$, car

$$\operatorname{Spec}(\operatorname{diag}(x_i^{-1})_i) = \{x_1^{-1}, \dots, x_N^{-1}\} \quad , \quad \operatorname{Spec}(Z_0) = \{0\}$$

sont disjoints. □

Le circulant est diagonalisable *via* la TFD :

PROPOSITION 1.12. *Soit $\omega = e^{-2i\pi/N}$ et $F_N = [\omega^{ij}]_{0\le i,j\le N-1}$ la matrice de la TFD, alors*

$$Z_1 = F_N \cdot \operatorname{diag}(\omega^j)_{0\le j\le N-1} \cdot F_N^{-1}.$$

DÉMONSTRATION. On a

$$Z_1 v = \lambda v \quad \Longleftrightarrow \quad \begin{aligned} v_2 &= \lambda v_1, \\ v_3 &= \lambda v_2, \\ &\vdots \\ v_1 &= \lambda v_N; \end{aligned}$$

d'où

$$\lambda = \omega^j \quad , \quad v = (1, \omega^j, \dots, (\omega^j)^{N-1})$$

pour quelque $0 \le j \le N-1$, donc

$$Z_1 \cdot F_N = F_N \cdot \operatorname{diag}(1, \omega, \cdots, \omega^{N-1}).$$

□

Ceci entraîne la réduction des matrices type Toeplitz au type Vandermonde : posons $D := \mathrm{diag}(\omega^j)_{0\le j\le N-1}$, pour $A \in \mathbb{K}^{N\times N}$

$$\nabla_{Z_1,Z_0}(A) = F_N \cdot \nabla_{D,Z_0}(F_N^{-1}\cdot A) = \frac{1}{N} F_N \cdot \nabla_{D,Z_0}(F_N^{*}\cdot A),$$

donc si A est type Toeplitz alors $F_N^* \cdot A$ est type Vandermonde avec le même rang de déplacement, et si $\nabla_{Z_1,Z_0}(A) = G\cdot B^T$ alors

$$\nabla_{D,Z_0}(F_N^*\cdot A) = (NF_N\cdot G)\cdot B^T.$$

De plus la multiplication $F_N \cdot G$ n'est pas chère car elle se fait en $1.5\alpha N\log(N)$ ops avec la TFR.

Posons

$$J = \begin{bmatrix} & & 1 \\ & \diagup & \\ 1 & & \end{bmatrix}.$$

Ceci est une permutation et on a

$$JJ^T = J^2 = 1 \quad , \quad Z_0^T = J\cdot Z_0\cdot J^{-1}.$$

Avec ceci on réduit facilement le type Hankel au type Toeplitz, et *a fortiori* au type Vandermonde : pour $A \in \mathbb{K}^{N\times N}$,

$$\nabla_{Z_1,Z_0^T}(A) = \nabla_{Z_1,Z_0}(A\cdot J)\cdot J$$

donc si $\nabla_{Z_1,Z_0^T}(A) = G\cdot B^T$ alors

$$\nabla_{Z_1,Z_0}(A\cdot J) = G\cdot(B^T\cdot J).$$

2. Algorithmes rapides

Les outils sont en place pour la version "rapide" de l'algorithme de résolution pour des matrices structurées. L'idée est d'opérer à niveau des générateurs et non pas sur les matrices elles mêmes ; seulement on se permets de reconstruire et d'opérer avec de petits morceaux des matrices sous-jacentes. Pour la décomposition LU sans pivotage d'une matrice A fortément inversible, l'algorithme consiste à calculer les *générateurs* des compléments de Schur $A_{(i)}$ successifs, puis de reconstruire à chaque étape les premières ligne et colonne de $A_{(i)}$, à fin de produire les facteurs L et U. En accord avec ça, on suppose que la matrice d'entrée A est donnée par ses générateurs et non pas par sa forme dense.

Dans une première approche, on se restreindra à des déplacements $S, T \in \mathbb{K}^{N\times N}$ triangulaires inférieure et supérieure respectivement (pour pouvoir appliquer la proposition 1.6) et tels que $\mathrm{Spec}(S)\cap\mathrm{Spec}(T) = \emptyset$.

Algorithme de décomposition LU de matrices structurées :

Entrée : $S, T \in \mathbb{K}^{N\times N}$ **resp. triangulaires inférieure et supérieure, telles que** $\mathrm{Spec}(S)\cap\mathrm{Spec}(T) = \emptyset$ **;**

$G_0, B_0 \in \mathbb{K}^{N\times\alpha}$ **générateurs d'une matrice** $A_{(0)} := A$ **fortément inversible ;**

Sortie : $L, U \in \mathbb{K}^{N\times N}$ **décomposition** LU **de** A**.**

For i **from 1 to** N **do**

(1) reconstruire le pivot et les premières ligne et colonne

$$A_{(i-1)}(i,i) \in \mathbb{K}^{\times},$$

$$A_{(i-1)}(i, i+1:N) \in \mathbb{K}^{1\times(N-i)},$$

$$A_{(i-1)}(i+1:N, i) \in \mathbb{K}^{(N-i)\times 1}$$

de $A_{(i-1)} \in \mathbb{K}^{(N-i+1)\times(N-i+1)}$ **vérifiant** $\nabla(A_{(i-1)}) = G_{i-1} \cdot B_{i-1}^T$ **;**

(2) construire la i**-ème colonne de** L **et la** i**-ème ligne de** U **:**

$$L(i,i) \leftarrow 1,$$
$$L(i+1:N,i) \leftarrow A_{(i-1)}(i,i)^{-1} \cdot A_{(i-1)}(i+1:N,i),$$
$$U(i,i:N) \leftarrow A_{(i-1)}(i:N,i);$$

(3) actualiser les générateurs :

$$G_i \leftarrow G_{i-1}(i+1:N) - A_{(i-1)}(i+1:N,i) \cdot A_{(i-1)}(i,i)^{-1} \cdot G_i(i),$$
$$B_i^T \leftarrow B_{i-1}(i+1:N) - (A_{(i-1)}(i,i)^{-1} \cdot A_{(i-1)}(i,i+1:N))^T \cdot B_i(i);$$

od ;
$L(N,N) \leftarrow 1, \quad U(N,N) \leftarrow G_N \cdot B_N$ **;**
end.

Pour introduire le pivotage, considerons l'effet d'une permutation P sur le déplacement :

$$P{\cdot}\nabla_{S,T}(A) = P{\cdot}S{\cdot}A - P{\cdot}A{\cdot}T = P{\cdot}S{\cdot}P^*{\cdot}(P{\cdot}A) - (P{\cdot}A){\cdot}T = \nabla_{P\cdot S\cdot P^*,T}(P{\cdot}A). \tag{31}$$

Donc la matrice pivotée $P \cdot A$ reste structurée, seulement S doit être conjugée par la permutation. Si l'on veut que $P \cdot S \cdot P^*$ reste triangulaire inférieure pour toute permutation P, il nous faut que S soit diagonale. Le pivotage partiel restreint encore l'application de l'algorithme rapide à des deplacements S *diagonal* et T *triangulaire supérieure*; et introduit le pas supplémentaire :

(0.5) reconstruire la première colonne du $(i-1)$**-ème complément de Schur**

$$A_{(i-1)}(i:N,i) \in \mathbb{K}^{(N-i+1)\times 1}$$

et déterminer $i \le k \le N$ **tel que** $|A_{(i-1)}(k,i)|$ **soit maximale. Garder** P_i **la permutation correspondante ; interchanger les lignes** i **et** k **dans le générateur** G_i **et conjuger** S **suivant** P_i**.**

Le pas (1) se réduit alors au calcul de la première ligne de $A_{(i-1)}$, puisque sa première colonne est déjà calculée. Il nous reste d'expliciter comment fait-on le pas de reconstruction des premières ligne et colonne de $A_{(i-1)}$. Voici le procédé pour le type Cauchy :

PROPOSITION 2.1. *Soient* $x, y \in \mathbb{K}^N$ *tels que* $x_i \neq y_j$ *pour tout* i, j*, et* $A \in \mathbb{K}^{N\times N}$ *tel que*

$$\mathrm{diag}(x) \cdot A - A \cdot \mathrm{diag}(y) = G \cdot B^T$$

pour des certains $G, B \in \mathbb{K}^{N\times\alpha}$*, alors*

$$A_{i,j} = \frac{1}{x_i - y_j} \sum_{k=1}^{\alpha} g_{i,k} b_{j,k}.$$

DÉMONSTRATION. C'est une vérification directe à partir de la proposition 1.10 ; faisons-le pour $N = 3$. Pour $1 \leq k \leq \alpha$

$$\begin{bmatrix} g_{1,k} & & \\ & g_{2,k} & \\ & & g_{3,k} \end{bmatrix} \cdot \begin{bmatrix} \frac{1}{x_1 - y_1} & \frac{1}{x_1 - y_2} & \frac{1}{x_1 - y_3} \\ \frac{1}{x_2 - y_1} & \frac{1}{x_2 - y_2} & \frac{1}{x_2 - y_3} \\ \frac{1}{x_3 - y_1} & \frac{1}{x_3 - y_2} & \frac{1}{x_3 - y_3} \end{bmatrix} \cdot \begin{bmatrix} b_{1,k} & & \\ & b_{2,k} & \\ & & b_{3,k} \end{bmatrix}$$

$$= \begin{bmatrix} \frac{g_{1,k}b_{1,k}}{x_1 - y_1} & \frac{g_{1,k}b_{2,k}}{x_1 - y_2} & \frac{g_{1,k}b_{3,k}}{x_1 - y_3} \\ \frac{g_{2,k}b_{1,k}}{x_2 - y_1} & \frac{g_{2,k}b_{2,k}}{x_2 - y_2} & \frac{g_{2,k}b_{3,k}}{x_2 - y_3} \\ \frac{g_{3,k}b_{1,k}}{x_3 - y_1} & \frac{g_{3,k}b_{2,k}}{x_3 - y_2} & \frac{g_{3,k}b_{3,k}}{x_3 - y_3} \end{bmatrix}$$

donc pour $1 \leq i, j \leq 3$

$$A_{i,j} = \frac{1}{x_i - y_j} \sum_{k=1}^{\alpha} g_{i,k} b_{j,k}.$$

□

Pour le type Vandermonde :

LEMME 2.2. *Soient $x_1, \ldots, x_N \in \mathbb{K}^\times$ et $A \in \mathbb{K}^{N \times N}$ tels que*

$$\operatorname{diag}(x_i^{-1})_{1 \leq i \leq N} \cdot A - A \cdot Z_0 = G \cdot B^T$$

pour des certains $G, B \in \mathbb{K}^{N \times \alpha}$, alors

$$A_{i,1} = \sum_{k=1}^{\alpha} g_{i,k} x_k b_{1,k} \quad , \quad i = 1, \ldots, N;$$

$$A_{1,j} = x_1 A_{1,j-1} + \sum_{k=1}^{\alpha} g_{1,k} x_1 b_{j,k} \quad , \quad j = 2, \ldots, N.$$

DÉMONSTRATION. C'est une vérification directe à partir de la proposition 1.11 ; faisons-le pour $N = 3$. Soit $1 \leq k \leq \alpha$, alors

$$\begin{bmatrix} g_{1,k}x_1 & & \\ & g_{2,k}x_2 & \\ & & g_{3,k}x_3 \end{bmatrix} \cdot \begin{bmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ 1 & x_3 & x_3^2 \end{bmatrix} \cdot \begin{bmatrix} b_{1,k} & b_{2,k} & b_{3,k} \\ & b_{1,k} & b_{2,k} \\ & & b_{1,k} \end{bmatrix}$$

$$= \begin{bmatrix} g_{1,k}x_1b_{1,k} & g_{1,k}x_1b_{2,k} + g_{1,k}x_1^2b_{1,k} & g_{1,k}x_1b_{3,k} + g_{1,k}x_1^2b_{2,k} + g_{1,k}x_1^3b_{1,k} \\ g_{2,k}x_2b_{1,k} & * & * \\ g_{3,k}x_3b_{1,k} & * & * \end{bmatrix};$$

à partir d'ici on vérifie facilement les formules proposées. □

Les déplacements sont sujets à des conditions qui restreintent l'application de cet algorithme aux types Vandermonde et Cauchy, parmi les quattre structures classiques. On peut sauter en quelque sorte cette restriction, si l'on réduit le type Hankel au type Toeplitz en le post-multiplicant par une permutation convénable, et on réduit le type Toeplitz au type Vandermonde en le preconditionnant par une TFD. Dans la sous-section 2.2 on généralisera l'algorithme à des déplacements S quasi-diagonal par blocs et T quasi-triangulaire par blocs, ce qu'étends son rang d'application et en particulier permets une approche unifiée à les quattre structures classiques.

PROPOSITION 2.3. *Le coût de l'algorithme de décomposition PLU pour une matrice $A \in \mathbb{K}^{N \times N}$ type Toeplitz/Hankel/Vadermonde/Cauchy avec $\operatorname{rang}_{S,T}(A) \leq \alpha$ est de*

$$O(\alpha N^2) \quad \text{ops}.$$

DÉMONSTRATION. On fait la démonstration en détail pour la structure type Vandermonde. Pour chaque $1 \leq i \leq N-1$ on fait

(1) $2\alpha(N-i+1)+\alpha-1$ ops pour la reconstruction de la première colonne du complément de Schur $A_{(i)}$;
(2) le pivotage demande $N-i+1$ comparaisons ;
(3) la reconstruction de la première ligne de $A_{(i)}$ demande $2\alpha(N-i+1)+\alpha+1$ ops ;
(4) le calcul des générateurs du complément de Schur suivant $A(i+1)$ demande $\alpha+2\alpha(N-i+1)$.

Le coût total s'estime en

$$\sum_{i=1}^{N-1}(2\alpha(N-i+1)+\alpha-1)+(N-i+1)+(2\alpha(N-i+1)+\alpha+1)$$
$$+(\alpha+2\alpha(N-i+1))=\left(3\alpha+\frac{1}{2}\right)N^2+O(\alpha N).$$

□

EXERCICE 5.1. ◁ Étendre l'algorithme de décomposition PLU à des matrices structurées rectangulaires quelconques. ▷

2.1. Compression de générateurs. La longueur des générateurs augmente avec les opérations de somme, multiplication, projections aux blocs et complément de Schur, ce qui conduit à des générateurs trop longs pour des matrices potentiellement à petit rang de déplacement. Cette situation apparaîtra naturellement lorsqu'on voudra étendre l'algorithme rapide à des déplacements S et T respectivement quasi-diagonal par blocs et quasi-triangulaire par blocs : l'actualisation des générateurs des compléments de Schur successifs ne sera plus directe, et on les obtiendra de la formule (21). Si on procède directement, ceci produira des générateurs de plus en plus longs au cours du processus d'élimination, pourtant on sait que tous les compléments de Schur considérés sont à rang de déplacement uniformément borné. Ce phénomène sera encore plus marqué dans l'obtention d'algorithmes super-rapides.

Il y a deux approches pour la compression de générateurs, suivant que l'on soit dans un contexte numérique ou algébrique. Dans le premier cas, on préférera le calcul des générateurs orthogonaux *via* la DVS comme dans l'expression (17). Si par contre, on travaille en exacte ou on est sur un corps $\mathbb{F}$ outre que $\mathbb{R}$ ou $\mathbb{C}$, on fera la compression *via* l'algorithme d'élimination. On tire cette exposition de [**30**, §4.6].

Compression de générateurs :
Entrée : $G, B \in \mathbb{K}^{N\times\ell}$ **;**
Sortie : $H, C \in \mathbb{K}^{N\times\alpha}$ **tel que** $H\cdot C^T = G\cdot B^T$ **et** $\alpha = \operatorname{rang}(G\cdot B^T)$**.**

(1) On calcule la décomposition

$$B^T = P_B\cdot L_B\cdot U_B$$

avec $P_B \in \mathbb{F}^{\ell\times\ell}$ **permutation,** $L_B \in \mathbb{F}^{\ell\times\ell}$ **triangulaire inférieure avec 1s dans la diagonal, et** $U_B \in \mathbb{F}^{\ell\times N}$ **échélonnée supérieure.**

(2) On pose

$$\beta \leftarrow \operatorname{Card}\{\ \textbf{lignes}\ \neq 0\ \textbf{de}\ U_B\} = \operatorname{rang}(B)$$

et

$$F \leftarrow (G\cdot P_B\cdot L_B)(1:N, 1:\beta) \quad , \quad D \leftarrow U_B^T(1:N, 1:\beta) \quad \in \mathbb{F}^{N\times\beta}.$$

(3) On calcule la décomposition

$$F^T = P_F \cdot L_F \cdot U_F$$

avec $P_F \in \mathbb{F}^{\beta\times\beta}$ permutation, $L_F \in \mathbb{F}^{\beta\times\beta}$ triangulaire inférieure avec 1s dans la diagonal, et $U_F \in \mathbb{F}^{\beta\times N}$ echélonnée supérieure.

(4) On pose

$$\alpha \leftarrow \mathrm{Card}\{ \textbf{ lignes } \neq 0 \textbf{ de } U_F\} = \mathrm{rang}(F)$$

et

$$H \leftarrow U_F^T(1:N, 1:\alpha) \quad , \quad C \leftarrow (U_B \cdot P_F \cdot L_F)(1:N, 1:\alpha) \quad \in \mathbb{F}^{N\times\alpha}.$$

PROPOSITION 2.4. *L'algorithme ci-dessus calcule un système minimal de générateurs en $O(\ell^2 N)$ ops.*

DÉMONSTRATION. On montre d'abord la correctitude de l'algorithme. On vérifie

$$G \cdot B^T = (G \cdot P_B \cdot L_B) \cdot U_B = F \cdot D^T$$

et similairement

$$F \cdot D^T = U_F^T \cdot (D \cdot P_F \cdot L_F)^T = H \cdot C^T.$$

Par construction $C^T : \mathbb{F}^N \to \mathbb{F}^\alpha$ est un epimorphisme et $H : \mathbb{F}^\alpha \to \mathbb{F}^N$ est un monomorphisme, donc

$$\alpha = \mathrm{rang}(H \cdot C^T) = \mathrm{rang}(G \cdot B^T)$$

ce qui certifie la minimalité des générateurs H, C. La complexité de $O(\ell^2 N)$ ops est celle de l'algorithme d'élimination sur des matrices de taille $N \times \ell$. □

2.2. Extension de l'algorithme rapide. Pour longtemps, c'était un fait accepté que la mise en œuvre de la méthode de rang de déplacement était restreinte à S triangulaire inférieure (ou diagonal si besoin il y avait de pivotage) et T triangulaire supérieure.

Comme on l'a déjà vu, pour les déplacements quasi-diagonales le rang des compléments de Schur reste uniformément borné, et on peut actualiser les générateurs de ces compléments de Schur [**30**, § 4.6]. De plus, la quasi-diagonalité est (essentiellement) invariante par conjugaison par des permutations, donc on peut incorporer pivotage partiel ou total pour rendre l'algorithme numériquement stable.

Cependant, pour un souci de simplicité dans l'exposition, on se centrera dans le cas des déplacements triangulaires supérieure (ou carrément diagonal) et inférieure respectivement.

Dans l'article [**21**] on montre comment la méthode s'étends à des déplacement S et T Hessenberg inférieure et supérieure respectivement. On dit que une matrice S est dîte *Hessenberg inférieure* (resp. *Hessenberg supérieure*) si $S_{i,j} = 0$ pour $j \geq i+2$ (resp. si $S_{i,j} = 0$ pour $i \geq j+2$) c'est-à-dire si S est triangulaire sauf pour les diagonales secondaires. Malhereusement l'algorithme de Heinig et Olshevsky n'admets pas de stratégie de pivotage, et par conséquent est numériquement inestable.

L'algorithme de décomposition LU dans cette section admets des versions hermitiennes, au moins pour le cas de structure type Toeplitz, *voir* [**24**].

3. Algorithmes super-rapides

Le premier algorithme super-rapide fut proposé par Morf citemorf80 (*voir* aussi [**27**]) et Bitmead et Anderson [**2**] pour la résolution de systèmes type Toeplitz et Hankel, en $O(N \log^2(N))$ ops. Ceci résulte de la combinaison de l'algorithme de type "divide-and-conquer" de Strassen pour l'inversion de matrices, avec l'idée de rang de déplacement.

On démontre (théorème 3.4) que la complexité de l'inversion de matrices est dominée par celle de la multiplication de matrices. L'exposant de la multiplication de matrices est tellement elusif qu'il a mérité un nom : ω. Formellement, ω c'est le réel le plus petit tel que $O(N^{\omega+\varepsilon})$ ops suffissent, pour tout $\varepsilon > 0$.

L'algorithme standard pour multiplier matrices de taille $N \times N$ utilise $O(N^3)$ ops. Puisque la matrice a N^2 entrées on a $2 \leq \omega \leq 3$. En 1969 Strassen surprit tout le monde en exhibant un algorithme pour multiplier matrices avec $\omega = \log_2(7) = 2.81$ [**36**]. L'algorithme le plus rapide que l'on connaît est celui de D. Coppersmith et S. Winograd de 1987 qui montre que $\omega \leq 2.37$ [**8**], et la plupart de chercheurs pense que $\omega = 2$.

Récemment, C. Umans et H. Cohn ont dévisé une nouvelle approche basé en représentations de groupes, et en collaboration avec R. Kleiberg et B. Szegedy ont montré que cette approche permets de retrouver les meilleurs estimations connues pour ω [**6, 31**]. Cette méthode est promettante, et peut se voir comme une sorte d'analogue non commutative de la transformée de Fourier rapide.

Dans ce qui suit on considérera le cas d'une matrice fortément inversible. Le cas général se réduit probabilistiquement à ceci soit par symmétrization soit avec l'application d'un preconditionneur structuré [**30**, Ch. 5].

Soit $A \in \mathbb{K}^{N\times N}$ une matrice fortément inversible. Pour $1 \leq i \leq N$ considérons sa décomposition en blocs

$$A = \begin{matrix} \begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{bmatrix} & \begin{matrix} i \\ N-i \end{matrix} \\ \begin{matrix} i & N-i \end{matrix} & \end{matrix} ;$$

alors

$$A = \begin{bmatrix} \mathbf{1}_i & \\ A_{2,1}A_{1,1}^{-1} & \mathbf{1}_{N-i} \end{bmatrix} \cdot \begin{bmatrix} A_{1,1} & A_{1,2} \\ & S_i \end{bmatrix} \tag{32}$$

où

$$S_i = A_{2,2} - A_{2,1}A_{1,1}^{-1}A_{1,2}$$

est le i-ème complément de Schur.

LEMME 3.1. *Soit $A \in \mathbb{K}^{N\times N}$ une matrice fortément inversible, alors $A_{1,1}$ et S_i sont fortément inversibles aussi.*

DÉMONSTRATION. Pour $1 \leq k \leq i$ la factorisation 32 entraîne

$$\det(A^{(k)}) = \det(A_{1,1}^{(k)} \neq 0$$

donc $A_{1,1}$ est fortément inversible. Pour $i \leq k \leq N$

$$\det(A^{(k)}) = \det(A_{1,1}) \cdot \det(S_i^{(k-i)}) \neq 0$$

et donc S_i est fortément inversible. □

LEMME 3.2. *Soit*

$$A = \begin{matrix} \begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{bmatrix} & \begin{matrix} i \\ N-i \end{matrix} \\ \begin{matrix} i & N-i \end{matrix} & \end{matrix}$$

tel que A et $A_{1,1}$ soient inversibles, et soit $S_i \in \mathbb{K}^{(N-i)\times(N-i)}$ le i-ème complément de Schur, alors

$$A^{-1} = \begin{bmatrix} A_{1,1}^{-1} + A_{1,1}^{-1}A_{1,2}S_i^{-1}A_{2,1}A_{1,1}^{-1} & -A_{1,1}^{-1}A_{1,2}S_i^{-1} \\ -S_i^{-1}A_{2,1}A_{1,1}^{-1} & S_i^{-1} \end{bmatrix}.$$

DÉMONSTRATION. On inverse A *via* la factorisation 32, alors

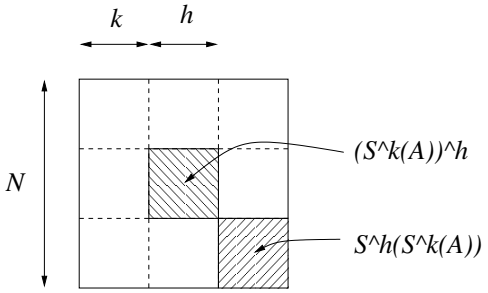
$$A^{-1} = \begin{bmatrix} A_{1,1} & A_{1,2} \\ & S_i \end{bmatrix}^{-1} \cdot \begin{bmatrix} \mathbf{1}_i & \\ A_{2,1}A_{1,1}^{-1} & \mathbf{1}_{N-i} \end{bmatrix}^{-1}$$
$$= \begin{bmatrix} A_{1,1}^{-1} & -A_{1,1}^{-1}A_{1,2}S_i^{-1} \\ & S_i^{-1} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{1}_i & \\ -A_{2,1}A_{1,1}^{-1} & \mathbf{1}_{N-i} \end{bmatrix};$$

on obtient l'expression pour A^{-1} en faisant la multiplication par blocs. □

LEMME 3.3. *Soit A fortément inversible, alors pour $h, k \geq 0$:*

$$\mathcal{S}^{(h)}(\mathcal{S}^{(k)}(A)) = \mathcal{S}^{(h+k)}(A) \quad , \quad (\mathcal{S}^{(k)}(A))^{(h)} = \mathcal{S}^{(k)}(A^{(h+k)}).$$

Ces matrices correspondent aux blocs indiqués graphiquement :



DÉMONSTRATION. Le complément de Schur $\mathcal{S}^{(h)}(\mathcal{S}^{(k)}(A))$ est la matrice obtenue après h pas de l'algorithme d'élimination sur $\mathcal{S}^{(k)}(A)$; soit celle qu'on obtient après $h + k$ pas de l'algorithme d'élimination sur A, donc elle coïncide avec $\mathcal{S}^{(h+k)}(A)$. En outre, on a

$$\begin{aligned}(\mathcal{S}^{(k)}(A))^{(h)} &= (A_{2,2} - A_{2,1}A_{1,1}^{-1}A_{1,2})^{(h)} \\ &= A_{2,2}^{(h)} - A_{2,1}^{(h)}A_{1,1}^{-1}A_{1,2}^{(h)} \\ &= \mathcal{S}^{(k)}(A^{(h+k)}).\end{aligned}$$

□

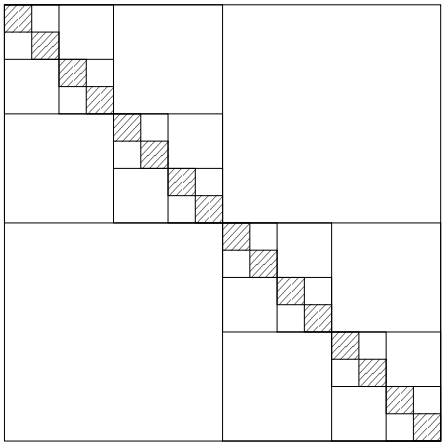
Grâce au lemme 3.2, on peut calculer l'inversion d'une matrice A fortément inversible à celle du bloc $A_{1,1}$ et du complément de Schur $\mathcal{S}^{(k)}(A)$, et ce procédé peut continuer recursivement jusqu'à arriver à des blocs 1×1. En fait, le calcul se fait au sens inverse, par un procédé de remontée, qui commence avec l'inversion de $A^{(1)} = [A_{1,1}]$ et de $A_{2,2} \in \mathbb{K}^\times$, ensuite le $1\times$-complément de Schur de $A^{(2)}$, finalement on inverse $A^{(2)} \in \mathbb{K}^{2\times 2}$ *via* les formules du lemme 3.2, etc.. Pour que ce schéma soit efficace, il faut que la partition soit *balancée*, c'est-à-dire $k = \lfloor N/2 \rfloor$.

Ceci conduit à un arbre binaire. La figure ci-dessous graphique en noir les 1×1 blocs qu'on inverse directement, et en blancs ceux qu'on inverse recursivement, et donc le calcul ne fait appel qu'à des multiplications de matrices :

Algorithme d'inversion de Strassen :
Entrée : $A \in \mathbb{K}^{N\times N}$;
Sortie : $A^{-1} \in \mathbb{K}^{N\times N}$;

(1) Construction d'une partition balancée : soit $k = \lfloor N/2 \rfloor$ et soit

$$A = \begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{bmatrix} \begin{matrix} k \\ N-k \end{matrix}$$
$$\begin{matrix} k & N-k \end{matrix}$$



la partition associée ;

(2) Calcul de

$$A_{1,1}^{-1}$$

directement si $k = 1$ **; sinon on le fait recursivement ;**

(3) Calcul du complément de Schur

$$S \leftarrow A_{2,2} - A_{2,1}A_{1,1}^{-1}A_{1,2};$$

(4) Calcul de

$$S^{-1}$$

directement si $N - k = 1$ **; sinon on le fait recursivement ;**

(5) Calcul de A^{-1} *via* **les formules du lemme 3.2.**

end.

THÉORÈME 3.4. *Soit $m(N)$ la complexité de multiplier deux matrices $N \times N$, et suppossons $m(N) = O(N^\omega)$ pour un $2 \leq \omega \leq 3$, alors l'algorithme d'inversion recursive calcule A^{-1} en $O(N^\beta)$ ops.*

DÉMONSTRATION. Notons $a(N) := \mathcal{C}_{Strassen}(N)$ la complexité de l'algorithme d'inversion recursive. Le calcul du complément de Schur S on calcule d'abord

$$\alpha := A_{2,1} \cdot A_{1,1}^{-1} \quad , quad\beta := \alpha \cdot A_{1,2} = A_{2,1}A_{1,1}^{-1}A_{1,2},$$

puis on obtient le complément de Schur avec une somme. Pour le calcul de A^{-1} *via* les formules du lemme 3.2 on calcule

$$\gamma := A_{1,1}^{-1} \cdot A_{1,2}, \quad \delta := S^{-1} \cdot \alpha, \quad \varepsilon := \gamma \cdot \delta, \quad \theta := \gamma \cdot S^{-1}.$$

Donc

$$a(N) \leq 2a(\lceil N/2 \rceil) + 6m(N) + O(N^2) = 2a(\lceil N/2 \rceil) + O(N^\omega).$$

Si l'on suppose par recurrence $a(\lceil N/2 \rceil) \leq c\lceil N/2 \rceil^\omega$ alors

$$a(N) \leq 2c\lceil N/2 \rceil^\beta \leq cN^\omega.$$

□

3.1. Inversion super-rapide de matrices structurées. Soit $A \in \mathbb{K}^{N\times N}$ une matrice fortement inversible et structurée. Pour calculer l'inverse en temps quasi-linéaire, on applique l'algorithme d'inversion recursive aux générateurs.

Algorithme d'inversion recursive de matrices structurées :

Entrée : $S, T \in \mathbb{K}^{N\times N}$ **triangulaires inférieure et supérieure respectivement telles que** $\mathrm{Spec}(S) \cap \mathrm{Spec}(T) = \emptyset$ **et** $G, B \in \mathbb{K}^{N\times\alpha}$ **générateurs d'une matrice** A **fortément inversible ;**

Sortie : $\widetilde{G}, \widetilde{B} \in \mathbb{K}^{N\times\alpha}$ (T, S)**-générateurs de** A^{-1}**.**

(1) Construction d'une partition balancée : soit $k = \lfloor N/2 \rfloor$ **et soit**

$$G = \begin{bmatrix} G_1 \\ G_2 \end{bmatrix}, \qquad B = \begin{bmatrix} B_1 \\ B_2 \end{bmatrix} \quad \begin{matrix} k \\ N-k \end{matrix}$$

la partition associée ;

(2) Recursivement on calcule

$$\widetilde{G}_1, \widetilde{B}_1$$

des générateurs de $\nabla_{T_{1,1},S_{1,1}}(A_{1,1}^{-1})$ **;**

(3) Calcul des $(S_{2,2}, T_{2,2})$**-générateurs de déplacement** $\nabla_{S_{2,2},T_{2,2}}(S)$ **du complément de Schur** *via* **les formules 23 ;**

(4) Recursivement on calcule des générateurs de $\nabla_{T_{2,2},S_{2,2}}(S^{-1})$ **;**

(5) Calcul de A^{-1} *via* **les formules du lemme 3.2.**

end.

Pour obtenir des algorithmes super-rapides, on est obligé de traiter les entrées et sorties en forme compacte en termes de générateurs : déjà l'écriture dense des matrices coûte N^2 ops.

Le coût de cet algorithme dépends essentiellement de la multiplication d'un vecteur avec une matrice structurée : par exemple, le calcul des générateurs pour le complément de Schur donne

$$\nabla_{S_{2,2},T_{2,2}}(A_{(k)}) = (G_2 - A_{2,1}A_{1,1}^{-1}G_1)(B_2 - B_1A_{1,1}^{-1}A_{2,1})$$

donc il faut multiplier chacune des α colonnes de G_1 par une matrice structurée.

PROPOSITION 3.5. *Le coût de l'algorithme d'inversion recursive de matrices structurées est de* $O(\alpha^2(N + m_{S,T}(N))\log(N))$.

3.2. Multiplication matrice-vecteur pour des matrices structurées. Le coût de la multiplication matrice-vecteurs pour les structures classiques de dimension N est de

$O(N\log(N))$ pour les matrices Toeplitz et Hankel et

$O(N\log^2(N))$ pour les matrices Vandermonde et Cauchy.

Ces estimations s'étendent aux matrices *type* Toeplitz, etc et leurs inverses, *via* les expressions bilinéaires des matrices en termes de générateurs.

Pour les matrices de Toeplitz, la multiplication matrice-vecteur est essentiellement une convolution, et donc se calcule en $O(N\log(N))$ ops grâce à la TFR. Faisons le cas $N = 3$: la multiplication

$$\begin{bmatrix} a_0 & a_{-1} & a_{-2} \\ a_1 & a_0 & a_{-1} \\ a_2 & a_1 & a_0 \end{bmatrix} \cdot \begin{bmatrix} v_0 \\ v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix}$$

se calcule comme les coefficients des termes de degré 0,1,2 dans le produit

$$(a_{-2}x^{-2} + a_{-1}x^{-1} + a_0 + a_1x + a_2x^2)(v_0 + v_1x + v_2x^2).$$

Notons que le codage consiste en une multiplication du mesagge par une matrice de Vandermonde, donc peut se faire en temps quasi-linéaire $O(N \log^2(N) \log\log(N)))$.

4. Application aux codes correcteurs d'erreurs

Références pour cette section : [**9, 38**]. Dans la page **(adresse)** on peut jouer avec une implémentation interactive de décodage des codes de Reed-Solomon, ce qui nous permettra une première approche concrète aux notions de codification, bruit et décodage.

Dans la pratique, la transfer de l'information se fait par des canaux "bruyants" introduisant des erreurs au cours la transmission. Exemples de cette situation :

- Transmission de données *via* satellite ;
- Stockage d'information sur un support (données numériques, musique, images) genre cassette ou CD, pour sa récuperation ultérieure ;
- Transmission d'information entre des ordinateurs (Internet).

La solution est de *coder* le message envoyé, a fin de pouvoir détecter et corriger les erreurs produits au cours de la transmission. On étudiera des codes où le *message* ou *mot* à envoyer est une suite de longueur k de symboles d'un alphabet fixé L. Ces messages sont codés avant la transmission, le *message (ou mot) codé* sera une suite de longueur n de symboles du même alphabet L. On a besoin d'une certaine redondance pour que le message originel soit récupérable, donc on suppose $n \geq k$.

En théorie algébrique de codes, l'alphabet de transmission forme un corps fini

$$L = \mathbb{F}_q \qquad (q = p^r).$$

Vue la construction des ordinateurs, il est naturel de considérer un alphabet $\mathbb{F}_2 = \{0, 1\}$ ou encore $\mathbb{F}_{2^r} = \{0, 1\}^r$, toutefois les constructions qu'on fera seront valables pour un alphabet $\mathbb{F}_q$ pour $q = p^r$ quelconque. La *codification* est une fonction injective

$$E : \mathbb{F}_q^k \hookrightarrow \mathbb{F}_q^n.$$

On se restreindra au cas des codes *linéaires* dont la codification E est une fonction linéaire. On notera $[E] \in \mathbb{F}^{n\times k}$ la *matrice génératrice* de la codification. Souvent une codification est de la forme $[E] = [\mathbf{1}_k|P]$ où P est une matrice de taille $k \times (n-k)$. Soit $Ex = y$, dans ce cas pour $1 \leq i \leq k$ les coordonnées y_i sont les *positions d'information*, et pour $k+1 \leq i \leq n$ les y_i sont les *parity checks*. Ceci se révèle utile lorsqu'il n'y a pas eu d'erreurs dans la transmission, car alors on peut récupérer le message à partir des positions d'information, sans avoir à le décoder.

L'ensemble de mots codés

$$\mathcal{C} := E(\mathbb{F}_q^k) \subset \mathbb{F}_q^n$$

est par définition le *code*, c'est un sous-espace vectoriel de $\mathbb{F}_q$ de dimension k. Le *décodage* est une rétraction de E, c'est-à-dire une fonction $D : \mathbb{F}_q^n \to \mathbb{F}_q^k$ telle que $D \circ E = \mathbf{1}_{\mathbb{F}_q^k}$.

La *distance de Hamming* entre deux mots $v, w \in \mathbb{F}_q^n$ est

$$\mathrm{dist}(v, w) := \mathrm{Card}\{i \ : \ v_i \neq w_i\}.$$

Pour $v \in \mathbb{F}_q^n$ on désigne $B_\rho(v)$ la boule de rayon ρ (par rapport à la distance de Hammings) et centrée en v :

$$B_\rho(v) := \{w \in \mathbb{F}_q^n \ : \ \mathrm{dist}(v, w) \leq \rho\}.$$

Autrement-dit, c'est l'ensemble des w différant de v en au plus ρ coordonnées. La *distance minimale* de $\mathcal{C}$ est

$$\begin{aligned}\operatorname{dist}(\mathcal{C}) &:= \min\{\operatorname{dist}(v,w) \ : \ v,w \in \mathcal{C}, v \neq w\} \\ &= \min\Big\{ \operatorname{dist}(v,0) \ : \ v \in \mathcal{C} \setminus \{0\}\Big\} \\ &= \min\Big\{\operatorname{Card}\{i \ : \ v_i \neq 0\} : v \in \mathcal{C} \setminus \{0\}\Big\},\end{aligned}$$

la deuxième égalité étant une conséquence de la linéarité. Un code $\mathcal{C}$ sur un alphabet $\mathbb{F}_q$, de *dimension* (longueur du message) k, *longueur* (du mot codée) n et distance minimale d, est nommé un $[n,k,d]_q$*-code*.

Soit $y \in \mathbb{F}_q^n$ tel que $\operatorname{dist}(y,v) \leq d-1$ pour un certain $v \in \mathcal{C}$, alors $y \in \mathcal{C}$ si et seulement si $y = v$. De plus, si $d = 2\,t+1$ et $\operatorname{dist}(y,v) \leq t$, alors v est l'unique élément $w \in \mathcal{C}$ tel que $\operatorname{dist}(y,w) \leq t$. On en déduit la proposition suivante :

Proposition 4.1. *Soit $\mathcal{C}$ un code à distance minimale d, alors $\mathcal{C}$ peut détecter jusqu'à $d-1$ erreurs. Si $d \geq 2\,t+1$, alors $\mathcal{C}$ peut corriger jusqu'à t erreurs.*

Comme exemple, considérons les *codes de répétition* :

— Deux répétitions :

$$(a,b,c) \mapsto (a,b,c,a,b,c).$$

On vérifie aisément que c'est un $[6,3,2]_q$-code ; donc il détecte jusqu'à un erreur, mais il ne peut pas le corriger.

— Trois répétitions :

$$(a,b,c) \mapsto (a,b,c,a,b,c,a,b,c)$$

. C'est un $[(9,3,3]_q$-code ; donc il détecte 2 erreurs et il corrige 1.

Faisons l'analyse des codes de répétition. Disons qu'on veut passer un message de longueur k sur un alphabet $\mathbb{F}_q$; la solution bêtement proposé par ces codes est de le répéter ℓ fois, c'est-à-dire

$$(a_1,\dots,a_k) \mapsto (\underbrace{a_1,\dots,a_k,a_1,\dots,a_k,a_1,\dots,a_k}_{k\,\ell}).$$

C'est un $[k\,\ell,k,\ell]_q$-code, et donc il peut détecter jusqu'à ℓ erreurs et corriger $\lfloor(\ell-1)/2\rfloor$. Le quotient

$$\frac{d}{n} = \frac{\ell}{k\,\ell} = \frac{1}{k}$$

est constant par rapport au nombre ℓ des répétitions, et beaucoup trop petit pour être utile en pratique. Les "bons" codes sont ceux pour lesquels le *quotient d'information* k/n n'est pas trop petit, et qu'en même temps d est grand.

Notons finalement que pour un code linéaire, la codification se réduit à la multiplication matrice-vecteur, donc prends au plus

$$O(k\,n) \text{ de } \mathbb{F}_q.$$

Voici quelques exercices sur la structure des corps finis, pour l'étudiant non habitué à ces objets. La notation $\mathbb{F}_p$ désigne le corps $\mathbb{Z}/p\mathbb{Z}$ (p premier). On admettra que tout corps (commutatif) possède une clôture algébrique, et que deux clôtures algébriques d'un même corps sont isomorphes.

EXERCICE 5.2. ◁ Soit k un corps quelconque et $g \in k[X]$ un polynôme *irréductible*, c'est à dire un polynôme non constant qui n'est pas le produit de deux autres polynômes non constants. Montrer que l'idéal $(g) \subset k[X]$ engendré par g est maximal. En déduire que $k[X]/(g)$ est un corps.

Indication : utiliser le théorème de Bézout qui énonce que si $g, h \in k[X]$ sont premiers entre eux, alors ils existent $u, v \in k[X]$ tels que

$$gu + hv = 1.$$

▷

EXERCICE 5.3. ◁

(i) Montrer que $g = x^4 + x + 1 \in \mathbb{F}_2[x]$ est irréductible.

(ii) Combien d'éléments y a-t-il dans $\mathbb{F} := \mathbb{F}_2[x]/(g)$?

(iii) Notons $\overline{x}$ la classe de x dans le corps quotient $\mathbb{F}$. Calculer les différentes puissances de $\overline{x}$. Vérifier que $1, \overline{x}, \overline{x}^2, \overline{x}^3$ est une base de $\mathbb{F}$ comme espace vectoriel sur $\mathbb{F}_2$.

(iv) Montrer que $\{0, 1, \overline{x}^5, \overline{x}^{10}\}$ est un sous-corps de $\mathbb{F}$ à quatre éléments.

(v) Y a-t-il un sous-corps de $\mathbb{F}$ à huit éléments ? Y a-t-il d'autres sous-corps ?

▷

EXERCICE 5.4. ◁ Soit $\mathbb{F}$ un corps fini. Montrer qu'il existe un nombre premier p et un entier n tel que $\mathrm{Card}(\mathbb{F}) = p^n$. Indication : montrer que le sous corps engendré par 1 est un $\mathbb{F}_p$ et que $\mathbb{F}$ est un espace vectoriel de dimension finie sur ce corps. ▷

EXERCICE 5.5. ◁ Notons $\overline{\mathbb{F}_p}$ la clôture algébrique de $\mathbb{F}_p$, et soit n un entier quelconque. On pose $q = p^n$, et on considère l'ensemble

$$Z_q = \{x \in \overline{\mathbb{F}_p} : x^q = x\}.$$

Montrer que $\mathrm{Card}(Z_q) = q$, et que Z_q est un corps. Indication : montrer que $f(X) = X^q - X$ est un polynôme sans racines multiples. ▷

EXERCICE 5.6. ◁ Soit $\mathbb{F}$ un corps fini contenu dans $\overline{\mathbb{F}_p}$, de cardinal $q = p^n$. Montrer que tout $x \in \mathbb{F}^\times$ vérifie $x^{q-1} = 1$. En conclure que $\mathbb{F} = Z_q$. ▷

EXERCICE 5.7. ◁ Soit $\mathbb{F}_{p^n}$ le seul sous-corps de $\overline{\mathbb{F}_p}$ à p^n éléments. Montrer que

$$\mathbb{F}_p = \bigcup_{n \geq 1} \mathbb{F}_{p^n}.$$

Indication : utilizer que pour $\xi \in \overline{\mathbb{F}_p}$, le corps engendré $\mathbb{F}_p(\xi)$ est fini. ▷

EXERCICE 5.8. ◁ Dans cet exercice on donne plusieurs bornes pour les paramètres associés aux codes. Une façon de produire des bons codes est de fixer une longueur n et une distance minimale d, puis d'essayer de maximiser k en prenant les mots du code une par une, tout en gardant $\mathrm{dist}(v, w) \geq d$.

(1) Montrer que pour tout $c \in \mathbb{F}_q^n$

$$b(n, d) := \mathrm{Card}(B_{d-1}(c)) = \sum_{i=0}^{d-1} \binom{n}{i} (q-1)^i.$$

(2) Soit $d \geq 1$ et $\mathcal{C} \subset \mathbb{F}_q^n$ un sous-ensemble (pas forcément un sous-espace linéaire) tel que $\mathrm{dist}(v, w) \geq d$ pour tout $v \neq w$ dans avec $v, w \in \mathcal{C}$. Supposons que pour tout $z \in \mathbb{F}_q^n \setminus \mathcal{C}$ on a $\mathrm{dist}(z, c) \leq d - 1$ pour un certain $c \in \mathcal{C}$. Montrer que

$$b(n, d) \cdot \mathrm{Card}(\mathcal{C}) \geq q^n.$$

Ceci est la *estimation de Gilbert-Varshamov*. Indication : De façon équivalente, montrer que si $b(n,d) \cdot \mathrm{Card}(\mathcal{C}) < q^n$ alors il existe z tel que la distance minimale de $\mathcal{C} \cup \{z\}$ est encore $\geq d$.

(3) Montrer que si
$$b(n,d) < q^{n-k+1}$$
pour un certain k, alors il existe un $[n,k,d]_q$-code linéaire. Indication : par récurrence, on peut supposer qu'il existe un $[n,k-1,d]_q$-code linéaire $\mathcal{C}$. En appliquant la partie (2), considérez le code linéaire $\mathcal{C}'$ engendré par $\mathcal{C}$ et z, où la distance de z à n'importe quelle mot de $\mathcal{C}$ est $\geq d$, et montrer que $\mathcal{C}'$ a encore distance minimale d.

(4) Dans l'autre direction, montrer que pour tout code linéaire on a
$$d \leq n - k + 1.$$
Ceci est la *Singleton bound*. Indication : considérez ce qu'il se passe quand un sous-ensemble de $d-1$ coordonnées est effacé de chacune des mots du code.

▷

4.1. Codes de Reed-Solomon. Pour $q = p^r$ et $k \leq n \leq q-1$ considérons une suite de points deux à deux distincts
$$x_1, \ldots, x_n \in \mathbb{F}_q^\times.$$
On identifie $\mathbb{F}_q^k$ avec $\mathbb{F}_q[x]_{\leq k-1}$, l'espace vectoriel des polynômes sur $\mathbb{F}_q$ de degré au plus $k-1$ et considérons l'application linéaire injective
$$E : \mathbb{F}_q[x]_{\leq k-1} \to \mathbb{F}_q^n \quad , \quad f \mapsto (f(x_1), \ldots, f(x_n)).$$
On pose
$$\mathrm{RS}(k, q; x_1, \ldots, x_n) := E(\mathbb{F}_q[x]_{\leq k-1})$$
le *code de Reed-Solomon* sur $\mathbb{F}_q$ de dimension k et longueur n associé aux points $x_1, \ldots, x_n$. Un polynôme non nul f de degré au plus $k-1$ ne peut pas avoir plus de $k-1$ zéros, et donc
$$\mathrm{dist}(v, 0) \geq n - k + 1 \quad \text{pour toute mot } v \in \mathcal{C} \setminus \{0\}.$$
Comme on peut construire $f \in \mathbb{F}_q[x]_{\leq k-1}$ avec $k-1$ zéros parmi les x_is, on a
$$\mathrm{dist}(\mathrm{RS}(k, q; x_1, \ldots, x_n)) = n - k + 1.$$
La *Singleton bound* (exercice 5.8(d)) montre que les codes de Reed-Solomon ont la distance minimale d la plus grande possible, parmi tout les codes de longueur n et dimension k. Les codes avec cette propriété s'appellent *codes séparables à distance maximale* (en anglais : *maximum distance separable codes*) dans la littérature.

Exercice 5.9. ◁

(1) Montrer que
$$g = x^3 + x + 1$$
est irréductible sur $\mathbb{F}_2$. En déduire que
$$\mathbb{F}_8 = \mathbb{F}_2[x]/g,$$
et que $1, \overline{x}, \overline{x}^2$ est une base de $\mathbb{F}_8$ comme $\mathbb{F}_2$-espace linéaire.

(2) Énumérer le code de Reed-Solomon sur $\mathbb{F}_8$ de longueur 4 et dimension 2, associé aux éléments
$$x_1 := 1, \quad x_2 := 1 + \overline{x}, \quad x_3 := 1 + \overline{x} + \overline{x}^2, \quad x_4 := 1 + \overline{x}^2.$$

▷

EXERCICE 5.10. ◁ Soit $\mathcal{C}$ une code de Reed-Solomon de longueur n et dimension k, sur un alphabet $\mathbb{F}_{2^r}$. Ainsi chaque mot codée peut se représenter comme une suite de $r\,n$ bits, car chaque symbole de $\mathbb{F}_{2^r}$ est représenté par r bits.

Montrer qu'une suite de $r\,\ell$ erreurs consécutifs à niveau bit, change au plus $\ell+1$ des symboles d'un mot codée, à niveau $\mathbb{F}_{2^r}$. En déduire que si

$$\ell + 1 \leq \lfloor (n-k)/2 \rfloor,$$

le code $\mathcal{C}$ peut corriger une suite de $r\,\ell$ erreurs consécutifs. ▷

4.2. Décodage rapide de codes de Reed-Solomon. Référence pour cette section : [**28**]. Le décodage des codes de Reed–Solomon se fait *via* l'algorithme de Berlekamp et Massey. Considérons un alphabet $\mathbb{F}_q$ ($q = p^r$) et soit m un message de longueur k qu'on idéntifie à un polynôme de degré borné par $k-1$:

$$f = m_1 + m_2\, x + \cdots + m_k\, x^{k-1} \in \mathbb{F}_q[x]_{k-1}.$$

Soient $x_1, \ldots, x_n \in \mathbb{F}_q^\times$ des points distincts et soit

$$E(f) := (f(x_1), f(x_2), \ldots, f(x_n)) \in \mathbb{F}_q^n$$

et notons $t(E(m)) = (t_1, \ldots, t_n)$ le message reçu. Notre but est de récuperer f comme l'unique élément de $\mathbb{F}_q[x]_{k-1}$ tel que

$$\operatorname{dist}(t, E(f)) \leq \frac{d-1}{2} = \frac{n-k}{2}.$$

Autrement-dit, f est l'unique polynôme dans $\mathbb{F}_q[x]_{k-1}$ tel que $f(x_i) = t_i$ pour au moins $n - \dfrac{n-k}{2} = \dfrac{n+k}{2}$ des x_is.

PROPOSITION 4.2. *Soit $k \equiv n \pmod 2$ et soient g, h des polynômes tels que $h \neq 0$,*

$$\deg(g) < \frac{n+k}{2} \quad , \qquad \deg(h) \leq \frac{n-k}{2},$$

et tels que $g(x_i) + t_i \cdot h(x_i) = 0$ pour $i = 1, \ldots, n$, alors $f = -g/h$.

Notons que g, h existent puisqu'il s'agit d'un système de n équations linéaires en $n+1$ variables.

DÉMONSTRATION. Posons $H := g + f \cdot h$, alors

$$\deg(H) \leq \max(\deg(g), \deg(f \cdot h)) \leq \max\left(\frac{n+k}{2} - 1, k - 1 + \frac{n-k}{2}\right) \leq \frac{n+k}{2} - 1.$$

En outre

$$\operatorname{Card}\{\xi \in \mathbb{F}_q \,:\, H(\xi) = 0\} \geq \operatorname{Card}\{x_i \,:\, f(x_i) = t_i\} \geq \frac{n+k}{2}$$

et donc $H \equiv 0$, c'est-à-dire $f = -g/h$. □

Le système linéaire associé est

$$\mathrm{BM} \cdot \begin{bmatrix} g^T \\ h^T \end{bmatrix} = \begin{bmatrix} 1 & x_1 & \cdots & x_1^{(n+k)/2-1} & y_1 & y_1\,x_1 & \cdots & y_1\,x_1^{(n-k)/2} \\ \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\ 1 & x_n & \cdots & x_n^{(n+k)/2-1} & y_n & y_n\,x_n & \cdots & y_n\,x_n^{(n-k)/2} \end{bmatrix} \cdot \begin{bmatrix} g_0 \\ \vdots \\ h_{(n-k)/2} \end{bmatrix}.$$

La matrice $\mathrm{BM} \in \mathbb{F}_q^{n \times (n+1)}$ n'est pas une matrice de Vandermonde mais presque ! Posons

$$S := \operatorname{diag}(x_1^{-1}, \ldots, x_n^{-1}) \in \mathbb{F}_q^{n \times n} \quad , \quad T := Z_0 \in \mathbb{F}_q^{(n+1)\times(n+1)}$$

on vérifie

$$\nabla_{S,T}(\mathrm{BM}) = \begin{bmatrix} x_1^{-1} & 0 & \cdots & 0 & (y_1\, x_1^{-1} - x_1^{(n+k)/2-1}) & 0 & \cdots & 0 \\ \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\ x_n^{-1} & 0 & \cdots & 0 & (y_n\, x_n^{-1} - x_n^{(n+k)/2-1}) & 0 & \cdots & 0 \end{bmatrix} \tag{33}$$

qui est une matrice de rang 2. Le calcul du couple (g, h) revient à trouver un élément non nul dans le noyau de la matrice BM, ce qu'on fait en calculant la décomposition $\mathrm{BM} = P \cdot L \cdot U$ car $\ker(\mathrm{BM}) = \ker(U)$. La résolution consiste en trois étapes :

(1) décomposition $\mathrm{BM} = P^{(n\times n)}\, L^{(n\times n)}\, U^{(n\times(n+1))}$;
(2) détermination de $(g, h) \in \ker(U) \setminus \{0\}$;
(3) division de polynômes $f = -g/h$.

La partie la plus coûteuse est le calcul de la décomposition PLU. Pour cela, on tirera profit du fait que BM est structurée : avec un algorithme rapide, cette décomposition se fait en

$$O(\mathrm{rang}_{S,T}(\mathrm{BM})\, n^2) = O(n^2) \quad \text{ops de } \mathbb{F}_q.$$

Le calcul du noyau de $U \in \mathbb{F}_q^{n\times(n+1)}$ se fait par *backward substitution* en n^2 ops, puisqu'il s'agit d'un système triangulaire. Finalement, on a $\deg(g), \deg(h) = O(n)$ donc la division g/h se fait en $O(n^2)$ ops par l'algorithme standard, ou en $O(n \log(n))$ ops par l'analogue polynomial de l'algorithme de Schonhäge-Strassen. Le coût total reste en

$$O(n^2) \quad \text{ops de } \mathbb{F}_q.$$

On a implémenté la méthode sur `Maple9` pour la tester sur un exemple petit, *voir* le fichier `ReedSolomon.mw` joint. L'exemple considéré (un message de longueur 4 et dimension 2 sur l'alphabet $\mathbb{F}_8$) est instructif et on le décrira avec un certain détail.

Le polynôme $x^3 + x + 1 \in \mathbb{F}_2[x]$ est irréductible donc

$$\mathbb{F}_8 = \mathbb{F}_2[x]/(x^3 + x + 1).$$

Posons $a := \overline{x}$; tout élément $b \in \mathbb{F}_8$ s'écrit alors comme

$$b = b_0 + b_1 a + b_2 a^2 \qquad \text{avec } b_i = 0, 1.$$

Le message à envoyer est $m = (1, 1)$, et on le représente comme le polynôme $f = 1 + x \in \mathbb{F}_8[x]$. Prenons quatre éléments témoins dans $\mathbb{F}_8^\times$:

$$x_1 = 1, \quad x_2 = 1 + a, \quad x_3 = 1 + a + a^2, \quad x_4 = 1 + a^2,$$

et disons que le mot reçu est

$$y_1 = 0, \quad y_2 = a, \quad y_3 = a, \quad y_4 = a^2.$$

Il y a un seul erreur a niveau $\mathbb{F}_8$: $y_3 \neq f(x_3)$, et notre code de Reed-Solomon peut le corriger car $(n - k)/2 = 1$. La matrice de Berlekamp-Massey associée à ces x_is et y_js est

$$\begin{aligned} \mathrm{BM} &= \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 1+a & (1+a)^2 & a & a(1+a) \\ 1 & 1+a+a^2 & (1+a+a^2)^2 & a & a(1+a+a^2) \\ 1 & 1+a^2 & (1+a^2)^2 & a^2 & a^2(1+a^2) \end{bmatrix} \\ &= \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 1+a & 1+a^2 & a & a+a^2 \\ 1 & 1+a+a^2 & 1+a & a & 1+a^2 \\ 1 & 1+a^2 & 1+a+a^2 & a^2 & a \end{bmatrix}. \end{aligned}$$

On calculera se décomposition LU *via* l'algorithme rapide. pour vérification ultérieure, le résultat est

$$L = \begin{bmatrix} 1 & & & \\ 1 & 1 & & \\ 1 & 1+a & 1 & \\ 1 & a & 1 & 1 \end{bmatrix}, \quad U = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ & a & a^2 & a & a+a^2 \\ & & 1+a^2 & a^2 & a^2 \\ & & & a^2 & 1 \end{bmatrix}.$$

Avec ceci on calcule $\ker(\mathrm{BM}) = \ker(U)$; ce noyau est engendré par le vecteur

$$(v_1, \ldots, v_5) = (1 + a + a^2, a^2 + a, 1, 1 + a + a^2, 1) \in \mathbb{F}_8^5.$$

On construit les polynômes de Berlekamp-Massey associés

$$g = v_1 + v_2 x + v_3 x^2 = 1 + a + a^2 + (a^2 + a)x + x^2, \quad h = v_4 + v_5 x = 1 + a + a^2 + x;$$

le message reconstruit est donc $f = -g/h = 1 + x$.

C'est très bien, mais ce qui nous intéresse c'est de voir marcher l'algorithme rapide ! Les matrices de déplacement sont

$$S = \mathrm{diag}(x_1^{-1}, \ldots, x_4^{-1}) = \begin{bmatrix} 1 & & & \\ & a+a^2 & & \\ & & a^2 & \\ & & & a \end{bmatrix}$$

et $T = Z_0$ un bloc 5×5 de Jordan. On calcule des générateurs pour le déplacement de BM à l'aide de la formule (33), sans avoir à expliciter BM :

$$G(1) = \begin{bmatrix} 1 & 1 \\ a+a^2 & a \\ a^2 & 0 \\ a & a^2 \end{bmatrix}, \quad B(1) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}.$$

On construit progressivement la matrice $U = [U_{i,j}]_{1 \le i \le 4, 1 \le j \le 5}$. On devra calculer aussi L, puisqu'on en a besoin pour l'actualisation des générateurs du déplacement des compléments de Schur successifs. Suivant l'algorithme rapide, à chaque étape on construit une ligne de U, puis on actualise les générateurs. La première ligne de U coïncide avec celle de BM :

$$U_{1,1} = 1, \quad U_{1,2} = 1, \quad U_{1,3} = 1, \quad U_{1,4} = 0, \quad U_{1,5} = 0.$$

Puis on calcule des générateurs pour BM(2), le déplacement du premier complément de Schur, *via* les formules (23) :

$$G(2) = \begin{bmatrix} 1+a+a^2 & 1+a \\ 1+a^2 & 1 \\ 1+a & 1+a^2 \end{bmatrix}, \quad B(2) = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

Passons à la deuxième itération de la boucle. Maintenant on veut calculer la deuxième ligne de U. Pour cela il faut reconstruire la première ligne de BM(2), donc on calcule la première ligne de son déplacement :

$$\nabla(BM(2)) = \begin{bmatrix} 1+a+a^2 & 1+a+a^2 & 1+a & 0 \\ * & * & * & * \\ * & * & * & * \end{bmatrix}$$

puis on reconstruit la première ligne de BM(2) à l'aide des formules du lemme (2.2) :

$$\mathrm{BM}(2) = \begin{bmatrix} a & a^2 & a & a+a^2 \\ * & * & * & * \\ * & * & * & * \end{bmatrix}.$$

Ainsi la deuxième ligne de U est

$$U_{2,1} = 0, \quad U_{2,2} = a, \quad U_{2,3} = a^2, \quad U_{2,4} = a, \quad U_{2,5} = a + a^2.$$

L'actualisation des générateurs donne

$$G(3) = \begin{bmatrix} 1 + a + a^2 & a^2 \\ a + a^2 & 1 + a \end{bmatrix}, \quad B(3) = \begin{bmatrix} 1 + a & 1 & 1 + a \\ 0 & 1 & 0 \end{bmatrix}.$$

Et c'est reparti ! La première ligne du déplacement de BM(3) est

$$\nabla(BM(3)) = \begin{bmatrix} a & 1 + a & a \\ * & * & * \end{bmatrix}$$

et donc $\mathrm{BM}(3) = \begin{bmatrix} 1 + a^2 & a^2 & a^2 \\ * & * & * \end{bmatrix}$, alors la troisième ligne de U est

$$U_{3,1} = 0, \quad U_{3,2} = 0, \quad U_{3,3} = 1 + a^2, \quad U_{3,4} = a^2, \quad U_{3,5} = a^2.$$

L'actualisation des générateurs donne

$$G(4) = \begin{bmatrix} 1 & 1 + a + a^2 \end{bmatrix}, \quad B(4) = \begin{bmatrix} a^2 & a + a^2 \\ 1 & 0 \end{bmatrix}.$$

Pour finir, on calcule encore la première ligne du déplacement de BM(4)

$$\nabla(BM(4)) = \begin{bmatrix} 1 + a & a + a^2 \end{bmatrix}$$

puis $\mathrm{BM}(4) = \begin{bmatrix} a^2 & 1 \end{bmatrix}$, la dernière ligne de U est donc

$$U_{4,1} = 0, \quad U_{4,2} = 0, \quad U_{4,3} = 0, \quad U_{4,4} = a^2, \quad U_{3,5} = 1;$$

et avec ceci on a fini le calcul.

4.3. Décodage super-rapide.

5. À faire ou à mettre à point

- Approximants de Padé comme exemple d'apparition naturelle de matrices structurées dans la pratique [**30**, § 2.11].
- Formule de Gohberg-Semencul pour l'inversion d'une matrice Toeplitz [**17**]. Le premier algorithme super-rapide pour la résolution d'un système Toeplitz fut obtenu par réduction à approximants de Padé *via* la formule de GS.
- Stabilité de l'algorithme rapide, à ce respect *voir* [**29**].
- Estimer la complexité *en exacte* des algorithmes rapide et super-rapide.
- Comme direction de recherche : étendre ces algorithmes à des matrices structurées autres que les classes classiques. Notamment, pour les matrices Toeplitz bloc Toeplitz ou de Toeplitz bloc Toeplitz bloc Toeplitz, qui apparaissent de façon naturelle dans la discrétisation des EDP elliptiques à coefficients constants sur une grille uniforme (thèse de Houssan).

6. Exercices

EXERCICE 5.11. ◁ Soient

$$d_0, \ldots, d_\ell; n \in \mathbb{N} \quad , \quad d = d_0 + \cdots + d_\ell \quad , \quad x_1, \ldots, x_n \in \mathbb{K}^\times$$

et posons

$$A := \begin{bmatrix} 1 & x_1 & \cdots & x_1^{d_0-1} & y_1 & y_1x_1 & \cdots & y_1x_1^{d_1-1} & \cdots & y_1^\ell & y_1^\ell x_1 & \cdots & y_1^\ell x_1^{d_\ell-1} \\ \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots & & \vdots & \vdots & & \vdots \\ 1 & x_n & \cdots & x_n^{d_0-1} & y_n & y_nx_n & \cdots & y_nx_n^{d_1-1} & \cdots & y_n^\ell & y_n^\ell x_n & \cdots & y_n^\ell x_n^{d_\ell-1} \end{bmatrix} \in \mathbb{K}^{n \times d}.$$

Trouvez des déplacements $S \in \mathbb{K}^{n \times n}$ et $T \in \mathbb{K}^{d \times d}$ à spectre disjoint, tels que

$$\nabla_{S,T}(A) = S \cdot A - A \cdot T$$

soit de rang au plus $\ell + 1$, et déterminez des générateurs pour $\nabla_{S,T}(A)$. ▷

6.1. Matrices de Toeplitz infinies et semi-infinies. Soit $\tau := (t_k)_{k\in\mathbb{Z}}$ une sucession de nombres complexes et posons

$$T_t := [t_{i-j}]_{i,j\in\mathbb{Z}} = \begin{bmatrix} \ddots & \ddots & & \\ \ddots & t_0 & t_{-1} & \\ & t_1 & t_0 & \ddots \\ & & \ddots & \ddots \end{bmatrix}$$

la matrice de Toeplitz bi-infinie et

$$\tau(z) := \sum_{k\in\mathbb{Z}} t_k z^k$$

le *symbole* associés. On suppossera $\tau(z)$ convergente dans un petit anneau autour du circle unité.

EXERCICE 5.12. ◁ Soient $\sigma := (s_k)_{k\in\mathbb{Z}}$ et $\tau := (t_k)_{k\in\mathbb{Z}}$, montrer que

(1) $T_\sigma + T_\tau = T_{\sigma+\tau}$;
(2) $\lambda \cdot T_\tau = T_{\lambda\cdot\tau}$;
(3) $T_\sigma \cdot T_\tau = T_{\sigma\cdot\tau}$;
(4) si $\tau(z) \neq 0$ pour tout $z \in S^1$, alors $T_\tau^{-1} = T_{\tau^{-1}}$.

▷

EXERCICE 5.13. ◁ Montrer qu'on peut factoriser le symbole

$$\tau(z) = \ell(z) \cdot u(z)$$

avec

$$\ell(z) = 1 + \sum_{k\geq 1} \ell_k z^k \quad , \quad u(z) = \sum_{k\leq 0} u_k z^k$$

respectivement analytique dans le disque unité et analytique en dehors du disque unité. En déduire

$$T_\tau = T_\ell \cdot T_u,$$

en particulier la décomposition LU d'une Toeplitz bi-infinie est Toeplitz. ▷

EXERCICE 5.14. ◁ Montrer que T_τ n'a pas des valeurs propres, et que

$$\mathrm{Spec}(T_\tau) = \{\tau(z) : z \in S^1\}.$$

▷

EXERCICE 5.15. ◁ Montrer que $||T_\tau||_2 = |\tau(z)|_\infty$. ▷

EXERCICE 5.16. ◁ Théorème de Szegö-Grenander : suppossont T_τ normale, c'est-à-dire $T^* \cdot T = T \cdot T^*$. Soit

$$T_N := [\tau_{i-j}]_{1\leq i,j\leq N} \in \mathbb{C}^{N\times N}$$

et considérons la mésure discrète

$$\mu_N(z) = \frac{1}{N} \sum_{\lambda\in\mathrm{Spec}(T_N)} \delta(z - \lambda).$$

Montrer que μ_N converge faiblement vers la mésure supportée et equidistribuée sur $\mathrm{Spec}(T_\tau)$. ▷

Le but des exercices suivants est d'étudier les matrices de Toeplitz semi-infinies ; en particulier on établira le celebre théorème de l'indice de Gohberg et . . ., antécedant direct du théorème de l'indice de Atiyah et Singer.

EXERCICE 5.17. ◁On identifie le cercle unité à $\mathbb{R}/2\pi\mathbb{Z}$. Notons $C^0(S^1)$ l'ensemble des fonctions complexes continues, périodiques de période 2π. On suppose $a \in C^0(S^1)$ et on définit un opérateur de multiplication M_a dans $L^2(S^1)$ par

$$M_a u = au.$$

On rappelle que le spectre d'un opérateur A d'un espace de Hilbert dans lui même est l'ensemble des $z \in \mathbb{C}$ tels que $z - A$ possède un inverse partout défini et continu. Montrer que le spectre de M_a est exactement l'image de a. ▷

EXERCICE 5.18. ◁ Soit $b \in C^0(S^1)$; calculer $M_a M_b$. ▷

EXERCICE 5.19. ◁ On note $\hat{a}_j$ le j-ième coefficient de Fourier de a :

$$\widehat{a}_j = \int_0^{2\pi} a(\theta) e^{-\mathrm{i}j\theta} \, \mathrm{d}\theta.$$

Montrer que dans la base de Fourier formée des $e_k : \theta \mapsto \exp(\mathrm{i}k\theta)$, pour $k \in \mathbb{Z}$, l'opérateur M_a se met sous forme de matrice infinie, qu'on donnera explicitement en fonction des $\hat{a}_j$. ▷

EXERCICE 5.20. ◁ On appelle $H^2(S^1)$ l'ensemble des fonctions dans $L^2(S^1)$ dont tous les coefficients de Fourier d'indice négatif sont nuls. Vérifier que $H^2(S^1)$ est un sous-espace fermé de $L^2(S^1)$; en déduire que c'est un espace de Hilbert pour le produit scalaire induit par celui de $L^2(S^1)$. L'espace $H^2(S^1)$ est un espace de Hardy ; attention, ce n'est *pas* un espace de Sobolev. ▷

EXERCICE 5.21. ◁ On note P la projection orthogonale de $L^2(S^1)$ sur $H^2(S^1)$. Soit $a \in C^0(S^1)$; on définit un opérateur T_a dans $H^2(S^1)$ par

$$T_a u = P(M_a u).$$

Montrer que T_a est un opérateur borné de $H^2(S^1)$ dans lui-même. Donner sa matrice dans la base de Fourier. ▷

EXERCICE 5.22. ◁Soit b dans $C^0(S^1)$; est ce que, en général, T_a et T_b commutent ? ▷

EXERCICE 5.23. ◁ On note $H^2_-(S^1)$ l'ensemble des fonctions de $L^2(S^1)$ dont les coefficients de Fourier d'indice positif ou nul sont nuls, et Q la projection orthogonale de $L^2(S^1)$ sur $H^2_-(S^1)$. On définit l'application linéaire J de $L^2(S^1)$ par son expression dans la base de Fourier :

$$(\widehat{Jx})_k = \hat{x}_{-k-1}.$$

Trouver la représentation dans la base de Fourier des opérateurs

$$H_a = PM_aQJ\,|_{H^2(S^1)} \text{ et } \widetilde{H}_a = JQM_aP\,|_{H^2(S^1)}\,.$$

Montrer que H_a et $\widetilde{H}_a$ sont des opérateurs bornés, pour $a \in C^0(S^1)$. ▷

EXERCICE 5.24. ◁Soit a dans $C^0(S^1)$. Montrer que H_a est un opérateur compact.

Indication : comme a est une fonction continue, et que les polynômes trigonométriques sont denses dans $C^0(S^1)$, on considère une suite de polynômes trigonométriques a_n convergeant uniformément vers a ; on montrera alors que la suite des H_{a_n} converge en norme d'opérateur vers H_a et que les H_{a_n} sont de rang fini. ▷

EXERCICE 5.25. ◁ Soient a et b dans $C^0(S^1)$. Montrer que $T_aT_b - T_{ab}$ est compact.
Indication : montrer l'identité

$$T_{ab} = T_aT_b + H_a\widetilde{H}_b.$$

▷

EXERCICE 5.26. ◁ Calculer le spectre de T_{e_1}.
Indication : si $|z| \leq 1$, calculer la solution de l'équation $zx - T_ax = e_0$, et montrer qu'elle n'est pas dans $\ell^2(\mathbb{N})$. Si $|z| > 1$, calculer la solution de l'équation $zx - T_ax = y$, avec y quelconque dans $\ell^2(\mathbb{N})$; on pourra vérifier que la série

$$\sum_{j=0}^{\infty} \frac{k}{|z|^{2k}}$$

converge. ▷

EXERCICE 5.27. ◁ On considère une troncation de dimension finie n de l'opérateur T_{e_1} de la question 5.26 : c'est le bloc $n \times n$ en haut à gauche de la matrice infinie de T_{e_1} dans la base de Fourier, on le note S_n. Le ε-pseudospectre de S_n est l'ensemble des z dans $\mathbb{C}$ tels que $z - S_n$ n'est pas inversible, ou la norme de $(z - S_n)^{-1}$ est supérieure à $1/\varepsilon$. Montrer que le ε-pseudospectre de S_n est inclus dans un disque de centre 0 et de rayon $R_{n,\varepsilon}$ et contient un disque de centre 0 et de rayon $r_{n,\varepsilon}$ et que ces deux rayons tendent vers 1 quand n tend vers l'infini.
Indication : utiliser une norme $\| \|_\infty$ et le résultat de comparaison entre cette norme et la norme $\| \|_2$. ▷

EXERCICE 5.28. ◁ On note $n(A)$ la dimension du **n**oyau d'un opérateur A et $m(A)$ la dimension de l'orthogonal de son i**m**age ; un opérateur est dit de Fredholm si son noyau est de dimension finie et son image est fermée et de codimension finie ; dans ce cas, son indice est défini par

$$\mathrm{ind}(A) = n(A) - m(A).$$

Montrer que l'indice de T_{e_m} est égal à $-m$. ▷

EXERCICE 5.29. ◁ Soit A un opérateur de Fredholm ; montrer que si E est de norme assez petite, alors $A + E$ est encore de Fredholm, et son indice est le même que celui de A. ▷

EXERCICE 5.30. ◁ Soit a dans $C^0(S^1)$; on suppose que l'image de a ne contient pas 0 ; on rappelle que l'indice du chemin a par rapport à z est l'intégrale

$$\mathrm{ind}(0, a) = \int_0^{2\pi} \frac{a'(\theta)}{a(\theta) - z}\, \mathrm{d}\theta$$

si a est de classe C^1 ; si a est seulement continu, c'est l'intégrale

$$\mathrm{ind}(0, b) = \int_0^{2\pi} \frac{b'(\theta)}{b(\theta) - z}\, \mathrm{d}\theta$$

pour b de classe C^1 et suffisamment proche en norme uniforme de a. Enfin, on peut trouver une homotopie à valeur dans $\mathbb{C} \setminus \{0\}$ reliant a et e_m, c'est à dire qu'il existe une fonction $A(\theta, t)$ continue de $S^1 \times [0, 1]$ dans $\mathbb{C} \setminus \{0\}$ telle que $A(\cdot, 0) = e_m$ et $A(\cdot, 1) = a$.
Montrer que l'indice de T_a est l'opposé de l'indice de 0 par rapport à a. ▷

EXERCICE 5.31. ◁ Montrer que le spectre de T_a est la réunion de l'image de a et de l'ensemble des z dont l'indice par rapport à a n'est pas nul. ▷

Chapitre 6

Méthodes itératives basiques

Ce chapitre est largement basé sur [**10**, Chapter 6]. On y présentera les méthodes itératives de base : Jacobi, Gauss-Seidel et sur-relaxation successive (SOR) ; dans les chapitres suivants on étudiera les méthodes des méthodes plus sophistiqués (et performants) des gradients conjugués et multigrille. On illustrera ces méthodes par leur application à l'équation de Poisson sur le cube $[0,1]^d$ avec des conditions de Dirichlet nulles, problème modèle s'il y en a.

Cependant notons que pour dans une situation particulière, il n'est pas toujours clair quelle est la meilleur stratégie. Pour s'y orienter, on peut faire appel à de l'aide en ligne en `http//www.netlib.org/templates`. Ce directoire contient un bouquin et des implementations pour tous les méthodes itératives qu'on traitera, en Matlab, Fortran et C++.

1. L'équation de Poisson

L'équation de Poisson et l'opérateur de Laplace interviennent dans la modélisation de l'éléctromagnetisme, le chaleur, diffusion, mécanique quantique, etc.. Dans cette section on étudiera la discrétisation de cette équation *via* différences finies standard, surtout en dimension 1 et 2.

1.1. L'équation de Poisson en dimension 1. Soit

$$\Delta = \frac{\partial^2}{\partial x_1^2} + \cdots + \frac{\partial^2}{\partial x_d^2}$$

l'opérateur de Laplace ou *laplacien* en dimension d. L'équation de Poisson de dimension d sur le cube unité avec des conditions de Dirichlet nulles est

$$-\Delta u = f \quad \text{pour } x \in [0,1]^d \text{ et } \quad u \equiv 0 \text{ pour } x \in \partial([0,1]^d)$$

pour une fonction $f : [0,1]^d \to \mathbb{R}$ donnée. Pour $d = 1$ se réduit à

$$-u'' = f \quad \text{pour } x \in \Omega = [0,1] \text{ et } \quad u(0) = u(1) = 0.$$

On discrétise cette équation avec des différences finies centrées : pour $N \in \mathbb{N}$ on pose $h := (N+1)^{-1}$ et on considère la grille

$$\Omega_h := \{jh : j = 1, \ldots, N\}$$

des nodes *à l'intérieur* de l'intervalle $[0,1]$ divisé en $N+1$ sous-intervalles :

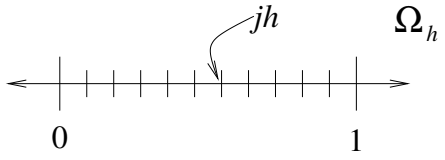


FIGURE 1. La grille Ω_{10}

On pose

$$\mathcal{G}(\Omega_h) : \Omega_h \to \mathbb{R}$$

pour l'ensemble des fonctions réelles de cet ensemble. On étends $u \in \mathcal{G}(\Omega_h)$ aux bords par la convention $u(0) = u(1) = 0$. Pour $x \in \Omega_h$

$$\begin{aligned} u'(x + \frac{h}{2}) &\approx \frac{u(x+h) - u(x)}{h}, \\ u'(x - \frac{h}{2}) &\approx \frac{u(x) - u(x-h)}{h}, \\ u''(x) &\approx \frac{2u(x) - u(x+h) - u(x-h)}{h^2}. \end{aligned}$$

La discrétisation de $-\Delta$ qui en résulte est l'opérateur

$$L_h : \mathcal{G}(\Omega_h) \to \mathcal{G}(\Omega_h) \quad , \quad L_h(u)(x) = h^{-2}\Big(u(x-h) - 2u(x) + u(x+h)\Big).$$

Alternativement on peut exprimer L_h en *notation stencil*

$$L_h = h^{-2}[-1 \quad 2 \quad -1]_h.$$

C'est une approximation d'ordre 2 du laplacien : pour une fonction $u \in \mathcal{C}^4(\Omega)$ et $h \to 0$

$$u(x+h) = u(x) + u'(x)h + \frac{u''(x)}{2}h^2 + \frac{u'''(x)}{6}h^3 + O(||u^{(4)}||_\infty)h^4$$

d'où $L_h(u(x)) = -u''(x) + h^2 O(||u^{(4)}||_\infty)$.

L'équation approchée $L_h u_h = f_h$ se traduit dans le système linéaire Toeplitz bande symétrique d'ordre $N \times N$

$$h^{-2}\begin{bmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{bmatrix}\begin{bmatrix} u(h) \\ u(2h) \\ \vdots \\ u((N-1)h) \\ u(Nh) \end{bmatrix} = \begin{bmatrix} f(h) \\ f(2h) \\ \vdots \\ f((N-1)h) \\ f(Nh) \end{bmatrix},$$

Alternativement on écrira L_N, u_N, f_N pour L_h, u_h, f_h, respectivement.

LEMME 1.1. *Les valeurs propres de $T_N := h^2 L_h$ sont*

$$\lambda_\ell = 2\left(1 - \cos\left(\frac{\pi\ell}{N+1}\right)\right) = 4\sin^2\left(\frac{\pi\ell}{2(N+1)}\right) \text{ pour } \ell = 1, \ldots, N$$

avec comme pour base ortonormale des fonctions propres

$$\varphi_\ell : \mathcal{G}(\Omega_N) \to \mathcal{G}(\Omega_N) \quad , \quad x \mapsto \left(\frac{2}{N+1}\right)^{1/2} \sin(\pi\ell x).$$

Identifions φ_ℓ avec le vecteur dans $\mathbb{R}^N$ correspondant et posons $Z := [\varphi_1 \cdots \varphi_N] \in \mathbb{R}^{N\times N}$, le lemme peut alors se récrire comme que

$$T_N = Z \cdot \operatorname{diag}(\lambda_1, \ldots, \lambda_N) \cdot Z^T.$$

DÉMONSTRATION. Soit

$$B_N := 2 \cdot \mathbf{1}_N - T_N = \begin{bmatrix} 0 & 1 & & & \\ 1 & 0 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & 0 & 1 \\ & & & 1 & 0 \end{bmatrix} \in \mathbb{R}^{N\times N}.$$

La détermination d'un vecteur propre $v = (v_1, \dots, v_N) \in \mathbb{C}^N \setminus \{0\}$ de B_N et son valeur propre $\lambda \in \mathbb{C}$ équivaut à résoudre le système

$$\begin{aligned} v_2 &= \lambda v_1 \\ v_1 + v_3 &= \lambda v_2 \\ v_2 + v_4 &= \lambda v_3 \\ &\vdots \\ v_{N-2} + v_{N-3} &= \lambda v_{N-1} \\ v_{N-1} &= \lambda v_N. \end{aligned}$$

C'est la récurrence linéaire

$$v_{j+1} - \lambda v_j + v_{j-1} = 0.$$

L'équation caractéristique est

$$z^2 - \lambda z + 1 = 0$$

et notons z_1, z_2 les racines de cette équation. Le discriminant étant $\lambda^2 - 4$, si $\lambda = \pm 2$ alors $z_1 = z_2 = 1$ et

$$v_j = a + bj \quad \text{pour certains } a, b \in \mathbb{C}.$$

Or les conditions au bord $v_0 = a = 0$ et $v_N = a + bN = 0$ entraînent $a = b = 0$ et par conséquent $v = 0$. On peut donc supposer $\lambda \neq \pm 2$, alors $z_1 \neq z_2$ et

$$v_j = az_1^j + bz_2^j \quad \text{pour certains } a, b \in \mathbb{C}.$$

La condition $v_0 = a + b = 0$ entraîne $b = -a$ et puisqu'on ne cherche que des solutions non triviales, la deuxième condition

$$v_{N+1} = az_1^{N+1} + bz_2^{N+1} = a(z_1^{N+1} - z_2^{N+1}) = 0$$

entraîne $z_1^{2(N+1)} = 1$ (car $z_1 z_2 = 1$). On en déduit que les couples des différentes solutions possibles sont

$$(z_{1,\ell}, z_{2,\ell}) = (e^{\pi i \ell/(N+1)}, e^{-\pi i \ell/(N+1)}) \quad , \quad \ell = 0, \dots, N.$$

Le vecteur propre associé est (modulo le choix du facteur scalaire)

$$(v_\ell)_j = \frac{1}{2i}(z_1^j - z_2^j) = \sin\Big(\frac{\pi \ell}{N+1} j\Big) \quad , \quad j = 1, \dots, N$$

pour $\ell = 1, \dots, N$, car le cas $\ell = 0$ ne donne que la solution triviale $v_\ell = 0$. On vérifie que $||v_\ell||_2 = \sqrt{(N+1)/2}$ **(faut le faire !)** et il en résulte la fonction propre

$$\varphi_\ell : \mathcal{G}(\Omega_N) \to \mathcal{G}(\Omega_N) \quad , \quad x \mapsto \Big(\frac{2}{N+1}\Big)^{1/2} \sin(\pi \ell x).$$

La valeur propre de B_N correspondante est

$$\lambda_\ell(B_N) = z_1 + z_2 = 2\cos(\frac{\pi \ell}{N})$$

tandis que pour T_N

$$\lambda_\ell(T_N) = 2 - \lambda_\ell(B_N) = 2\left(1 - \cos\Big(\frac{\pi \ell}{N+1}\Big)\right) = 4\sin^2\Big(\frac{\pi \ell}{N+1}\Big).$$

□

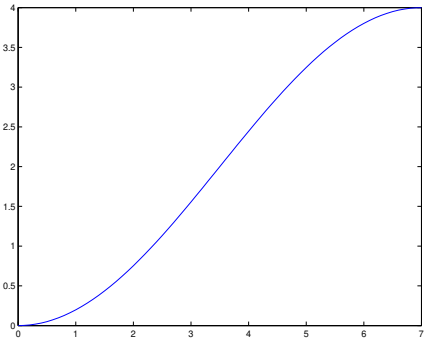


FIGURE 2. Valeurs propres de T_6

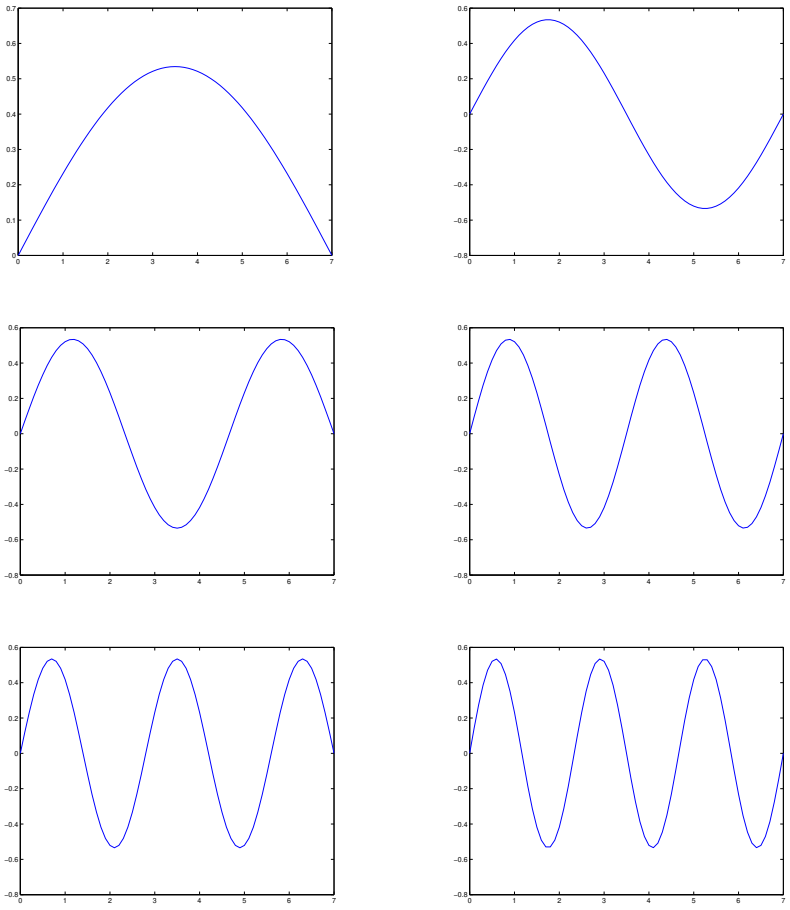


FIGURE 3. Fonctions propres de T_6

On a $\lambda_N \approx 4$ et $\lambda_1 \approx 4(\frac{\pi}{2(N+1)})^2 = O(h^2)$. La figure suivante montre les valeurs propres et les fonctions propres correspondantes pour $N = 6$:

À présent on peut estimer l'écart entre la solution exacte $\widehat{u}$ de l'opérateur discret et la vraie solution u de l'équation différentielle, pour $f \in \mathcal{C}^2(\Omega)$: on a

$$L_h(u) = -u'' + h^2 O(||u^{(4)}||_\infty) = f + h^2 O(||f''||_\infty)$$

donc

$$||u - \widehat{u}||_2 = h^2 O(||f''||_\infty)||L_h^{-1}||_2 = O(h^2||f^{(2)}||_\infty)$$

car la plus grande valeur propre de L_h^{-1} est $h^2\lambda_1^{-1} = O(1)$.

Les opérateurs $-\Delta$ et $L_N = (N+1)^2 T_N$ partagent des similarités qualitatives, en particulier ses fonctions propres se ressemblent : on sait que les fonctions propres de $-\Delta$ sont forcément de la forme

$$z(x) = \alpha \sin(\sqrt{\lambda}x) + \beta \cos(\sqrt{\lambda}x)$$

et les conditions aux bords entraînent $\beta = 0$ et $\sqrt{\lambda} = k\pi$ avec $k \in \mathbb{Z}$. Il s'en suit que ces fonctions propres sont

$$z_\ell(x) = \sin(\ell\pi) \quad \text{pour } \ell \in \mathbb{N},$$

c'est-à-dire que φ_ℓ est (sauf pour un facteur scalaire) la restriction de z_ℓ à la grille Ω_h. La valeur propre correspondante est

$$\lambda_\ell^\infty = \ell^2\pi^2 \approx \lambda_\ell(L_N) = (N+1)^2 \left(4\sin^2\left(\frac{\pi\ell}{2(N+1)}\right)\right) \quad \text{pour } N \to \infty.$$

1.2. L'équation de Poisson en dimension 2 et plus. On étudiera en détail l'équation de Poisson en dimension 2

$$-u_{xx} - u_{yy} = f \quad \text{sur } \Omega^2 := [0,1]^2 \text{ et } u \equiv 0 \quad \text{sur } \partial\Omega^2.$$

On utilise le même pas de discrétisation $h = (N+1)^{-1}$ et différences finies centrées

$$-u_{xx}(x,y) \approx \frac{2u(x,y) - u(x+h,y) - u(x-h,y)}{h^2},$$
$$-u_{yy}(x,y) \approx \frac{2u(x,y) - u(x,y+h) - u(x,y-h)}{h^2}$$

donc

$$-\Delta(u)(x,y) \approx L_h(u) := \frac{4u(x,y) - u(x+h,y) - u(x+h,y) - u(x,y+h) - u(x,y-h)}{h^2},$$

et comme précédemment on peut voir que l'erreur est quadratique en h :

$$L_h(u)(x,y) = -\Delta(u)(x,y) + O(||u^{(4)}||_\infty)h^2 \quad \text{pour } u \in \mathcal{C}^4(\Omega).$$

Cet discrétisation est peut-être mieux décrite à l'aide du *stencil à 5 points* :

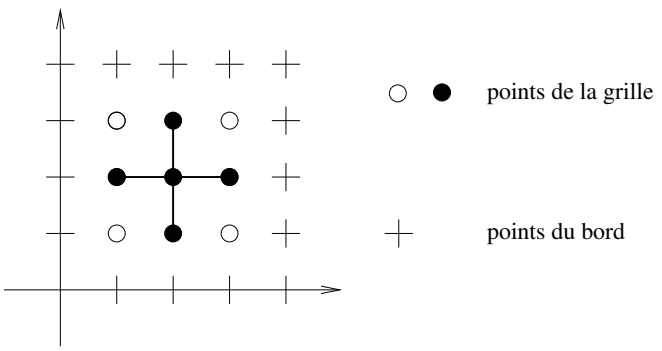


FIGURE 4. Stencil à 5 points

Posons $\Omega := [0,1]^2$, pour $u \in \mathcal{G}(\Omega)$ on pose $u_{i,j} := u(ih, jh)$ et similairement pour f. La discrétisation de l'équation différentielle est donc le système de $n = N^2$ équations linéaires

$$4u_{i,j} - u_{i+1,j} - u_{i-1,j} - u_{i,j+1} - u_{i,j-1} = h^2 f_{i,j} \quad \text{pour } 1 \leq i, j \leq N \tag{34}$$

qui tient en compte les conditions aux bords $u_{i,j} = 0$ si i ou j sont égaux à 0 ou N.

Il sera convenable de regarder l'inconnue (solution de l'équation discrétisée) comme une matrice $U := [u_{i,j}]_{1 \leq i,j \leq N}$ et de même on pose $F = [f_{i,j}]_{1 \leq i,j \leq N}$. On a

$$2u_{i,j} - u_{i+1,j} - u_{i-1,j} = (T_N \cdot U)_{i,j},$$
$$2u_{i,j} - u_{i,j+1} - u_{i,j-1} = (U \cdot T_N)_{i,j},$$

donc le système (34) est de type Sylvester :

$$T_N \cdot U - U \cdot T_N = h^2 F.$$

On a déjà montré ailleurs que ceci entraîne que les valeurs propres de $T_{N\times N} := h^2 L_h$ sont

$$\lambda_\ell(T_N) + \lambda_m(T_N) \quad \text{pour } 1 \leq \ell, m \leq N.$$

Soient $\varphi_\ell, \varphi_m \in \mathbb{R}^N$ les fonctions propres de T_N correspondantes, alors

$$T_N \varphi_\ell \varphi_m^T + \varphi_\ell \varphi_m^T T_N = (\lambda_\ell + \lambda_m)(\varphi_\ell \varphi_m^T)$$

et cette matrice $\varphi_\ell \varphi_m^T$ corresponds à la fonction sur la grille

$$\Omega_h \to \mathbb{R} \quad , \quad (x, y) \mapsto \varphi_\ell(x) \cdot \varphi_m(y).$$

Ce sont des fonctions linéairement indépendantes et donc elles forment une base (orthonormal) de fonctions propres. Ces fonctions propres et valeurs propres sont donc

$$\frac{2}{N+1} \sin(\pi \ell x) \cdot \sin(\pi m x) \quad \text{et} \quad (\ell^2 + m^2)\pi^2 \quad \text{pour } 1 \leq \ell, m \leq N.$$

En particulier, le plus petit valeur propre de $T_{N\times N}$ est $2\lambda_1(T_N)$ et le plus grand $2\lambda_N(T_N)$ donc

$$\kappa_2(T_{N\times N}) = \kappa_2(T_N) = \frac{4\sin^2\left(\frac{\pi N}{2(N+1)}\right)}{4\sin^2\left(\frac{\pi}{2(N+1)}\right)} \approx \frac{4(N+1)^2}{\pi^2}.$$

Alternativement on peut écrire le système de façon vectoriel

$$\mathrm{Vect}(T_N \cdot U - U \cdot T_N) = h^2 \mathrm{Vect}(F).$$

L'opérateur Vect numérote les variables suivant un ordre lexicographique :

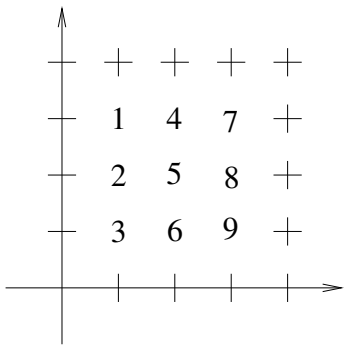


FIGURE 5. Ordre lexicographique

La vectorisation du système s'écrit en termes des produits tensoriels

$$(\mathbf{1}_N \otimes \mathrm{Vect}(T_N) - \mathrm{Vect}(T_N)\mathbf{1}_N)\mathrm{Vect}(U) = h^2\mathrm{Vect}(F),$$

c'est-à-dire

$$\begin{bmatrix} T_N & & \\ & \ddots & \\ & & T_N \end{bmatrix} - \begin{bmatrix} 2\cdot\mathbf{1}_N & -\mathbf{1}_N & & \\ -\mathbf{1}_N & \ddots & \ddots & \\ & \ddots & \ddots & -\mathbf{1}_N \\ & & -\mathbf{1}_N & 2\cdot\mathbf{1}_N \end{bmatrix}$$

$$= \begin{bmatrix} T_N - 2\cdot\mathbf{1}_N & \mathbf{1}_N & & \\ \mathbf{1}_N & \ddots & \ddots & \\ & \ddots & \ddots & \mathbf{1}_N \\ & & \mathbf{1}_N & T_N - 2\cdot\mathbf{1}_N \end{bmatrix}.$$

Par exemple pour $N = 3$:

$$\begin{bmatrix} 4 & -1 & & -1 & & & & & \\ -1 & 4 & -1 & & -1 & & & & \\ & -1 & 4 & & & -1 & & & \\ -1 & & & 4 & -1 & & -1 & & \\ & -1 & & -1 & 4 & -1 & & -1 & \\ & & -1 & & -1 & 4 & & & -1 \\ & & & -1 & & & 4 & -1 & \\ & & & & -1 & & -1 & 4 & -1 \\ & & & & & -1 & & -1 & 4 \end{bmatrix}$$

C'est une matrice Toeplitz bloc Toeplitz symétrique qui est *sparse* mais *pas bande*. La discrétisation du laplacien en dimension $d = 3$ s'écrit

$$T_{N\times N\times N} = T_N \otimes \mathbf{1}_N \otimes \mathbf{1}_N + \mathbf{1}_N \otimes T_N \otimes \mathbf{1}_N + \mathbf{1}_N \otimes \mathbf{1}_N \otimes T_N$$

et similairement en dimension supérieure.

1.3. Méthodes itératives sur l'équation de Poisson. Voici un sommaire de la complexité des différentes méthodes appliquées au 2-Poisson ($n = N^2$) :

Méthode	Ops	Espace	Directe/Itérative
Gauss	n^3	n^2	D
Gauss bande	n^2	$n^{3/2}$	D
Jacobi	n^2	n	I
Gauss-Seidel	n^2	n	I
SOR	$n^{3/2}$	n	I
Gradients conjugués	$n^{3/2}$	n	I
FFT	$n\log(n)$	n	D
Multigrille	n	n	I
Borne inférieure	n	n	

Les méthodes directes donnent une réponse exacte modulo les erreurs de troncations, tandis que les méthodes itératives ne donnent qu'une approximation de la solution. Il n'est donc pas facile de comparer les méthodes. Pour les itératives, cette

table indique le coût d'approximer la solution jusqu'à un seuil constant. Alternativement, on peut itérer jusqu'à que l'erreur soit $O(h^2) = O((N+1)^{-2})$, de l'ordre de l'erreur introduit par la discrétisation. Ceci augmente le coût des méthodes itératives d'un facteur $\log(n)$.

2. Méthodes itératives basiques

C'est-à-dire Jacobi, Gauss-Seidel et relaxation successive (SOR). Typiquement leur application directe n'est pas efficace, mais ce sont des des composantes des méthodes plus sophistiqués et efficaces comme le multigrille.

Soit $A \in \mathbb{K}^{N\times N}$ et considérons une décomposition du type

$$A = M - K \quad \text{avec } M \text{ inversible.}$$

Une telle décomposition donne une méthode itérative

$$x^{m+1} = M^{-1}Kx^m + M^{-1}b.$$

Le système d'équations $Ax = Mx - Kx = b$ équivaut à $x = M^{-1}Kx + M^{-1}b$ donc si l'itération converge, c'est forcément vers la solution cherchée.

Or, ceci n'est pas toujours le cas : considérons l'itération

$$x^{m+1} = 2x^m - 1.$$

La solution du système associé est $x^\infty = 1$. Posons $x^m = 1 + \varepsilon^m$ pour un certain $\varepsilon^m \neq 0$, alors

$$1 + \varepsilon^{m+1} = 2(1 + \varepsilon^m) - 1$$

donc $\varepsilon^{m+1} = 2\varepsilon^m$!

THÉORÈME 2.1. *L'itération $x^{m+1} = Rx^m + c$ converge pour tout point initial $x^0 \in \mathbb{K}^N$ si et seulement si $\rho(R) < 1$.*

DÉMONSTRATION. Si $\rho(R) = \max_j |\lambda_j(R)| \geq 1$ prenons $v \in \mathbb{C}^N \setminus \{0\}$ tel que $Rv = \lambda v$ avec $|\lambda| \geq 1$. Soit x^∞ la solution du système associé et prenons $x^0 := v + x^\infty$. Si l'on écrit $x^m = x^\infty + \varepsilon^m$ pour un certain $\varepsilon^m \in \mathbb{K}^N$ alors

$$\varepsilon_m = \lambda^m v$$

donc $(x^m)_m$ ne converge pas.

Supposons au contraire que $\rho(R) < 1$ et soit $||\cdot||$ une norme d'opérateurs subordonnée à une norme vectorielle quelconque $|\cdot|$, alors

$$\rho(R) = \lim_{n\to\infty} ||R^n||^{1/n}.$$

Il s'en suit qu'il existe n_0 tel que $||R^n|| < 1$ pour $n \geq n_0$. Alors

$$x^{m+1} - x^\infty = Rx^m + c - x^\infty = R(x^m - x^\infty)$$

car $x^\infty = Rx^\infty + c$ c'est le point fixe de l'itération. Écrivons $m = qn_0 + r$ avec $0 \leq r \leq n_0 - 1$, alors

$$|x^m - x^\infty| \leq ||R^m|| \cdot |x^0 - x^\infty| \leq ||R^{n_0}||^q \cdot ||R^r|| \cdot |x^0 - x^\infty| \to 0.$$

□

DÉFINITION 2.2. La *raison de convergence* de l'itération $x^{m+1} = Rx^m + c$ est

$$r(R) := -\log_2 \rho(R).$$

C'est le nombre de bits corrigés *per* itération, car

$$\log_2 ||x^m - x^\infty|| - \log_2 ||x^{m+1} - x^\infty|| \geq r(R) + o(1).$$

Le but du jeu c'est de trouver des décompositions $A = M - K$ fournissant des bonnes itérations, et pour cela il faut remplir les conditions :

(1) $M^{-1}Kx$ et $M^{-1}b$ faciles à calculer ;

(2) $\rho(R)$ petit.

Le choix $M = \mathbf{1}_N$ vérifie (1) mais pas (2), et réciproquement le choix $M = A^{-1}$ vérifie (2) et pas (1).

Pour toutes les décompositions dans ce chapitre on utilise la notation

$$A = D - L - U = D(\mathbf{1}_N - \overline{L} - \overline{U})$$

où D est la diagonale (supposée inversible) de A, L et U les parties triangulaire inférieure et supérieure de A respectivement, puis $\overline{L} = D^{1-}L$ et $\overline{U} = D^{-1}U$.

2.1. Méthode de Jacobi. Corresponds à la décomposition $A = D-(L+U)$. L'itération est donc

$$x^{m+1} = R_J x^m + c_J.$$

avec $R_J = D^{-1}(L+U) = \overline{L} + \overline{U}$ et $c_J = D^{-1}b$. Le pseudo-code est

Itération de Jacobi :

for j **from** 1 **to** N **do**

(1) $x_j^{m+1} \leftarrow \frac{1}{a_{j,j}} \left(b_j - \sum_{k \neq j} a_{j,k} x_k^m \right)$ **;**

od ; end.

Pour le problème modèle du 2-Poisson, ceci se simplifie en

$$u_{i,j}^{m+1} = \frac{1}{4} \left(u_{i-1,j}^m + u_{i+1,j}^m + u_{i,j-1}^m + u_{i,j-1}^m + h^2 f_{i,j} \right).$$

Chaque nouveau valeur de $u_{i,j}$ s'obtient donc en moyennant ses points voisins et rajoutant $mofrac14h^2 f_{i,j}$.

2.2. Méthode de Gauss-Seidel. Dans la méthode de Jacobi, au moment du j-ème pas on déjà corrigé les $j-1$ variables précédentes. L'idée de la méthode de Gauss-Seidel est d'en profiter pour améliorer la correction de la j-ème variable :

Itération de Gauss-Seidel :

for j **from** 1 **to** N **do**

(1) $x_j^{m+1} \leftarrow \frac{1}{a_{j,j}} \left(b_j - \sum_{k=1}^{j-1} a_{j,k} x_k^{m+1} - \sum_{k=j+1}^{N} a_{j,k} x_k^m \right)$ **;**

od ; end.

Ceci corresponds à une décomposition $A = (D-L)+U$; en notation matricielle l'itération de Gauss-Seidel est

$$x^{m+1} = R_{GS} x^m + c_{GS}$$

avec $R_{GS} = (D-L)^{-1}U = (\mathbf{1}_N - \overline{L})\overline{U}$ et $c_{GS} = (D-L)^{-1}b$.

Contrairement à la méthode de Jacobi, cette itération *dépends* de l'ordre des variables et en général n'est pas bien parallelisable. Pour le 2-Poisson il y a deux choix usuelles, la numérotation lexicographique standard (comme dans la figure 1.2) et la numérotation rouge-noir. Ce deuxième ordre sera le plus important dans la suite.

La numérotation rouge-noir consiste à marquer les éléments d'un "couleur" ou d'un autre de façon alternée, comme dans une table à échecs. Ensuite on ordonne toutes les variables rouges avant que les variables noires, comme dans la figure suivante :

Ici on a ordonné les variables de chaque couleur suivant un ordre lexicographique, mais cela n'aura pas d'importance dans la suite. Ce qui est important c'est qu'on a d'abord toutes les variables d'un couleurs et puis les variables de l'autre

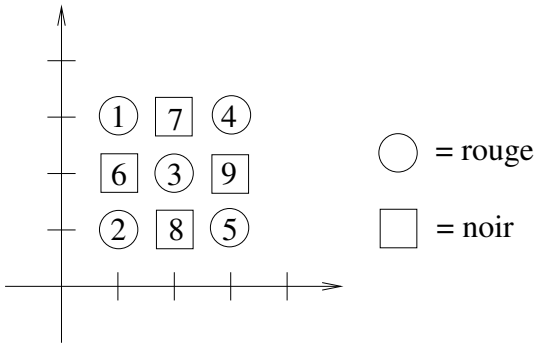


FIGURE 6. Numérotation rouge-noir

couleur. La numérotation rouge-noir possède l'appréciable avantage de rendre parallelisable la méthode de Gauss-Seidel sur le 2-Poisson : l'actualisation des nodes rouges ne dépends que de ses voisins, qui sont noirs, et similairement pour les nodes noirs. On peut donc actualiser les nodes rouges en parallèle :

$$v_{i,j}^{m+1} \leftarrow \frac{1}{4}\left(v_{i-1,j}^{m} + v_{i+1,j}^{m} + v_{i,j-1}^{m} + v_{i,j+1}^{m} + h^2 f_{i,j}\right) \quad \text{pour } (i,j) \text{ rouge,}$$

et ensuite faire pareil pour les nodes noirs :

$$v_{i,j}^{m+1} \leftarrow \frac{1}{4}\left(v_{i-1,j}^{m+1} + v_{i+1,j}^{m+1} + v_{i,j-1}^{m+1} + v_{i,j+1}^{m+1} + h^2 f_{i,j}\right) \quad \text{pour } (i,j) \text{ noir.}$$

2.3. Sur-relaxation successive (SOR). La méthode de sur-relaxation successive (en anglais : *successive overrelaxation (SOR)*) consiste à pondérer l'itération de Gauss-Seidel avec l'identité suivant un *paramètre de relaxation* ω. Soit

$$\widetilde{x}^{m+1} = R_{GS}x^m + c_{GS}$$

l'itération de Gauss-Seidel, alors la SOR(ω) est

$$x^{m+1} = (1-\omega)x^m + \omega\widetilde{x}^m.$$

Le pseudo-code résulte :

Méthode de sur-relaxation successive (SOR) :
for j **from** 1 **to** N **do**
(1) $x_j^{m+1} \leftarrow \omega x_j^m + \frac{\omega}{a_{j,j}}\left(b_j - \sum_{k=1}^{j-1} a_{j,k}x_k^{m+1} - \sum_{k=j+1}^{N} a_{j,k}x_k^m\right)$ **;**
od ; end.

On récrit cela en

$$a_{j,j}x_j^{m+1} + \omega\sum_{k=1}^{j-1} a_{j,k}x_k^{m+1} = (1-\omega)a_{j,j}x_j^m - \omega\sum_{k=j+1}^{N} a_{j,k}x_k^m + \omega b_j,$$

soit

$$(D-\omega L)x^{m+1} = \Big((1-\omega)D + \omega U\Big)x^m + \omega b$$

ou encore

$$\begin{aligned} x^{m+1} &= (D-\omega L)^{-1}\Big((1-\omega)D + \omega U\Big)x^m + \omega(D-\omega L)^{-1}b \\ &= (\mathbf{1}_N - \omega\overline{L})^{-1}\Big((1-\omega)\mathbf{1}_N + \omega\overline{U}\Big)x^m + \omega D^{-1}(\mathbf{1}_N - \omega\overline{L})^{-1}b \\ &= R_{SOR(\omega)}x^m + c_{SOR(\omega)}. \end{aligned}$$

Quand $\omega > 1$ on parle de *sur-relaxation* et quand $\omega < 1$ on parle de *sous-relaxation*; pour $\omega = 1$ on retrouve Gauss-Seidel. Dans les paragraphes suivants on montre (parmi d'autres choses) comment choisir le meilleur ω pour le 2-Poisson.

3. Convergence de Jacobi, Gauss-Seidel et SOR

Soit $T_{N\times N} = h^{-2}L_N$ la matrice de la discrétisation du 2-Poisson. Sa diagonale est $4 \cdot \mathbf{1}_{N\times N}$ donc la matrice de l'itération de Jacobi est

$$R_J = 1 - \frac{T_{N\times N}}{4}.$$

C'est une matrice symétrique ayant comme des valeurs propres

$$1 - \frac{\lambda_{i,j}(T_{N\times N})}{4} = 1 - \sin^2(\frac{\pi i}{2(N+1)}) - \sin^2(\frac{\pi j}{2(N+1)}) \quad , \quad 1 \le i,j \le N.$$

Son rayon spectral est

$$\rho(R_J) = 1 - 2\sin^2\left(\frac{\pi}{2(N+1)}\right) = \cos\left(\frac{\pi}{N+1}\right) \approx 1 - \frac{\pi^2}{2(N+1)^2}.$$

Comme N devient plus grand, le conditionnement de $T_{N\times N}$ empire et le rayon spectral $\rho(R_J)$ s'approche de 1. Le nombre m d'itérations nécessaires pour corriger 1 bit vérifie $-1 = m\log\left(1 - \frac{\pi^2}{2(N+1)^2}\right)$ donc

$$m \approx \frac{2(N+1)^2}{\pi^2} = O(N^2) = O(n).$$

Chaque pas de l'itération coûte $O(1)$ ops *per* composante, donc $O(n)$ en total. En tout, le coût de corriger 1 bit par la méthode de Jacobi appliquée au 2-Poisson est de $O(n^2)$ ops.

Ceci est un phénomène fréquent : plus le conditionnement du problème est mauvais, plus las méthodes itératives sont lentes. Cependant il y a des exceptions importantes, comme le multigrille.

On montrera que (toujours pour le 2-Poisson) le rayon spectral de la matrice de la méthode de Gauss-Seidel avec numérotation rouge-noir est

$$\rho(R_{GS}) = \rho(R_J)^2 = \cos^2\left(\frac{\pi}{N+1}\right).$$

L'erreur de l'approximation décroît deux fois plus rapide que pour Jacobi, donc le coût reste en $O(n^2)$ ops.

Toujours pour la numérotation rouge-noir, on montrera que le rayon spectral de la sur-relaxation successive avec le paramètre optimal

$$1 < \omega_{\text{opt}} = \frac{2}{1 + \sin\left(\frac{\pi}{N+1}\right)} < 2$$

est

$$\rho(R_{SOR(\omega_{\text{opt}})}) = \frac{\cos^2\left(\frac{\pi}{N+1}\right)}{\left(1 + \sin\left(\frac{\pi}{N+1}\right)\right)^2} \approx \frac{(1 - x^2/2)^2}{(1+x)^2} \approx 1 - 2x = 1 - \frac{2\pi}{N+1}.$$

Donc il suffit de $O(N) = O(n^{1/2})$ itérations pour corriger 1 bit, ce qui donne un total $O(n^{3/2})$ ops pour la complexité de corriger 1 bit pour la sur-relaxation successive avec un paramètre optimal.

3.1. Démonstrations et détails.

DÉFINITION 3.1. Une matrice M possède la *propriété A* s'il existe une permutation P tel que

$$P \cdot M \cdot P^T = \begin{bmatrix} M_{1,1} & M_{1,2} \\ M_{2,1} & M_{2,2} \end{bmatrix}$$

avec $M_{1,1}$ et $M_{2,2}$ diagonales.

Notons que $P \cdot M \cdot P^T$ corresponds à opérer une permutation simultanée des lignes et colonnes de M.

Un *graphe dirigé* est une collection finie de *nodes* connectés par une collection finie de *flèches* allant d'un node à l'autre. À une matrice $A \in \mathbb{K}^{N \times N}$ on associe le graphe dirigé $G(A)$ des nodes $1, 2, 3, \ldots, N$ et flèches d'un node i vers un node j si et seulement si $a_{i,j} \neq 0$. Par exemple, le graphe dirigé de

$$\begin{bmatrix} 2 & -1 & \\ -1 & 2 & -1 \\ & -1 & 2 \end{bmatrix}$$

est

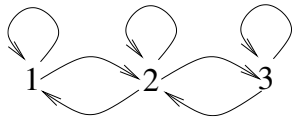


FIGURE 7. $G(A)$

On vérifie que pour une matrice M, la propriété A équivaut à ce que le graphe $G(A)$ soit bipartite, c'est-à-dire qu'on peut séparer les nodes de $G(A)$ dans une union disjointe $S_1 \sqcup S_2$ tel qu'il n'y ait pas des flèches entre des nodes différents de S_i $(i = 1, 2)$. C'est le cas de la numérotation rouge-noir : les ensembles S_1 et S_2 sont faits des nodes rouges et noirs respectivement. Il n'y a pas de connexion entre des nodes rouges ni entre des nodes noirs, sauf dans la diagonale, car chaque point de la grille n'a que des voisins du couleur contraire : c'est de l'intégration raciale parfaite !

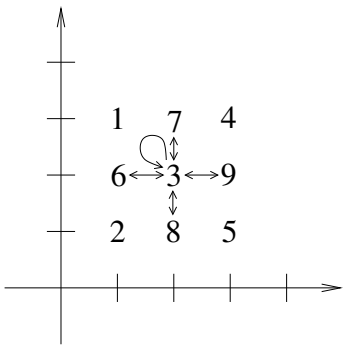


FIGURE 8. Le graphe de la numérotation rouge-noir est bipartite

Dans ce même exemple, voyons ce qu'il en est des matrices correspondantes :

$$P\cdot\begin{bmatrix} 4 & -1 & & -1 & & & & & \\ -1 & 4 & -1 & & -1 & & & & \\ & -1 & 4 & & & -1 & & & \\ -1 & & & 4 & -1 & & -1 & & \\ & -1 & & -1 & 4 & -1 & & -1 & \\ & & -1 & & -1 & 4 & & & -1 \\ & & & -1 & & & 4 & -1 & \\ & & & & -1 & & -1 & 4 & -1 \\ & & & & & -1 & & -1 & 4 \end{bmatrix}\cdot P^T$$

$$=\begin{bmatrix} 4 & & & & & & -1 & -1 & & \\ & 4 & & & & & -1 & & -1 & \\ & & 4 & & & & -1 & -1 & -1 & -1 \\ & & & 4 & & & & -1 & & -1 \\ & & & & 4 & & & & -1 & -1 \\ -1 & -1 & -1 & & & 4 & & & & \\ -1 & & -1 & -1 & & & 4 & & & \\ & -1 & -1 & & -1 & & & 4 & & \\ & & -1 & -1 & -1 & & & & 4 & \end{bmatrix}.$$

Soit M une matrice ayant la propriété A, on écrit $D_i := M_{i,i}$ et on a

$$P\cdot M\cdot P^T=\begin{bmatrix} D_1 & M_{2,1} \\ M_{2,1} & D_2 \end{bmatrix}=\begin{bmatrix} D_1 & \\ & D_2 \end{bmatrix}-\begin{bmatrix} & \\ -M_{2,1} & \end{bmatrix}-\begin{bmatrix} & -M_{2,1} \\ & \end{bmatrix}=D-L-U.$$

Pour $\alpha\neq 0$ on pose

$$R_J(\alpha):=\alpha\overline{L}+\frac{1}{\alpha}\overline{U};$$

notons que $R_J(1)=R_J$ c'est la matrice de l'itération de Jacobi.

LEMME 3.2. *Les valeurs propres de $R_J(\alpha)$ sont indépendants de α.*

DÉMONSTRATION.

$$R_J(\alpha)=\begin{bmatrix} & \frac{1}{\alpha}D_1^{-1}M_{1,2} \\ \alpha D_2^{-1}M_{2,1} & \end{bmatrix}$$

a les mêmes valeurs propres que la matrice similaire

$$\begin{bmatrix} \mathbf{1} & \\ & \frac{1}{\alpha}\cdot\mathbf{1} \end{bmatrix}\cdot R_J(\alpha)\cdot\begin{bmatrix} \mathbf{1} & \\ & \alpha\cdot\mathbf{1} \end{bmatrix}=-\begin{bmatrix} & D_1^{-1}T_{1,2} \\ D_2^{-1}T_{2,1} & \end{bmatrix}=R_J(1).$$

□

Soit M une matrice quelconque avec sa décomposition $M=D-L-U$ et posons

$$R(\alpha):=\alpha D^{-1}L+\frac{1}{\alpha}D^{-1}U.$$

Si les valeurs propres de $R(\alpha)$ ne dépendent pas de α on dit que M est *ordonnée de façon consistente*. Notons que si M possède la propriété A alors $P\cdot M\cdot P^T$ est ordonné de façon consistente.

Pour cette classe de matrices il y a des formules simples relationant les valeurs propres de R_J, R_{GS} et $R_{SOR(\omega)}$:

THÉORÈME 3.3. *Soit A une matrice ordonnée de façon consistente et $\omega\neq 0$, alors*

(1) si μ est une valeur propre de R_J, alors $-\mu$ l'est aussi ;

(2) soit μ une des valeurs propres de R_J et λ tel que

$$(\lambda+\omega-1)^2=\lambda\omega^2\mu^2, \tag{35}$$

alors λ est une valeur propre de $R_{SOR(\omega)}$;

(3) soit $\lambda\neq 0$ une valeur propre de $R_{SOR(\omega)}$, alors il existe une valeur propre μ de R_J vérifiant l'équation (35).

Démonstration.

(1) Par le lemme 3.2, $R_J(1)=R_J$ et $R_J(-1)=-R_J$ ont les mêmes valeurs propres.

(2) Si $\lambda=0$ satisfait l'équation (35) alors $\omega=1$ et 0 est bien une valeur propre de

$$R_{SOR}(1)=R_{GS}=(\mathbf{1}_N-\overline{L})^{-1}\overline{U}$$

car U est singulière. Sinon

$$\begin{aligned}\det(\lambda\cdot\mathbf{1}_N-R_{SOR(\omega)}) &= \det\Big((\mathbf{1}_N-\omega\overline{L})(\lambda\cdot\mathbf{1}_N-R_{SOR(\omega)})\Big)\\ &= \det(\lambda(\mathbf{1}_N-\omega\overline{L})-(1-\omega)\mathbf{1}_N-\omega\overline{U})\\ &= \det((\lambda+\omega-1)\mathbf{1}_N-\omega\lambda\overline{L}-\omega\overline{U})\\ &= \det\left(\sqrt{\lambda}\omega\Big(\frac{\lambda+\omega-1}{\sqrt{\lambda}\omega}\mathbf{1}_N-\sqrt{\lambda}\overline{L}-\frac{1}{\sqrt{\lambda}}\overline{U}\Big)\right)\\ &= (\sqrt{\lambda}\omega)^n\det\Big(\frac{\lambda+\omega-1}{\sqrt{\lambda}\omega}\mathbf{1}_N-\overline{L}-\overline{U}\Big),\end{aligned}$$

où la dernière identité est conséquence du lemme 3.2. Donc si μ est une valeur propre de $\overline{L}+\overline{U}=R_J$ alors

$$\mu=\frac{\lambda+\omega-1}{\sqrt{\lambda}\omega}$$

pour $\lambda\in\mathrm{Spec}(R_{SOR(\omega)})$ et donc

$$\lambda\omega^2\mu^2=(\lambda+\mu-1)^2.$$

(3) Si $\lambda\neq 0$ l'antérieur est réversible.

□

Corollaire 3.4. *Soit A est ordonnée de façon consistente, alors $\rho(R_{GS})=\rho(R_J)^2$.*

Démonstration. Gauss-Seidel corresponds au paramètre de relaxation $\omega=1$, dans ce cas $\lambda\mu^2=\lambda^2$ d'où $\lambda=\mu^2$. □

Lemme 3.5. $\rho(R_{SOR(\omega)})\geq|\omega-1|$.

Donc $0\leq\omega\leq 2$ est nécessaire pour que SOR(ω) converge.

Démonstration. Soient $\lambda_j(\omega)$ $(1\leq j\leq N)$ les valeurs propres de $R_{SOR(\omega)}$, alors

$$\det(R_{SOR})=\det\Big((\mathbf{1}_N-\omega\overline{L})((1-\omega)\mathbf{1}_N+\omega\overline{U})\Big)=\det((1-\omega)\mathbf{1}_N+\omega\overline{U})=(1-\omega)^n$$

donc $\rho(R_{SOR(\omega)})\max_j|\lambda_j(\omega)|\geq|1-\omega|$. □

THÉORÈME 3.6. *Soit A une matrice ordonnée de façon consistente et dont les valeurs propres de R_J soient réelles et $\rho = \rho(R_J) < 1$, alors*

$$\omega_{\text{opt}} = \frac{2}{1+\sqrt{1-\rho^2}}$$

$$\rho(R_{SOR}(\omega_{\text{opt}})) = \omega_{\text{opt}} - 1 = \frac{\rho^2}{(1+\sqrt{1-\rho^2})^2}$$

$$\rho(R_{SOR}(\omega)) = \begin{cases} \omega - 1 & \text{pour } \omega_{\text{opt}} \le \omega \le 2, \\ 1 - \omega + \frac{1}{2}\omega^2\rho^2 + \omega\rho\sqrt{1-\omega+\frac{1}{4}\omega^2\rho^2} & \text{pour } 0 < \omega \le \omega_{\text{opt}}. \end{cases}$$

DÉMONSTRATION. Soit $\mu \in \text{Spec}(R_J)$, on résout

$$(\lambda + \omega - 1)^2 = \lambda\omega^2\mu^2$$

en λ. Après quelques calculs tedieux on trouve

$$\lambda^{\pm} = 1 - \omega + \frac{1}{2}\omega^2\mu^2 \pm \omega\mu\sqrt{1-\omega+\frac{1}{4}\omega^2\mu^2}.$$

Maintenant supposons $-1 < \mu < 1$, alors le discriminant est ≥ 0 si et seulement si

$$0 \le \omega \le \frac{2}{1+\sqrt{1-\mu^2}} = 2 - O(N^{-1})$$

alors $|\lambda^+| \ge |\lambda^-|$ et

$$|\lambda^+| = 1 - \omega + \frac{1}{2}\omega^2\mu^2 + \omega\mu\sqrt{1-\omega+\frac{1}{4}\omega^2\mu^2}.$$

Sinon,

$$|\lambda^+|^2 = |\lambda^-|^2 = (1 - \omega + \frac{1}{2}\omega^2\mu^2)^2 + \omega^2\mu^2\Big(1 - \omega + \frac{1}{4}\omega^2\mu^2\Big) = (\omega - 1)^2.$$

Le minimum est atteint pour $\omega = \omega_{\text{opt}} = \dfrac{2}{1+\sqrt{1-\mu^2}} = 2 - O(N^{-1})$. □

Pour le 2-Poisson

$$\omega_{\text{opt}} = \frac{2}{1+\sin\Big(\dfrac{\pi}{N+1}\Big)} = 2 - O(N^{-1}),$$

$$\rho(R_{SOR}(\omega_{\text{opt}})) = \frac{\cos^2\Big(\dfrac{\pi}{N+1}\Big)}{1+\sin\Big(\dfrac{\pi}{N+1}\Big))^2} = 1 - O(N^{-1}).$$

La figure suivante montre la convergence de la sur-relaxation successive pour le 2-Poisson et $N = 16$ **(il manque dessiner le graphe de la fonction)**.

On observe que le minimum est très étroit, et qu'à défaut d'une expression précise pour ω_{opt} il convient plutôt de prendre $\omega \to 2$.

Chapitre 7

Gradient et gradient conjugué

1. Introduction

Le gradient conjugué est la méthode itérative de choix si l'on veut résoudre le système linéaire

$$Ax = b \tag{36}$$

quand A est une grande matrice, symétrique réelle ou hermitienne, définie positive.

La méthode du gradient conjugué peut être encore améliorée si l'on utilise un préconditionnement.

La présentation de l'algorithme sous la forme d'une recette de cuisine qu'on trouve dans tous les livres n'est pas très intéressante. Ce qui est intéressant, c'est d'essayer de comprendre

— la construction de la méthode
— ses propriétés analytiques et algébriques
— l'interprétation géométrique
— le rapport entre gradient conjugué, optimisation et espaces de Krylov.

On rappelle qu'un espace de Krylov est un espace engendré par les itérés successifs $x_0, Ax_0, A^2x_0, \ldots, A^kx_0$, pour x_0 donné.

Il est rare de trouver l'ensemble de ces informations dans un même écrit ; le présent chapitre doit beaucoup au remarquable exposé de Jonathan Shewchuk [**34**] et du livre de Yousef Saad.

2. Le début : la méthode du gradient, pourquoi elle marche et pourquoi elle ne marche pas

La première observation est l'équivalence entre résolution de (36) et problème de minimisation ;

LEMME 2.1. *Soit A une matrice symétrique réelle définie positive $d \times d$, et soit b un vecteur de $\mathbb{R}^d$. On note par le signe T en exposant la transposition. Posons*

$$f(x) = \frac{x^{\mathsf{T}}Ax}{2} - b^{\mathsf{T}}x. \tag{37}$$

Le vecteur x est solution de (36)*, si et seulement s'il minimise* (37) *sur $\mathbb{R}^d$.*

DÉMONSTRATION. La fonction f est quadratique sur $\mathbb{R}^d$. Sa dérivée $f'(x)$ est une forme linéaire donnée par

$$f'(x)y = x^{\mathsf{T}}Ay - b^{\mathsf{T}}y = y^{\mathsf{T}}Ax - b. \tag{38}$$

Donc, si x minimise (37), alors il résout (36). Comme A est symétrique définie positive, elle est inversible, et par conséquent cette solution x est unique. Réciproquement, si y est n'importe quel vecteur de $\mathbb{R}^d$, et si x est la solution de (36), on a

$$f(y) = \frac{y^{\mathsf{T}}Ay}{2} - x^{\mathsf{T}}Ay$$

parce que $b^\mathsf{T} = x^\mathsf{T} A$,

$$= \frac{y^\mathsf{T} A y - 2x^\mathsf{T} A y + x^\mathsf{T} A x - x^\mathsf{T} A x}{2}$$

en mettant en évidence un carré,

$$= \frac{(y-x)^\mathsf{T} A (y-x)}{2} - \frac{x^\mathsf{T} A x}{2}$$

ce qui montre bien que x est l'unique minimiseur de (37). □

C'est le lemme 2.1 qui motive l'utilisation de la méthode de gradient pour résoudre (36). On va donc commencer par la méthode de gradient, qui est une méthode de descente parmi beaucoup d'autres.

Qu'est-ce qu'une méthode de descente ? On part d'un point x_0, on se fixe une direction de descente d_0, et on cherche sur la droite $\alpha \mapsto x_0 + \alpha d_0$ le minimum de f. Comme la fonction f est strictement (et même uniformément) convexe, il existe un unique minimum, qui vérifie

$$f'(x_0 + \alpha d_0) d_0 = 0,$$

et en vertu de (38), elle doit vérifier

$$d_0^\mathsf{T} \big(A(x_0 + \alpha_0 d_0) - b\big) = 0. \tag{39}$$

Notons

$$r_0 = b - A x_0,$$

qui est le *résidu* en x_0. Le résidu mesure la qualité d'une solution : s'il est petit, on peut espérer que x_0 n'est pas trop loin de la solution x. La relation (39) équivaut alors à

$$\alpha_0 d_0^\mathsf{T} A d_0 - d_0^\mathsf{T} r_0 = 0,$$

ce qui nous fournit la valeur de α_0 :

$$\alpha_0 = \frac{d_0^\mathsf{T} r_0}{d_0^\mathsf{T} A d_0}.$$

Donc, dans une méthode de descente, on aura

$$\alpha_0 = \frac{d_0^\mathsf{T} r_0}{d_0^\mathsf{T} A d_0}, \tag{40a}$$

$$x_1 = x_0 + \alpha_0 d_0, \tag{40b}$$

$$r_1 = r_0 - \alpha_0 A d_0. \tag{40c}$$

On remarque que le calcul de (40a) et de (40c) fait intervenir le même vecteur Ad_0. Il est commode de lui donner un nom et de le stocker. Avec cette modification, l'itération (40) devient

$$p_0 = A d_0, \tag{41a}$$

$$\alpha_0 = \frac{d_0^\mathsf{T} r_0}{p_0^\mathsf{T} d_0}, \tag{41b}$$

$$x_1 = x_0 + \alpha_0 d_0, \tag{41c}$$

$$r_1 = r_0 - \alpha_0 p_0. \tag{41d}$$

Il reste évidemment à choisir la direction d_0 : la méthode du gradient, ou de plus grande descente, consiste à prendre comme d_0 l'opposé du gradient de f en x_0, soit

$$d_0 = r_0.$$

Il est maintenant possible la méthode de gradient :

(42a) x_0 donné,

(42b) $r_0 \leftarrow b - Ax_0$,

(42c) tant que $r_j \neq 0$ et jusqu'à précision donnée,

(42d) $p_j \leftarrow Ar_j$,

(42e) $\alpha_j \leftarrow \dfrac{r_j^\mathsf{T} r_j}{p_j^\mathsf{T} r_j}$,

(42f) $x_{j+1} \leftarrow x_j + \alpha_j r_j$,

(42g) $r_{j+1} \leftarrow r_j - \alpha_j p_j$,

(42h) fin

Maintenant, nous allons démontrer que cette méthode est convergente et évaluer son taux de convergence. Le mieux serait d'évaluer une norme bien choisie de l'erreur $e_k = x_k - x$ ou du résidu r_k, et d'obtenir par exemple une estimation de cette norme à l'indice $k+1$ en fonction de la norme à l'indice k.

Si nous combinons (42d), (42e) et (42g), nous obtenons une récurrence entre r_k et r_{k+1} :

$$r_{k+1} = r_k - \frac{r_k^\mathsf{T} r_k}{r_k^\mathsf{T} A r_k} A r_k.$$

Le problème est de choisir la bonne norme. Nous allons voir sur un exemple en dimension $d = 2$ qu'il faut un peu réfléchir. Après changement de base, on peut toujours mettre A sous la forme

$$A = \begin{pmatrix} \lambda & 0 \\ 0 & \mu \end{pmatrix},$$

avec $0 < \lambda < \mu$. Supposons que nous choisissions la norme euclidienne de $\mathbb{R}^d$, c'est à dire $\sqrt{r_k^\mathsf{T} r_k}$. Calculons le carré scalaire de r_{k+1} :

$$r_{k+1}^\mathsf{T} r_{k+1} = r_k^\mathsf{T} r_k - 2\frac{r_k^\mathsf{T} r_k}{r_k^\mathsf{T} A r_k} r_k^\mathsf{T} A r_k + \frac{(r_k^\mathsf{T} r_k)^2}{(r_k^\mathsf{T} A r_k)^2} r_k^\mathsf{T} A^2 r_k,$$

et il y a une simplification évidente, ce qui donne

$$= r_k^\mathsf{T} r_k \left(\frac{(r_k^\mathsf{T} A^2 r_k)(r_k^\mathsf{T} r_k)}{(r_k^\mathsf{T} A r_k)^2} - 1 \right). \tag{43}$$

Ce qui serait bien, ce serait si le terme entre parenthèses, en facteur de r_k était strictement inférieur à 1. Nous allons voir qu'il n'en est rien : si nous choisissons

$$r_k = \begin{pmatrix} 1/\sqrt{2\lambda} \\ 1/\sqrt{2\mu} \end{pmatrix},$$

alors

$$r_k^\mathsf{T} A r_k = \frac{\lambda}{2\lambda} + \frac{\mu}{2\mu} = 1, \quad r_k^\mathsf{T} r_k = \frac{1}{2\lambda} + \frac{1}{2\mu}, \quad r_k^\mathsf{T} A^2 r_k = \frac{\lambda}{2} + \frac{\mu}{2}.$$

Par conséquent, avec ce choix particulier, notre terme entre parenthèses vaut

$$\frac{(\lambda+\mu)^2}{4\lambda\mu} - 1,$$

et quand le rapport μ/λ tend vers l'infini, ce terme devient arbitrairement grand...

Moralité : ce choix n'est pas le bon.

On peut essayer $r_k^\mathsf{T} A r_k$, mais c'est encore pire, parce que dans la formule donnant $r_{k+1}^\mathsf{T} r_{k+1}$ en fonction de $r_k^\mathsf{T} r_k$, on trouve les $r_k^\mathsf{T} A^n r_k$ avec n allant de 0 à 3, ce qui ne laisse pas beaucoup d'espoir.

On peut essayer une puissance négative ; calculons donc $r_{k+1} A^{-1} r_{k+1}$ en fonction de $r_k A^{-1} r_k$:

$$r_{k+1} A^{-1} r_{k+1} = r_k^\mathsf{T} A^{-1} r_k - 2 r_k^\mathsf{T} r_k \frac{r_k^\mathsf{T} r_k}{r_k^\mathsf{T} A r_k} + \frac{(r_k^\mathsf{T} r_k)^2}{(r_k^\mathsf{T} A r_k)^2} r_k^\mathsf{T} A r_k,$$

qui se simplifie en

$$r_{k+1}^\mathsf{T} r_{k+1} = r_k^\mathsf{T} A r_k \left(1 - \frac{(r_k^\mathsf{T} r_k)^2}{(r_k^\mathsf{T} A r_k)(r_k^\mathsf{T} A^{-1} r_k)} \right). \tag{44}$$

Le terme entre les grands parenthèses dans (44) est énormément plus sympathique que le terme analogue dans (43). En effet, on sait d'avance qu'il est positif ou nul, et en plus, comme il est de la forme 1 − terme positif, on voit qu'il est au plus égal à 1. Mais on peut mieux faire ; en effet, si x n'est pas nul, l'expression

$$g(x) = \frac{(x^\mathsf{T} x)^2}{(x^\mathsf{T} A x)(x^\mathsf{T} A^{-1} x)}$$

est homogène de degré 0 ; donc sa borne inférieure est égale à sa borne inférieure sur la sphère unité euclidienne, qui est un compact ; elle est donc atteinte, et strictement positive :

$$\beta = \inf_{x \neq 0} g(x) = \inf_{x^\mathsf{T} x = 1} g(x) > 0.$$

De là il résulte que la convergence est géométrique pour la méthode de descente, de taux $1 - \beta$.

Ceci étant, on peut faire mieux :

LEMME 2.2. *Soit A une matrice symétrique réelle, définie, positive, dont on note λ_1 la plus petite valeur propre et λ_d la plus grande. Alos, on a l'identité suivante :*

$$\beta = \inf \frac{(x^\mathsf{T} A x)(x^\mathsf{T} A^{-1} x}{(x^\mathsf{T} x)^2} = \frac{4 \lambda_1 \lambda_d}{(\lambda_1 + \lambda_d)^2},$$

et par conséquent

$$1 - \beta = \frac{(1 - \lambda_1/\lambda_d)^2}{(1 + \lambda_1/\lambda_d)^2}. \tag{45}$$

DÉMONSTRATION. Il revient au même de trouver la borne inférieure de $g(x)$, ou la borne supérieure de $1/g(x)$. Nous avons donc un problème de minimisation avec contraintes :

$$\text{maximiser } (x^\mathsf{T} A x)(x^\mathsf{T} A^{-1} x) \text{ sous la condition } x^\mathsf{T} x = 1.$$

Commençons par chercher les extrema de

$$x \mapsto (x^\mathsf{T} A x)(x^\mathsf{T} A^{-1} x) - \sigma x^\mathsf{T} x,$$

σ étant un multiplicateur de Lagrange. La condition d'extrêmalité s'écrit

$$(x^\mathsf{T} A^{-1} x) A x + (x^\mathsf{T} A x) A^{-1} x - \sigma x = 0,$$

et si on pose

$$x^\mathsf{T} A x = \rho, \quad x^\mathsf{T} A {-1} x = \tau,$$

alors x vérifie la relation

$$\rho A^2 x - \sigma A x + \tau x = 0;$$

par conséquent, x ne peut avoir de composantes que sur au plus deux vecteurs propres correspondant à des valeurs propres distinctes. Soient λ et μ ces valeurs

propres, et soient y et z les composantes respectives de x sur les vecteurs propres correspondants.

Maximisons

$$(\lambda y^2 + \mu z^2)(y^2/\lambda + z^2/\lambda)$$

sous la contrainte $y^2 + z^2 = 1$. Il suffit de maximiser sur $[0, 1]$

$$\begin{aligned}(\lambda Y + \mu(1-Y))(Y/\lambda + (1-Y)/\mu) &= Y^2 + (1-Y)^2 + (\lambda/\mu + \mu/\lambda)Y(1-Y) \\ &= 1 + \left(\frac{\lambda}{\mu} + \frac{\mu}{\lambda} - 2\right) Y(1-Y).\end{aligned}$$

Par conséquent, on trouve immédiatement que le maximum est au plus égal à

$$\frac{(\lambda+\mu)^2}{4\lambda\mu}, \tag{46}$$

et ce maximum est atteint comme on le voit en prenant

$$x = \begin{pmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{pmatrix}.$$

Si on maximise (46) par rapport à toutes les valeurs propres, on trouve

$$\frac{(\lambda_1 + \lambda_d)^2}{4\lambda_1\lambda_d},$$

et la conclusion (45) est claire. □

Remarque 2.3. Il est intéressant de remarquer que $r_k^{\mathsf{T}} A^{-1} r_k$ est égal à $(x_k - x)^{\mathsf{T}} A (x_k - x)$, ce qui veut dire qu'on a mesuré l'erreur dans une norme naturelle du problème : celle qui est définie par le produit scalaire associé à la matrice A.

Nous déduisons du lemme 2.2 que la méthode du gradient converge géométriquement avec un taux au plus égal à la valeur donnée par (45) ; il est intéressant de remarquer que pour $\kappa = \lambda_d/\lambda_1 \gg 1$, le taux de convergence est approximativement égal à $\exp(-4/\kappa)$. Par conséquent le nombre de pas nécessaire pour diviser la norme de $r_k^{\mathsf{T}} A^{-1} r_k$ par 2 est au plus égal à $(\kappa \log 2)/4$.

xxx *Ici, ou avant, faire des dessins pour faire comprendre la situation, et en particulier la lenteur de la convergence.*

3. Un meilleur choix de directions de descente

Visiblement, la méthode de descente est lente, parce qu'elle choisit toujours les mêmes directions de descente. Supposons qu'on puisse fabriquer aisément des directions de descente orthogonales les unes aux autres :

$$\text{si } j \neq k, \text{ alors } d_j^{\mathsf{T}} d_k = 0. \tag{47}$$

On va alors viser pour tuer successivement les composantes de l'erreur $e_k = x_k - x$ le long des d_j ; pour cela, il suffit d'imposer :

$$d_j^{\mathsf{T}} e_{j+1} = 0, \tag{48}$$

et par conséquent, comme $x_{j+1} = x_j + \alpha_j d_j$, cela revient à choisir α_j tel que

$$d_j^{\mathsf{T}}(e_j + \alpha_j d_j) = 0,$$

d'où l'on tire :

$$\alpha_j = -\frac{d_j^{\mathsf{T}} e_j}{d_j^{\mathsf{T}} d_j}. \tag{49}$$

Un argument par récurrence prouve que $d_i^{\mathsf{T}} e_j$ est nul pour tout $i = 0, \ldots, j-1$. Pour $j = 1$, cela résulte de (48) ; supposons la relation vraie jusqu'à un certain j, et

voyons ce qu'il en est pour $j+1$. La relation $d_j^\mathsf{T} e_{j+1}$ est vérifiée par construction. Pour $0 \leq i \leq j-1$, on a

$$d_i^\mathsf{T} e_{j+1} = d_i^\mathsf{T}(e_j + \alpha_j d_j);$$

en vertu de la relation de récurrence, $d_i^\mathsf{T} d_j$ est nul, et par ailleurs $d_i^\mathsf{T} d_j$ est nul par hypothèse.

Par conséquent la méthode ainsi esquissée convergerait en un nombre de pas au plus égal à la dimension d de l'espace. C'est parfait, sauf que...on ne peut pas calculer les e_j : si on pouvait calculer une seule des erreurs e_j, on obtiendrait immédiatement la solution ! Ce serait bien trop beau.

Donc il faut affiner l'idée, et réutiliser le fait que le problème possède une norme et un produits scalaires naturels, associés à la matrice A.

Au lieu d'utiliser des directions de descente orthogonales relativement au produit scalaire euclidien de $\mathbb{R}^d$, rien ne nous empêche d'essayer de travailler avec des directions de descentes A-orthogonales, c'est à dire au lieu de (47)

$$\text{si } j \neq k, \text{ alors } d_j^\mathsf{T} A d_k = 0. \tag{50}$$

On choisit $x_{j+1} = x_j + \alpha_j d_j$, et à chaque pas, on va essayer d'annuler le A-produit scalaire de e_{j+1} avec d_j :

$$d_j^\mathsf{T} A(x_j + \alpha_j d_j - x) = 0.$$

Mais on remarque que

$$A(x_j + \alpha_j d_j - x) = Ax_j - b + \alpha_j A d_j = \alpha_j A d_j - r_j.$$

En d'autres termes :

$$d_j r_{j+1}^\mathsf{T} = 0. \tag{51}$$

La condition qui définit α_j s'écrit à présent :

$$\alpha_j = \frac{d_j^\mathsf{T} r_j}{d_j^\mathsf{T} A d_j}. \tag{52}$$

Si nous comparons (49) et (52), nous voyons que maintenant, α_j peut être calculé à partir de la connaissance du résidu r_j.

Donc, si nous savons générer une suite de directions A-orthogonales d_j (on dit aussi A-conjuguées), l'algorithme

$$\begin{aligned}
&x_0 \text{ donné},\\
&r_0 \leftarrow b - Ax_0,\\
&\text{tant que } r_j \text{ n'est pas nul}\\
&\quad p_j \leftarrow Ad_j,\\
&\quad \alpha_j \leftarrow \frac{d_j^\mathsf{T} r_j}{p_j^\mathsf{T} d_j},\\
&\quad x_{j+1} \leftarrow x_j + \alpha_j d_j,\\
&\quad r_{j+1} \leftarrow r_j - \alpha_j p_j, \qquad \text{(53a)}\\
&\text{fin}
\end{aligned}$$

converge en au plus d itérations.

Maintenant, il reste à savoir comment générer les directions d_j, parce que pour le moment, tout ceci n'est guère pratique. La première idée consiste à prendre n'importe quelle base $v_0, \ldots, v_{d-1}$ de $\mathbb{R}^d$, et à l'orthogonaliser (relativement au A-produit scalaire) par la méthode de Gram-Schmidt.

Cela revient à écrire l'algorithme suivant :

$$
\begin{aligned}
& d_0 \leftarrow v_0, \\
\text{(54a)} \qquad & p_0 \leftarrow Ad_0, \\
& m_0 \leftarrow d_0^{\mathsf{T}} p_0, \\
& \text{pour } j \text{ de } 1 \text{ à } d-1, \\
& \quad d_j \leftarrow v_j, \\
& \quad \text{pour } i \text{ de } 0 \text{ à } j-1, \\
\text{(54b)} \qquad & \quad\quad \beta_{ji} \leftarrow p_i^{\mathsf{T}} v_j / m_i, \\
& \quad\quad d_j \leftarrow d_j - \beta_{ji} d_i, \\
& \quad \text{fin}, \\
\text{(54c)} \qquad & \quad p_j \leftarrow Ad_j, \\
\text{(54d)} \qquad & \quad m_j \leftarrow d_j^{\mathsf{T}} p_j, \\
& \text{fin}.
\end{aligned}
$$

Si on compte le nombre de multiplications matrice par vecteur dans l'algorithme (54), on en trouve aux lignes (54a) et (54c) ; en fait il y en a une en trop, et il aurait été mieux de programmer différemment le dernier pas, de façon à éviter le calcul de p_{d-1}, dont on n'a pas besoin. Donc nous avons $d-1$ multiplications matrice par vecteur, $d-1$ produits scalaires de p_i par d_i et $(d-1)(d-2)/2$ produits scalaires de p_i par v_j, soit un total de $d(d-1)/2$ produits scalaires, et enfin $(d-2)(d-1)/2$ opérations de la forme $v + \alpha w$. Si on ne tient pas compte des multiplications matrice vecteur, cela fait déjà $2d^3 + O(d^2)$ multiplications ou additions...ce qui est beaucoup plus que pour une élimination gaussienne, dont le coût est $2d^3/3 + O(d^2)$. De plus, orthogonaliser une base relativement au A produit scalaire, c'est tout à fait équivalent à résoudre : en effet, une fois qu'on a les d_i, il suffit de faire le produit scalaire de l'équation (36) par d_i, et l'on obtient

$$x = \sum_{j=0}^{d-1} \frac{d_j^{\mathsf{T}} b}{m_j} d_j,$$

les d_j étant comme dans (54d).

Ce compte peut être divisé par 2 si on choisit comme v_i les vecteurs de la base canonique, car dans ce cas-là, les produits scalaires de p_i par v_j ne coûtent rien. Mais ce n'est toujours pas compétitif par rapport à lélimination gaussienne.

Donc cet algorithme est très mauvais en terme de comtpes d'opérations. Si l'on se souvient en plus que le procédé d'orthogonalisation de Gram-Schmidt est très instable, on n'est guère avancé.

La raison pour laquelle cet algorithme marche si mal est le choix arbitraire des v_i. On peut faire mieux ! Au lieu d'orthogonaliser une base qui n'a pas de sens particulier pour le problème, on va orthogonaliser la base des r_j.

Bon, mais comment sait-on que les r_j forment une base ? On ne le sait pas, on le prouve :

LEMME 3.1. *Supposons que les d_i soient obtenus par orthogonalisation de la suite des r_i. Soit j le plus grand indice tel que r_i ne soit pas nul. Alors les résidus $r_0, \ldots, r_j$ sont orthogonaux relativement au produit scalaire canonique.*

DÉMONSTRATION. Supposons que r_0 n'est pas nul. Au premier pas, $d_0 = r_0$ et il résulte de (51) que $d_0^{\mathsf{T}} r_1 = r_0^{\mathsf{T}} r_1$ est nul. Supposons que jusqu'à l'indice $l < j$, la condition d'orthogonalité suivante soit remplie,

$$0 \leq i < k \leq l \Longrightarrow r_i^{\mathsf{T}} r_k = 0.$$

Calculons $r_i^\mathsf{T} r_{l+1}$ pour $i = 0, \ldots, l-1$; il résulte des relations (51) et (53a) que

$$r_i^\mathsf{T}(r_l - \alpha_l A d_l)$$

s'annule parce que $r_i^\mathsf{T} r_l$ est nul par hypothèse de résurrence et parce que d_l est orthgonal à tous les r_i pour $i \leq l$. Maintenant pour $i = l$, on doit calculer

$$r_l^\mathsf{T}(r_l - \alpha_l A d_l) = r_l^\mathsf{T} r_l - \frac{d_l^\mathsf{T} r_l}{d_l^\mathsf{T} A d_l} d_l^\mathsf{T} A r_l. \tag{55}$$

Dans le deuxième membre de (55), on constate que $d_l^\mathsf{T} A d_l$ et $d_l^\mathsf{T} A r_l$ coïncident, puisque la différence entre d_l et r_l est une combinaison de d_i pour $i \leq l-1$; comme ces d_i sont A-orthogonaux à d_l, on en déduit que $d_l^\mathsf{T} A d_l$ est égal à $d_l^\mathsf{T} A r_l$. De la même façon, $d_l - r_l$ est aussi une combinaison linéaire des r_i pour $i \leq l-1$, et donc, en vertu de l'hypothèse de récurrence, $r_l^\mathsf{T}(d_l - r_l)$ est nul. On voit à présent que l'expression (55) est nulle, ce qui démontre le lemme. □

Mais pour le moment, on n'a pas encore gagné le gros lot. C'est pour maintenant. Si nous reprenons le calcul des β_{ji}, donnés par (54b), nous avons besoin pour le faire de la valeur de $p_i^\mathsf{T} v_j = d_i^\mathsf{T} A d_j$; or, il résulte de la relation (53a) que

$$A d_i = \frac{r_i - r_{i+1}}{\alpha_i},$$

et par conséquent, pour $i < j-1$, β_{ji} est nul ; de plus, pour $i = j-1$,

$$\begin{aligned}
\beta_{j,j-1} &= \frac{(r_{j-1} - r_j)^\mathsf{T} r_j}{\alpha_{j-1} d_{j-1}^\mathsf{T} A d_{j-1}} \\
&= -\frac{r_j^\mathsf{T} r_j}{d_{j-1}^\mathsf{T} A d_{j-1}} \frac{d_{j-1}^\mathsf{T} A d_{j-1}}{d_{j-1}^\mathsf{T} r_{j-1}} \\
&= -\frac{r_j^\mathsf{T} r_j}{r_{j-1}^\mathsf{T} r_{j-1}},
\end{aligned}$$

puisque $d_{j-1}^\mathsf{T} r_{j-1}$ est égal à $r_{j-1}^\mathsf{T} r_{j-1}$.

Maintenant, le calcul des d_j devient très simple et nettement plus rapide que le calcul analogue pour une orthogonalisation à partir d'une base arbitraire :

$$\beta_{j+1} = \frac{r_{j+1}^\mathsf{T} r_{j+1}}{r_j^\mathsf{T} r_j},$$
$$d_{j+1} = r_{j+1} + \beta_{j+1} d_j.$$

Donc, la création de la suite des d_j à partir de la suite des d_j coûte $d-1$ produits scalaires et $d-1$ opérations de la forme $v + \beta w$, soit un total d'opérations de $4d^2 + O(d)$.

Écrivons l'algorithme complet du gradient conjugué :

$$
\begin{aligned}
&x^0 \text{ donné},\\
&r_0 \leftarrow b - Ax_0,\\
&d_0 \leftarrow r_0,\\
&p_0 \leftarrow Ad_0,\\
&m_{\text{ancien}} \leftarrow r_0^{\mathsf{T}} r_0,\\
&m_{\text{nouveau}} \leftarrow m_{\text{ancien}},\\
&\text{tant que } r_j \text{ n'est pas nul},\\
&\quad \alpha_j \leftarrow \frac{d_j^{\mathsf{T}} r_j}{d_j^{\mathsf{T}} p_j},\\
&\quad x_{j+1} \leftarrow x_j + \alpha_j d_j,\\
&\quad r_{j+1} \leftarrow r_j - \alpha_j p_j,\\
&\quad m_{\text{ancien}} \leftarrow m_{\text{nouveau}},\\
&\quad m_{\text{nouveau}} \leftarrow r_{j+1}^{\mathsf{T}} r_{j+1},\\
&\quad \beta_{j+1} \leftarrow \frac{m_{\text{nouveau}}}{m_{\text{ancien}}},\\
&\quad d_{j+1} \leftarrow r_{j+1} + \beta_{j+1} d_j,\\
&\text{fin}
\end{aligned}
\tag{56}
$$

Nous voyons à la lecture de (56) que nous devrons effectuer $d+1$ multiplications de la matrice A par un vecteur et $10d^2 + O(d)$ multiplications ou additions de nombres dans les autres étapes de l'algorithme. Par conséquent, si le coût d'une multiplication de A par un vecteur est petit devant d^2, ce qui sera vérifié pour une matrice creuse, et si nous notons qd le nombre d'opérations nécessaires, nous trouvons que le coût de l'algorithme (56) est $(q + 10)d^2 + o(d^2)$.

3.1. Gradient conjugé preconditionnée.

Exercice 7.1. ◁

(1) Soient $A, L \in \mathbb{K}^{N\times N}$ deux matrices symétriques définies positives en dimension d, réelle ou complexe ; on pose

$$\langle x, y\rangle_L = x^* Ly \text{ et } |x|_L = (x^* Lx)^{1/2}.$$

Montrer que $L^{-1}A$ est autoadjoint relativement au produit scalaire $(\cdot,\cdot)_L$.

(2) Calculer $\|B\|_L$ pour tout opérateur $B \in \mathbb{K}^{N\times N}$, en termes de norme d'opérateur $\|\ \|_2$ appliquée à des combinaisons convenables de B, son adjoint et de puissances appropriées de L. Calculer $\kappa_L(L^{-1}A) = \|L^{-1}A\|_L \|A^{-1}L\|_L$. On rappelle que toute matrice symétrique définie positive M possède pour chaque s réel une unique puissance M^s symétrique définie positive.

(3) On suppose qu'il existe deux nombres λ et μ tels que, pour tout vecteur x,

$$\lambda x^* Lx \le x^* Ax \le \mu x^* Lx.$$

Donner une estimation de $\kappa_L(L^{-1}A)$ en fonction de λ et μ. Indication : raisonner en termes de valeurs propres.

▷

EXERCICE 7.2. ◁ On rappelle l'algorithme du gradient conjugué :

$$
\begin{array}{l}
x_0 \text{ donné,} \\
r_0 \leftarrow b - Ax_0, \\
d_0 \leftarrow r_0, \\
p_0 \leftarrow Ad_0, \\
m_{\text{ancien}} \leftarrow r_0^* r_0, \\
m_{\text{nouveau}} \leftarrow m_{\text{ancien}}, \\
\text{tant que } r_j \text{ n'est pas nul,} \\
\quad q_j \leftarrow Ad_j, \\
\quad \alpha_j \leftarrow \dfrac{\langle r_j, r_j \rangle}{\langle d_j, p_j \rangle}, \\
\quad x_{j+1} \leftarrow x_j + \alpha_j d_j, \\
\quad r_{j+1} \leftarrow r_j - \alpha_j p_j, \\
\quad m_{\text{ancien}} \leftarrow m_{\text{nouveau}}, \\
\quad m_{\text{nouveau}} \leftarrow \langle r_{j+1}, r_{j+1} \rangle, \\
\quad \beta_{j+1} \leftarrow \dfrac{m_{\text{nouveau}}}{m_{\text{ancien}}}, \\
\quad d_{j+1} \leftarrow r_{j+1} + \beta_{j+1} d_j, \\
\text{fin}
\end{array}
\tag{57}
$$

Appliquer cet algorithme à l'équation

$$L^{-1}Ax = L^{-1}b$$

en remplaçant les produits scalaires canoniques par $(\cdot,\cdot)_L$. on prendra soin de mettre en évidence le vraie résidu

$$r_j = b - Ax_j.$$

Montrer que ceci devient l'algorithme du gradient conjugué preconditionné par L, equivalent (moyennant des changements de variables convenables) au gradient conjugué ordinaire appliqué à l'opérateur $L^{-1/2}AL^{-1/2}$. ▷

EXERCICE 7.3. ◁

(1) Montrer que l'algorithme de gradient conjugué préconditionné qu'on vient d'obtenir demande par pas une multiplication matrice-vecteur par A et une résolution de système de matrice L. En supposant que la multiplication de la matrice A par un vecteur demande kd opérations et que la résolution du système de matrice L demande $k'd\log d$ opérations, évaluer le nombre d'opérations par pas.

(2) On rappelle le résultat de convergence pour le gradient conjugué :

$$\|e_i\|_A \leq \left[\left(\frac{\sqrt{\kappa_2(A)}+1}{\sqrt{\kappa_2(A)}-1} \right)^i + \left(\frac{\sqrt{\kappa_2(A)}-1}{\sqrt{\kappa_2(A)}+1} \right)^i \right]^{-1} \|e_0\|_A.$$

Énoncer le résultat de convergence pour le gradient conjugué preconditionné en termes de $\|e_i\|_A$ et de $\kappa_2(L^{-1/2}AL^{-1/2})$.

(3) On se donne une suite de $n+1$ nombres strictement positifs $(b_i)_{0\leq i\leq n}$, et on pose $a_i = b_{i-1} + b_i$. Soit A la matrice donnée par

$$A = \begin{pmatrix} a_1 & -b_1 & 0 & 0 & \dots & 0 \\ -b_1 & a_2 & -b_2 & 0 & \dots & 0 \\ 0 & -b_2 & a_3 & -b_3 & \ddots & 0 \\ & \ddots & \ddots & \ddots & & \vdots \\ 0 & 0 & & & -b_n & a_n \end{pmatrix}.$$

Trouver une matrice L particulièrement simple, telle que les constantes α et β de l'exercice puissent être calculée explicitement en fonction des b_j. Si on résout le système de matrice L au moyen d'une transformation de Fourier rapide, quel est le nombre d'opérations nécessaires pour diviser l'erreur par 10^{-10} ? La norme de l'erreur sera prise aussi commodément que possible.

(4) Combien faut-il d'opérations pour diviser l'erreur par 10^{10} sous les hypothèses faites dans les questions précédentes ?

▷

Corrigé.

Exercice 7.4. ◁

(1) On a pour tous x et y de dimension d :

$$(x, L^{-1}Ay)_L = x^*LL^{-1}Ay = x^*Ay \text{ et } (L^{-1}Ax, y)_L = x^*(L^{-1}A)^*Ly = x^*Ay.$$

(2) La norme $\|B\|_L$ est par définition égale à

$$\|B\|_L = \sup_{x\neq 0} \frac{|Bx|_L}{|x|_L},$$

et par conséquent

$$\|B\|_L^2 = \sup_{x\neq 0} \frac{x^*B^*LBx}{x^*Lx};$$

si on pose $L^{1/2}x = y$, alors l'expression ci-dessus devient

$$\|B\|_L^2 = \sup_{x\neq 0} \frac{y^*L^{-1/2}BL^{1/2}L^{1/2}BL^{-1/2}y}{y^*y},$$

et par conséquent, $\|B\|_L = \|L^{1/2}BL^{-1/2}\|_2$.

On voit alors que

$$\|L^{-1}A\|_L = \|L^{-1/2}AL^{-1/2}\|_2$$

et que

$$\|A^{-1}L\|_L = \|L^{1/2}A^{-1}L^{1/2}\|_2.$$

Par conséquent, $\kappa_L(L^{-1}A) = \kappa_2(L^{-1/2}AL^{-1/2})$.

(3) Si on fait le changement de variable indiqué dans l'énoncé, alors

$$\forall y, \quad \lambda y^*y \leq y^*L^{-1/2}AL^{-1/2}y.$$

Par conséquent, les valeurs propres de l'opérateur $L^{-1/2}AL^{-1/2}$ sont toutes au moins égales à λ, et donc les valeurs propres de son inverse $L^{1/2}A^{-1}L^{1/2}$ sont toutes au plus égales à $1/\lambda$. On aura ainsi

$$y^*L^{1/2}A^{-1}L^{1/2}y \leq y^*y/\lambda;$$

si on fait le changement de variable $L^{1/2}y = x$, on obtient immédiatement la conclusion désirée.

(4) Puisque $L^{-1/2}AL^{-1/2}$ est autoadjoint pour le produit scalaire canonique, le supremum

$$\sup \frac{x^* L^{-1/2} A L^{-1/2} x}{x^* x}$$

est atteint, et est égal à la norme $\|L^{-1/2}AL^{-1/2}\|_2$. Par changement de variable $L^{-1/2}x = y$, il est aussi égal à

$$\sup_{y \neq 0} \frac{y^* A y}{y^* L y}$$

et en vertu de l'hypothèse de la question, il est au plus égal à μ. Le même raisonnement nous montre que

$$\|L^{1/2}A^{-1}L^{1/2}\|_2 = \sup \frac{y^* A^{-1} y}{y^* L^{-1} y}.$$

En utilisant le résultat (2), on conclut que $\|L^{1/2}AL^{1/2}\|_2$ est au plus égal à $1/\lambda$, et par conséquent

$$\kappa_L(L^{-1}A) \leq \mu/\lambda.$$

▷

EXERCICE 7.5. ◁ Écrivons le gradient conjugué pour $L^{-1}Ax = L^{-1}b$, en suivant les indications de l'énoncé :

x_0 donné,
$\widetilde{r}_0 \leftarrow L^{-1}(b - Ax_0)$,
$\widetilde{d}_0 \leftarrow \widetilde{r}_0$,
$\widetilde{q}_0 \leftarrow L^{-1}A\widetilde{d}_0$,
$m_{\text{ancien}} \leftarrow \widetilde{r}_0^* L \widetilde{r}_0$,
$m_{\text{nouveau}} \leftarrow m_{\text{ancien}}$,
tant que $\widetilde{r}_j$ n'est pas nul,
  $\widetilde{q}_j \leftarrow L^{-1}A\widetilde{d}_j$,
  $\alpha_j \leftarrow \dfrac{\widetilde{d}_j^* L \widetilde{r}_j}{\widetilde{d}_j^* L \widetilde{q}_j}$,
  $x_{j+1} \leftarrow x_j + \alpha_j \widetilde{d}_j$,
  $\widetilde{r}_{j+1} \leftarrow \widetilde{r}_j - \alpha_j \widetilde{q}_j$,
  $m_{\text{ancien}} \leftarrow m_{\text{nouveau}}$,
  $m_{\text{nouveau}} \leftarrow \widetilde{r}_{j+1}^* L \widetilde{r}_{j+1}$,
  $\beta_{j+1} \leftarrow \dfrac{m_{\text{nouveau}}}{m_{\text{ancien}}}$,
  $\widetilde{d}_{j+1} \leftarrow \widetilde{r}_{j+1} + \beta_{j+1}\widetilde{d}_j$,
fin

Comme l'énoncé demande de mettre en évidence le vrai résidu, on transforme cet algorithme en le faisant apparaître ; pour des raisons d'homogénéité, il est commode de

faire apparaître $q_j = A\widetilde{d}_j = \widetilde{q}_j$ au lieu de $\widetilde{q}_j$; on a alors :

x_0 donné,
$r_0 \leftarrow b - Ax_0$,
$\widetilde{d}_0 \leftarrow L^{-1}r_0$,
$q_0 \leftarrow A\widetilde{d}_0$,
$m_{\text{ancien}} \leftarrow r_0^*\widetilde{d}_0$,
$m_{\text{nouveau}} \leftarrow m_{\text{ancien}}$,
tant que r_j n'est pas nul,
$q_j \leftarrow A\widetilde{d}_j$,
$\alpha_j \leftarrow \dfrac{\widetilde{d}_j^* r_j}{\widetilde{d}_j^* q_j}$,
$x_{j+1} \leftarrow x_j + \alpha_j \widetilde{d}_j$,
$r_{j+1} \leftarrow r_j - \alpha_j q_j$,
$m_{\text{ancien}} \leftarrow m_{\text{nouveau}}$,
$s_{j+1} \leftarrow L^{-1}r_{j+1}$,
$m_{\text{nouveau}} \leftarrow r_{j+1}^* s_{j+1}$,
$\beta_{j+1} \leftarrow \dfrac{m_{\text{nouveau}}}{m_{\text{ancien}}}$,
$\widetilde{d}_{j+1} \leftarrow s_{j+1} + \beta_{j+1}\widetilde{d}_j$,
fin

Si on passe maintenant au même algorithme mais pour le problème autoadjoint pour le produit scalaire canonique

$$\hat{A}\hat{x} = L^{-1/2}b = \hat{b}, \text{avec } \hat{A} = L^{-1/2}AL^{-1/2},$$

le gradient conjugué va s'écrire

$\hat{x}_0$ donné,
$\hat{r}_0 \leftarrow \hat{b} - \hat{A}\hat{x}_0$,
$\hat{d}_0 \leftarrow \hat{r}_0$,
$\hat{q}_0 \leftarrow \hat{A}\hat{d}_0$,
$m_{\text{ancien}} \leftarrow \hat{r}_0^*\hat{r}_0$,
$m_{\text{nouveau}} \leftarrow m_{\text{ancien}}$,
tant que $\hat{r}_j$ n'est pas nul,
$\hat{q}_j \leftarrow \hat{A}\hat{d}_j$,
$\alpha_j \leftarrow \dfrac{\hat{d}_j^*\hat{r}_j}{\hat{d}_j^*\hat{q}_j}$,
$\hat{x}_{j+1} \leftarrow \hat{x}_j + \alpha_j\hat{d}_j$,
$\hat{r}_{j+1} \leftarrow \hat{r}_j - \alpha_j\hat{q}_j$,

$$
\begin{aligned}
&m_{\text{ancien}} \leftarrow m_{\text{nouveau}},\\
&m_{\text{nouveau}} \leftarrow \hat{r}_{j+1}^{*}\hat{r}_{j+1},\\
&\beta_{j+1} \leftarrow \frac{m_{\text{nouveau}}}{m_{\text{ancien}}},\\
&\hat{d}_{j+1} \leftarrow \hat{r}_{j+1} + \beta_{j+1}\hat{d}_j,
\end{aligned}
$$

fin

Les changements de variable $\hat{x}_j = L^{1/2}x_j$, $\hat{r}_j = L^{-1/2}r_j$, $\widetilde{d}_j = L^{-1/2}\hat{d}_j$ et $q_j = L^{1/2}\hat{q}_j$ permettent de retrouver l'algorithme précédent. ▷

EXERCICE 7.6. ◁

(1) Le fait qu'on a une multiplication de A par un vecteur se lit sur le système, ainsi que le nombre de résolution de systèmes de matrice L. Il y a en plus : 3 produits scalaires et trois opérations de la forme $x + \gamma y$ avec x et y des vecteurs, ainsi que deux divisions, qu'on négligera. Si on compte $2d + O(1)$ opérations pour les produits scalaires et pour les opérations $x + \gamma y$, on voit qu'il y a par pas $d(k + 10 + k' \log d)$ opérations par pas.

(2) Estimons l'erreur pour le schéma autoadjoint :

$$\|\hat{e}_i\|^2_{L^{-1/2}AL^{-1/2}} = \hat{e}_i^{*} L^{-1/2} A L^{-1/2} \hat{e}_i,$$

et en vertu du changement de variable

$$\hat{e}_i = L^{1/2} e_i,$$

cette norme d'erreur est en fait $\|e_i\|_A$. On a donc, pour le gradient conjugué préconditionné

$$
\begin{aligned}
\|e_i\|_A &\leq \left(\frac{1 - 1/\sqrt{\kappa_2(L^{-1/2}AL^{-1/2})}}{1 + 1/\sqrt{\kappa_2(L^{-1/2}AL^{-1/2})}} \right)^i \|e_0\|_A\\
&\sim \exp\bigl(-2i/\sqrt{\kappa_2(L^{-1/2}AL^{-1/2})}\bigr)\|e_0\|_A.
\end{aligned}
$$

(3) Il résulte de l'analyse de la première question que $\kappa_2(L^{-1/2}AL^{-1/2}) \leq \mu/\lambda$. Par conséquent, pour que l'erreur soit divisée par 10^{-10}, il suffit que $i \geq 5(\log 10)\sqrt{\mu/\lambda}$. On aura donc besoin de $5d(k + 10 + k' \log d)(\log 10)\sqrt{\mu/\lambda}$ opérations pour diviser l'erreur par 10^{-10}.

▷

Chapitre 8

Méthodes multigrille

La méthode multigrille se propose de corriger les défauts des méthodes itératives classiques pour des systèmes provenant de la discrétisation d'équations aux dérivées partielles. C'est une méthode itérative récursive, la récurrence étant effectuée sur l'échelle de discrétisation spatiale.

Elle s'applique à la discrétisation d'équations aux dérivées partielles elliptiques, et plus généralement à des problèmes de réseau, des problèmes structurels et beaucoup d'autres. Ici nous nous restreindrons au problème modèle le plus simple, l'équation de Poisson en dimension 1 avec des conditions de Dirichlet nulles :

$$-u_{xx} = f \quad \text{pour } x \in \Omega = [0,1] \text{ et } \quad u(0) = u(1) = 0.$$

On discrétise cette équation avec des différences finies. Pour $N \in \mathbb{N}$ on considère la grille

$$\Omega_h := \{jh : j = 1, \ldots, N-1\}$$

des nodes *à l'intérieur* de l' l'intervalle $[0,1]$ divisé en N sous-intervalles, et on pose $\mathcal{G}(\Omega_h); \Omega_h \to \mathbb{R}$ pour l'ensemble des fonctions réelles de cet ensemble. L'opérateur discret qui en résulte est

$$L_h : \mathcal{G}(\Omega_h) \to \mathcal{G}(\Omega_h) \quad , \quad L_h(u)(x) = h^{-2}\Big(u(x-h) - 2u(x) + u(x+h)\Big)$$

avec la convention $u(0) = u(1) = 0$. Ceci est une approximation d'ordre 2 de l'opérateur laplacien

$$-u_{xx} - L_h(u) = O(h^2)$$

pour $h \to 0$ et des fonctions u suffisamment régulières (par exemple $u \in \mathcal{C}^4(\Omega)$). Alternativement on peut l'écrire en *notation stencil* comme

$$L_h = h^{-2}[-1 \quad 2 \quad -1]_h.$$

L'équation approchée $L_h u_h = f_h$ se traduit dans le système linéaire d'ordre $(N-1) \times (N-1)$

$$h^{-2}\begin{bmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{bmatrix}\begin{bmatrix} u(h) \\ u(2h) \\ \vdots \\ u((N-2)h) \\ u((N-1)h) \end{bmatrix} = \begin{bmatrix} f(h) \\ f(2h) \\ \vdots \\ f((N-2)h) \\ f((N-1)h) \end{bmatrix}.$$

Alternativement on écrira L_N, u_N, f_N pour L_h, u_h, f, h respectivement.

L'idée principale de la méthode multigrille est que certaines itérations classiques, telles les itérations de Jacobi amorties agissent comme des filtres passe-bas : elles amortissent beaucoup les hautes fréquences, alors que les basses fréquences changent peu en une itération ; donc, si nous voulons réduire efficacement le résidu, après l'avoir débarrassé de ses hautes fréquences, nous ferons une correction sur une grille plus grossière, ayant un pas d'espace deux fois plus grand.

Dans la méthode bigrille, nous supposons N pair ; nous effectuons une itération de Jacobi amortie avec un paramètre convenablement choisi sur la grille fine $h = 1/N$; nous relevons le résidu sur la grille grossière de pas $H = 2h = 2/N$,

nous résolvons l'équation sur la grille grossière, nous interpolons cette solution sur la grille fine pour obtenir une correction.

Le plus beau de la méthode bigrille est que le rayon spectral de la matrice des itérations ne dépend pas du pas d'espace. Cependant, ce n'est pas encore un schéma pratique puisque la résolution sur grille grossière reste coûteuse. La solution est d'approcher la solution sur la grille grossière, et la méthode de choix est encore la bigrille, appliquée sur la grille de pas H et $2H$.

Si nous supposons que N est une puissance de 2, le principe de la méthode multigrille est très simple : au lieu de résoudre sur la grille grossière, nous faisons un ou plusieurs balayages de Jacobi, et nous corrigeons sur une grille encore plus grossière sur laquelle nous exécutons un balayage, et ainsi de suite, jusqu'à atteindre une grille très simple, par exemple une grille à un point sur laquelle la résolution est triviale ; nous interpolons alors successivement les corrections sur toutes les grilles plus fines, tout en faisant éventuellement des balayages supplémentaires à chaque passage.

Diverses combinaisons sont possibles et les méthodes multigrille sont encore un sujet actif de recherche : ce sont des objets fascinants, dont les idées sont proche de celles de la transformation de Fourier rapide et des algorithmes d'ondelettes.

Plusieurs livres traitent la méthode multigrille ; une liste non exhaustive est [**20, 37, 26, 4, 10**].

Dans cette section nous introduirons les idées de base de cette méthode en utilisant le le problème modèle comme guide. Nous traiterons d'abord la description de la méthode bigrille et on démontrera son taux de convergence indépendant du pas de l'espace. Ensuite on décrira les méthodes multigrille et multigrille complète, dont on démontrera la propriété remarquable de calculer une approximation du même ordre que l'erreur de discrétisation en temps quasi-optimal $O(N\log(N))$.

1. Spectre d'une matrice de différences finies

On pose

$$L_N = h^{-2}(2\mathbf{1}_{N-1} - B_N) = N^2(2\mathbf{1}_{N-1} - B_N)$$

avec

$$B_N = \begin{bmatrix} 0 & 1 & & & \\ 1 & 0 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & 0 & 1 \\ & & & 1 & 0 \end{bmatrix} \in \mathbb{R}^{(N-1)\times(N-1)}.$$

La détermination des vecteurs et valeurs propres de B_N équivaut à résoudre le système

$$\begin{aligned} v_2 &= \lambda v_1 \\ v_1 + v_3 &= \lambda v_2 \\ v_2 + v_4 &= \lambda v_3 \\ &\vdots \\ v_{N-3} + v_{N-4} &= \lambda v_{N-2} \\ v_{N-2} &= \lambda v_{N-1} \end{aligned}$$

pour $v = (v_1, \ldots, v_{N-1}) \in \mathbb{C}^{N-1} \setminus \{0\}$ et $\lambda \in \mathbb{C}$. Ceci qui corresponds à résoudre la récurrence linéaire

$$v_{j+1} - \lambda v_j + v_{j-1} = 0.$$

L'équation caractéristique est

$$z^2 - \lambda z + 1 = 0$$

et notons z_1, z_2 les racines de cette équation. Le discriminant est $\lambda^2 - 4$, donc si $\lambda = \pm 2$ alors $z_1 = z_2 = 1$ et donc

$$v_j = a + bj$$

pour des certains $a, b \in \mathbb{C}$. Or $v_0 = a = 0$ et $v_N = a + bN = 0$ entraînent $a = b = 0$ et par conséquent $v = 0$. Supposons alors $\lambda \neq \pm 2$, alors $z_1 \neq z_2$ et donc

$$v_j = az_1^j + bz_2^j$$

pour des certains $a, b \in \mathbb{C}$. La condition $v_0 = a + b = 0$ entraîne $b = -a$ et puisque on cherche des solutions non triviales, la deuxième condition

$$v_N = az_1^N + bz_2^N = a(z_1^N - z_2^N) = 0$$

entraîne $z_1^{2N} = 1$ (car $z_1 z_2 = 1$). On en déduit que les couples des différentes solutions possibles sont

$$(z_{1,\ell}, z_{2,\ell}) = (e^{\pi i \ell/N}, e^{-\pi i \ell/N}) \quad , \quad \ell = 0, \dots, N-1.$$

Le vecteur propre associé est (modulo la choix du facteur scalaire)

$$(v_\ell)_j = \frac{1}{2i}(z_1^j - z_2^j) = \sin(\frac{\pi \ell}{N} j), \quad , \quad j = 1, \dots, N-1.$$

pour $\ell = 1, \dots, N-1$, car $\ell = 0$ donne la solution triviale $v_\ell = 0$. On écrit cette vecteur comme une *fonction* propre

$$\varphi_\ell : \mathcal{G}(\Omega_N) \to \mathcal{G}(\Omega_h) \quad , \quad x \mapsto \sin(\pi \ell x).$$

La valeur propre correspondante est

$$\lambda_\ell(B_N) = z_1 + z_2 = 2\cos(\frac{\pi \ell}{N}) \quad , \quad \ell = 1, \dots, N-1,$$

tandis que pour L_N

$$\lambda_\ell(L_N) = N^2(2 - \lambda_\ell(B_N)) = N^2 \left(2 - 2\cos(\frac{\pi \ell}{N})\right) = 4N^2 \sin^2(\frac{\pi \ell}{N}).$$

2. Itération de Jacobi amortie

Le système à résoudre est $N^2(2\mathbf{1}_{N-1} - B_N)u_N = f_N$. Posons

$$u' := \frac{1}{2} B_N u^m + \frac{1}{2N^2} f_N,$$

l'*itération de Jacobi* consiste à faire $u^{m+1} = u'$. L'*itération de Jacobi amortie* ou ω-*Jacobi* est la moyenne pondérée de l'itération de Jacobi standard avec l'approximation précédente :

$$u^{m+1} = (1-\omega)u^m + \omega u' = \left((1-\omega)\mathbf{1}_{N-1} + \frac{\omega}{2} B_N\right) u^m + \frac{\omega}{2N^2} f_N. \tag{58}$$

L'opérateur du ω-Jacobi est $J_{N,\omega} = (1-\omega)\mathbf{1}_{N-1} + \frac{\omega}{2} B_N$ avec valeurs propres

$$\lambda_\ell(J_{N,\omega}) = 1 - \omega + \frac{\omega}{2}\lambda_\ell(B_N) = 1 - \omega \left(1 - \cos(\frac{\pi \ell}{N})\right) = 1 - 2\omega \sin^2(\frac{\pi \ell}{2N})$$

pour $\ell = 1, \dots, N-1$. Donc pour $0 < \omega \leq 1$ le rayon spectral est

$$\rho(J_{N,\omega}) = 1 - 2\omega \sin^2(\frac{\pi}{2N}) = 1 - O(h^2)$$

et de fait est minimal pour $\omega = 1$; par contre pour $\omega > 1$ on a $\rho(J_\omega) > 1$ pour $h \to 0$ et donc la méthode n'est pas convergente. Par contre le *facteur de lissage* est sensiblement meilleur. Posons

$$\mu(h;\omega) := \max\{\lambda_\ell(J_{N,\omega}) : \frac{N}{2} \leq \ell \leq N-1\},$$
$$\mu(\omega) := \overline{\lim}_{h\to 0}\mu(h;\omega).$$

On a

$$\mu(h;\omega) = \max\{|1-\omega|, |1-2\omega\sin^2(\frac{\pi(N-1)}{2N})|\}$$

et donc

$$\mu(\omega) = \max\{|1-\omega|, |1-2\omega|\}.$$

Le minimum se réalise en $\omega = 2/3$: $J_{N,2/3}$ est donc un filtre passe-bas qui amorti les hautes fréquences d'un facteur $1/3$.

3. Méthode bigrille

On introduit le schéma générale d'une itération basée sur la résolution approchée de l'équation du résidu. Pour une approximation u^m d'une équation linéaire $Lu = f$, l'*erreur* et le *résidu* sont respectivement

$$e^m = u - u^m \quad , \quad r^m = f - Lu^m.$$

L'erreur est zéro si et seulement si le résidu l'est aussi (si L est inversible) mais par contre le résidu peut être petit sans que l'erreur le soit tellement. La relation entre erreur et résidu est quantifiée par la notion de nombre de conditionnement.

L'*équation du résidu*

$$Le^m = r^m,$$

est équivalente à l'équation originale puisque $u = u^m + e^m$. Le procédé est résumé par le diagramme

$$u^m \to r^m = f - Lu^m \to e^m = L^{-1}r^m \to u = u^m + e^m.$$

Bien entendu l'erreur est aussi inaccessible que la solution elle-même, et ce procédé n'est nullement un algorithme. Cependant, si l'on substitue L par un opérateur $\widehat{L}$ proche de L mais plus simple à inverser, alors

$$\widehat{e}^m = \widehat{L}r^m$$

fournira une nouvelle approximation

$$u^{m+1} = u^m + \widehat{e}^m.$$

Cette modification donne une méthode itérative

$$u^m \to r^m = f - Lu^m \to \widehat{e}^m = \widehat{L}^{-1}r^m \to u^{m+1} = u^m + \widehat{e}^m;$$

de façon plus compacte

$$u^{m+1} = (\mathbf{1} - \widehat{L}^{-1}L)u^m + \widehat{L}^{-1}f.$$

Une des idées de la méthode bigrille consiste à observer qu'après un ou plusieurs balayages des hautes fréquences *via* un certain opérateur de lissage $\mathcal{S}_h$, on sait que l'erreur devient assez régulier. La résolution de l'équation du résidu sur une grille plus grossière Ω_H sera alors une bonne approximation de l'erreur. Voici la description sommaire du *cycle bigrille* :

(1) Lissage :
   (a) $\widehat{u}_h^m \leftarrow \mathcal{S}_h^{(\nu)}(u_h^m, L_h, f_h)$;
(2) Correction *via* la grille grossière :
   (a) Calcul du résidu : $d_h^m \leftarrow f_h - h^{-2}L_h\widehat{u}_h^m$;

(b) Restriction à Ω_H : $d_H^m \leftarrow R_h^H d_h^m$;
(c) Résolution sur Ω_H : $e_H^m \leftarrow L_H^{-1} d_H^m$;
(d) Interpolation sur Ω_h : $e_h^m \leftarrow P_H^h e_H^m$;
(e) Correction de l'approximation : $u_h^{m+1} \leftarrow \widehat{u}_h^m + e_h^m$.

On pourra visualiser mieux la structure de ce cycle avec le diagramme

$$\begin{array}{lllll} u_h^m \to \widehat{u}_h^m \to & d_h^m = f_h - h^{-2} L_h \widehat{u}_h^m & & e_h^m & \longrightarrow u_h^{m+1} = \widehat{u}_h^m + e_h^m \\ & \downarrow & & \uparrow & \\ & d_H^m & \longrightarrow e_H^m = (H^{-2} L_H)^{-1} d_H^m & & \end{array}$$

dont le premier niveau corresponds à la grille fine Ω_h et le deuxième à la grille grossière Ω_H ; les passages se font avec les opérateurs de restriction et d'interpolation. Le cycle bigrille dépends de plusieurs composantes :

— l'opérateur de lissage $\mathcal{S}_h : \mathcal{G}(\Omega_h) \to \mathcal{G}(\Omega_h)$;
— le nombre ν des pas de lissage ;
— la grille grossière Ω_H ;
— l' opérateur de restriction $R_h^H : \mathcal{G}(\Omega_h) \to \mathcal{G}(\Omega_H)$;
— l'opérateur L_H sur la grille grossière ;
— l' opérateur d'interpolation $P_H^h : \mathcal{G}(\Omega_H) \to \mathcal{G}(\Omega_h)$.

Pour notre petite équation de Poisson en dimension 1, le lissage sera fait par le 1/2-Jacobi

$$\mathcal{J}_N(u) = J_N u + \frac{1}{4N^2} f_N$$

dont l'opérateur est $J_N = \mathbf{1}_{N-1}/2 + B_N/4 \in \mathbb{R}^{(N-1)\times(N-1)}$. Nous supposerons dorénavant N pair, et nous posons $h = 1/N$ et $H = 2h = 2/N$.

L'opérateur d'*interpolation bilinéaire* ou de prolongation est

$$P_{N/2}(e)(x) = P_H^h(e)(x) = \begin{cases} e(x) & \text{si } x \in \Omega_H ; \\ \dfrac{1}{2}(e(x-h) + e(x+h)) & \text{si } \in \Omega_h \setminus \Omega_H \end{cases}$$

pour $e \in \mathcal{G}(\Omega_H)$ (on suppose $e(0) = e(1) = 0$). Pour l'*opérateur de restriction* on pourrai choisir la restriction standard des fonctions puisque Ω_H est un sous-ensemble de Ω_h. Or, il s'avère plus convenable de prendre en compte les valeurs au points voisins ; pour $d \in \mathcal{G}(\Omega_h)$ on pose

$$R_N(d)(x) = R_h^{2h}(d)(x) = \frac{1}{4}\Big(d(x-h) + 2d(x) + d(x+h)\Big) \quad , \quad x \in \Omega_h \tag{59}$$

dans cette définition on suppose aussi $d(0) = d(1) = 0$. Ceci est dual de l'interpolation bilinéaire : $R_N = P_{N/2}^*/2$.

L'itération bigrille s'écrit de façon plus compacte comme

$$u_N^{m+1} = \mathcal{J}_N^{(\nu)}(u_N^m) + P_{N/2} L_{N/2}^{-1} R_N (f_N - L_N \mathcal{J}_N^{(\nu)})(u_N^m);$$

l'*opérateur bigrille* est donc

$$T_N = (\mathbf{1}_{N-1} - P_{N/2} L_{N/2}^{-1} R_N L_N) J_N^{(\nu)} \in \mathbb{R}^{(N-1)\times(N-1)}.$$

Exercice 8.1. ◁

(1) Montrer que la composition de deux méthodes itératives consistantes pour un même système $Ax = b$, est aussi consistante.
(2) Donner un exemple de deux itérations convergentes pour un même système, dont la composition ne l'est pas.

Remarque : Noter cependant que si les deux itérations sont de rayon spectrale < 1, alors il est de même pour la composition, qui sera alors convergente.

▷

Le cycle bigrille est donc un schéma consistent puisque composition de deux schémas consistants : le lissage et la correction *via* la grille grossière.

4. Taux de convergence de la méthode bigrille

Ou ce qui est équivalent, on calcule le rayon spectral de T_N :

THÉORÈME 4.1. $\rho(T_{N-1}) = 1/2$.

EXERCICE 8.2. ◁ Montrer que

$$\sum_{j=1}^{N-1} \sin\left(\frac{\pi\ell}{N}j\right)\sin\left(\frac{\pi m}{N}j\right) = \begin{cases} N/2 & \text{si } \ell = m, \\ 0 & \text{si non.} \end{cases}$$

▷

On en déduit que

$$(2/N)^{1/2}\varphi_\ell \quad , \quad \ell = 1, \ldots, N-1$$

est une base orthonormale de $\mathbb{R}^{N-1}$. La matrice Q faite des $(2/N)^{1/2}\varphi_\ell$ comme les colonnes, est unitaire et donc

$$\rho(T_N) = \rho(QT_NQ^*) \quad , \quad ||T_N||_2 = ||QT_NQ^*||_2.$$

Le reste de cette section est dédié à la démonstration de ce résultat. découpera l'espace dans des sous-espaces T_N-invariants

$$E_\ell = \text{Vect}(\varphi_\ell, \varphi_{N-\ell} \quad , \quad \ell = 1, \ldots, N/2.$$

Pour $\ell = N/2$ on a $\dim(E_{N/2}) = 1$, autrement $\dim(E_\ell) = 2$.

Considérons d'abord l'action du 1/2-Jacobi $J_N = \mathbf{1}_{N-1}/2 + B_N/4 \in \mathbb{R}^{(N-1)\times(N-1)}$; le valeur propre correspondant à φ_ℓ est

$$1 - \sin^2\left(\frac{\pi\ell}{2N}\right) = \cos^2\left(\frac{\pi\ell}{2N}\right) \quad , \quad \ell = 1, \ldots, N-1.$$

On se souvient que les valeurs propres de L_N sont

$$4N^2\sin^2\left(\frac{\pi\ell}{2N}\right) \quad , \quad \ell = 1, \ldots, N-1.$$

Considérons maintenant l'interpolation : soit $1 \le \ell \le N/2 - 1$ et posons $\alpha_\ell = \pi\ell/N$ et $\varphi_\ell^{N/2}$) pour les fonctions propres correspondant à la grille $\Omega_{N/2}$, alors

$$P_N(\varphi_\ell^{N/2}) = \Big(\frac{1}{2}\sin(\alpha_\ell 0) + \frac{1}{2}\sin(\alpha_\ell 2), \sin(\alpha_\ell 2), \frac{1}{2}\sin(\alpha_\ell 2) + \frac{1}{2}\sin(\alpha_\ell 4), \ldots,$$
$$\sin(\alpha_\ell(N-2)), \frac{1}{2}\sin(\alpha_\ell(N-2)) + \frac{1}{2}\sin(\alpha_\ell N)\Big).$$

On a

$$\varphi_{N-\ell}(x) = \sin\left(\frac{\pi(N-\ell)}{N}x\right)$$
$$= \sin(\pi x - \alpha_\ell x) = \sin(\pi x)\cos(\alpha_\ell x) - \cos(\pi x)\sin(\alpha_\ell x)$$

donc

$$\varphi_{N-\ell}(x) = \begin{cases} -\sin(\alpha_\ell) & \text{si } x \in \Omega_{N/2}, \\ \sin(\alpha_\ell) & \text{si } x \in \Omega_N \setminus \Omega_{N/2}. \end{cases}$$

On a aussi

$$\frac{1}{2}\sin(\alpha_\ell(2j)) + \frac{1}{2}\sin(\alpha_\ell(2j+2)) = \cos(\alpha_\ell)\sin(\alpha_\ell(2j+1)) = \cos(\alpha_\ell)\varphi_\ell((2j+1)h)$$

donc

$$P_N(\varphi_\ell^{N/2}) = \frac{\cos(\alpha_\ell)+1}{2}\varphi_\ell + \frac{\cos(\alpha_\ell)-1}{2}\varphi_{N-\ell} = \cos^2(\alpha_\ell/2)\varphi_\ell - \sin^2(\alpha_\ell/2)\varphi_{N-\ell}.$$

L'opérateur de restriction est

$$R_N(\varphi_\ell)(x) = \frac{1}{4}\Big(\varphi_\ell(x-h) + 2\varphi_\ell(x) + \varphi_\ell(x+h)\Big);$$

sa matrice est

$$\frac{1}{4}\begin{bmatrix} 1 & 2 & 1 & & & & & \\ & & 1 & 2 & 1 & & & \\ & & & & & \ddots & & \\ & & & & & & 1 & 2 & 1 \end{bmatrix} = P_N^*/2 = \frac{1}{4}\begin{bmatrix} 1 & & & \\ 2 & & & \\ 1 & 1 & & \\ & 2 & & \\ & 1 & & \\ & & \ddots & \end{bmatrix}.$$

Par dualité on a

$$\langle R_N(\varphi_\ell), \varphi_m^{N/2}\rangle = \langle \varphi_\ell, \frac{1}{2}P_N^*(\varphi_m^{N/2})\rangle = \langle \varphi_\ell, \frac{1}{2}\Big(\cos^2(\alpha_m/2)\varphi_m - \sin^2(\alpha_m/2)\varphi_{N-m}\Big)\rangle$$

donc pour $\ell = 1, \ldots, N/2-1$ on a

$$\langle R_N(\varphi_\ell), \varphi_m^{N/2}\rangle = \frac{1}{2}\cos^2(\alpha_\ell)||\varphi_\ell||^2 = \cos^2(\alpha_\ell)||\varphi_\ell^{N/2}||^2$$

pour $m = \ell$ et $= 0$ sinon, tandis que pour $\ell = N/2+1, \ldots, N-1$ on a

$$\langle R_N(\varphi_\ell), \varphi_m^{N/2}\rangle = -\frac{1}{2}\sin^2(\alpha_\ell)||\varphi_\ell||^2 = -\sin^2(\alpha_\ell)||\varphi_\ell^{N/2}||^2$$

pour $m = N-\ell$ et $= 0$ sinon. En conclusion

$$R_N(\varphi_\ell) = \begin{cases} \cos^2(\alpha_\ell) & \text{pour } \ell = 1, \ldots, N/2-1, \\ -\sin^2(\alpha_\ell) & \text{pour } \ell = N/2+1, \ldots, N-1. \end{cases}$$

Pour $\ell = N/2$ on a $\dim(E_{N/2}) = 1$ et dans ce cas $R_N(\varphi_{N/2}) = 0$ et donc

$$T_N(\varphi_{N/2}) = J_N(\varphi_{N/2}) = \frac{1}{2}\varphi_{N/2}. \tag{60}$$

Écrivons $T_N = K_N J_N$ avec $K_N = \mathbf{1}_{n-1} - P_N L_{N/2} R_N L_N$. Pour $1 \le \ell \le N/2-1$ on a

$$\begin{aligned} K_N(\varphi_\ell) &= \varphi_\ell - P_N L_{N/2}^{-1} R_N L_N(\varphi_\ell) \\ &= \varphi_\ell - 4N^2\sin^2(\alpha_\ell/2) P_N L_{N/2}^{-1} R_N(\varphi_\ell) \\ &= \varphi_\ell - \frac{4N^2\sin^2(\alpha_\ell/2)\cos^2(\alpha_\ell/2)}{4(N/2)^2\sin^2(\alpha_\ell)} P_N(\varphi_\ell^{N/2}) \\ &= \varphi_\ell - (\cos^2(\alpha_\ell)\varphi_\ell - \sin^2(\alpha_\ell)\varphi_{N-\ell}) \\ &= \sin^2(\alpha_\ell)(\varphi_\ell + \varphi_{N-\ell}). \end{aligned}$$

Un calcul similaire montre que

$$K_N(\varphi_{N-\ell}) = \cos^2(\alpha_\ell)(\varphi_\ell + \varphi_{N-\ell}).$$

En définitive

$$\begin{bmatrix} K_N\varphi_\ell \\ K_N\varphi_{N-\ell} \end{bmatrix} = \begin{bmatrix} \sin^2(\alpha_\ell) & \sin^2(\alpha_\ell) \\ \cos^2(\alpha_\ell) & \cos^2(\alpha_\ell) \end{bmatrix} \begin{bmatrix} \varphi_\ell \\ \varphi_{N-\ell} \end{bmatrix}$$

et donc

$$T_{N,\ell} = \begin{bmatrix} T_N\varphi_\ell \\ T_N\varphi_{N-\ell} \end{bmatrix} = \sin^2(\alpha_\ell/2)\cos^2(\alpha_\ell/2) \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} \varphi_\ell \\ \varphi_{N-\ell} \end{bmatrix}.$$

Les valeurs propres de la matrice au milieu sont 0 et 2, donc pour $\ell = 1, \dots, N/2-1$ les valeurs propres de $T_{N,\ell}$ sont

$$2\sin^2(\alpha_\ell/2)\cos^2(\alpha_\ell/2) = \frac{1}{2}\sin^2(\alpha_\ell)$$

et 0. Les valeur propre le plus grand est pris pour $\ell = N/2-1$ et est égal à

$$\frac{1}{2}\sin^2(\frac{\pi(N/2-1)}{N}) \to_N \frac{1}{2}.$$

Plus généralement, si l'on effectue ν pas de lissage dans chaque itération, le bloc correspondant sera

$$\begin{aligned} T_{N,\ell} &= \begin{bmatrix} \sin^2(\alpha_\ell/2) & \sin^2(\alpha_\ell/2) \\ \cos^2(\alpha_\ell/2) & \cos^2(\alpha_\ell/2) \end{bmatrix} \begin{bmatrix} \cos^2(\alpha_\ell/2) & \\ & \sin^2(\alpha_\ell/2) \end{bmatrix}^\nu \\ &= \sin^2(\alpha_\ell/2)\cos^2(\alpha_\ell/2) \begin{bmatrix} \cos^{2\nu-2}(\alpha_\ell/2) & \sin^{2\nu-2}(\alpha_\ell/2) \\ \cos^{2\nu-2}(\alpha_\ell/2) & \sin^{2\nu-2}(\alpha_\ell/2) \end{bmatrix}. \end{aligned}$$

Le rayon spectral et la norme sont

$$\rho(T_\ell) = F(\sin^2(\alpha_\ell/2)) \quad , \quad ||T_\ell|| = G(\sin^2(\alpha_\ell/2))$$

avec

$$F(x) = x(1-x)^\nu + (1-x)x^\nu \quad , \quad G(x) = \sqrt{2(x^2(1-x)^{2\nu} + (1-x^2)x^{2\nu})},$$

donc

$$\begin{aligned} \rho(T_N) &= \max\{F(\sin^2(\alpha_\ell/2)) : 1 \le \ell \le N/2-1\} \le \max\{F(x) : x \in [0,1/2]\}, \\ ||T_N|| &= \max\{G(\sin^2(\alpha_\ell/2)) : 1 \le \ell \le N/2-1\} \le \max\{G(x) : x \in [0,1/2]\} \end{aligned}$$

du fait que $0 < \sin^2(\alpha_\ell/2) < 1/2$. On estime ces quantités :

LEMME 4.2.

$$\rho(T_N) = \frac{1}{e\nu} + O(\frac{1}{\nu^2}) \quad , \quad ||T_N|| = \frac{\sqrt{2}}{e\nu} + O(\frac{1}{\nu^2}).$$

DÉMONSTRATION. On calcule d'abord le maximum de F sur l'intervale $[0,1/2]$. La dérivée est

$$F'(x) = (1-x)^\nu - \nu x(1-x)^{\nu-1} + \nu x^{\nu-1}(1-x) - x^\nu = x^\nu \cdot P(u)$$

avec $P(u) := u^\nu - \nu u^{\nu-1} + \nu u - 1$ et $u := x^{-1} - 1$. Donc $F(x_0) = 0$ si et seulement si $P(u_0) = 0$ et notons que la condition $0 < x \le 1/2$ équivaut à ce que $u \ge 1$. On

$$P(\nu-1) = -(\nu-1)^{\nu-1} + \nu(\nu-1) - 1 \le 0 \quad \text{et} \quad P(\nu) = \nu^2 - 1 \ge 0$$

donc P possède une racine $u(\nu) \in [\nu-1, \nu]$. En outre, $P(1) = 0$ donc il n'y a plus d'autres racinces positives, grâce à la règle de Descartes. Soit

$$x(\nu) := \frac{1}{u(\nu)+1},$$

on en déduit que F croît entre 0 et $x(\nu)$ et decroît entre $x(\nu)$ et $1/2$. Le maximum est atteint en $x(\nu)$; on a

$$\frac{1}{\nu+1} \le x(\nu) \le \frac{1}{\nu}$$

et donc

$$\begin{aligned} \max\{F(x) : x \in [0,1/2]\} &= F(\nu^{-1}) + O(\nu^{-2}) = \nu^{-1}(1-\nu^{-1})^\nu + \nu^{-\nu}(1-\nu^{-1}) \\ &= \frac{1}{e\nu} + O(\frac{1}{\nu^2}). \end{aligned}$$

Finalement

$$\rho(T_N) = \max\{F(\sin^2(\alpha_\ell/2)) : 1 \leq \ell \leq N/2-1\} = \max\{F(x) : x \in [0,1/2]\}+O(\frac{1}{\nu^2})$$

car $F \in \mathcal{C}^2([0,1/2])$. Pour la estimation de la norme 2, on observe que

$$\sqrt{2}x(1-x)^\nu \leq G(x) \leq \sqrt{F(x)},$$

d'où

$$\sqrt{2}\frac{1}{\nu}(1-\frac{1}{\nu})^\nu \leq \max\{G(x) : x \in [0,1/2]\} \leq \sqrt{2}\max\{F(x) : x \in [0,1/2]\}$$

et on en déduit

$$||T_N|| = \max\{G(x) : x \in [0,1/2]\} + O(\frac{1}{\nu^2}) = \frac{\sqrt{2}}{e\nu} + O(\frac{1}{\nu^2}).$$

□

EXERCICE 8.3. ◁ D´écrire un algorithme bigrille dans le cas des différences finies sur un rectangle et calculer le rayon spectral de la matrice correspondante d'itérations. ▷

5. Méthode multigrille

Donc le taux de convergence de la méthode bigrille est indépendant du pas de discrétisation, donc elle converge très rapidement. Cependant, pour l'appliquer on est conduit à résoudre à chaque itération un système avec la moitié de variables, ce qui n'est pas efficace.

Cette désavantage est remédié par la méthode multigrille. Dans la méthode multigrille, on ne résout pas cette équation mais on l'approxime itérativement en utilisant la même méthode bigrille, cette fois appliquée aux grilles $\Omega_{N/2}$ et $\Omega_{N/4}$. Un nouveaux problème auxiliaire apparaît

$$L_{N/4}e_{N/4} = r_{N/4},$$

qu'on résout encore par un coup de bigrille appliqué aux niveaux $N/4$ et $N/8$, etc., jusque arriver à un niveau suffisamment bas pour que la résolution directe soit compétitive.

On suppose maintenant $N = 2^k$ pour un certain $k \in \mathbb{N}$, et pour $0 \leq j \leq k$ on pose $\Omega_j, L_j, J_j, P_j, R_j, u_j, f_j$ pour les objets correspondant au niveau j. La récursion qui résulte de l'antérieur est le *cycle multigrille* Φ_k, dont on résume sa forme algorithmique :

Cycle multigrille $u_k^{m+1} \leftarrow \Phi_k(\gamma, u_k^m, L_k, f_k, \nu_1, \nu_2)$:

Si $k = 0$ alors $u_0 \leftarrow L_0^{-1} f_0$, sinon

(1) Prelissage :

  (a) $\widehat{u}_k^m \leftarrow \mathcal{S}_k^{(\nu_1)}(u_k^m, L_k, f_k)$;

(2) Correction *via* la grille grossière (CGG) :

  (a) Calcul du résidu : $r_k^m \leftarrow f_k - L_k\widehat{u}_k^m$;

  (b) Restriction à Ω_{k-1} : $r_{k-1}^m \leftarrow R_k d_k^m$;

  (c) Résolution sur Ω_{k-1} par application de $\gamma \geq 1$ cycles multigrille d'ordre $k-1$, utilisant la fonction 0 comme première approximation :

$$e_H^m \leftarrow \Phi_{\ell-1}^{(\gamma)}(\gamma, 0, L_{k-1}, r_{k-1}^m, \nu_1, \nu_2);$$

  (d) Interpolation sur Ω_k : $e_k^m \leftarrow P_k e_{k-1}^m$;

  (e) Correction de l'approximation : $\widehat{u}_k^{m+1} \leftarrow \widehat{u}_k^m + e_k^m$.

(3) Post-lissage :

(a) $\widehat{u}_k^m \leftarrow \mathcal{S}^{(\nu_2)}(\widehat{u}_k^m, L_k, f_k)$;

La procédure récursive fini après k pas, quand on atteint le niveau 0 ; l'algorithme est donc bien défini. Pour la CGG on applique γ pas de l'itération $\Phi_{\ell-1}$; dans la pratique $\gamma \leq 2$ suffit. Le multigrille avec $\gamma = 1$ est appelé *V-cycle* et celui avec $\gamma = 2$ est appelé *W-cycle*, pour des raisons évidentes, *voir* les figures ci-dessous.

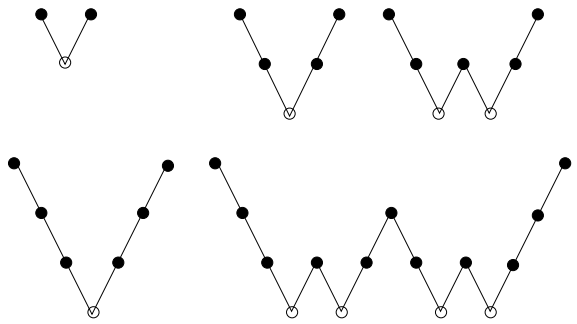


FIGURE 1. Structure des cycles multigrille pour $k = 1, 2, 3$ et $\gamma = 1, 2$

6. Complexité du multigrille

Estimer la complexité de l'itération multigrille n'est pas immédiat, à cause de la nature récursive de sa définition. Posons $n_k := \#\Omega_k + 1 = 2^k$. Le coût de chaque étape est estimé en

$C_L n_k$ opérations pour le *lissage* $\mathcal{S}(u_k, f_k)$,
$C_C n_k$ opérations pour le *résidu* $R_k(f_k - L_k u_k)$,
$C_P n_k$ opérations pour la *correction de l'approximation* $u_k - P_k e_k$.

La proportionnalité à la dimension est une conséquence de la esparsité de la matrice L_k. On a

$$n_k = 2n_{k-1}.$$

Soit $\nu = \nu_1 + \nu_2$; la complexité du V-cycle ($\gamma = 1$) est donc estimé par

$$C_V(k) \leq \sum_{j=0}^{k} (\nu C_L + C_R + C_P) 2^j \leq 2(\nu C_L + C_R + C_P)N = O(N).$$

La complexité du W-cycle ($\gamma = 2$) est estimé par

$$C_W(k) \leq \sum_{j=0}^{j} (\nu C_L + C_R + C_P) 2^k \leq (\nu C_L + C_R + C_P) N \log(N) = Os(N \log(N)).$$

Le 1/2-Jacobi d'ordre k est défini par

$$(\widehat{u}_k)_j = \frac{(u_k)_j}{2} + \frac{1}{4}((u_k)_{j-1} + (u_k)_{j+1}) + \frac{1}{4N^2}(f_k)_j,$$

donc

$$C_L = 5.$$

Des vérifications du même type pour le résidu et la correction donnent $C_R = 8$ et $C_P = 4$; le coût total est estimé en

$$C_V(k) = (10\nu + 24)N.$$

7. Méthode multigrille complet

La présence d'une hiérarchie d'équations $L_k u_k = f_k$ est nécessaire à la mise en œuvre du multigrille. Cette situation peut encore s'exploiter d'une autre façon.

Évidemment le résultat d'une itération est d'autant plus précis si le point de départ est mieux. Habituellement, la solutions u_k^∞ de $L_k u_k = f_k$ s'approche vers la solution continue u de $Lu = f$ avec un certain *ordre de consistance* $O(h^\kappa)$ ($h = 2^{-k}$). Par l'inégalité du triangle on a

$$||u_k - \widetilde{P}_k u_{k-1}|| = O(h^\kappa);$$

$\widetilde{P}_k$ étant un opérateur d'interpolation convenable, qui n'est pas forcément le même que celui qu'on utilise dans l'itération multigrille.

Cette observation suggère de commencer Φ_k avec le résultat de l'itération Φ_{k-1}. Soit $m \in \mathbb{N}$ un paramètre fixé ; l'algorithme du *multigrille complet* est le suivant :

(1) $\widetilde{u}_0 \leftarrow L_0^{-1} f_0$;

(2) for $j = 1, \ldots, k$ do begin $\widetilde{u}_j \leftarrow \widetilde{P}_j u_{j-1}$;

  (a) for $\ell = 1, \ldots, m$ do $\widetilde{u}_j \leftarrow \Phi_j(\gamma, \widetilde{u}_j, L_j, f_j, \nu_1, \nu_2)$;

(3) end.

Notons que ceci n'est pas une itération proprement dîte, mais un processus *fini*.

Si l'on note $C_{\mathrm{MG},V} N$ la complexité de l'itération multigrille Φ_k correspondant au V-cycle, il en résulte une complexité

$$C_{\mathrm{MGC},V}(N) \le m C_{\mathrm{MG},V}(n_0 + n_1 + \cdots + n_k) = 2 C_{\mathrm{MG},V} n_k \le (20\nu + 48) N$$

pour le multigrille complet. Similairement la complexité du multigrille complet correspondant au W-cycle s'estime en

$$C_{\mathrm{MGC},W}(N) \le (10\nu + 24) N \log(N).$$

8. Analyse de l'erreur

Pour notre problème modèle l'ordre de consistance est $\kappa = 2$, c'est-à-dire

$$||u - u_k|| \le C h_k^2$$

pour une certaine constante $C > 0$. Pour le multigrille complet on utilisera aussi l'opérateur d'interpolation bilinéaire ; on a donc

$$\begin{aligned} ||u_k^\infty - P_k u_{k-1}^\infty|| &\le ||u_k^\infty - u|| + ||u - P_k u|| + ||P_k|| \cdot ||u - u_{k-1}^\infty|| \\ &\le C h_k^2 + \frac{||u''||}{2} h_k^2 + C h_{k-1}^2 = \Big(5C + \frac{||u''||}{2}\Big) h_k^2. \end{aligned}$$

puisque

$$\begin{aligned} |u(x) - P_k(u)(x)| &= \left\{ \begin{array}{ll} 0 & , \text{ pour } x \in \Omega_{k-1}; \\ \frac{1}{2}\Big| -u(x-h) + 2u(x) - u(x+h)\Big| & , \text{ pour } x \in \Omega_k \setminus \Omega_{k-1}; \end{array} \right\} \\ &\le \frac{1}{2} \sup\{|u''(x)| : x \in [0,1]\} h_k^2. \end{aligned}$$

Écrivons M_k l'opérateur de l'itération multigrille. Comme on verra bientôt, ceci possède un taux de convergence qu'on peut estimer indépendamment du pas de discrétisation

$$||M_k|| \le \zeta < 1.$$

THÉORÈME 8.1. *Supposons $\zeta < 1/4$, alors le multigrille complet produit des approximations $\widetilde{u}_k$ tels que*

$$||u_k - \widetilde{u}_k|| \leq C_1 \zeta^m (1 - 4\zeta^m)^{-1} h_k^2$$

avec $C_1 = C_1(u) = 5C + ||u''||/2$.

Pour des méthodes multigrille avec une taux de convergence $\zeta \leq 1/5 = 0.20$, on aura

$$||u_k - \widetilde{u}_k|| \leq C_1 h_k^2.$$

DÉMONSTRATION. On raisonne par récurrence sur k. Posons $\gamma(\zeta^m) = \zeta^m(1 - 4\zeta^m)^{-1}$ et $u_k^0 = P_k \widetilde{u}_{k-1}$, alors

$$\begin{aligned} ||u_k^\infty - u_k^0|| &\leq ||u_k^\infty - P_k u_{k-1}^\infty|| + ||P_k|| \cdot ||u_{k-1}^\infty - \widetilde{u}_{k-1}^m|| \\ &\leq C_1 h_k^2 + C_1 \gamma(\zeta^m) h_{k-1}^2 \\ &= C_1 h_k^2 (1 + 4\gamma(\zeta^m)). \end{aligned}$$

Après m itérations de la méthode multigrille on trouve

$$||u_k^\infty - \widetilde{u}_k^m|| \leq \zeta^m ||u_k^\infty - u_k^0|| \leq C_1 h_k^2 \zeta^m (1 + 4\gamma(\zeta^m)) = C_1 \gamma(\zeta^m) h_k^2.$$

□

EXERCICE 8.4. ◁ Comment doit-on choisir m variable dans le cas où le taux de convergence dépens du pas de discrétisation, par exemple quand $\zeta_k = 1 - O(h_k^\eta)$? ▷

9. Matrice de l'itération multigrille

Puisque l'itération est définie récursivement, il est de même pour la matrice :

THÉORÈME 9.1. *L'itération multigrille est consistante, et on a*

$$M_k = J_k^{\nu_2} \Big(\mathbf{1} - P_k (\mathbf{1} - M_{k-1}^\gamma)(L_{k-1})^{-1} R_k L_k \Big) J_k^{\nu_2} \quad , \quad k \in \mathbb{N}.$$

DÉMONSTRATION. Notons d'abord que pour une itération consistante $u^{m+1} \leftarrow Mu^m + v$ pour l'équation $L_k u = f_k$, le vecteur v est déterminé par

$$v = (\mathbf{1} - M) L_k^{-1} f_k.$$

Par la définition du multigrille

$$\Phi_k(u, L_k, f_k) = \mathcal{J}_k^{(\nu_2)} \Big(\mathcal{J}^{(\nu_1)}(u) + P_k \circ \Phi_{k-1}^{(\gamma)}(0, L_{k-1}, R_k(f_k - L_k \mathcal{J}^{(\nu_1)}(u)) \Big).$$

On a

$$\begin{aligned} &\Phi_{k-1}^{(\gamma)}(0, L_{k-1}, R_k(f_k - L_k \mathcal{J}^{(\nu_1)}(u)) \\ &\qquad = M_{k-1} \cdot 0 + (\mathbf{1} - M_{k-1}^\gamma)(L_{k-1})^{-1} R_k((f_k - L_k \mathcal{J}^{(\nu_1)}(u)), \end{aligned}$$

d'où l'on tire l'expression cherchée. □

10. Analyse du taux de convergence

Pour compléter l'analyse, il ne nous reste que d'établir le taux de convergence de l'opérateur multigrille M_k. Pour le W-cycle ($\gamma = 2$) l'indépendance du taux de convergence par rapport à h peut se déduire facilement de celui pour la méthode bigrille. Le V-cycle satisfait aussi une estimation du même tabac pour le taux de convergence, cependant l'analyse à faire est plus délicat, *voir* [**20**, § 10.7].

THÉORÈME 10.1. *Supposons $||T_k|| \leq \tau$ pour un certain $\tau < 1$, alors*

$$||M_k|| \leq \frac{2\sqrt{2}}{(} 1 - 2^{5/2}\tau).$$

DÉMONSTRATION. À l'aide des expressions matricielles pour les opérateur bigrille et multigrille on trouve

$$M_k = T_k + A_k M_{k-1}^2 B_k$$

avec

$$A_k = J^{\nu_2} P_k \quad , \quad B_k = L_{k-1}^{-1} R_k L_k J^{\nu_1}.$$

Pour l'équation de Poisson en dimension 1 on a **(vérifier)**

$$||A_k|| \leq ||J||^{\nu_2} ||P_k|| \leq 1,$$

$$||B_k|| \leq ||L_{k-1}^{-1}|| ||R_k|| ||L_k|| ||J||^{\nu_1} \leq \sqrt{2},$$

d'où par récurrence sur k

$$||M_k|| \leq \tau + \sqrt{2}||M_{k-1}||^2 \leq \frac{2\sqrt{2}}{(}1 - 2^{5/2}\tau).$$

□

Un calcul simple montre que pour $\nu = 4$ on a

$$\tau(\nu) \sim \frac{\sqrt{2}}{e\nu} = 0.13$$

et donc

$$\sigma(\nu) := \frac{2\sqrt{2}}{(}1 - 2^{5/2}\tau(\nu)) = 0.171 < 1/5.$$

On conclu que la méthode multigrille complète associé au W-cycle produit des approximations $\widetilde{u}_k$ tels que

$$||u_k - \widetilde{u}_k|| \leq C_1 h_k^2$$

avec $C_1 = C_1(u) = 5C + ||u''||/2$, avec complexité ($N = 2^k$)

$$C_{\mathrm{MGC},W}(N) \leq (10\nu + 24) N \log(N) = 64 N \log(N).$$

Chapitre 9

Les résultats d'Alan Edelman sur le nombre de conditionnement des matrices

1. Énoncé des résultats

2. Une application numérique pour des problèmes elliptiques discrétisés

3. Les transformations matricielles et leur régularité

On ne fera pas de différence de terminologie entre matrices symétriques réelles et matrices hermitiennes complexes ; l'adjoint d'une matrice complexe est bien sûr sa transposée conjuguée et l'adjoint d'une matrice réelle est sa transposée.

Nous avons besoin de quelques notations pour alléger la lecture : les matrices hermitiennes complexes (respectivemetn réelles) forement un sous-espace vectoriel de $\mathbb{C}^{n\times n}$ (respectivement $\mathbb{R}^{n\times n}$ qui sera noté $H^n_{\mathbb{C}}$ (respectivement $H^n_{\mathbb{R}}$) ; le cône des matrices hermitiennes définies positives sera noté respectivement $H^{n,+}_{\mathbb{C}}$ et $H^{n,+}_{\mathbb{R}}$. L'espace des matrices triangulaires inférieures $n\times n$ sera noté respectivement $T^n_{\mathbb{C}}$ et $T^n_{\mathbb{R}}$; le cône des matrices triangulaires inférieures à diagonale strictement positive sera, lui, noté respectivement $T^{n,+}_{\mathbb{C}}$ et $T^{n'+}_{\mathbb{R}}$.

Certaines transformations matricielles jouent un rôle essentiel dans la démonstration des résultats d'Edelman. Ils sont relatifs aux décompositions suivantes ; on pose $\mathbb{K}=\mathbb{R}$ ou $\mathbb{C}$:

(1) la décomposition de Cholesky, à une permutation près : pour $A\in H^{n,+}_{\mathbb{K}}$, il existe un unique $C\in T^{n,+}_{\mathbb{K}}$ tel que $A=CC^*$;

(2) la décomposition QR, ou plutôt son analogue la décomposition LQ : pour toute matrice A non singulière, il existe une unique matrice $L\in T^{n,+}_{\mathbb{C}}$ et une unique matrice Q orthogonale si $\mathbb{K}=\mathbb{R}$ et unitaire si $\mathbb{K}=\mathbb{C}$ telles que $A=LQ$;

(3) la décomposition d'une matrice $A\in H^n_{\mathbb{K}}$ en un produit $Q\Lambda Q^*$, Λ étant une matrice diagonale réelle et Q une matrice orthogonale dans le cas réel, unitaire dans le cas complexe.

Pour les deux premières décompositions, il y a existence et unicité, et pour la troisième, on n'a unicité qu'en dehors d'une réunion finie d'hypersurfaces, et en se restreignant à imposer l'ordre des valeurs propres dans Λ et la positivité stricte des premiers coefficients de chacun des colonnes de Q.

On peut calculer les jacobiens de ces transformations ; dans le premier cas, celui de la décomposition de Cholesky, il n'y a pas de difficultés particulières, mais dans le deuxième et le troisième, on a besoin de savoir quelle mesure prendre sur une variété, qui est dans le cas réel le groupe orthogonal et dans le cas complexe le groupe unitaire.

Les jacobiens de ces transformations sont obtenus dans les cas réels et complexes ; il y a d'ailleurs quelques différences entre les deux calculs, bien que les principes en soient tout à fait analogues.

Or pour calculer ces jacobiens de manière efficace, nous avons besoin d'un ingrédient qui n'est pas forcément tout à fait familier aux gens qui ont une formation de mathématiques appliquées : le produit extérieur des formes multilinéaires alternées, la différentielle extérieure, et les calculs de volume sur les variétés. Comme il ne s'agira que de variétés plongées, les préliminaires techniques seront réduits au minimum, mais sans cet investissement, l'obtention des résultats d'Edelman est beaucoup plus difficile. Les personnes ayant appris ces parties des mathématiques pourront sauter les sections *il faudra dire lesquelles.*

Rappelons ce que sont ces décompositions, et démontrons qu'on a bien des difféomorphismes comme annoncé :

LEMME 3.1. *Soit $\mathbb{K} = \mathbb{R}$ ou $\mathbb{C}$. Pour toute matrice $A \in H_{\mathbb{K}}^{n,+}$ il existe une unique matrice triangulaire inférieure $C \in T_{\mathbb{K}}^{n,+}$ telle que $A = CC^*$. De plus, l'application $A \mapsto C$ de l'ensemble des matrices hermitiennes définies positives dans l'ensemble des matrices triangulaires supérieures à diagonale strictement positive est un difféomorphisme analytique réel. Dans le cas complexe, cet énoncé signifie que la dépendance des parties réelle et imaginaire de C est analytique réelle par rapport aux parties réelle et imaginaire de C, et réciproquement.*

DÉMONSTRATION. La première partie de l'énoncé est la classique décomposition de Cholesky des matrices hermitiennes définies positives. La deuxième partie sera déduite de la démonstration de la première, qu'on a donc besoin de rappeler concisément, afin de pouvoir obtenir le résultat de difféomorphisme.

Commençons par prouver l'unicité : supposons qu'il existe B et C dans $T_{\mathbb{K}}^{n,+}$ telles que

$$A = BB^* = CC^*. \tag{61}$$

Alors, comme B et C sont inversibles, on déduit de (61) que

$$C^{-1}B = C^*(B^*)^{-1}$$

est une matrice à diagonale positive, à la fois triangulaire inférieure et triangulaire supérieure, donc diagonale à coefficients strictement positifs. Notons-la Λ. Alors

$$A = \Lambda A \Lambda,$$

et si on spécialise cette relation en prenant le coefficient de la ligne k et de la colonne k, on trouve $A_{kk} = \Lambda_{kk}^2 A_{kk}$. Comme A est dans $H_{\mathbb{K}}^{n,+}$, A_{kk} est strictement positif, et on a montré que Λ est en fait la matrice identité.

La preuve d'existence se fait par récurrence par rapport à la dimension. Pour $n = 1$, A n'a qu'un seul coefficient A_{11}, qui est forcément strictement positif ; on peut donc prendre $C = (C_{11})$ avec $C_{11} = \sqrt{A_{11}}$. Il est immédiate que l'application $x \mapsto x^2$ est un difféomorphisme analytique réel de $(0, \infty)$ sur lui même.

Supposons maintenant que le résultat soit vrai jusqu'en dimension $n-1$. Alors nous pouvons écrire la décomposition par blocs suivante :

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \text{ and } C = \begin{pmatrix} C_{11} & 0 \\ C_{21} & C_{22} \end{pmatrix}.$$

Ici, les blocs A_{11} et C_{11} sont des blocs 1×1 ne comportant qu'un réel positif, les blocs $A_{21} = A_{12}^*$ et C_{21} sont des blocs $(n-1) \times 1$ et les blocs $A_{22} = A_{22}^*$ et C_{22} sont des blocs $(n-1) \times (n-1)$. On a alors

$$C^*C = \begin{pmatrix} C_{11} & 0 \\ C_{21} & C_{22} \end{pmatrix} \begin{pmatrix} C_1 1 & C_{21}^* \\ 0 & C_{22}^* \end{pmatrix} = \begin{pmatrix} C_{11}^2 & C_{11}C_{21}^* \\ C_{11}C_{21} & C_{21}C_{21}^* + C_{22}C_{22}^* \end{pmatrix}.$$

On doit donc réaliser les relations

$$A_{11} = C_{11}^2, \tag{62}$$

$$A_{21} = C_{11}C_{21}, \tag{63}$$

$$A_{22} = C_{21}C_{21}^* + C_{22}C_{22}^*. \tag{64}$$

On choisit donc $C_{11} = \sqrt{A_{11}}$, ce qui fait de C_{11} une fonction analytique réelle de A_{11} sur $(0, \infty)$, et donc une fonction analytique réelle de A hermitien défini positif. On pose ensuite

$$C_{21} = A_{21}/A_{11}$$

ce qui fournit immédiatement C_{21} comme fonction analytique réelle de A_{11} et de A_{21}, donc aussi de A. Il reste à vérifier que $A_{22} - C_{21}C_{21}^* = A_{22} - A_{21}A_{21}^*/A_{11}$ est une fonction analytique réelle de A, ce qui est immédiat, et que c'est un élément de $H_{\mathbb{K}}^{n-1,+}$, afin d'appliquer l'hypothèse de récurrence. Le vecteur

$$x = \begin{pmatrix} -A_{12}x_2/A_{11} \\ x_2 \end{pmatrix}$$

est non nul si et seulement si x_2 n'est pas nul ; un calcul direct montre que

$$x^*Ax = x_2^*(A_{22} - A_{21}A_{21}^*/A_{11})x_2$$

et que par conséquent $A_{22} - A_{21}A_{21}^*/A_{11}$ est hermitien défini positif ; par conséquent il existe, en vertu de notre hypothèse de récurrence une matrice $C_{22} \in T_{\mathbb{K}}^{n,+}$ telle que (64) ait lieu, ce qui démontre que C dépend analytiquement de A. La dépendance réciproque est évidente. □

Avant de démontrer les lemmes suivants, nous avons besoin de comprendre la nature de l'ensemble $V_{\mathbb{K}}^{n,m}$ des matrices P de $\mathbb{K}^{n\times m}$, $n > m$ telles que

$$P^*P = \mathbf{1}_m. \tag{65}$$

L'ensemble $\mathbb{K}^{n\times m}$ des matrices à n lignes et m colonnes est un espace vectoriel à nm dimensions sur $\mathbb{K}$. La relation (65) contient m^2 relations, mais celles-ci ne sont pas indépendantes. En effet, si on note $p_1, \ldots, p_m$ les colones de P, on remarque que pour $k \neq l$, $p_k^*p_l = 0$ équivaut à $p_l^*p_k = 0$. Il n'y a donc apparemment que m relations réelles :

$$|p_1|^2 = \cdots = |p_m|^2 = 1$$

et $m(m-1)$ relations dans $\mathbb{K}$:

$$1 \leq i < j \leq m \Longrightarrow p_i^*p_j = 0.$$

Cela veut dire qu'on espère que $V_{\mathbb{R}}^{n,m}$ sera de dimension réelle $mn - m(m+1)/2$ et que $V_{\mathbb{C}}^{n,m}$ sera de dimension réelle $2mn - m - m(m-1) = 2mn - m^2$.

De fait, comme nous allons le montrer, nous avons bien affaire à une variété de la bonne dimension.

LEMME 3.2. *Pour $n \geq m$, $V_{\mathbb{K}}^{n,m}$ est une variété analytique réelle qu'on appelle variété de Stiefel ; si $Ka = \mathbb{R}$, sa dimension réelle est $mn - m(m+1)/2$ et si $\mathbb{K} = \mathbb{C}$, sa dimension réelle est $2mn - m^2$.*

DÉMONSTRATION. On peut compléter la matrice P en une matrice orthogonale ou unitaire, en adjoignant à la liste des vecteurs colonnes $p_1, \ldots, p_m$ de P une liste supplémentaire de $n-m$ vecteurs colonnes $p_{m+1}, \ldots, p_n$. En adoptant une notation par blocs, on peut donc écrire

$$P = \begin{pmatrix} p_1 & p_2 & \ldots & p_m \end{pmatrix}, P_1 = \begin{pmatrix} p_{m+1} & p_{m+2} & \ldots & p_n \end{pmatrix} \text{ et } P_2 = \begin{pmatrix} P & P_1 \end{pmatrix}.$$

Bien sur, P_2 vérifie la relation $P_2^* P_2 = P_2 P_2^* = \mathbf{1}_n$. Si nous savions déjà que $V_{\mathbb{K}}^{n,m}$ était une variété, son espace tangent en P serait donné par l'ensemble des matrices $Q \in \mathbb{K}^{n\times m}$ telles que

$$Q^* P + P^* Q = 0, \tag{66}$$

ce qu'on obtient en linéarisant l'équation (65). Posons

$$X = P_2^* Q$$

ce qui est légitime parce que P_2 est dans $\mathbb{K}^{n\times n}$ et parce que Q est dans $\mathbb{K}^{n\times m}$. Avec cette transformation, la relation (66) devient

$$X^* P_2^* P + P^* P_2 X = 0. \tag{67}$$

Or un calcul immédiat donne

$$P^* P_2 = \begin{pmatrix} \mathbf{1}_m \\ 0_{(n-m)\times m} \end{pmatrix}$$

et si on note $\hat{X}$ la matrice formée des m premières lignes de X, la relation (67) est équivalente à $\hat{X} + \hat{X}^* = 0$. À présent, il est clair que la dimension de l'espace des matrices $Q \in \mathbb{K}^{n\times m}$ vérifiant (66) est précisément la dimension de l'espace des matrices $X \in \mathbb{K}^{n\times m}$ dont le bloc formé des m premières lignes est antiadjoint ; la dimension réelle de cet espace dans le cas $\mathbb{K} = \mathbb{R}$ est $m(m-1)/2 + m(n-m)$ et dans le cas $\mathbb{K} = \mathbb{C}$ elle est $m^2 + 2m(n-m)$.

Maintenant, nous voyons que pour tout $P \in V_{\mathbb{K}}^{n,m}$, le système (65) est un système de $m(m+1)$ équations réelles à mn inconnues réelles (respectivement si $\mathbb{K} = \mathbb{C}$ m^2 équations réelles à $2mn$ inconnues réelles), et que le rang du système est donné par la dimension de l'espace des solutions de (66), et qu'on vient de la calculer. Comme la somme du nombre d'équations et du rang du système est égale au nombre d'inconnues, nous savons d'après le théorème du rang que $V_{\mathbb{K}}^{n,m}$ est une variété plongée dans $\mathbb{R}^{m\times n}$ si $\mathbb{K} = \mathbb{R}$ et dans $\mathbb{C}^{n\times m} \approx \mathbb{R}^{n\times 2m}$ si $\mathbb{K} = \mathbb{C}$.

Comme les équations définissant la variété sont analytiques réelles, la variété est analytique réelle. □

Il est intéressant de savoir faire des cartes locales des variétés de Stiefel, et de comprendre ce que serait la mesure sur ces variétés. Remarquons que les groupe unitaire (dans le cas complexe) ou orthogonal (dans le cas réel) opère sur une variété de Stiefel. En effet, soit P un élément de $V_{\mathbb{K}}^{n,m}$, et soit $Q \in \mathbb{K}^{n\times n}$, vérifiant $Q^* Q = \mathbf{1}_n$. Alors QP est encore dans $V_{\mathbb{K}}^{n,m}$, comme on le vérifie immédiatement ; si au contraire Q est dans $\mathbb{K}^{m\times m}$ et vérifie $QQ^* = \mathbf{1}_m$, alors c'est PQ qui est encore dans $V^{n,m}$. Il résulte de ces observations que si on trouve une carte locale de $V^{n,m}$ au voisinage d'un P particulier, alors on obtient immédiatement un recouvrement par des cartes de $V_{\mathbb{K}}^{n,m}$, qui est de toute évidence une variété compacte.

Vérifions maintenant que la paramétrisation suivante décrit localement au voisinage de

$$P_0 = \begin{pmatrix} \mathbf{1}_m \\ 0_{(n-m)\times m} \end{pmatrix}$$

la variété $V_{\mathbb{K}}^{n,m}$: soit X une matrice appartenant à $\mathbb{K}^{n\times m}$, dont on note X_1 le bloc $m \times m$ supérieur, et X_2 le bloc restant. On suppose que $X_2 \in \mathbb{K}^{(n-m)\times m}$ vérifie

$$\|X_2\|_2^2 = \rho(X_2^* X_2) \leq \varepsilon;$$

si $\varepsilon < 1$ alors $\mathbf{1}_m - X_2^* X^2$ est hermitien défini positif, et donc il possède une unique racine carrée hermitienne définie positive, qu'on va noter $\sqrt{\mathbf{1}_m - X_2^* X_2}$. Si maintenant $X_1 \in \mathbb{K}^{m\times m}$ est antiadjoint, la matrice par blocs

$$\begin{pmatrix} \phi(X) = \exp(X_1)\sqrt{\mathbf{1}_m - X_2^* X^2} \\ X_2 \end{pmatrix}$$

appartient à $V_{\mathbb{K}}^{n,m}$, comme on le vérifie immédiatement. Dans le cas réel, X_1 est décrit par $m(m-1)$ paramètres, la matrice X_2 est décrite par $(n-m)m$ paramètres, le total étant la dimension de $V^{n,m}\mathbb{R}$; un calcul analogue pour $\mathbb{K}=\mathbb{C}$ donne m^2 paramètres réels pour X_1 et $2(n-m)m$ paramètres pour X_2, ce qui est bien la dimension de $V_{\mathbb{C}}^{n,m}$.

Le rang de l'application ϕ en P_0 se calcule en obtenant le rang de sa dérivée :

$$D\phi(X)Y = \begin{pmatrix} \exp(X_1)\big[Y_1 + D(\sqrt{\mathbf{1}_m - X_2^* X_2})Y_2\big] \\ Y_2 \end{pmatrix}.$$

Il n'est pas nécessaire de calculer explicitement la dérivée de $X_2 \mapsto \sqrt{\mathbf{1}_m - X_2^* X_2}$, puisque $P_0 = \phi(0_{n\times m})$, et qu'en $X_2 = 0$ cette dérivée est nulle. On a donc

$$D\phi(0_{n\times m})Y = Y$$

ce qui nous prouve que le rang réel de la dérivée de ϕ en 0 est $(n-m)m+m(m-1)/2$ dans le cas réel et $2(n-m)m+m^2$ dans le cas complexe, et par continuité, le rang de $D\phi$ est constant au voisinage de $0_{n\times(m-n)}$.

Nous aurons aussi besoin d'une mesure sur les variétés de Stiefel ; pour cela, il faut d'abord savoir comment fabriquer une mesure sur une variété plongée dans $\mathbb{R}^l$; le résultat qui suit est bien sûr un résultat classique de géométrie différentielle, mais comme nous n'avons pas posé de prérequis de géométrie différentielle et que la plupart des livres de géométrie différentielle imposent d'abord de se battre avec des dizaines de pages fort abstraites avant d'arriver au résultat, nous avons préféré montrer le résultat directement par une technique de passage à la limite.

Définition 3.3. Soit V une variété de dimension k plongée dans $\mathbb{R}^l$. Le volume induit par la mesure de Lebesgue de $\mathbb{R}^l$ sur V est défini comme suit : si K est un compact de V, et si vol_p désigne le volume en dimension p calculé au moyen de la mesure de Lebesque, et si B_p est la boule unité euclidienne de dimension p, on pose

(68)
$$\mathrm{vol}_V(K) = \lim_{\varepsilon\downarrow 0} \frac{1}{\varepsilon^{l-k}\,\mathrm{vol}_{l-k}(B_{l-k})}\,\mathrm{vol}_l\{x\in\mathbb{R}^l : \mathrm{dist}(x,K) = |x-y| \le \varepsilon, y\text{intérieur à } K\}.$$

Remarque 3.4. La définition 3.3 est naturelle dans le cas d'une hypersurface ; pensons pour simplifier au cas d'une surface V dans $\mathbb{R}^3$; on dessine sur cette surface un morceau compact K, et on définit un tube autour de K, comme l'ensemble des points de $\mathbb{R}^l$ à distance au plus ε de K, et tels que la distance minimale soit réalisée en un point intérieur à K ; ce tube est donc formé de points qui sont sur une normale à la surface V, et à distance au plus ε de V ; le volume de cet ensemble de points, c'est 2ε fois l'aire de K plus des termes en $O(\varepsilon^2)$, qui comportent, eux, les termes de courbure. Comme le volume de la boule unité de $\mathbb{R}$ est 2, le passage à la limite donne le résultat attendu. Voir la figure 1 pour mieux se représenter la situation.

Lemme 3.5. *Soit Ω un ouvert de $\mathbb{R}^k$, et soit f une application de classe C^2 et de Ω dans $\mathbb{R}^l$ ($l \ge k$). On suppose que pour tout x dans Ω, $Df(x)$ est de rang k et que f est une bijection sur son image. Alors la mesure induite par le volume euclidien de $\mathbb{R}^l$ sur l'image de f est $\sqrt{\det(Df(x)^* Df(x))}\,\mathrm{d}x$.*

Démonstration. Comme $Df(x)$ est de rang k pour tout x, sur toute boule assez petite $B(x_0,\varepsilon)$, on peut compléter la matrice $Df(x)$ par $l-k$ vecteurs n_j tels que

— les vecteurs n_j forment un système orthonormal de $\mathbb{R}^l$;
— l'espace engendré par les n_j est orthonormal à l'image de $Df(x)$;
— les n_j sont continûment différentiables par rapport à x.

En effet, il suffit de prendre un tel système de vecteurs au centre x^0 de la boule : il en existe de toute évidence ; ensuite de quoi, on peut projeter les vecteurs $n_j(x_0)$

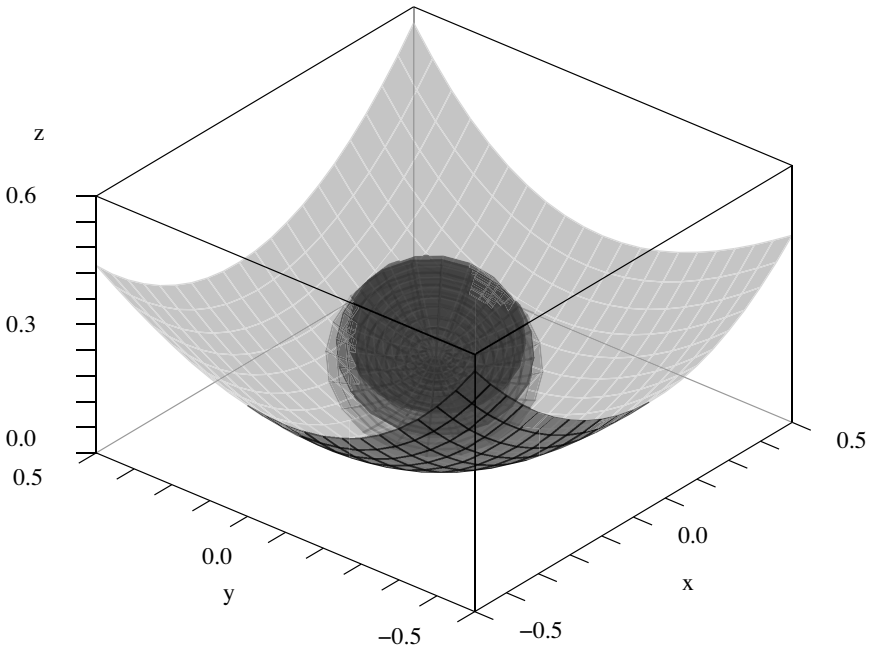


FIGURE 1. Un morceau de la surface $z = x^2 + y^2 + xy^3$, un compact : l'ensemble des points de la surface tels que $|x-0.15|^2+|y-0.15|^2 \leq 0.04$ et le tube de demi-épaisseur 0.05 autour de ce compact : on n'a représenté que les surfaces inférieures et supérieures, pas les côtés du tube.

sur l'orthogonal de l'image de $Df(x)$, ce qui donnera un système de vecteurs indépendants par continuité, pour x assez proche de x_0 ; il suffit alors d'orthonormaliser ce système par le procédé de Gram-Schmidt pour obtenir les $n_j(x)$, et comme $Df(x)$ est de classe C^1 par rapport à x, il en est de même des $n_j(x)$. La matrice $l \times (l-k)$ dont les colonnes sont les n_i sera notée $N(x)$.

Fabriquer la mesure induite par le volume euclidien sur la variété plongée $f(\Omega)$ est un problème local ; soit alors B la boule unité de dimension $l-k$; nous définissons un tube autour de $f(B(x_0,\varepsilon))$ comme suit :

$$T(x_0,\alpha,\varepsilon) = \{f(x) + N(x)y : x \in B_k(x_0,\alpha), y \in \varepsilon B_{l-k}(0)\}.$$

Si nous posons

$$z = \begin{pmatrix} x & y \end{pmatrix} \text{ et } F(z) = \begin{pmatrix} f(x) + N(x)y, \end{pmatrix}$$

nous pouvons calculer la dérivée de F ; elle est donnée par

$$DF(z)z' = Df(x)x' + (DN(x)x')y + N(x)y',$$

et nous vérifions immédiatement que pour y nul, cette dérivée peut être mise sous forme matricielle $\begin{pmatrix} Df(x) & N(x) \end{pmatrix} = G(x)$ qui est inversible par construction, pourvu que ε soit assez petit. Par conséquent, pour ε assez petit, $DF(z)$ est aussi inversible ; de plus, comme on a supposé que f est une bijection sur son image, pour ε assez petit, F est un difféomorphisme de $B_k(x_0,\alpha) \times B_{l-k}(0,\varepsilon)$ sur son image $T(x_0,\alpha,\varepsilon)$.

Le déterminant de $DF(z)$ admet le développement asymptotique

$$\det DF(z) = \det G(x) + O(\varepsilon);$$

comme nous ne sommes intéressés que par la valeur absolue de ce déterminant dans le cas réel, et que $\det G(x) = \det G^*(x)$, il suffit de calculer $\det(G^*(x)G(x)$ et d'en

prendre la racine carrée ; mais par construction

$$G^*(x)G(x) = Df(x)^*Df(x),$$

et nous aurons donc

$$|\det DF(x)| = \sqrt{\det(Df(x)^*Df(x))} + O(\varepsilon).$$

Le volume du tube $T(x_0, \alpha, \varepsilon)$ est donné par le changement de variable $Z = F(z)$, et il vaut donc

$$\begin{aligned}\int_{T(x_0,\alpha,\varepsilon)} \mathrm{d}Z &= \int_{B_k(x_0,\alpha)\times B_{l-k}(0,\varepsilon)} \sqrt{\det(Df(x)^*Df(x))}\,\mathrm{d}x\,\mathrm{d}y + O(\varepsilon^{k-l+1}) \\ &= \int_{B_k(x_0,\alpha)} \sqrt{\det(Df(x)^*Df(x))}\,\mathrm{d}x\varepsilon^{l-k}\,\mathrm{vol}_{l-k}(B_{l-k}(0,1)) + O(\varepsilon^{l-k+1}).\end{aligned}$$

Il reste à évaluer le volume de l'ensemble des points qui sont à distance au plus ε de $f(B_k(x_0, \alpha))$, mais qui ne sont pas dans $T(x_0, \alpha, \varepsilon)$: il est majoré par l'aire $k-1$-dimensionnelle de la boule$B_k(x_0, \alpha)$ multipliée par $O(\varepsilon^{l-k+1})$, ce qui démontre le résultat annoncé. □

LEMME 3.6. *Soit A une matrice $n \times m$ réelle ou complexe de rang m, en supposant bien sûr $n \geq m$; alors il existe une unique matrice triangulaire inférieure L réelle dans le cas réel et complexe dans le cas complexe, à coefficients diagonaux strictement positifs, et une unique matrice P réelle dans le cas réel et complexe dans le cas complexe, dont les vecteurs colonnes sont orthonormaux, telles que*

$$A = LP. \tag{69}$$

LEMME 3.7. *Toute matrice hermitienne A n'a que des valeurs propres réelles et peut être diagonalisée dans une base orthonormale, c'est à dire qu'il existe une matrice Q orthogonale (dans le cas réel) ou unitaire (dans le cas complexe) ainsi qu'une matrice diagonale réelle Λ telles que*

$$A = Q\Lambda Q^*. \tag{70}$$

De plus, sauf sur un ensemble fini d'hypersurfaces de l'espace des matrices hermitienne, A a toutes ses valeurs propres distinctes, la première composante de chacun des vecteurs de Q est non nulle, et si on fixe les conditions suivantes :

- *les valeurs propres de Λ sont ordonnées par ordre décroissant ;*
- *la première composante de chacune des colonnes de Q est strictement positive,*

alors la décomposition (70) *est unique.*

DÉMONSTRATION. □

4. Calcul extérieur : les formes multilinéaires alternées

On rappelle qu'une forme p-multilinéaire alternée sur un espace V de dimension n sur $\mathbb{K}$ est une application a de V^p dans $\mathbb{K}$ qui a les propriétés suivantes :

- elle est séparément linéaire par rapport à chacun de ses arguments ;
- si τ est une transposition qui échange deux indices i et $j \neq i$, alors, quels que soient les vecteurs $u_1, \ldots. u_p$ dans V on a la relation

$$a(u_{\tau(1)}, u_{\tau(2)}, \ldots, u_{\tau(p)}) = -a(u_1, u_2, \ldots, u_p).$$

Remarquons que si deux arguments sont identiques parmi les u_i alors a s'annule sur le p-uple $u_1, \ldots, u_i$. Cette propriété est vraie dès lors que le corps $\mathbb{K}$ n'est pas de caractéristique 2. Comme nous ne travaillons dans ce chapitre qu'avec $\mathbb{R}$ ou $\mathbb{C}$, il n'y a bien évidememnt pas de problème de ce type dans ce chapitre.

Les formes p-linéaires alternées sur V forment de toute évidence un espace vectoriel, qu'on notera $\bigwedge^p(V)$, ou $\bigwedge^p$ plus simplement, s'il n'y a pas de risque de confusion.

Pour $p = 1$, $\bigwedge^1(V)$ est simplement le dual de V : l'espace des formes linéaires sur V.

Nous introduisons maintenant une notation importante, mais pas définitive : si $\{e_1, \dots, e_n\}$ désigne une base de V, nous noterons $\{e^1, \dots, e^n\}$ *la* base duale : elle est définie par les conditions

$$e^i(e_j) = \delta_{ij}. \tag{71}$$

On a souvent coutume de noter e_i^* la base duale de la base e_i, mais il est plus commode de mettre des indices en haut pour noter tout ce qui vit dans un espace de formes, comme on le verrra un peu plus loin.

Si a est une forme linéaire sur V, alors on peut décrire son action en donnant sa valeur sur chacun des vecteurs de base e_i : sous la forme

$$a(e_i) = a_i,$$

et par conséquent, si x est un vecteur de V de coordonnées x^i dans la base des e_i, c'est à dire

$$x = \sum_{j=1}^{n} x^j e_j,$$

on aura

$$a(x) = \sum_{j=1}^{n} a_j x^j.$$

En d'autres termes, par définition de la base duale,

$$e^j(x) = x_j$$

et

$$a = \sum_{j=1}^{n} a_j e^j.$$

Ceci est un résultat bien connu : une base du dual de V est la base duale définie par (71).

Voyons maintenant la situation pour les formes bilinéaires alternées. Si V est de dimension 1, c'est un espace réduit à 0 : en effet, une forme bilinéaire alternée vérifie $a(e_1, e_1) = 0$, et comme un espace de dimension 1 n'a qu'un seul vecteur de base, il ne reste plus grand chose comme forme bilinéaires alternées. On suppose donc que n est au moins égal à 2.

Nous allons refaire le même raisonnement ci-dessus : a peut être décrit par son action sur les couples de vecteurs de base de V :

$$a(e_i, e_j) = a_{ij};$$

mais maintenant, il faut tenir compte de ce que la forme est bilinéaire et alternée ; on aura donc

$$a_{jj} = 0, \text{ et pour } i \neq j, a_{ji} = -a_{ij}.$$

Maintenant, si x et y sont des vecteurs appartenant à V de coordonnées respectives x^i et y^i dans la base des e_i, nous aurons

$$a(x, y) = \sum_{1 \leq i < j \leq n} a_{ij}(x_i y_j - x_j y_i).$$

Nous allons introduire une notation : si $J = \{i, j\}$, nous définissons une forme bilinéaire $e^J = e^{ij}$ en posant

$$e^J(e_k, e_l) = e_i(e_k)e_j(e_l) - e^i(e_l)e^j(e_k).$$

Il est alors clair que

$$a = \sum_{1 \leq i < j \leq n} a_{ij} e^{ij}.$$

Les e^{ij} forment une base de $\bigwedge^2(V)$: nous savons déjà qu'elles forment un système générateur de cet espace, et il suffit de vérifier qu'elles sont indépendantes ; soit en effet une combinaison linéaire nulle des e^{ij} :

$$\sum_{1 \leq i < j \leq n} a_{ij} e^{ij} = 0.$$

Si nous appliquons cette forme bilinéaire au couple formé de e_k et de e_l, nous trouvons

$$e^{ij}(e_k, e_l) = \delta_{ik}\delta_{jl} - \delta_{il}\delta_{jk}.$$

Cette expression est non nulle que si la liste $\{i, j\}$ coïncide avec la liste $\{k, l\}$.

Par conséquent $\bigwedge^2(V)$ est un espace de dimension $n(n-1)/2$.

Nous dirons à partir de dorénavant que e^{ij} est le produit extérieur de e^i et de e^j et nous noterons

$$e^{ij} = e^i \wedge e^j.$$

Nous pouvons maintenant définir par linéarité le produit extérieur de deux formes linéaires quelconques a et b comme suit :

$$a = \sum_{i=1}^{n} a_i e^i, \quad b = \sum_{i=1}^{n} b_i e^i, a \wedge b = \sum_{1 \leq i < j \leq n} a_i b_j e^{ij}.$$

Remarquons qu'en dimension n supérieure ou égale à 4, il y a des formes bilinéaires qui ne sont pas de la forme $a \wedge b$ pour a et b des formes linéaires. En effet, en dimension 2, la base de l'espace $\bigwedge^1(V)$ n'a qu'un élément, et donc toutes les formes bilinéaires sont multiples de cet élément. En dimension 3, étant donné une forme bilinéaire c on peut trouver deux formes linéaires a et b telles que

$$a_1 b_2 - a_2 b_1 = c_{12}, \quad a_1 b_3 - a_3 b_1 = c_{13}, \quad a_2 b_3 - a_3 b_2 = c_{23}.$$

Si c est identiquement nulle, la situation est claire et il n'y a rien à démontrer. Sans perte de généralité, on peut supposer que c_{23} n'est pas nul ; sinon, on peut se ramener à ce cas en renumérotant les éléments de la base. Cherchons une décomposition sous la forme

$$a = a_1 e^1 + a_2 e^2 \text{ et } b = b_1 e^1 + b_3 e^3.$$

Pour avoir $a \wedge b = c$, nous devons avoir

$$(a_1 e^1 + a_2 e^2) \wedge (b_1 e^1 + b_3 e^3) = a_1 b_3 e^{13} - a_2 b_1 e^{12} + a_2 b_3 e^{23} = c_1 2 e^{12} + c_{13} e^{13} + c_{23} e^{23},$$

ce qui conduit à un sytème qu'on peut résoudre (de façon non unique) en posant

$$a_1 = c_{13}, \quad a_2 = c_{23}, \quad b_1 = -c_{13}/c_{23}, \quad b_3 = 1.$$

Par contre en dimension $n = 4$, si on choisit

$$c = e^{12} + e^{34}$$

il nous faudrait vérifier à la fois

$$a_1 b_2 - a_2 b_1 = a_3 b_4 - a_4 b_3 = 1, \tag{72}$$

et

$$a_1 b_3 - a_3 b_1 = a_1 b_4 - a_4 b_1 = a_2 b_3 - a_3 b_2 = a_2 b_4 - a_4 b_2 = 0. \tag{73}$$

Les relations (73) impliquent que a et b sont linéairement dépendantes ; par contre, les relations (72) ne peuvent être vraies dans ce cas.

Plus généralement, on a les résultats suivants : si $p \leq n$, on peut définir un produit extérieure de p formes linéaires $a_1, \ldots, a_p$; il est noté

$$a^1 \wedge a^2 \wedge \cdots \wedge a^p$$

et il a les propriétés suivantes : si σ est une permutation de $\{1, \ldots, p\}$, alors

$$a^{\sigma(1)} \wedge a^{\sigma(2)} \wedge \cdots \wedge a^{\sigma(p)} = \epsilon(\sigma) a^1 \wedge a^2 \wedge \cdots \wedge a^p \tag{74}$$

avec $\epsilon(\sigma)$ la signature de la permutation σ. L'espace $\bigwedge^p_{\mathbb{K}}$ a pour base les $e^J = e^{j_1} \wedge e^{j_p}$, avec $J = \{j_1, \ldots, j_p\}$ et $j_1 < j_2 < \cdots < j_p$.

Nous noterons $\mathcal{J}_{p,n}$ l'ensemble des p-uples ordonnés d'indices distincts pris dans $\{1, \ldots, n\}$.

De plus, si λ appartient à $\bigwedge^p$ et μ à $\bigwedge^l$, le produit extérieur de λ et μ est obtenu par linéarité et en utilisant la propriété (74) : on peut écrire

$$\lambda = \sum_{J \in \mathcal{J}_p} \lambda_J e^J \text{ et } \mu = \sum_{K \in \mathcal{J}_q} \mu_K e^K$$

alors

$$\lambda \wedge \mu = \sum_{\substack{J \in \mathcal{J}_{p,n}, K \in \mathcal{J}_{q,n} \\ \mathcal{J}p,n \cap \mathcal{J}q,n = \emptyset}} \lambda_J \mu_k e^J \wedge e^K.$$

Il faut ensuite regrouper les $e^J \wedge e^K$ qui coïncident au signe près. À cette fin, on note *trouver la notation canonique* les permutations σ de $\{1, \ldots, p+q\}$ qui ont la propriété suivante :

$$\sigma(1) < \sigma(2) < \cdots < \sigma(p) \text{ et } \sigma(p+1) < \sigma(p+2) < \cdots < \sigma(p+q).$$

Cet ensemble est appelé l'ensemble des p, q battages (de $p+q$ éléments), par analogie avec ce qu'on fait pour battre des cartes : on sépare le paquet en deux parties et on réintègre les deux parties en les interacalant ; il est noté $\text{battage}_{p,n}$ Le regroupement des termes donne alors

$$(\lambda \wedge \mu)_L = \sum_{\sigma \in \text{battage}_{p,n}} \epsilon(\sigma) \lambda_J \mu_K$$

si L est la liste ordonnée des indices appartenant à J ou à K.

On peut également donner une formule directe du produit extérieure de deux formes de degré respectifs p et q, comme suit : pour tous vecteurs $u_1, \ldots, u_{p+q}$ dans V, on aura

$$(\lambda \wedge \mu)(u_1, \ldots, u_{p+q}) = \sum_{\sigma \in \text{battage}_{p,q}} \epsilon(\sigma) \lambda(u_{\sigma(1)}, u_{\sigma(2)}, \ldots, u_{\sigma(p)}) \mu(u_{\sigma(p+1)}, u_{\sigma(p+2)}, \ldots, u_{\sigma(p+q)}). \tag{75}$$

EXERCICE 9.1. ◁ Combien l'ensemble $\text{battage}_{p,q}$ a-t-il d'éléments ? ▷

EXERCICE 9.2. ◁ Démontrer la formule (75). ▷

Tous ces faits seront démontrés dans la section 4. Mais pour calculer effectivement avec le produit extérieur, ce qui est notre principal objectif maintenant, il suffit de connaître les informations suivantes :

- si V est un espace de dimension n sur $\mathbb{K}$, l'espace des formes p-multilinéaires alternées sur V $\bigwedge^p(V)$ est un espace vectoriel sur $\mathbb{K}$ de dimension $\binom{n}{p}$ si $0 \leq n \leq p$; il est réduit à $\{0\}$ pour $p \geq n+1$.
- pour tous p et q, on peut définir le produit extérieur $\lambda \in \bigwedge^p(V)$ et $\mu \in \bigwedge^q(V)$; ce produit a les propriétés suivantes : il est à valeurs dans $\bigwedge^{p+q}(V)$,

il commute avec la multiplication scalaire et avec l'addition, c.-à-d.

$$(s\lambda) \wedge \mu = \lambda \wedge (s\mu) = s(\lambda \wedge \mu) \tag{76}$$

$$(\lambda + \lambda') \wedge \mu = \lambda \wedge \mu + \lambda' \wedge \mu, \quad \lambda \wedge (\mu + \mu') = \lambda \wedge \mu + \lambda \wedge \mu'. \tag{77}$$

Il est associatif : pour tous $\lambda \in \bigwedge^p(V)$, $\mu \in \bigwedge^q(V)$ et $\nu \in \bigwedge^r(V)$,on a

$$(\lambda \wedge \mu) \wedge \nu = \lambda \wedge (\mu \wedge \nu), \tag{78}$$

et donc on notera systématiquement $\lambda \wedge \mu \wedge \nu$ la valeur des deux membres de (78). De plus il est anticommutatif au sens suivant :

$$\mu \wedge \lambda = (-1)^{pq} \lambda \wedge \mu; \tag{79}$$

c'est le produit des degrés qui apparaît dans l'exposant de -1, parce que c'est la signature de la permutation

$$\begin{pmatrix} 1 & 2 & \dots & p & p+1 & p+2 & \dots & q \\ p+1 & p+2 & \dots & p+q & 1 & 2 & \dots & p \end{pmatrix}.$$

— soit e^i la base duale d'une base de V ; alors une base de $\bigwedge^p(V)$ est donnée par les $e^J = e^{j_1} \wedge e^{j_2} \wedge \cdots \wedge e^{j_p}$, pour $j_1 < j_2 < \cdots < j_p$.

Il est bon de savoir qu'on peut mettre tous les espaces $\bigwedge^p$ dans une grande pochette surprise notée $\bigwedge(V)$; l'espace $\bigwedge(V)$, ou algèbre extérieure de V est la somme formelle de tous les espaces $\bigwedge^p(V)$, pour p variant de 0 à n. Un élément de $\bigwedge(V)$ est donc simplement une somme formelle

$$\lambda_0 + \lambda_1 + \cdots + \lambda_n, \tag{80}$$

chacun des λ_j étant de degré j : c'est une forme j-linéaire alternée.

Notons que l'espace des formes 0-linéaires alternées est le corps des scalaires, $\mathbb{K}$.

On peut additionner les éléments de $\bigwedge(V)$ terme à terme, et les multiplier par un scalaire. On peut aussi les multiplier au moyen du produit extérieur : le terme de degré j ddu produit de (80) avec

$$\mu_0 + \mu_1 + \cdots + \mu_n$$

est simplement

$$\sum_{i=0^j} \lambda_i \wedge \mu_{j-i}.$$

On dira que l'algèbre $\bigwedge(V)$ est une algèbre graduée, parce que c'est une somme directe d'espaces $\bigwedge^p(V)$, qui ont tous un degré : l'entier p, et que le produit envoie $\bigwedge^p(V) \times \bigwedge^q(V)$ dans $\bigwedge^{p+q}(V)$.

Il reste une question importante avant de conclure cette section : l'effet des applications linéaires sur les formes multilinéaires alternées.

Commençons par un cas particulier simple le cas bilinéaire.

Soit A une application linéaire de V dans lui-même. Alors, si a est une forme bilinéaire alternée sur V, il est immédiat que $(u_1, u_2) \mapsto a(Au_1, Au_2)$ est aussi une forme bilinéaire alternée sur V. Nous pouvions d'ailleurs raisonner plus généralement : si A est une application linéaire de W, espace de dimension m sur $\mathbb{K}$ dans V, alors $(u_1, u_2) \mapsto a(Au_1, Au_2)$ est une forme bilinéaire alternée sur W.

L'application qui envoie $a \in \bigwedge^2(V)$ sur $a(A\cdot, A\cdot) \in \bigwedge^2(W)$ sera notée $A \wedge A$ ou $A^{\wedge 2}$. C'est une application linéaire de $\bigwedge^2(V)$ dans $\bigwedge^2(W)$ comme on le vérifie immédiatement (il suffit de se poser la question pour y répondre !)

Nous pouvons trouver la matrice de cette application en supposant que les f_i forment une base de W et les f^i la base duale. Soient en effet $A_i j$ les coefficients de la matrice de A, c'est-à-dire

$$Af_j = \sum e_i A_i j, \quad 1 \le i \le n, \quad 1 \le j \le m.$$

Alors

$$\begin{aligned} a(Af_l, Af_p) &= a\left(\sum_{i=1}^{n} e_i A_{il}, \sum_{j=1}^{n} e_j A_{jp}\right) \\ &\sum_{1\le r<s\le n} a_{rs} e^r \wedge e^s \left(\sum_{i=1}^{n} e_i A_{il}, \sum_{j=1}^{n} e_j A_{jp}\right) \\ &= \sum_{1\le r<s\le n} a_{rs}(A_{rl}A_{sp} - A_{rp}A_{sl}. \end{aligned}$$

Par conséquent la matrice de $A^{\wedge 2}$ est donnée par

$$(A^{\wedge 2})_{lp,rs} = A_{rl}A_{sp} - A_{rp}A_{sl},$$

avec comme bas dans l'espace de départ $\bigwedge^2(V)$ les $e^r \wedge e^s$ pour $1 \le r < s \le n$ et dans l'espace de départ $\bigwedge^2(W)$ les $f^l \wedge f^p$, pour $1 \le l < p \le m$.

Conformément aux règles générales du calcul en algèbre linéaire, on peut écrire maintenant

$$A^{\wedge 2} e^r \wedge e^s = \sum_{1\le l<p\le m} f^l \wedge f^p A_{lp,rs}.$$

On peut bien entendu définir la puissance extérieure de n'importe quelle application linéaire, cette définition étant laissée au lecteur.

Le résultat le plus important dans les applications à venir est celui qui concerne le cas $p = n = m$: soit A une application linéaire de V de dimension n dans lui-même ; quelle est la puissance extérieure p-ième de A ? La réponse est donnée par le lemme suivant :

LEMME 4.1. *La n-ième puissance extérieure d'une application linéaire de V de dimension n dans lui-même s'identifie au déterminant de cette application.*

DÉMONSTRATION. L'espace $\bigwedge^n(V)$ est de dimension 1, parce que $\mathcal{J}_n$ ne contient que l'élément $\{1, 2, \dots, n\}$. Par conséquent, $A^{\wedge n}$, qui est une application linéaire, doit être une multiplication par un scalaire, qu'il s'agit d'identifier. Notons-le, pour le moment, $s(A)$; nous pouvons nous restreindre à travailler avec une forme n-linéaire alternée particulière, la forme $e^N = e^{1,\dots,n}$. nous aurons donc,

$$\big(A^{\wedge n}\big)a^N(e_1, \dots, e_n) = s(A)e^N(e_1, \dots, e_n) = s(A) = e^N(Ae_1, \dots, Ae_n).$$

Soit alors B une autre application linéaire de V dans V ; on remarque que

$$\begin{aligned} \big((AB)^{\wedge n}\big)e^N(e_1, \dots, e_n) &= e^N(ABe_1, \dots ABe_n) \\ &= \big(A^{\wedge n}\big)e^N(Be_1, \dots, Be_n) \\ &= s(a)\big(B^{\wedge n}\big)e^N(e_1, \dots, e_n) \\ &= s(a)s(b). \end{aligned}$$

En d'autres termes, pour toutes applications linéaires A et B, nous avons la relation

$$s(AB) = s(A)s(B). \tag{81}$$

Si A est la matrice P_σ d'une permutation σ de $\{1, \dots, n\}$, il est clair que

$$(P_\sigma^{\wedge n} e^N)(e_1, \dots, e_n) = e^N(e_{\sigma(1)}, \dots, e_{\sigma(n)} = \epsilon(\sigma),$$

donc

$$s(P_\sigma) = \epsilon(\sigma). \tag{82}$$

Si, dans la base des e_i, la matrice de A est triangulaire supérieure, c'est à dire A_ij nul pour $i \geq j$, alors

$$\left(A^{\wedge n}e^N\right)(e_1,\dots,e_n) = e^N\left(\sum_{i=1}^n e_iA_{ij}, \sum_{i=2}^n e_iA_{ij},\dots,A_{nn}e_n\right),$$

comme e^N s'annule quand deux de ses arguments sont répétés, on constate que pour A triangulaire supérieure,

$$s(A) = \prod_{i=1}^n A_{ii} = \det(A).$$

Il en est bien sûr de même pour toute matrice triangulaire supérieure. Nous savons que toute matrice A admet la décomposition

$$A = P_\sigma LU$$

avec P_σ une matrice de permutation, L triangulaire inférieure et U triangulaire supérieure. Les matrices L et U ne sont pas nécessairement inversible : on a ce qu'on appelle la décomposition sous forme échelon *Martin : il serait bon de parler ailleurs de la forme échelon et de référer à cela ici.* Par conséquent,

$$s(A) = \epsilon(\sigma)\det(L)\det(U) = \det(P_\sigma LU) = \det(A).$$

□

5. Calcul extérieur : les formes différentielles

Si on sait ce qu'est un champ de vecteurs, on sait ce qu'est une forme différentielle : c'est un champ de vecteurs un peu particuliers, puisque les vecteurs sont des formes multilinéaires alternées ; par conséquent une forme différentielle ω de degré p sur un ouvert $\mathcal{O}$ de $\mathbb{R}^n$ est un champ de vecteur sur $\mathcal{O}$, à valeurs dans $\bigwedge^p(\mathbb{R}^n)$. On note $\Omega^p(\mathcal{O})$ cet espace. Pour bien faire, il faudrait également donner sa régularité ; dans tout ce qui suit, nous ne considérerons que des formes différentielles lisses, c'est à dire aussi régulière que nécessaire dans les calculs ; si on tient à préciser, elles seront infinment différentiables par rapport à la variable dans $\mathcal{O}$ et le plus souvent analytiques réelles dans les calculs à venir.

Du point de vue fonctionnel, une forme ω de degré p est une bestiole qui peut agir sur $p+1$ arguments de dimension n : on peut déjà mettre en évidence sa valeur en $x \in \mathcal{O}$; on peut aussi la faire agir sur p vecteurs $u_1,\dots,u_p$ de $\mathbb{R}^n$; on obtient ainsi une fonction à valeurs réelles

$$x \mapsto \omega(x)(u_1,\dots,u_p);$$

si les vecteurs u_j dépendent de façon lisse de x, on peut même définir la fonction scalaire

$$x \mapsto \omega(x)(u_1(x),\dots,u_p(x)).$$

On peut définir sur les formes différentielles toutes les opérations qu'on a défini sur les formes multilinéaires alternées : il suffit de les définir ponctuellement. Par exemple si ω et ω' sont dans $\Omega^p(\mathcal{O})$ et si s est un scalaire quelconque,

$$(s\omega+\omega')(x) = s\omega(x)+\omega'(x).$$

De même, si ω est dans $\Omega^p(\mathcal{O})$ et μ dans $\Omega^q(\mathcal{O})$, alors $\omega\wedge\mu$ est défini par

$$(\omega\wedge\mu)(x) = \omega(x)\wedge\mu(x).$$

Bien entendu, on va retrouver toutes les propriétés du produit extérieur des formes multilinéaires alternées sur les formes différentielles.

Il y a une opération supplémentaire qu'on peut faire sur les formes différentielles : la dérivation extérieure. Plutôt que de faire une description formelle, nous allons

aborder progressivement cet opérateur, et voir l'intérêt de bien savoir le manipuler pour faire des calculs effectifs.

Soient $x^1, \dots, x^n$ les coordonnées dans $\mathbb{R}^n$; la base canonique associée est notée e_i, et sa base duale e^i, comme dans la section 4.

Si f est une fonction lisse sur $\mathcal{O}$, donc une fonction des n variables $x^1, \dots, x^n$, sa dérivée extérieure est par définition la forme de degré 1 suivante.

$$\mathrm{d}f(x) = \sum_{i=1}^{n} \frac{\partial f}{\partial x^i}(x) e^i. \tag{83}$$

L'espace des fonctions sur $\mathcal{O}$ peut être noté $\Omega^0(\mathcal{O})$, puisque les scalaires sont des formes 0-linéaires alternées. Donc l'opérateur d envoie $\Omega^0(\mathcal{O})$ dans $\Omega^1(\mathcal{O})$.

Maintenant, il y a des fonctions particulières qui jouent un rôle fondamental : les fonctions coordonnées ; en effet, $x \mapsto x^i$ est la i-ème coordonnée du vecteur x, c'est clairement une fonction lisse sur $\mathcal{O}$, et nous pouvons calculer immédiatement sa dérivée extérieure :

$$dx^i = e^i. \tag{84}$$

La relation (84) est tellement simple et commode qu'on utilisera uniquement désormais la notation dx^i à la place de e^i, et si nos coordonnées ne s'appellent pas x^i, les éléments de la base duale adopteront le nom des coordonnées correspondantes.

Nous allons voir immédiatement la commodité de cette notation.

Prenons pour exemple les coordonnées polaires :

$$x = r\cos\theta, \quad y = r\sin\theta.$$

Si nous considérons x et y comme des fonctions de r et θ, alors nous pouvons calculer $\mathrm{d}x$ et $\mathrm{d}y$, au moyen de la formule (83) :

$$\mathrm{d}x = \cos\theta \mathrm{d}r - r\sin\theta \mathrm{d}\theta,$$
$$\mathrm{d}y = \sin\theta \mathrm{d}r + r\cos\theta \mathrm{d}\theta.$$

On a bien compris dans la section 4 que le produit extérieur a à voir avec le déterminant, et donc avec le volume ; tentons de calculer $\mathrm{d}x \wedge \mathrm{d}y$ grâce à la formule ci-dessus, en tenant compte de ce que, bien évidemment, $\mathrm{d}r \wedge \mathrm{d}r$ et $\mathrm{d}\theta \wedge \mathrm{d}\theta$ s'annulent :

$$\mathrm{d}x \wedge \mathrm{d}y = \cos^2\theta r \mathrm{d}r \wedge \mathrm{d}\theta - \sin^\theta r \mathrm{d}\theta \wedge \mathrm{d}r = r\mathrm{d}r \wedge \mathrm{d}\theta.$$

On constate que la simple manipulation des produits extérieurs de formes différentielles permet de trouver le jacobien du changement de coordonnées $(r, \theta) \mapsto (x, y)$.

C'est une situation générale :

THÉORÈME 5.1. *Soit ϕ une application lisse d'un ouvert $\mathcal{O}$ de $\mathbb{R}^n$ dans $\mathbb{R}^n$. Si l'on note $x^1, \dots, x^n$ les coordonnées dans $\mathcal{O}$ et $y^j = \phi^j(x)$ les coordonnées dans l'image, alors le jacobien de la transformation ϕ est donné par le coefficient de $\mathrm{d}x^1 \wedge \dots \mathrm{d}x^n$ dans l'expression de $\mathrm{d}y^1 \wedge \dots \wedge \mathrm{d}y^n$.*

Ce théorème sera démontré dans la section 7. Pour le moment, il ressemble à une recette de cuisine, et nous allons en voir nombre d'applications. Mais comme nous serons amenés à appliquer la dérivatione extérieure à des formes différentielles de degré plus élevé, nous allons commencer par définir la dérivation extérieure des formes de n'importe quel degré.

La première règle est que d est de carré nul :

$$\mathrm{d}(\mathrm{d}\omega) = 0. \tag{85}$$

La deuxième règle est fondée sur l'idée suivante : si on sait définir la dérivation extérieure d'un produit extérieur de formes, alors on pourra définir la dérivation

extérieure de formes de n'importe quel degré puisque toute forme ω de degré p est la somme d'expressions du type

$$\omega_J \mathrm{d}x^J$$

pour $J \in \mathcal{J}_{p,n}$. Supposons que la forme ω est dans $\Omega^p(\mathcal{O})$ et que la forme μ est dans $\Omega^q(\mathcal{O})$; alors la règle algébrique est la suivante :

$$\mathrm{d}(\omega \wedge \mu) = (xd\omega) \wedge \mu + (-1)^p \omega \wedge (\mathrm{d}\mu). \tag{86}$$

Remarquons avant de poursuivre que si *omega* est de degré 0, c'est simplement une fonction sur $\mathcal{O}$, et il importe que la règle (86) soit encore vraie ; donc par abus de notation, dans ce cas, on notera indifféremment $\omega \wedge \mu$ ou $\omega\mu$. Au cas où μ est aussi de degré 0, on retrouve la règle familière de Leibniz.

Nous démontrerons dans la section 7 que les règles (83), (85) et (86) suffisent à spécifier complètement la dérivée extérieure des formes différentielles, pourvu qu'on fasse l'hypothèse que cet opérateur est linéaire. Cet opérateur applique $\Omega^p(\mathcal{O})$ dans $\Omega^{p+1}(\mathcal{O})$. En particulier, la dérivée extérieure d'une n forme sur $\mathcal{O} \subset \mathbb{R}^n$ est toujours nulle.

Donnons quelques calculs vectoriels classiques dans le langage de la dérivation extérieure.

Soient a_1, a_2 et a_3 des fonctions lisses sur $\mathcal{O} \subset \mathbb{R}^3$; la dérivée extérieure de la forme

$$\omega = a_1 \mathrm{d}x^1 + a_2 \mathrm{d}x^2 + a_3 \mathrm{d}x^3$$

est calculée comme suit :

$$\begin{aligned}
\mathrm{d}(a_1 \mathrm{d}x^1) &= \mathrm{d}(a_1 \wedge \mathrm{d}x^1) \\
&= \mathrm{d}(a_1) \wedge \mathrm{d}x^1 + a_1 \mathrm{d}(\mathrm{d}x^1) \\
&= \left(\sum_{i=1^3} \frac{\partial a_1}{\partial x_i} \mathrm{d}x^i \right) \wedge \mathrm{d}x^1 \\
&= \frac{\partial a_1}{\partial x^2} \mathrm{d}x^2 \wedge \mathrm{d}x^1 + \frac{\partial a_1}{\partial x^3} \mathrm{d}x^3 \wedge \mathrm{d}x^1 ;
\end{aligned}$$

en additionnant les résultats analogues pour les deux autres morceaux de la forme ω, on obtient

$$\mathrm{d}\omega = \left(\frac{\partial a_2}{\partial x^3} - \frac{\partial a_3}{\partial x^2} \right) \mathrm{d}x^2 \wedge \mathrm{d}x^3 + \text{analogues obtenus par permutation circulaire.}$$

On reconnaît que le coefficient de $\mathrm{d}x^2 \wedge \mathrm{d}x^3$ est la première composante du rotationnel du champs de vecteurs $a = (a_1\, a_2\, a_3)^\mathsf{T}$; par permutation circulaire, il en sera de même des deux autres.

Supposons maintenant, a étant comme dans le paragraphe précédent, que la forme ω soit donnée par

$$\omega = a_1 \mathrm{d}x^2 \wedge \mathrm{d}x^3 + a_2 \mathrm{d}x^3 \wedge \mathrm{d}x^1 + a_3 \mathrm{d}x^1 \wedge \mathrm{d}x^2.$$

Calculons la dérivée extérieure du terme $a_1 \mathrm{d}x^1 \wedge \mathrm{d}^2$:

$$\begin{aligned}
\mathrm{d}(a_1 \mathrm{d}x^2 \wedge \mathrm{d}x^3) &= (\mathrm{d}a_1) \wedge (\mathrm{d}x^2 \wedge \mathrm{d}x^3) + a_1 \mathrm{d}(\mathrm{d}x^1 \wedge \mathrm{d}x^2) \\
&= \frac{\partial a_1}{\partial x_1} \mathrm{d}x^1 \wedge \mathrm{d}x^2 \wedge \mathrm{d}x^3.
\end{aligned}$$

En ajoutant les termes analogues obtenus par permutation circulaire, et en tenant compte de la règle (74), nous obtenons

$$\mathrm{d}\omega = (\operatorname{div} a) \mathrm{d}x^1 \wedge \mathrm{d}x^2 \wedge \mathrm{d}x^3.$$

Par conséquent, la dérivation extérieure résume en un seul opérateur les opérateurs de gradient, de rotationnel et de divergence, à condition de choisir les bonnes traductions depuis la version champs de vecteurs vers la version formes différentielles.

Voyons comment les calculs traditionnellement pénibles de changement de variables en coordonnées polaires, cylindriques ou sphériques deviennent incomparablement plus simples en utilisant le formalisme du calcul extérieur.

5.1. Le rotationnel en coordonnées polaires. Supposons a un champ de vecteur de $\mathbb{R}^2$, et cherchons le rotationnel de a en coordonnées polaires. Il est utile de poser

$$X(r,\theta) = \begin{pmatrix} r\cos\theta \\ r\sin\theta \end{pmatrix}.$$

Alors, si $(x_1\, x_2)^{\mathsf{T}} = X(r,\theta)$, nous avons

$$\begin{aligned} &a_1(x^1,x^2)\mathrm{d}x^1 + a_2(x_1,x_2)dx^2 \\ &\quad = a_1\circ X(r,\theta)(\cos\theta \mathrm{d}r - r\sin\theta \mathrm{d}\theta) + a_2\circ X(r,\theta)(\sin\theta \mathrm{d}r + r\cos\theta \mathrm{d}\theta) \\ &\quad = b_1(r,\theta)\mathrm{d}r + b_2(r,\theta)\mathrm{d}\theta, \end{aligned}$$

les fonctions b_1 et b_2 étant données par

$$b_1 = a_1\circ X\cos\theta + a_2\circ X\sin\theta, \quad b_2 = r(-a_1\circ X\sin\theta + a_2\circ X\cos\theta).$$

À ce moment-là,

$$\big(\mathrm{d}(a_1\mathrm{d}x^1 + a_2\mathrm{d}x^2)\big)\circ\phi = \left(\frac{\partial a_2}{\partial x^1} - \frac{\partial a_1}{\partial x^2}\right)\circ\phi \mathrm{d}x^1\wedge\mathrm{d}x^2 = \left(\frac{\partial b_2}{\partial r} - \frac{\partial b_1}{\partial\theta}\right)\mathrm{d}r\wedge\mathrm{d}\theta;$$

mais comme $\mathrm{d}x^1\wedge\mathrm{d}x^2 = r\mathrm{d}r\wedge\mathrm{d}\theta$, on obtient finalement l'expression suivante du rotationnel en coordonnées polaires

$$\frac{1}{r}\frac{\partial}{\partial r}\left[\frac{\partial}{\partial r}\big(r(-a_1\circ\phi\sin\theta + a_2\circ\phi\cos\theta)\big) - \frac{\partial}{\partial\theta}(a_1\circ\phi\cos\theta + a_2\circ\phi\sin\theta)\right]. \tag{87}$$

Étudions un instant cette relation en prenant un point de vue de physicien : les quantités a_1 et a_2 sont mesurées en des unités hypothétiques, les OC ; il faut donc s'attendre à ce que les unités de mesure du rotationnel de a soient des OC/m. L'expression (87) est mesurée en

6. Démonstration complète des résultats 4

7. Annexe : démonstration complète des énoncés de la section 5

Bibliographie

[1] Alfred V. Aho, John E. Hopcroft, and Jeffrey D. Ullman. *The design and analysis of computer algorithms*. Addison-Wesley Publishing Co., Reading, Mass.-London-Amsterdam, 1975. Second printing, Addison-Wesley Series in Computer Science and Information Processing.

[2] Robert R. Bitmead and Brian D. O. Anderson. Asymptotically fast solution of Toeplitz and related systems of linear equations. *Linear Algebra Appl.*, 34 :103–116, 1980.

[3] Lenore Blum, Felipe Cucker, Michael Shub, and Steve Smale. *Complexity and real computation*. Springer-Verlag, New York, 1998. With a foreword by Richard M. Karp.

[4] William L. Briggs, Van Emden Henson, and Steve F. McCormick. *A multigrid tutorial*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, second edition, 2000.

[5] Peter Bürgisser, Michael Clausen, and M. Amin Shokrollahi. *Algebraic complexity theory*, volume 315 of *Grundlehren der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences]*. Springer-Verlag, Berlin, 1997. With the collaboration of Thomas Lickteig.

[6] Henry Cohn, Robert Kleinberg, Balzs Szegedy, and Christopher Umans. A group-theoretic approach to fast matrix multiplication. *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 438–449, 2003.

[7] James W. Cooley and John W. Tukey. An algorithm for the machine calculation of complex Fourier series. *Math. Comp.*, 19 :297–301, 1965.

[8] Don Coppersmith and Shmuel Winograd. Matrix multiplication via arithmetic progressions. *J. Symbolic Comput.*, 9(3) :251–280, 1990.

[9] David A. Cox, John Little, and Donal O'Shea. *Using algebraic geometry*, volume 185 of *Graduate Texts in Mathematics*. Springer, New York, second edition, 2005.

[10] James W. Demmel. *Applied numerical linear algebra*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1997.

[11] Alan Edelman. The complete pivoting conjecture for gaussian elimination. *The Mathematica Journal*, 2(2) :58–61, 1992.

[12] B. Friedlander, M. Morf, T. Kailath, and L. Ljung. New inversion formulas for matrices classified in terms of their distance from Toeplitz matrices. *Linear Algebra Appl.*, 27 :31–60, 1979.

[13] Carl Friedrich Gauss. *Theoria motus corporum coelestium in sectionibus conicis solem ambientium*. Perthes and Besser, Hamburg, Germany, 1809.

[14] Carl Friedrich Gauss. *Theoria combinationis observationum erroribus minimis obnoxiae/Theory of the combination of observations least subject to errors*, volume 11 of *Classics in Applied Mathematics*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1995. Pars prior. Pars posterior. Supplementum/Part One. Part Two. Supplement, Dual Latin-English text, Translated and with an introduction and afterword by G. W. Stewart.

[15] I. Gohberg, T. Kailath, and I. Koltracht. Efficient solution of linear systems of equations with recursive structure. *Linear Algebra Appl.*, 80 :81–113, 1986.

[16] I. Gohberg, T. Kailath, I. Koltracht, and P. Lancaster. Linear complexity parallel algorithms for linear systems of equations with recursive structure. *Linear Algebra Appl.*, 88/89 :271–315, 1987.

[17] I. C. Gohberg and A. A. Semencul. The inversion of finite Toeplitz matrices and their continual analogues. *Mat. Issled.*, 7(2) :201–223, 290, 1972.

[18] Gene H. Golub and Charles F. Van Loan. *Matrix computations*. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, Baltimore, MD, third edition, 1996.

[19] Nick Gould. On growth in Gaussian elimination with complete pivoting. *SIAM J. Matrix Anal. Appl.*, 12(2) :354–361, 1991.

[20] Wolfgang Hackbusch. *Iterative solution of large sparse systems of equations*, volume 95 of *Applied Mathematical Sciences*. Springer-Verlag, New York, 1994. Translated and revised from the 1991 German original.

[21] G. Heinig and V. Olshevsky. The Schur algorithm for matrices with Hessenberg displacement structure. In *Structured matrices in mathematics, computer science, and engineering, II (Boulder, CO, 1999)*, volume 281 of *Contemp. Math.*, pages 3–15. Amer. Math. Soc., Providence, RI, 2001.

[22] Georg Heinig and Karla Rost. *Algebraic methods for Toeplitz-like matrices and operators*, volume 13 of *Operator Theory : Advances and Applications*. Birkhäuser Verlag, Basel, 1984.

[23] Thomas Kailath, Sun Yuan Kung, and Martin Morf. Displacement ranks of matrices and linear equations. *J. Math. Anal. Appl.*, 68(2) :395–407, 1979.

[24] Thomas Kailath and Ali H. Sayed. Displacement structure : theory and applications. *SIAM Rev.*, 37(3) :297–386, 1995.

[25] Igor Kaporin. A practical algorithm for faster matrix multiplication. *Numer. Linear Algebra Appl.*, 6(8) :687–700, 1999.

[26] Rainer Kress. *Numerical analysis*, volume 181 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 1998.

[27] Martin Morf. *Fast algorithms for multivariable systems, Ph.D. Thesis*. Department of Electrical Engineering, Stanford University, Stanford, CA, 1974.

[28] V. Olshevsky and M. Amin Shokrollahi. A displacement approach to decoding algebraic codes. In *Fast algorithms for structured matrices : theory and applications (South Hadley, MA, 2001)*, volume 323 of *Contemp. Math.*, pages 265–292. Amer. Math. Soc., Providence, RI, 2003.

[29] Vadim Olshevsky. Pivoting for structured matrices and rational tangential interpolation. In *Fast algorithms for structured matrices : theory and applications (South Hadley, MA, 2001)*, volume 323 of *Contemp. Math.*, pages 1–73. Amer. Math. Soc., Providence, RI, 2003.

[30] Victor Y. Pan. *Structured matrices and polynomials*. Birkhäuser Boston Inc., Boston, MA, 2001. Unified superfast algorithms.

[31] Sara Robinson. Toward an optimal algorithm for matrix multiplication. *SIAM News*, 38(9), 2005.

[32] Michelle Schatzman. *Analyse numérique*. InterEditions, Paris, 1991. Cours et exercices pour la licence. [Course and exercises for the bachelor's degree].

[33] A. Schönhage and V. Strassen. Schnelle Multiplikation grosser Zahlen. *Computing (Arch. Elektron. Rechnen)*, 7 :281–292, 1971.

[34] Jonathan Richard Shewchuk. An introduction to the conjugate gradient method without the agonizing pain. Technical report, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, États-Unis, August 1994. `www.cs.cmu.edu/~quake-papers/painless-conjugate-gradient.ps`.

[35] G. W. Stewart. On the early history of the singular value decomposition. *SIAM Rev.*, 35(4) :551–566, 1993.

[36] Volker Strassen. Gaussian elimination is not optimal. *Numer. Math.*, 13 :354–356, 1969.

[37] U. Trottenberg, C. W. Oosterlee, and A. Schüller. *Multigrid*. Academic Press Inc., San Diego, CA, 2001. With contributions by A. Brandt, P. Oswald and K. Stüben.

[38] Jacobus Hendricus van Lint. *Introduction to coding theory*, volume 86 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 1982.

[39] J. H. Wilkinson. *The algebraic eigenvalue problem*. Clarendon Press, Oxford, 1965.