Analyse numérique

Gilles Vilmart
Université de Genève

inspiré des polycopiés de
M. Gander, E. Hairer, F. Kwok, B. Vandereycken et G. Wanner

2017 – 2018

Table des matières

Avant-propos

Ces notes du cours “Analyse numérique” révisées récemment par Bart Vandereycken (2015–2018) sont inspirées des polycopiés enseignés par Martin Gander, Gerhard Wanner, Ernst Hairer, Felix Kwok, Gilles Vilmart et Bart Vandereycken pendant plusieurs années à l'université de Genève. Le contenu exigible à l'examen est seulement le matériel présenté en classe (dont les exercices et travaux pratiques) et ces notes complètent le matériel présenté en cours. De plus, elles peuvent être modifiées de façon mineure à tout moment. Vérifiez donc que vous avez une version relativement récente (voir la version au bas de la page).

Nous remercions les assistant(e)s et les étudiant(e)s (en particulier, Sandie Moody, Cyprien Rivier et Marco Sutti) pour leur relecture attentive à la création de ces notes. Nous vous remercions également d'envoyer à

gilles vilmart@unige.ch

toutes suggestions et erreurs, tant typographiques que mathématiques. Bonne lecture et bon semestre!

Gilles Vilmart
Genève, 2017–2018

Chapitre 1

Interpolation polynomiale

1.1 Les polynômes de Lagrange et la formule de Newton

Soit $f : [a,b] \to \mathbb{R}$ une fonction dont on connaît les valeurs $y_i = f(x_i)$ aux points $x_0, x_1, \ldots, x_n \in [a,b]$. On cherche une approximation de f par un polynôme p tel que

$$p(x_i) = y_i, \qquad i = 0, 1, \ldots, n.$$

Sans conditions, ce problème n'est pas bien posé car il existe, par exemple, une infinité de polynômes qui passent par un seul point. Notre premier résultat est l'existence et l'unicité d'un tel polynôme sous certaines conditions.

Théorème 1.1. Soient les $n+1$ points $(x_0, y_0), \ldots, (x_n, y_n)$ où les x_i sont distincts. Alors, il existe un polynôme unique p_n de degré $\leq n$, appelé *le polynôme d'interpolation*, tel que $p_n(x_i) = y_i$ pour $i = 0, 1, \ldots, n$.

Démonstration. Pour l'existence, on considère les *polynômes de Lagrange*

$$\begin{aligned} n = 0 : &\quad \ell_0(x) = 1 \\ n > 0 : &\quad \ell_i(x) = \prod_{\substack{j=0 \\ j \neq i}}^{n} \frac{x - x_j}{x_i - x_j}, \qquad 0 \leq i \leq n \end{aligned}$$

dont le degré de chacun est n et qui satisfont pour $n > 0$

$$\ell_i(x_j) = \begin{cases} 1, & \text{si } i = j, \\ 0, & \text{si } i \neq j. \end{cases}$$

Alors, le polynôme $p_n(x) = \sum_{i=0}^{n} y_i \ell_i(x)$ est de degré n et passe par (x_i, y_i), $i = 0, 1, \ldots, n$.

Pour l'unicité, soient p et q deux polynômes de degrés $\leq n$ qui passent par (x_i, y_i) pour $0 \leq i \leq n$. Alors $d = p - q$ est aussi un polynôme de degré $\leq n$, avec

$$d(x_0) = d(x_1) = \cdots = d(x_n) = 0.$$

Mais un polynôme non-nul de degré $\leq n$ ne peut avoir plus de n racines. On a donc

$$d(x) \equiv 0 \implies p(x) = q(x) \quad \forall x \in \mathbb{R}. \qquad \square$$

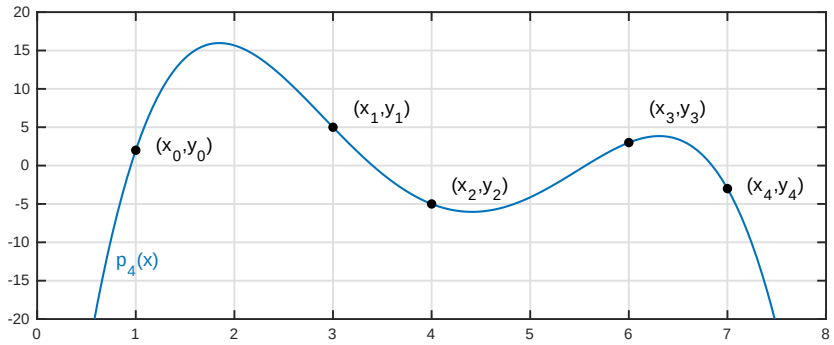


FIGURE 1.1 – Polynôme d'interpolation.

La démonstration précédente nous donne immédiatement une manière d'évaluer $p_n(x)$ en chaque $x \in \mathbb{R}$ par *la formule de Lagrange*

$$p_n(x) = \sum_{i=0}^{n} y_i \ell_i(x) \qquad \text{où} \qquad \ell_i(x) = \prod_{\substack{j=0 \\ j\neq i}}^{n} \frac{x - x_j}{x_i - x_j}. \tag{1.1}$$

Un exemple pour le cas $n = 5$ est donné dans la Fig. 1.1.

La formule (1.1) exprime p_n comme une combinaison linéaire de ℓ_i. Autrement dit, on utilise que l'espace $\mathbb{P}_n$ des polynômes de degré $\leq n$ satisfait

$$\mathbb{P}_n = \text{Vect}\{\ell_0, \ell_1, \ldots, \ell_n\}.$$

Il y a aussi d'autres bases pour $\mathbb{P}_n$, par exemple,

$$p_n(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_0$$

Une expression plus utile est définie par les différences divisées.

Définition 1.2 (différences divisées)**.** Soient (x_i, y_i) pour $i = 0, 1, \ldots, n$ (x_i distincts), on définit

$$\delta y[x_i, x_{i+1}] = \frac{y_{i+1} - y_i}{x_{i+1} - x_i}$$

et pour $k = 2, 3, \ldots$

$$\delta^k y[x_i, \ldots, x_{i+k}] = \frac{\delta^{k-1} y[x_{i+1}, \ldots, x_{i+k}] - \delta^{k-1} y[x_i, \ldots, x_{i+k-1}]}{x_{i+k} - x_i}.$$

Pour simplifier la notation, on utilise aussi $\delta^0 y[x_i] = y[x_i] = y_i$ et $\delta^1 y[x_i, x_{i+1}] = \delta y[x_i, x_{i+1}]$.

On peut montrer (voir la série d'exercices) que les différences divisées sont des fonctions symétriques de leurs arguments ; par exemple, $\delta^3 y[x_1, x_2, x_3] = \delta^3 y[x_2, x_1, x_3] = \delta^3 y[x_3, x_2, x_1]$.

Pour calculer les différences divisées, on utilise le schéma suivant :

$$\begin{array}{c|l} x_0 & y_0 \\ & \searrow \\ x_1 & y_1 \to \delta y[x_0,x_1] \\ & \searrow \quad \searrow \\ x_2 & y_2 \to \delta y[x_1,x_2] \to \delta^2 y[x_0,x_1,x_2] \\ & \searrow \quad \searrow \quad \searrow \\ x_3 & y_3 \to \delta y[x_2,x_3] \to \delta^2 y[x_1,x_2,x_3] \to \delta^3 y[x_0,\ldots,x_3] \end{array} \tag{1.2}$$

Maintenant, cherchons p_n en exprimant

$$\mathbb{P}_n = \text{Vect}\{1, x-x_0, (x-x_0)(x-x_1), \ldots, (x-x_0)\cdots(x-x_{n-1})\}.$$

Théorème 1.3 (Formule de Newton, 1669). Soit

$$p_n(x) = c_0 + c_1(x-x_0) + \cdots + c_n(x-x_0)\cdots(x-x_{n-1})$$

le polynôme unique de degré $\leq n$ passant par les $n+1$ points $(x_0,y_0),(x_1,y_1),\ldots,(x_n,y_n)$ où les x_i sont distincts. Alors

$$c_k = \delta^k y[x_0,\ldots,x_k], \qquad k = 0,1,\ldots,n.$$

Démonstration. On procède par récurrence sur n : pour $n=0$ le théorème est vrai car $c_0 = y[x_0]$ donne $p_0(x) = y_0$. On suppose maintenant que le résultat pour $n-1$ est vrai. Observer que $p_n(x)$ est nécessairement de la forme

$$p_n(x) = p_{n-1}(x) + \alpha(x-x_0)(x-x_1)\cdots(x-x_{n-1}) \tag{1.3}$$

où p_{n-1} est le polynôme de degré $\leq n-1$ qui passe par $(x_0,y_0),\ldots,(x_{n-1},y_{n-1})$ et α est déterminé par $p_n(x_n) = y_n$. Par hypothèse de récurrence, on a

$$p_{n-1}(x) = y[x_0] + \delta y[x_0,x_1](x-x_0) + \cdots + \delta^{n-1} y[x_0,\ldots,x_{n-1}](x-x_0)\cdots(x-x_{n-2}).$$

Il suffit donc de montrer que $\alpha = c_n$.

Pour ce but, considérons le polynôme q_{n-1} de degré $\leq n-1$ qui passe par $(x_1,y_1),\ldots,(x_n,y_n)$,

$$q_{n-1}(x) = y[x_1] + \delta y[x_1,x_2](x-x_1) + \cdots + \delta^{n-1} y[x_1,\ldots,x_n](x-x_1)\cdots(x-x_{n-1}),$$

et observer que p_n peut aussi être écrit comme

$$p_n(x) = \frac{x_n - x}{x_n - x_0} p_{n-1}(x) + \frac{x - x_0}{x_n - x_0} q_{n-1}(x). \tag{1.4}$$

(En effet, comme pour (1.3) il suffit de vérifier que la membre à droite est de degré $\leq n$ et passe par (x_i,y_i), $i = 0,1,\ldots,n$.) Puisque α est le coefficient le plus élevé dans (1.3), c.-à-d.,

$$p_n(x) = \alpha x^n + r_{n-1}(x), \qquad r_{n-1} \in \mathscr{P}_{n-1},$$

il nous reste à déterminer ce coefficient dans (1.4). Par la Déf. 1.2, on obtient

$$\alpha = \frac{-\delta^{n-1} y[x_0,\ldots,x_{n-1}] + \delta^{n-1} y[x_1,\ldots,x_n]}{x_n - x_0} = \delta^n y[x_0,\ldots,x_n] = c_n. \qquad \square$$

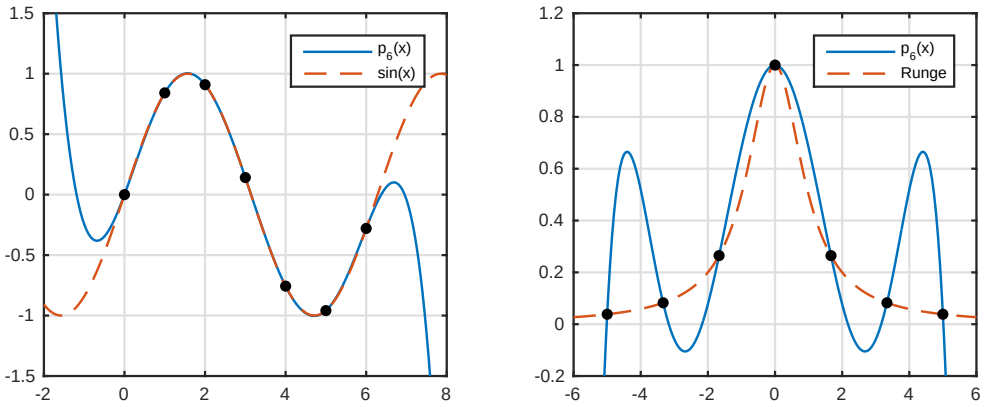


FIGURE 1.2 – Polynôme d'interpolation pour $f(x) = \sin(x)$ et pour la fonction de Runge, $f(x) = 1/(1+x^2)$.

1.2 Erreur de l'interpolation

On aimerait approcher une fonction f par le polynôme d'interpolation p_n qui passe par $(x_0, f(x_0)), \ldots, (x_n, f(x_n))$. Deux exemples sont donnés dans la Fig. 1.2 pour $x_0, \ldots, x_1$ une division équidistante. A gauche, p_n approche bien la fonction $f(x) = \sin(x)$ pour tout $x \in [x_0, x_n] = [0, 6]$ même si le degré est seulement 6. En revanche à droite, p_n n'est pas une bonne approximation de la fonction de Runge, $f(x) = 1/(1+x^2)$.

En général, quelle est l'erreur de l'approximation $f(x) - p_n(x)$ aux points $x \neq x_i$? La formule de Newton et les différences divisées nous aident à l'étudier.

Lemme 1.4. Soit $f\colon [a,b] \to \mathbb{R}$ n fois dérivable et soit $y_i = f(x_i)$ pour $x_0, x_1, \ldots, x_n \in [a,b]$ distincts. Alors il existe $\xi \in [a,b]$ tel que

$$\delta^n y[x_0, \ldots, x_n] = \frac{f^{(n)}(\xi)}{n!}.$$

Démonstration. Soit $p(x) = c_0 + c_1(x - x_0) + \cdots + c_n(x - x_0)\cdots(x - x_{n-1})$ le polynôme d'interpolation de degré $\leq n$ passant par $(x_0, f(x_0)), \ldots, (x_n, f(x_n))$ et notons $d = f - p$. Alors la fonction d a au moins $n+1$ racines distinctes dans $[a,b]$, car elle s'annule en $x_0, \ldots, x_n$. D'après le théorème de Rolle,

$$\begin{aligned} &d' \text{ a au moins } n \text{ zéros distincts dans } (a,b),\\ &d'' \text{ a au moins } n-1 \text{ zéros distincts dans } (a,b),\\ &\quad\vdots\\ &d^{(n)} \text{ a au moins un zéro dans } (a,b). \end{aligned}$$

Notons ce zéro par ξ. En utilisant la formule de Newton, on a alors

$$d^{(n)}(\xi) = 0 \implies f^{(n)}(\xi) = p^{(n)}(\xi) = n!\, c_n \implies \frac{f^{(n)}(\xi)}{n!} = \delta^n y[x_0, \ldots, x_n]. \qquad \square$$

Théorème 1.5. Soit $f\colon [a,b] \to \mathbb{R}$ $n+1$ fois dérivable et soit p le polynôme d'interpolation de degré $\le n$ passant par $(x_0, f(x_0)), \ldots, (x_n, f(x_n))$ (x_i distincts et $x_i \in [a,b]$). Alors pour $x \in [a,b]$, il existe $\xi \in [a,b]$ (qui dépend de x) tel que

$$f(x) - p(x) = (x-x_0)\cdots(x-x_n)\frac{f^{(n+1)}(\xi)}{(n+1)!}. \tag{1.5}$$

Démonstration. La formule est vraie en $x = x_i$ pour $i = 0, 1, \ldots, n$, car les deux côtés s'annulent. Il suffit donc de montrer le résultat pour $x = \bar{x} \in [a,b]$ différent des x_i. L'idée est de considérer le polynôme $\bar{p}(x)$ de degré $\le n+1$ *passant par* $(x_i, f(x_i))$, $0 \le i \le n$ *et aussi par* $(\bar{x}, f(\bar{x}))$. Par la formule de Newton, on a

$$\bar{p}(x) = p(x) + (x-x_0)\cdots(x-x_n)\delta^{n+1}y[x_0, \ldots, x_n, \bar{x}].$$

Mais par le lemme précédent, il existe $\xi \in [a,b]$ t.q.

$$\frac{f^{(n+1)}(\xi)}{(n+1)!} = \delta^{n+1}y[x_0, \ldots, x_n, \bar{x}].$$

On obtient alors

$$f(\bar{x}) = \bar{p}(\bar{x}) = p(\bar{x}) + (\bar{x}-x_0)\cdots(\bar{x}-x_n)\frac{f^{(n+1)}(\xi)}{(n+1)!},$$

c.-à-d. (1.5) pour $x = \bar{x}$. □

Exemple 1.6

Vérifions (1.5) *pour* $\sin(x)$ *dans la situation de la Fig. 1.2 où* $n = 6$. *Comme la* 7^e *dérivée de* $\sin x$ *et majorée par* 1, *on obtient que*

$$|p_6(x) - \sin x| \le |x(x-1)(x-2)\cdots(x-6)| \cdot \tfrac{1}{7!},$$

et ainsi on peut calculer que

$$\max_{x\in[0,6]} |p_6(x) - \sin x| \le 95.85 \cdot \tfrac{1}{7!} \le 0.01902,$$

qui explique la bonne correspondance entre $p_6(x)$ *et* $\sin x$ *dans la figure.*

En revanche, pour le deuxième exemple, $f(x) = 1/(1+x^2)$, *la* 7^e *dérivée est donnée par*

$$f^{(7)}(x) = -40320\frac{x(x^6 - 7x^4 + 7x^2 - 1)}{(1+x^2)^8}$$

qui est maximale pour $x_* \approx \pm 0.1763$ *avec* $|f^{(7)}(x_*)| \le 4392$. *On obtient ainsi*

$$\max_{x\in[-5,5]} \left| p_6(x) - \tfrac{1}{1+x^2} \right| \le 3424 \cdot \tfrac{4392}{7!} \le 2984.$$

Alors, pour la fonction de Runge, on ne peut pas conclure que l'erreur de l'interpolation soit petite ; en effet, la figure montre que ce n'est pas le cas.

1.3 Polynômes de Chebyshev

L'erreur de l'interpolation est un produit de $f^{(n+1)}(\xi)/(n+1)!$ (ξ inconnue) avec l'expression

$$(x-x_0)(x-x_1)\cdots(x-x_n),$$

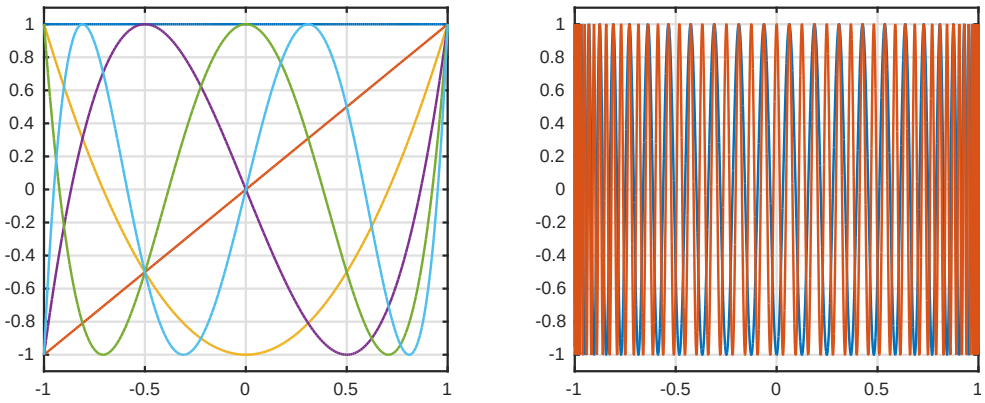


FIGURE 1.3 – Polynômes de Chebyshev : à gauche $T_0, \dots, T_5$ et á droite T_{50} et T_{100}.

qui ne dépend que de la division $x_0, x_1, \dots, x_n$. Un bon choix de la division serait alors $x_i \in [a,b]$ tel que

$$\max_{x\in[a,b]} |(x-x_0)\cdots(x-x_n)| \to \min. \tag{1.6}$$

La réponse à ce problème est donnée à l'aide des polynômes de Chebyshev (1854).

Définition 1.7. Les *polynômes de Chebyshev* $T_n(x)$ sont définis pour $n = 0,1,2,\dots$ par

$$T_n(x) = \cos(n \arccos(x)), \quad \text{pour tout } x \in [-1,1].$$

Quelques exemples de T_n sont visibles dans la figure 1.3.

Malgré sa définition curieuse, T_n est en fait un polynôme comme le théorème suivant le confirme.

Théorème 1.8. Propriétés des polynômes de Chebyshev T_n :

(a) T_n est un polynôme de degré n, et pour $n \geq 1$ de la forme $T_n(x) = 2^{n-1}x^n + \cdots$;

(b) T_n satisfait la récurrence $T_0(x) = 1, T_1(x) = x$,

$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x), \quad n \geq 1;$$

(c) $|T_n(x)| \leq 1$ pour $x \in [-1,1]$;

(d) $T_n\big(\cos \frac{(2k+1)\pi}{2n}\big) = 0$ pour $k = 0,1,\dots,n-1$;

(e) $T_n\big(\cos \frac{k\pi}{n}\big) = (-1)^k$ pour $k = 0,1,\dots,n$.

Démonstration. (b) D'après la définition, $T_0(x) = \cos(0) = 1$ et $T_1(x) = \cos(\arccos(x)) = x$.

Rappelons l'identité trigonométrique

$$\cos((n+1)\alpha) + \cos((n-1)\alpha) = 2\cos(\alpha)\cos(n\alpha).$$

Avec $\alpha = \arccos(x)$ pour $x \in [-1,1]$, nous obtenons

$$T_{n+1}(x) = \cos((n+1)\alpha) = 2\cos(\alpha)\cos(n\alpha) - \cos((n-1)\alpha) = 2xT_n(x) - T_{n-1}(x).$$

(a) Vrai pour T_0, T_1, T_2. Le cas général de T_n est démontré par la récurrence $T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x)$.

(c) D'après la définition et $-1 \le \cos(\alpha) \le 1$.

(d) $\cos(n\alpha) = 0 \Leftrightarrow n\alpha = \frac{(2k+1)\pi}{2}, \forall k \in \mathbb{N}$. Alors, avec $\alpha = \arccos(x)$ on obtient que $x = \cos(\alpha) = \cos\frac{(2k+1)\pi}{2n}$ sont toutes les racines de T_n.

(e) Analogue à (d) en utilisant $\cos(n\alpha) = \pm 1 \Leftrightarrow n\alpha = k\pi, \forall k \in \mathbb{N}$. □

Revenons maintenant au problème de départ qui est de trouver une division satisfaisant (1.6).

Lemme 1.9. Soient $n \ge 1$ et $q(x) = 2^{n-1}x^n + b_{n-1}x^{n-1} + \cdots + b_0 \not\equiv T_n(x)$, alors

$$\max_{x\in[-1,1]} |q(x)| > \max_{x\in[-1,1]} |T_n(x)| = 1.$$

Démonstration. Montrons la chose suivante : si $q(x) = 2^{n-1}x^n + \cdots$ satisfait

$$\max_{x\in[-1,1]} |q(x)| \le 1 = \max_{x\in[-1,1]} |T_n(x)|,$$

alors T_n doit être égal à q. Considérons la différence $d = T_n - q$. D'après (e) du Thm. 1.8

$$d(\cos(k\pi/n)) = (-1)^k - q(\cos(k\pi/n)) \begin{cases} \le 0, & k \text{ impair}, \\ \ge 0, & k \text{ pair}. \end{cases}$$

Alors, d s'annule au moins une fois dans chacun des intervalles fermés

$$[\cos((k+1)\pi/n), \cos(k\pi/n)], \quad k = 0, \ldots, n-1. \tag{1.7}$$

S'il existe une racine $\xi \in (-1,1)$ de d à l'extrémité de (1.7), on a $T_n(\xi) = q(\xi) = \pm 1$. Alors, ξ est un extremum de T_n et de q. Donc $T_n'(\xi) = q'(\xi) = 0 \implies d'(\xi) = 0$, et la multiplicité d'une telle racine est au minimum 2. En comptant les autres racines à l'intérieur de (1.7), on obtient au minimum n zéros (avec multiplicité) pour d. Mais d est un polynôme de degré $\le n-1$ (les coefficients dominants s'annulent). Donc $d(x) \equiv 0 \implies q(x) \equiv T_n(x), \forall x \in \mathbb{R}$. □

Le lemme montre que (1.6) est vrai si et seulement si

$$(x - x_0)\cdots(x - x_n) = \frac{1}{2^n} T_{n+1}(x) \implies x_k = \cos\frac{(2k+1)\pi}{2n+2}, \tag{1.8}$$

c.-à.d., les x_k sont les racines de T_{n+1}. Ces x_k sont appelés *les points de Chebyshev*. Pour l'intervalle $[a,b]$, il faut faire une transformation affine pour envoyer $[-1,1]$ sur $[a,b]$:

$$x \mapsto \frac{b+a}{2} + \frac{b-a}{2}x.$$

Théorème 1.10. L'expression $|(x - x_0)(x - x_1)\cdots(x - x_n)|$ est minimale pour $x \in [a,b]$ parmi toutes les divisions $a \le x_0 < x_1 < \cdots < x_n \le b$ si et seulement si

$$x_k = \frac{b+a}{2} + \frac{b-a}{2}\cos\frac{(2k+1)\pi}{2n+2}, \qquad k = 0, 1, \ldots, n.$$

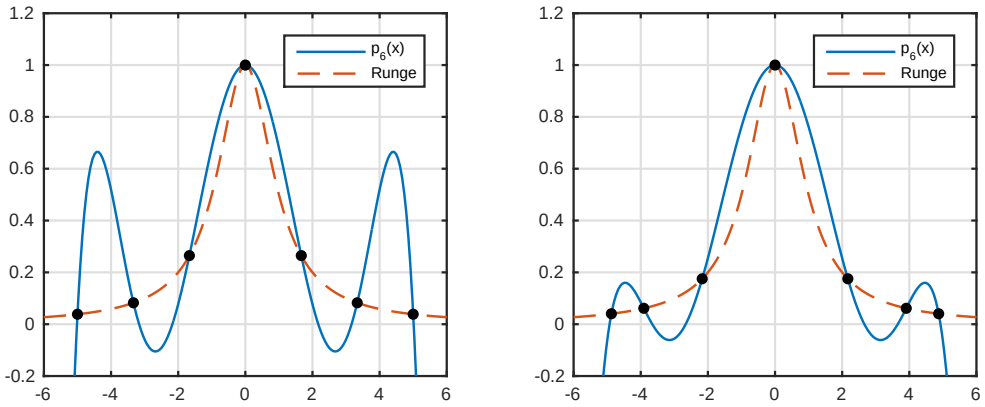


FIGURE 1.4 – Polynôme d'interpolation de la fonction de Runge avec des points équidistants (à gauche) et les points de Chebyshev (à droite).

En comparant les polynômes d'interpolation basés sur les points de Chebyshev avec ceux basés sur les points équidistants, on observe en général une nette amélioration ; voir la Fig. 1.4. En effet, il est possible de démontrer le résultat suivant. [1]

Théorème 1.11. Soit f une fois continûment dérivable sur $[a,b]$ et soit p_n le polynôme d'interpolation passant par $(x_k, f(x_k))$ où

$$x_k = \frac{b+a}{2} + \frac{b-a}{2}\cos\frac{(2k+1)\pi}{2n+2}, \quad k = 0,1,\ldots,n.$$

Alors,

$$\max_{x\in[a,b]} |f(x) - p_n(x)| \to 0 \qquad \text{pour } n \to \infty.$$

Ce théorème nous dit que *la suite de polynômes d'interpolation* (p_n) *avec les points de Chebyshev converge uniformément vers* f *sur l'intervalle d'interpolation* $[a,b]$. Pratiquement, on est sûr que pour chaque f (au moins une fois continûment dérivable), il existe un polynôme d'interpolation (de degré assez grand) qui approche f autant que l'on veut pour tout $x \in [a,b]$. Cette convergence est visible dans la Fig. 1.5.

Un résultat analogue pour les points équidistants n'existe pas. Par exemple, pour la fonction de Runge on peut observer dans la Fig. 1.6 que dans ce cas (p_n) ne converge pas vers f sur $[-5,5]$. Cette divergence avec les points équidistants peut être démontrée rigoureusement et elle est appelée le *phénomène de Runge* (1901).

1.4 Formules barycentriques

Le polynôme d'interpolation p_n est unique, mais il y a plusieurs façons de l'évaluer. La formule de Lagrange (1.1) est simple : pour chaque point x, il faut évaluer $\ell_i(x)$ ce qui demande $O(n)$ opérations. Ensuite, n de celles-là s'additionnent et donnent $p_n(x)$. Alors, le total est $O(n^2)$ opérations pour chaque x.

1. Voir Thm. 7.2 dans Trefethen (2013), *Approximation theory and approximation pratice*, SIAM.

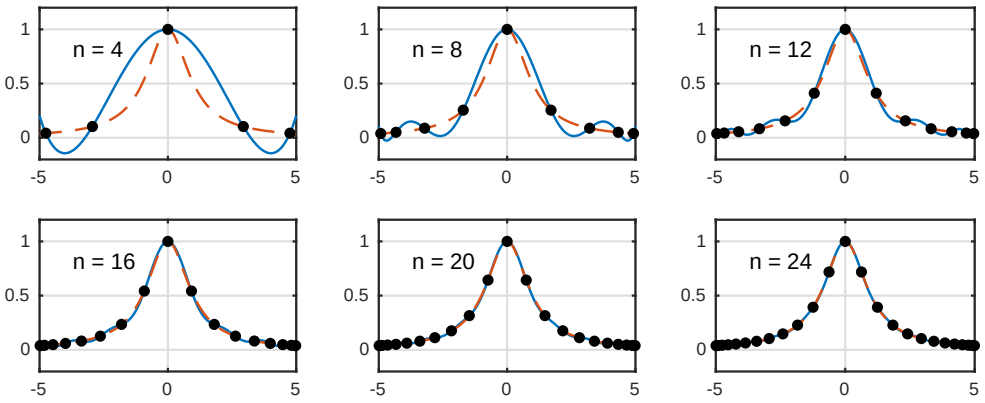


FIGURE 1.5 – Interpolation de la fonction $1/(1+x^2)$ avec des points de Chebyshev.

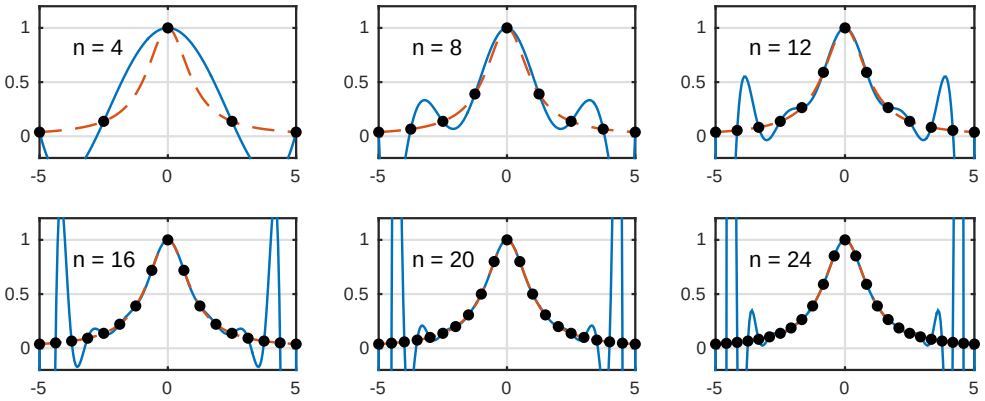


FIGURE 1.6 – Le phénomène de Runge : divergence de p_n pour $1/(1+x^2)$ avec les points équidistants.

Avec peu d'effort, on peut faire mieux. Remplaçons

$$\ell(x) = \prod_{j=0}^{n} (x - x_j), \qquad w_i = 1/\prod_{\substack{j=0 \\ j\neq i}}^{n} (x_i - x_j)$$

dans (1.1), on obtient la *formule barycentrique de la 1re espèce* :

$$p_n(x) = \ell(x) \sum_{i=0}^{n} y_i \frac{w_i}{x - x_i}. \tag{1.9}$$

Il est important de noter que dès qu'on connait les *poids* w_i, on peut évaluer $p_n(x)$ en chaque x en seulement $O(n)$ opérations. Calculer les poids avec la formule ci-dessus coûte $O(n^2)$ opérations mais il faut le faire seulement une fois. En outre, les poids sont parfois connus explicitement. Par exemple, on peut vérifier que pour une division équidistante de $[-1,1]$ avec $n+1$ points

$$w_i = (-1)^{n-i} \frac{n^n}{2^n n!} \binom{n}{i}$$

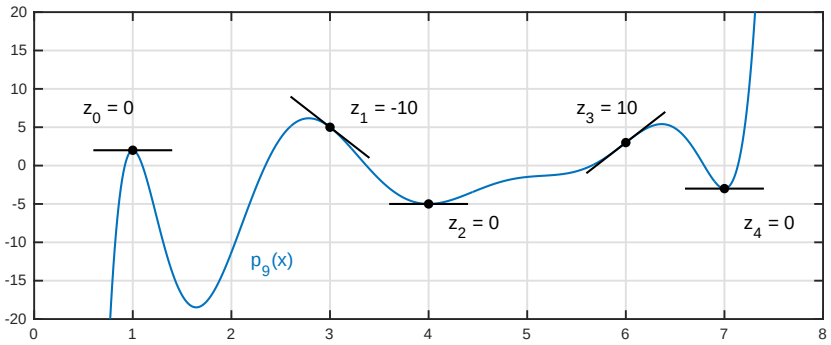


FIGURE 1.7 – Polynôme d'interpolation de Hermite.

et pour les $n+1$ points de Chebyshev (1.8)

$$w_i = C_n\,(-1)^i \sin\frac{(2i+1)\pi}{2n+2}$$

où C_n est une constante qui ne dépend pas de i.

Pour rendre les formules plus symétriques, interpolons la fonction $f(x) \equiv 1$. Parce que le polynôme d'interpolation p_n de celle-ci doit être 1 pour chaque $n \geq 0$ (pourquoi?), on obtient avec (1.9)

$$1 = \ell(x) \sum_{i=0}^{n} \frac{w_i}{x - x_i} \quad \Longrightarrow \quad \ell(x) = 1 \Big/ \sum_{i=0}^{n} \frac{w_i}{x - x_i}, \qquad x \neq x_i.$$

Alors, (1.9) devient la vraie *formule barycentrique* :

$$p_n(x) = \sum_{i=0}^{n} y_i \frac{w_i}{x - x_i} \Bigg/ \sum_{i=0}^{n} \frac{w_i}{x - x_i}\,, \qquad x \neq x_i, \tag{1.10}$$

et $p_n(x_i) = y_i$ pour $x = x_i$.

L'avantage de cette formule est qu'elle est invariante d'échelle. Il n'y a aucun changement si les poids w_i sont remplacés par $c \cdot w_i$, $c \neq 0$. Alors, la formule barycentrique peut utiliser les mêmes w_i quel que soit l'intervalle $[a,b]$. De plus, dans les travaux pratiques on verra que ces deux formules barycentriques ont des propriétés avantageuses dans un contexte numérique.

1.5 Interpolation de Hermite

Le problème d'interpolation de Hermite est une généralisation de celui de Lagrange : fixons une division $x_0, x_1, \ldots, x_n$ d'un intervalle $[a,b]$, on cherche un polynôme p tel que

$$p(x_i) = y_i, \qquad p'(x_i) = z_i, \qquad i = 0, 1, \ldots, n. \tag{1.11}$$

Autrement dit, le polynôme de Hermite n'interpole pas seulement les valeurs y_i mais aussi les dérivées premières z_i en x_i. Voir aussi la Fig. 1.7 où les points (x_i, y_i) sont les mêmes que ceux dans la Fig. 1.1, mais les dérivées en x_i sont clairement différentes.

Puisqu'il y a $2(n+1)$ conditions, on peut soupçonner qu'il existe toujours un polynôme de degré $2n+1$,

$$p_{2n+1}(x) = a_0 + a_1 x + \cdots + a_{2n+1} x^{2n+1},$$

satisfaisant (1.11). Le théorème suivant montre que c'est effectivement le cas.

Théorème 1.12. Soient les points $x_0, x_1, \ldots, x_n$ tous distincts. Soient les points $y_0, y_1, \ldots, y_n$ et $z_0, z_1, \ldots, z_n$. Alors, il existe un polynôme unique p_{2n+1} de degré $\le 2n+1$, appelé *le polynôme d'interpolation d'Hermite*, tel que $p_{2n+1}(x_i) = y_i$ et $p'_{2n+1}(x_i) = z_i$ pour $i = 0, 1, \ldots, n$.

Démonstration. La preuve est similaire à celle de Thm. 1.1. Supposons les polynômes h_i et k_i pour $i = 0, 1, \ldots, n$, qui satisfont les propriétés

$$h_i(x_j) = \begin{cases} 1 & i = j, \\ 0 & i \neq j, \end{cases} \qquad h'_i(x_j) = 0,$$

$$k_i(x_j) = 0, \qquad k'_i(x_j) = \begin{cases} 1 & i = j, \\ 0 & i \neq j. \end{cases}$$

Alors, si h_i et k_i sont de degré $\le 2n+1$, une solution de (1.11) est donnée par

$$p_{2n+1}(x) = \sum_{i=0}^{n} y_i\, h_i(x) + z_i\, k_i(x). \tag{1.12}$$

Après avoir réfléchi un peu, on obtient que

$$h_i(x) = [\ell_i(x)]^2\,[1 - 2\ell'_i(x_i)(x - x_i)]$$
$$k_i(x) = [\ell_i(x)]^2\,(x - x_i)$$

où ℓ_i sont les polynômes de Lagrange.

Pour démontrer l'unicité, soient p et q deux polynômes de degré $\le 2n+1$ qui satisfont (1.11). Alors, le polynôme $d = p - q$ est de dégré $\le 2n+1$ et il a $n+1$ racines distinctes en $x_0, x_1, \ldots, x_n$. D'après le théorème de Rolle, d' a (au moins) n racines qui entrelacent strictement les x_i, c.-à-d., il existe $\xi_i \in (x_i, x_{i+1})$ tel que $d'(\xi_i) = 0$ pour $i = 0, 1, \ldots, n$. En tenant compte de (1.11), on obtient que d' est un polynôme de degré $\le 2n$ avec $2n+1$ racines distincts. Alors, $d' \equiv 0$ et donc $d = p - q$ est constant. Puisque $d(x_i) = 0$, nous trouvons que $p(x) = q(x)$, $\forall\, x \in \mathbb{R}$. □

Au lieu de (1.12), montrons que p_{2n+1} peut aussi être obtenu comme la limite pour $\varepsilon \to 0$ de la suite de polynômes d'interpolation de Lagrange (q_ε) de degré $\le 2n+1$ satisfaisant

$$q_\varepsilon(x_i) = y_i, \qquad q_\varepsilon(x_i + \varepsilon) = y_i + \varepsilon z_i, \qquad i = 0, 1, \ldots, n.$$

Construisons d'abord q_ε avec la formule de Newton pour $\varepsilon \neq 0$. Le schéma (1.2) des différences divisées pour les données ci-dessus devient

$$\begin{array}{c|l}
x_0 & y_0 \\
 & \searrow \\
x_0 + \varepsilon & y_0 + \varepsilon z_0 \to \boxed{\delta y[x_0, x_0 + \varepsilon]} \\
 & \searrow \qquad\qquad \searrow \\
x_1 & y_1 \quad \to \delta y[x_0 + \varepsilon, x_1] \quad \to \quad \delta^2 y[x_0, x_0 + \varepsilon, x_1] \\
 & \searrow \qquad\qquad \searrow \qquad\qquad\qquad \searrow \\
x_1 + \varepsilon & y_1 + \varepsilon z_1 \to \boxed{\delta y[x_1, x_1 + \varepsilon]} \to \delta^2 y[x_0 + \varepsilon, x_1, x_1 + \varepsilon] \to \delta^3 y[x_0, x_0 + \varepsilon, x_1, x_1 + \varepsilon] \\
\vdots & \qquad\qquad\qquad\qquad\qquad\qquad \vdots
\end{array} \tag{1.13}$$

Afin d'obtenir q_*, on prend la limite $\varepsilon \to 0$:

$$\begin{array}{l|l} x_0 & y_0 \\ & \searrow \\ x_0 & y_0 \to \boxed{\delta y[x_0,x_0]} \\ & \searrow \qquad\quad \searrow \\ x_1 & y_1 \to \delta y[x_0,x_1] \to \delta^2 y[x_0,x_0,x_1] \\ & \searrow \qquad\quad \searrow \qquad\quad \searrow \\ x_1 & y_1 \to \boxed{\delta y[x_1,x_1]} \to \delta^2 y[x_0,x_1,x_1] \to \delta^3 y[x_0,x_0,x_1,x_1] \\ \vdots & \qquad\qquad\qquad\quad \vdots \end{array} \tag{1.14}$$

Puisque $x_i \neq x_j$ pour tout $i \neq j$, seulement les termes encadrés sont spéciaux car on ne peut pas directement employer la définition 1.2. En élaborant la limite, on les obtient comme

$$\delta y[x_i,x_i] = \lim_{\varepsilon\to 0} \delta y[x_i,x_i+\varepsilon] = \lim_{\varepsilon\to 0} \frac{(y_i+\varepsilon z_i)-y_i}{(x_i+\varepsilon)-x_i} = z_i.$$

Appelons le polynôme ainsi obtenu q_*, c.-à-d., $\lim_{\varepsilon\to 0} q_\varepsilon(x) = q_*(x)$, $\forall x \in \mathbb{R}$. En utilisant la formule de Taylor,

$$q_\varepsilon(x_i+\varepsilon) = q_\varepsilon(x_i) + \varepsilon q_\varepsilon'(x_i) + \frac{\varepsilon^2}{2} q_\varepsilon''(\xi_\varepsilon) \qquad (\xi_\varepsilon \in [x_i, x_i+\varepsilon])$$

on peut vérifier que q_* satisfait en effet les conditions (1.11) :

$$\begin{aligned} q_*(x_i) &= \lim_{\varepsilon\to 0} q_\varepsilon(x_i) = y_i \\ q_*'(x_i) &= \lim_{\varepsilon\to 0} q_\varepsilon'(x_i) = \lim_{\varepsilon\to 0} \frac{q_\varepsilon(x_i+\varepsilon)-q_\varepsilon(x_i)}{\varepsilon} - \frac{\varepsilon}{2} q_\varepsilon''(\xi_\varepsilon) \\ &= z_i - q_*''(x_i) \lim_{\varepsilon\to 0} \frac{\varepsilon}{2} = z_i. \end{aligned}$$

Puisque q_* est un polynôme de degré $\leq 2n+1$, on a $q_* = p_{2n+1}$ par l'unicité du polynôme d'interpolation de Hermite.

Cette relation avec la formule de Newton est utile pour étudier l'erreur de l'interpolation de Hermite :

Théorème 1.13. Soient $x_0, x_1, \ldots, x_n \in [a,b]$ tous distincts et soit $f\colon [a,b] \to \mathbb{R}$ $2n+2$ fois dérivable. Soit p le polynôme d'interpolation de Hermite qui satisfait (1.11) où $y_i = f(x_i)$ et $z_i = f'(x_i)$. Alors, pour $x \in [a,b]$, il existe $\xi \in [a,b]$ (qui dépend de x) tel que

$$f(x) - p(x) = (x-x_0)^2 \cdots (x-x_n)^2 \frac{f^{(2n+2)}(\xi)}{(2n+2)!}. \tag{1.15}$$

Démonstration. Considérons le polynôme q_ε qui satisfait

$$q_\varepsilon(x_i) = f(x_i), \qquad q_\varepsilon(x_i+\varepsilon) = f(x_i+\varepsilon), \qquad i = 0,1,\ldots,n.$$

D'après le Thm. 1.5, il existe $\xi_\varepsilon \in [x_0, x_n+\varepsilon]$ tel que

$$f(x) - q_\varepsilon(x) = (x-x_0)(x-x_0-\varepsilon)\cdots(x-x_n-\varepsilon)(x-x_n)\frac{f^{(2n+2)}(\xi)}{(2n+2)!}$$

Donc, pour $\varepsilon \to 0$,

$$f(x) - p(x) = (x - x_0)(x - x_0) \cdots (x - x_n)(x - x_n) \frac{f^{(2n+2)}(\xi)}{(2n+2)!}$$

où $p = \lim_{\varepsilon \to 0} q_\varepsilon$. □

Chapitre 2

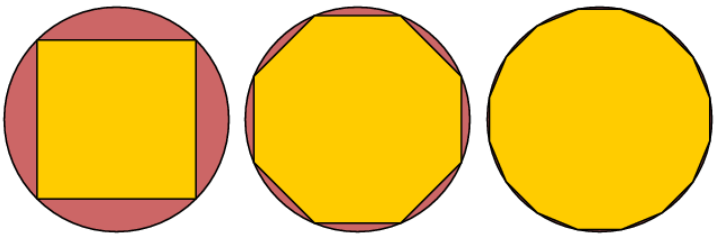
Analyse d'erreurs d'arrondi

2.1 Un exemple : calcul de π

Un problème très ancien, déjà étudié par les grecs, est la quadrature du cercle. On sait depuis Galois que ceci n'est pas possible avec la règle et le compas. Aujourd'hui, nous savons que l'aire A d'un cercle de rayon r est donnée par

$$A = \pi r^2$$

et une approximation peut être obtenue par l'aire des polygones réguliers inscrits dans le cercle comme dessiné dans la figure ci-dessous [1].



En 260 av. J.-C. Archimède a montré, en utilisant un polygone de 96 sommets, que π se trouve dans l'intervalle $(3\frac{1}{7}, 3\frac{10}{71})$ de longueur $1/497 = 0.00201207243$, assez de précision pour les applications de l'époque.

Pour calculer une telle approximation, supposons que $r = 1$. L'aire du triangle dans la Fig. 2.1 satisfait

$$F_n = \cos\frac{\alpha_n}{2}\sin\frac{\alpha_n}{2} = \frac{1}{2}\sin\alpha_n \qquad \text{où } \alpha_n = \frac{2\pi}{n}.$$

L'aire du polygone est donc

$$A_n = nF_n = \frac{n}{2}\sin\frac{2\pi}{n}.$$

Mais on ne connaît ni π, ni $\sin(2\pi/n)$ pour n arbitraire! Heureusement, pour $n = 6$, on sait que

$$\sin\frac{2\pi}{6} = \sin\frac{\pi}{3} = \frac{\sqrt{3}}{2}.$$

On peut donc utiliser la formule du demi-angle

$$\cos\alpha_n = 1 - 2\sin^2(\alpha_n/2)$$

1. Figure adaptée de `http://www.ams.org/samplings/feature-column/fc-2012-02`.

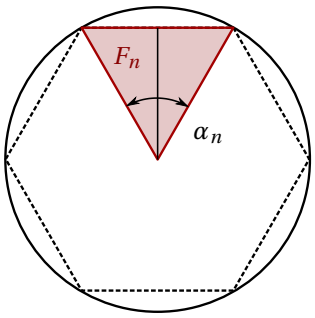


FIGURE 2.1 – Approcher l'aire du cercle par des polygones.

pour calculer $\sin(\alpha_n/2)$ à partir de $\sin\alpha_n$:

$$\sin(\alpha_n/2) = \sqrt{\frac{1-\cos\alpha_n}{2}} = \sqrt{\frac{1-\sqrt{1-\sin^2\alpha_n}}{2}}.$$

Cette formule nous permet de doubler successivement le nombre de sommets du polygone régulier pour calculer les approximations $A_6, A_{12}, A_{24}, A_{48}$ etc. En notant s comme $s_n = \sin\alpha_n$, on a l'algorithme suivant, écrit en langage MATLAB :

```
n = 6; s = sqrt(3)/2;
while n < 1e8
  n = n*2; s = sqrt((1-sqrt(1-s*s))/2);
  A = n*s/2;
   % Afficher l'approximation et l'erreur :
  fprintf('n = %10d, A_n = %.15f,  erreur = %+e \n', n, A, A-pi)
end
```

avec le résultat :

```
n =         12, A_n = 3.000000000000000,  erreur = -1.415927e-01
n =         24, A_n = 3.105828541230250,  erreur = -3.576411e-02
...
n =      24576, A_n = 3.141592618640789,  erreur = -3.494900e-08
n =      49152, A_n = 3.141592645321216,  erreur = -8.268577e-09
n =      98304, A_n = 3.141592645321216,  erreur = -8.268577e-09
n =     196608, A_n = 3.141592645321216,  erreur = -8.268577e-09
n =     393216, A_n = 3.141592645321216,  erreur = -8.268577e-09
n =     786432, A_n = 3.141593669849427,  erreur = +1.016260e-06
n =    1572864, A_n = 3.141592303811738,  erreur = -3.497781e-07
n =    3145728, A_n = 3.141608696224804,  erreur = +1.604264e-05
n =    6291456, A_n = 3.141586839655041,  erreur = -5.813935e-06
n =   12582912, A_n = 3.141674265021758,  erreur = +8.161143e-05
n =   25165824, A_n = 3.141674265021758,  erreur = +8.161143e-05
n =   50331648, A_n = 3.143072740170040,  erreur = +1.480087e-03
n =  100663296, A_n = 3.159806164941135,  erreur = +1.821351e-02
```

Théoriquement, on sait que $\lim_{n\to\infty} A_n = \pi$. En MATLAB on constate pourtant que l'erreur diminue avec n croissant jusqu'à $n = 49152$. Dès lors, l'erreur stagne et finit même par croître. Le but de ce chapitre est d'expliquer comment on peut obtenir avec une formule correcte et un programme correct un résultat faux.

2.2 Les nombres machines et la norme IEEE

Chaque nombre réel $x \in \mathbb{R}$ peut être exprimé en **virgule flottante** (similaire à la notation scientifique) :

$$x = \pm m \times B^e$$

où

— $B \in \mathbb{N}$ est la **base** de représentation (généralement 2 sur ordinateur, 10 sur calculatrice et pour nous) et $0 \leq D < B$ sont les chiffres,
— $e = \pm D\,D\,D\cdots \in \mathbb{Z}$ est l'**exposant**,
— $m = \pm\cdots D\,D\,.\,D\,D\cdots \in \mathbb{R}$ est la **mantisse**.

En général, on choisit de placer une virgule dans la mantisse à une position fixe. Par exemple :

— $m = \pm D\,.\,D\,D\cdots \in \mathbb{R}$ est la mantisse où la virgule est placée juste après le premier chiffre.

Si la virgule est flottante, on peut exprimer la même expression en faisant varier l'exposant :

$$13.254 = \underbrace{13.254}_{m} \times \underbrace{10}_{B}^{\overbrace{0}^{e}} = \underbrace{1.3254}_{m} \times \underbrace{10}_{B}^{\overbrace{1}^{e}} = \underbrace{13254}_{m} \times \underbrace{10}_{B}^{\overbrace{-3}^{e}}.$$

Pour fixer une représentation unique, il faut **normaliser** le nombre. Un choix courant pour $x \neq 0$ est que son premier chiffre est toujours différent de zéro :

$$\pi^4 = \underbrace{97.40909103\ldots}_{\text{dénormalisé}} = \underbrace{9.740909103\ldots \times 10^1}_{\text{normalisé}}$$
$$e^{-10} = \underbrace{0.0004539992976\ldots}_{\text{dénormalisé}} = \underbrace{4.539992976\ldots \times 10^{-5}}_{\text{normalisé}}$$

On appelle le sous-ensemble des nombres réels qui sont ainsi représentables sur un ordinateur les **nombres machines** (noté $\mathbb{M}$). Cette ensemble est fini car la mémoire d'un ordinateur est évidemment finie aussi. Cela veut dire que pour les nombres machines en virgule flottante, la mantisse et l'exposant sont généralement de taille fixée. La **précision** est le nombre de chiffres dans la mantisse et détermine le nombre de **chiffres significatifs** dans $\mathbb{M}$, c.-à-d., la précision relative. L'exposant détermine l'échelle des nombres machines.

Exemple 2.1

Une calculatrice manuelle en base 10 avec 5 chiffres dans la mantisse et 2 chiffres dans l'exposant. Quelques propriétés de $\mathbb{M}$ *:*

— *Le plus grand nombre machine :* $+9.9999 \times 10^{99}$
— *Le plus petit nombre machine :* -9.9999×10^{99}
— *Le plus petit nombre machine positif* normalisé *:* 1.0000×10^{-99}
— *Le plus petit nombre machine positif* dénormalisé *:* 0.0001×10^{-99}
— *Combien de nombres machines y a-t-il au total ?*
  — *Normalisés :* $2 \times 9 \times 10^4 \times 199 = 35\,820\,000$
  — *Dénormalisés :* $2 \times 10^4 - 1 = 19\,999$

Pour représenter un nombre réel $x \notin \mathbb{M}$, il faut l'arrondir au nombre machine *le plus proche*. Cette opération est la fonction $\mathrm{fl}\colon \mathbb{R} \to \mathbb{M}$. Régulièrement, on indique un tel arrondissement avec un tilde, c.-à-d, $\tilde{x} = \mathrm{fl}\,x$.

Appelons l'espacement entre 1 et le premier nombre normalisé dans $\mathbb{M}$ qui suit 1, la **précision de la machine** ou l'**epsilon de la machine**, noté $\varepsilon_{\text{mach}}$. Pour les réels dont la magnitude n'est pas trop grande ni trop petite, l'**erreur d'arrondi relatif** satisfait

$$\frac{|x - \text{fl}\, x|}{|x|} \leq \varepsilon_{\text{mach}}, \qquad x \neq 0.$$

Également,

$$\text{pour tout } x \in \mathbb{R} \text{ tel que } \delta_U < |x| < \delta_O, \text{ il existe } |\varepsilon| \leq \varepsilon_{\text{mach}} \text{ tel que } \text{fl}\, x = x(1+\varepsilon). \tag{2.1}$$

où δ_U et δ_O sont les bornes d'overflow et d'underflow (voir §2.5).

Il y a beaucoup de représentations à virgule flottante possibles, chacune avec leurs propres avantages. Lors de l'introduction des ordinateurs dans les années 1940–1970, de nombreuses représentations ont vu le jour (par exemple, en utilisant base 2, 16, 10 et même 3) mais toutes ces possibilités étaient un frein à la portabilité des programmes d'une machine à l'autre. En 1985, l'IEEE a introduit la norme IEEE 754 qui est utilisée par (presque) tous les ordinateurs d'aujourd'hui.

Exemple 2.2

IEEE ***simple précision*** *(*single precision*) : base* $B = 2$ *sur 32* ***bits*** *(*binary digits*)*

$$\begin{array}{cccc} S & EEEEEEEE & MMMMMM\ldots M \\ 0 & \underbrace{1 \qquad 8}_{e\ (8\ bits)} & \underbrace{9 \qquad\qquad 31}_{m\ (23\ bits)} \end{array}$$

Nombres normalisés *Si* $1 \leq e \leq 254 = 2^8 - 2$*, on obtient des nombres* normalisés *:*

$$\tilde{x} = (-1)^s \times 1.m \times 2^{e-127}$$

- *Le premier bit dans la mantisse n'est pas gardé car il est toujours* 1*. La mantisse contient donc* 24 *bits.*
- *Le plus petit nombre stockésrictement plus grand que 1 est* $1.\underbrace{00\ldots00}_{22}1$ *; la précision relative est donc* $\varepsilon_{mach} = 2^{-23} \approx 1.2 \times 10^{-7}$*.*
- *Un* ***biais*** *de* $b = 2^{8-1} - 1 = 127$ *dans l'exposant existe pour faciliter l'implémentation sur des circuits électroniques.*
- *Le plus grand nombre est* $1.1111\ldots1 \times 2^{254-127} \approx 3.4028 \times 10^{38}$

Nombres dénormalisés : *Les plus petites valeurs des nombres normalisés ne sont pas suffisantes dans bon nombre d'applications. Si* $e = 0$ *et* $m \neq 0$*, les nombres* dénormalisés

$$\tilde{x} = (-1)^s \times 0.m \times 2^{-126}$$

s'ajoutent dans le voisinage de zéro.

Exceptions : *Il est utile de pouvoir représenter* $\pm\infty$*, par exemple en divisant un nombre non nul par* 0*. De la même façon, on peut signaler que* $0/0$ *est une opération non valide.*

- *Si* $e = 255$ *et* $m \neq 0$*, on a un nombre non valide :* $\tilde{x} =$ ***NaN*** *(*Not a Number*)*
- *Si* $e = 255$*,* $m = 0$ *et* $s = 0$*, on a l'infini :* $\tilde{x} =$ ***Inf***
- *Si* $e = 255$*,* $m = 0$ *et* $s = 1$*, on a moins l'infini :* $\tilde{x} =$ ***-Inf***
- *Si* $e = 0$*,* $m = 0$ *et* $s = 1$*, on a* $\tilde{x} = -0$
- *Si* $e = 0$*,* $m = 0$ *et* $s = 0$*, on a* $\tilde{x} = +0$

Par défaut, les nombres machine dans MATLAB sont stockés en 64 bits.

Exemple 2.3

IEEE ***double précision*** *(*double precision*) : base $B = 2$ sur 64 bits.*

$$\begin{array}{cccc} S & EEEEEEEE & MMMMMM\ldots M \\ 0 & \underbrace{1 \qquad 11}_{e\ (11\ bits)} & \underbrace{12 \qquad 63}_{m\ (52\ bits)} \end{array}$$

Les nombres machine en double précision sont plus précis : $\varepsilon_{mach} = 2^{-52} \approx 2.2 \times 10^{-16}$.

2.3 Arithmétique flottante

Soient $\tilde{a}, \tilde{b} \in \mathbb{M}$, en général $c = \tilde{a} \times \tilde{b} \notin \mathbb{M}$ car c contient deux fois plus de chiffres que $\tilde{a}$ et $\tilde{b}$. Puisque c doit être gardé comme un nombre machine, une bonne solution serait de l'arrondir. Par exemple, avec 5 chiffres significatifs en base 10 :

$$c = (1.2345 \times 10^2) \times (6.7890 \times 10^4) = 8.38102050 \times 10^6 \implies \mathrm{fl}\, c = 8.3810 \times 10^6.$$

Alors, d'après (2.1), la multiplication entre deux nombres machine devrait satisfaire

$$\mathrm{fl}(\tilde{a} \times \tilde{b}) = (\tilde{a} \times \tilde{b})(1+\varepsilon), \qquad |\varepsilon| \le \varepsilon_{\mathrm{mach}}.$$

En fait, sur les ordinateurs d'aujourd'hui, toutes les opérations élémentaires $\circledast \in \{+, -, \times, /\}$ satisfont cette propriété importante [2] :

Axiome fondamental d'arithmétique flottante.
Soit $\circledast \in \{+, -, \times, /\}$. Pour tout $\tilde{a}, \tilde{b} \in \mathbb{M}$, il existe $|\varepsilon| \le \varepsilon_{\mathrm{mach}}$ tel que

$$\mathrm{fl}(\tilde{a} \circledast \tilde{b}) = (\tilde{a} \circledast \tilde{b})(1+\varepsilon).$$

Il sera utile de généraliser cette garantie aux toutes opérations élémentaires [3] comme

$$\sin(x), \cos(x), \tan(x), \arcsin(x), \ldots, e^x, \log_e(x), 10^x, \log_{10}(x), \ldots, (1+x)^n, x^{1/n}, x^n, (1+x)^{1/n}, \ldots$$

Observons que l'effet d'arrondir s'accumule en appliquant plusieurs opérations. Par exemple, pour calculer la somme des $a, b \in \mathbb{R}$, il faut d'abord les arrondir :

$$\begin{aligned} a+b \implies \mathrm{fl}(\mathrm{fl}\, a + \mathrm{fl}\, b) &= \mathrm{fl}(a(1+\varepsilon_a) + b(1+\varepsilon_b)), && |\varepsilon_a|, |\varepsilon_b| \le \varepsilon_{\mathrm{mach}} \\ &= (a(1+\varepsilon_a) + b(1+\varepsilon_b))(1+\varepsilon_c), && |\varepsilon_c| \le \varepsilon_{\mathrm{mach}} \\ &= \underbrace{a+b}_{\text{résultat exact}} + \underbrace{a\varepsilon_a + b\varepsilon_b + a\varepsilon_c + b\varepsilon_c + a\varepsilon_a\varepsilon_c + b\varepsilon_b\varepsilon_c}_{\text{perturbation}} \end{aligned}$$

Normalement, les erreurs d'arrondi sont négligeables en double precision car 16 chiffres significatifs ($\varepsilon_{\mathrm{mach}} \approx 10^{-16}$) sont suffisants dans presque toutes les applications. Cependant, il y a deux phénomènes importants pour lesquels le calcul en précision finie est très différent au calcul exact : l'annulation de chiffres significatifs et l'overflow.

2. Nous avons ignoré l'underflow et l'overflow. Dans ce cas, il n'y a pas de borne pour l'erreur d'arrondi relatif.
3. Voir `https://en.wikipedia.org/wiki/IEEE_floating_point` pour la liste complète par rapport à la norme IEEE.

2.4 Annulation de chiffres

Donnons un exemple en base 10 avec 3 chiffres significatifs et 2 chiffres dans l'exposant. Calculer $a^2 - b^2$ avec $a = 4.25$ et $b = 4.23$:

	Arithmétique exacte	Arithmétique flottante
a	4.25	4.25
b	4.23	4.23
a^2	18.0625	1.81×10^1
b^2	17.8929	1.79×10^1
$a^2 - b^2$	0.1696	2.00×10^{-1}

L'erreur d'arrondi est $\approx 18\%$, ce qui est beaucoup plus grand que $\varepsilon_{\text{mach}} = 0.01 = 1\%$. Lorsque l'on soustrait deux nombres proches qui ont déjà été arrondis, le résultat subit une **perte de précision** entraînée par l'extinction des chiffres significatifs.

Deux formules peuvent être égales mais leurs résultats en précision finie ne sont pas les mêmes en général. Pour l'exemple ci-dessus avec $(a-b)(a+b)$ au lieu de $a^2 - b^2$, l'erreur d'arrondi est seulement $\approx 0.2\%$ ce qui est très raisonnable étant donné $\varepsilon_{\text{mach}} = 1\%$.

	Arithmétique exacte	Arithmétique flottante
$a - b$	0.02	2.00×10^{-2}
$a + b$	8.48	8.48×10^0
$(a-b)(a+b)$	0.1696	1.70×10^{-1}

La conclusion est d'éviter la soustraction de nombres du même signe (ou l'addition de nombres de signes opposés) si leurs valeurs absolues sont très proches.

Exemple 2.4

Dans l'exemple du calcul de π, on observe le même phénomène :

$$\sin\frac{\alpha_n}{2} = \sqrt{\frac{1-\sqrt{1-\sin^2\alpha_n}}{2}}$$

Puisque $\alpha_n \to 0$ pour $n \to \infty$, on calcule au numérateur $1 - \sqrt{1-\varepsilon^2}$ avec annulation de chiffres pour ε petit. Pour éviter cette soustraction, on peut réécrire la formule

$$\sin^2\frac{\alpha_n}{2} = \frac{1-\sqrt{1-\sin^2\alpha_n}}{2} \cdot \frac{1+\sqrt{1-\sin^2\alpha_n}}{1+\sqrt{1-\sin^2\alpha_n}} = \frac{\sin^2\alpha_n}{2(1+\sqrt{1-\sin^2\alpha_n})},$$

avec un résultat beaucoup plus précis qu'avant :

```
n =         12, A_n = 3.000000000000000,  erreur = -1.415927e-01
n =         24, A_n = 3.105828541230249,  erreur = -3.576411e-02
...
n =    3145728, A_n = 3.141592653587701,  erreur = -2.091660e-12
n =    6291456, A_n = 3.141592653589268,  erreur = -5.249134e-13
n =   12582912, A_n = 3.141592653589660,  erreur = -1.332268e-13
n =   25165824, A_n = 3.141592653589758,  erreur = -3.552714e-14
```

```
n =   50331648, A_n = 3.141592653589782,  erreur = -1.065814e-14
n =  100663296, A_n = 3.141592653589788,  erreur = -4.884981e-15
```

2.5 Overflow et underflow

Les nombres machine sont bornés mais même si les données et le résultat d'une opération en arithmétique flottante sont représentables par les nombres machines, les quantités intermédiaires peuvent ne pas l'être. Ce phénomène est appelé l'**overflow**.

Continuons l'exemple en base 10 avec 3 chiffres significatifs et 2 chiffres dans l'exposant. Calculer $r = \sqrt{a^2 + b^2}$ avec $a = 3 \times 10^{52}$, $b = 4 \times 10^{52}$

	Arithmétique exacte	Arithmétique flottante
a^2	9×10^{104}	9.99×10^{99}
b^2	1.6×10^{105}	9.99×10^{99}
$a^2 + b^2$	2.5×10^{105}	9.99×10^{99}
$\sqrt{a^2 + b^2}$	5×10^{52}	9.99×10^{49}

L'erreur d'arrondi est tellement grande que le résultat en arithmétique flottante est inutile en général.

La solution est de diviser par la plus grande des deux quantités avant d'élever au carré :

$$r = \sqrt{a^2 + b^2} = b\sqrt{(\tfrac{a}{b})^2 + 1} \qquad \text{où } b = \max\{|a|, |b|\}.$$

avec un résultat dont tous les chiffres significatifs sont (par hasard) corrects :

	Arithmétique exacte	Arithmétique flottante
a/b	0.75	7.50×10^{-1}
$(a/b)^2$	0.5625	5.63×10^{-1}
$(a/b)^2 + 1$	1.5625	1.56×10^{0}
$\sqrt{(a/b)^2 + 1}$	1.25	1.25×10^{0}
$b\sqrt{(a/b)^2 + 1}$	5×10^{52}	5.00×10^{52}

On voit aussi l'**underflow** quand on essaie de représenter un nombre non nul trop petit. Cette situation ne pose presque jamais de problème parce que les nombres dénormalisés donnent beaucoup de précision additionnelle dans le voisinage de zéro. En outre, le zéro est toujours une bonne approximation de tels nombres.

Chapitre 3

Condition et stabilité

Au chapitre précédent, nous avons vu que certaines opérations sont très sensibles aux erreurs d'arrondi. Heureusement, on peut parfois reformuler un calcul afin d'obtenir un résultat assez précis. Mais une telle reformulation est-elle toujours possible et quelle est le meilleur résultat que l'on peut obtenir en calculant la solution d'un problème en arithmétique flottante ?

3.1 Condition d'un problème

Pour mieux comprendre ces questions, considérons abstraitement le calcul d'un problème comme une fonction $f : \mathbb{R}^n \to \mathbb{R}^m$ pour laquelle $f(\boldsymbol{x})$ est la solution pour les données $\boldsymbol{x}$. Par exemple, l'addition de deux nombres mène à la fonction $f(\boldsymbol{x}) = x_1 + x_2$ où $\boldsymbol{x} = (x_1, x_2)$. Remarquons qu'on utilise plutôt la notation $f(x_1, x_2)$ au lieu de $f(\boldsymbol{x})$. Les fonctions f peuvent représenter des problèmes bien plus compliqués comme résoudre des systèmes d'équations, évaluer des polynômes d'interpolation, calculer des intégraux définies, etc. En général, la fonction f est non-linéaire mais elle est normalement au moins continue si le problème est **bien posé au sens d'Hadamard**.

Afin de mesurer les erreurs dans l'espace vectoriel $\mathbb{R}^n$, rappelons la famille des normes p :

$$\|\boldsymbol{x}\|_p = \left(|x_1|^p + |x_2|^p + \cdots + |x_n|^p\right)^{1/p}, \qquad 1 \le p \le \infty,$$

dont les normes suivantes sont les plus utilisées :

la norme euclidienne ($p = 2$) :	$\|\boldsymbol{x}\|_2 = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}$
la norme 1 ($p = 1$) :	$\|\boldsymbol{x}\|_1 = \lvert x_1\rvert + \lvert x_2\rvert + \cdots + \lvert x_n\rvert$
la norme infinie ($p = \infty$) :	$\|\boldsymbol{x}\|_\infty = \max\{\lvert x_1\rvert, \lvert x_2\rvert, \ldots, \lvert x_n\rvert\}$

Il est intéressant d'étudier l'influence de perturbations dans $\boldsymbol{x}$ sur le résultat $f(\boldsymbol{x})$.

Définition 3.1. Soient $\boldsymbol{x} \in \mathbb{R}^n$ et $f : \mathbb{R}^n \to \mathbb{R}^m$ tels que $\boldsymbol{x} \neq \boldsymbol{0}$ et $f(\boldsymbol{x}) \neq \boldsymbol{0}$. La **condition** $\kappa = \kappa(f(\boldsymbol{x}))$ du problème f en $\boldsymbol{x}$ est le plus petit nombre tel que

$$\forall \varepsilon > 0, \forall \boldsymbol{y} \neq \boldsymbol{x}: \qquad \frac{\|\boldsymbol{y} - \boldsymbol{x}\|}{\|\boldsymbol{x}\|} \le \varepsilon \quad \Longrightarrow \quad \frac{\|f(\boldsymbol{y}) - f(\boldsymbol{x})\|}{\|f(\boldsymbol{x})\|} \le \kappa \cdot \varepsilon + o(\varepsilon). \tag{3.1}$$

On dit que le problème f est **bien conditionné** en $\boldsymbol{x}$, si κ n'est pas trop grand (par rapport à $\varepsilon_{\text{mach}}$). Sinon, il est **mal conditionné**.

Ci-dessus, $o(\cdot)$ est le petit-o. On rappelle [1] qu'une fonction $u(\varepsilon) \in o(\varepsilon)$ signifie $u(\varepsilon)/\varepsilon \to 0$ pour $\varepsilon \to 0$. Donc, $o(\varepsilon)$ représente des termes qui sont négligeables par rapport à ε lorsque ε est petit. Par exemple, les fonctions ε^2, $100\varepsilon^{3/2} + \varepsilon^3$ et $10^4\varepsilon^{12}$ sont toutes petit-o de ε.

Remarquons que *la condition κ dépend de la donnée $\boldsymbol{x}$, du problème f, et des normes $\|\cdot\|$ sur $\mathbb{R}^n$ et $\mathbb{R}^m$*, mais qu'elle ne dépend pas de l'algorithme avec lequel on calcule $f(\boldsymbol{x})$. Parfois, on dénote cette dépendance par $\kappa_{f,\boldsymbol{x}}$ ou $\kappa(f(\boldsymbol{x}))$.

Expliquons l'importance de la condition par rapport à l'arithmétique en virgule flottante. Supposons que toutes les composantes d'un vecteur donné $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ sont arrondies vers les nombres machines ce qui produit le vecteur $\tilde{\boldsymbol{x}} = (\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_n)$. D'après (2.1), on obtient qu'un tel $\tilde{\boldsymbol{x}}$ satisfait l'inégalité dans la définition 3.1 pour $\|\cdot\| = \|\cdot\|_1$ et $\varepsilon = \varepsilon_{\text{mach}}$:

$$\|\tilde{\boldsymbol{x}} - \boldsymbol{x}\|_1 = \sum_{i=1}^{n} |\tilde{x}_i - x_i| = \sum_{i=1}^{n} |\operatorname{fl} x_i - x_i| = \sum_{i=1}^{n} |x_i|\,|\varepsilon_i| \le \varepsilon_{\text{mach}} \|\boldsymbol{x}\|_1.$$

Alors, d'après la définition de la condition κ, le résultat *exact* du problème f pour la donnée *perturbée* $\tilde{\boldsymbol{x}}$ satisfait

$$f(\tilde{\boldsymbol{x}}) = f(\boldsymbol{x}) + \|f(\boldsymbol{x})\| \cdot \boldsymbol{\delta}, \quad \boldsymbol{\delta} \in \mathbb{R}^m \text{ t.q. } \|\boldsymbol{\delta}\| \approx \kappa \cdot \varepsilon_{\text{mach}},$$

où nous avons négligé le terme $o(\varepsilon_{\text{mach}})$ car $\varepsilon_{\text{mach}}$ est un nombre très petit. Grâce au terme $\|f(\boldsymbol{x})\|$, ce résultat est une garantie du perturbation au sens relatif. Par exemple, pour $\mathbb{R}^m = \mathbb{R}$ et $\|\cdot\| = |\cdot|$, on a ainsi

$$f(\tilde{\boldsymbol{x}}) = f(\boldsymbol{x})(1 + \delta), \quad |\delta| \approx \kappa \cdot \varepsilon_{\text{mach}}.$$

On constate que si $\kappa \approx 1$ (donc f est bien conditionné), $f(\tilde{\boldsymbol{x}})$ est une perturbation de $f(\boldsymbol{x})$ de la même grandeur que les erreurs d'arrondi, ce qui est très raisonnable pour l'arithmétique en précision finie. Autrement dit, le problème f n'est pas tellement sensible aux erreurs d'arrondi dans la donnée. Cependant, quand $\kappa \gg 1$ (donc f est mal conditionné), l'erreur dans $f(\tilde{\boldsymbol{x}})$ est très grande. Le problème f est donc très sensible aux erreurs d'arrondi.

En résumé, la condition κ mesure la difficulté intrinsèque du problème f. Lorsqu'un problème est mal conditionné, une petite perturbation dans les données (ce qui est presque toujours le cas) peut provoquer des erreurs importantes. Dans ces conditions, il est en général impossible de résoudre le problème avec beaucoup de précision, quel que soit l'algorithme en précision finie utilisé pour calculer la solution.

Trouvons maintenant la condition de l'addition.

Exemple 3.2

L'addition $f(x_1, x_2) = x_1 + x_2$ où $x_1, x_2 \in \mathbb{R}$. Puisque $\|\boldsymbol{y} - \boldsymbol{x}\|_\infty \le \varepsilon \|\boldsymbol{x}\|_\infty$ ssi $|y_i - x_i| \le \varepsilon \|\boldsymbol{x}\|_\infty$, pour tout i, on a

$$\left| \frac{(y_1 + y_2) - (x_1 + x_2)}{x_1 + x_2} \right| = \frac{|(y_1 - x_1) + y_2 - x_2)|}{|x_1 + x_2|} \le \frac{2\varepsilon \|\boldsymbol{x}\|_\infty}{|x_1 + x_2|} = \kappa \cdot \varepsilon.$$

Pour les normes $\|\cdot\|_\infty$ et $|\cdot|$, on a ainsi obtenu $\kappa = 2\max\{|x_1|, |x_2|\}/|x_1 + x_2|$. Si $\operatorname{sgn}(x_1) = \operatorname{sgn}(x_2)$, on a une vraie addition et $\kappa \le 2$; l'addition est donc bien conditionnée. Mais si $\operatorname{sgn}(x_1) = -\operatorname{sgn}(x_2)$, on a effectivement une soustraction. Alors κ devient très grand si $x_1 \approx -x_2$ car $|x_1 + x_2| \ll \max\{|x_1|, |x_2|\}$. Dans ce cas, l'addition est très mal conditionnée.

L'exemple ci-dessus explique l'annulation des chiffres dans le premier tableau de §2.4. Malgré le fait que la calculation en arithmétique flottante de la soustraction entre $\tilde{a}^2 = 18.1$ et

1. Voir aussi le fichier "O et o.pdf" sur Chamilo.

$\tilde{b}^2 = 17.9$ était faite correctement, les nombres $\tilde{a}^2$ et $\tilde{b}^2$ eux-mêmes sont les résultats des arrondissements $\tilde{a}^2 = \mathrm{fl}(18.0625)$ et $\tilde{b}^2 = \mathrm{fl}(17.8929)$. Puisque la condition d'une telle soustraction est assez grande en comparaison de $\varepsilon_{\text{mach}} = 0.01$,

$$\kappa = 2\frac{|18.1|}{|18.1 + (-17.9)|} \approx 181,$$

dans le pire des cas on peut s'attendre à perdre $\log_{10}(181) \approx 2.25$ "chiffres" significatifs pendant le calcul en arithmétique flottante. En effet, il reste seulement un chiffre significatif dans le tableau pour $\mathrm{fl}(\tilde{a}^2 - \tilde{b}^2)$.

3.2 Condition des problèmes différentiables

Le calcul de la condition se simplifie beaucoup si le problème est différentiable. Rappelons (voir le cours d'Analyse) qu'une fonction $f\colon \mathbb{R}^n \to \mathbb{R}^m$ à plusieurs variables est différentiable en $\boldsymbol{x} \in \mathbb{R}^n$ s'il existe une application linéaire $f'(\boldsymbol{x})\colon \mathbb{R}^n \to \mathbb{R}^m$ telle que

$$f(\boldsymbol{y}) = f(\boldsymbol{x}) + f'(\boldsymbol{x})(\boldsymbol{y} - \boldsymbol{x}) + o(\|\boldsymbol{y} - \boldsymbol{x}\|), \qquad \|\boldsymbol{y} - \boldsymbol{x}\| \to 0. \tag{3.2}$$

Cette application linéaire est donnée par la **matrice jacobienne** composée de toutes les dérivées partielles premières de f :

$$f'(\boldsymbol{x}) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1}(\boldsymbol{x}) & \cdots & \frac{\partial f_1}{\partial x_n}(\boldsymbol{x}) \\ \vdots & & \vdots \\ \frac{\partial f_m}{\partial x_1}(\boldsymbol{x}) & \cdots & \frac{\partial f_m}{\partial x_n}(\boldsymbol{x}) \end{bmatrix}, \tag{3.3}$$

où $f(\boldsymbol{x}) = (f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_m(\boldsymbol{x}))$ et $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$. Noter que f est différentiable si toutes ses dérivées partielles premières existent et sont continues.

Pour le théorème 3.5 ci-dessous on a aussi besoin des normes matricielles. Pour une matrice à m lignes et n colonnes,

$$A = \begin{bmatrix} a_{11} & \ldots & a_{1n} \\ \vdots & & \vdots \\ a_{m1} & \ldots & a_{mn} \end{bmatrix} \in \mathbb{R}^{m\times n},$$

on définit la norme d'opérateur :

Définition 3.3 (Norme d'opérateur). Soit $A \in \mathbb{R}^{m\times n}$. On définit

$$\|A\|_{p\to q} = \max_{\|\boldsymbol{x}\|_p = 1} \|A\boldsymbol{x}\|_q = \max_{\boldsymbol{x}\neq \boldsymbol{0}} \frac{\|A\boldsymbol{x}\|_q}{\|\boldsymbol{x}\|_p}, \tag{3.4}$$

c.-à-d., $\|A\|_{p\to q}$ est le plus petit nombre tel que $\|A\boldsymbol{x}\|_q \le \|A\|_{p\to q}\,\|x\|_p$ pour tout $\boldsymbol{x}$.

Remarquons que $\|A\|_{p\to q}$ dépend des normes $\|\cdot\|_p$ et $\|\cdot\|_q$ sur $\mathbb{R}^m$ et $\mathbb{R}^n$. Habituellement, nous écrirons simplement $\|A\|$ si le choix des normes sur $\mathbb{R}^m$ et $\mathbb{R}^n$ est clair. La norme $\|A\|$ satisfait toutes les propriétés d'une norme.

Théorème 3.4 (Propriétés des normes d'opérateurs)**.** Soit $A \in \mathbb{R}^{m\times n}$. Alors, la norme d'opérateur $\|A\|$ est une norme matricielle :

(a) $\|A\| \geq 0$

(b) $\|A\| = 0 \iff A = 0$

(c) $\|A+B\| \leq \|A\| + \|B\|$

(d) $\|\alpha A\| = |\alpha|\|A\|$ pour tout $\alpha \in \mathbb{R}$

(e) $\|A\boldsymbol{x}\| \leq \|A\|\,\|\boldsymbol{x}\|$ pour tout $\boldsymbol{x} \in \mathbb{R}^n$

De plus, elle est sous-multiplicative :

(f) $\|A\cdot B\| \leq \|A\|\,\|B\|$ pour tout $B \in \mathbb{R}^{n\times p}$ et $\|B\|$ une norme d'opérateur

Démonstration. Voir les cours d'Algèbre et d'Analyse. □

Il y a des situations où l'on connaît des formules explicites pour $\|A\|$. Par exemple, si l'on prend la même norme sur $\mathbb{R}^m$ et $\mathbb{R}^n$:

$$\|A\|_{2\to 2} = \|A\|_2 = \sqrt{\text{plus grande valeur propre de } A^T A} \tag{3.5}$$

$$\|A\|_{1\to 1} = \|A\|_1 = \max_{j=1,\dots,n}\left(\sum_{i=1,\dots,m} |a_{ij}|\right)$$

$$\|A\|_{\infty\to\infty} = \|A\|_\infty = \max_{i=1,\dots,m}\left(\sum_{j=1,\dots,n} |a_{ij}|\right) \tag{3.6}$$

Observons que $\|A\|_2 = \|A^T\|_2$ et $\|A\|_1 = \|A^T\|_\infty$.

On peut montrer les formules ci-dessus de la manière suivante. Par exemple, pour $\|A\|_\infty$, on montre d'abord que pour tout $\|\boldsymbol{x}\|_\infty = 1$:

$$|(A\boldsymbol{x})_i| = \left|\sum_{j=1}^n a_{ij}x_j\right| \leq \sum_{j=1}^n |a_{ij}||x_j| \leq \sum_{j=1}^n |a_{ij}|.$$

Ainsi on a $\|A\boldsymbol{x}\|_\infty \leq \max_{1\leq i\leq m}\sum_{j=1}^n |a_{ij}|$. On montre maintenant que cette borne est atteinte. Soit $i = I$ la ligne avec la somme maximale. Posons $\boldsymbol{v} = (\pm 1,\dots,\pm 1)^T$, où on choisit pour v_j le même signe que a_{Ij}. Alors $\|\boldsymbol{v}\|_\infty = 1$ et

$$(A\boldsymbol{v})_I = \sum_{j=1}^n a_{Ij}v_j = \sum_{j=1}^n |a_{Ij}|$$

$$\text{et pour } i \neq I, \qquad (A\boldsymbol{v})_i = \sum_{j=1}^n a_{ij}v_j \leq \sum_{j=1}^n |a_{ij}| \leq \sum_{j=1}^n |a_{Ij}|.$$

On a donc $\|A\|_\infty = \max_{1\leq i\leq m}\sum_{j=1}^n |a_{ij}|$.

Nous avons ainsi le résultat suivant :

Théorème 3.5. Soit $f\colon \mathbb{R}^n \to \mathbb{R}^m$ différentiable. Alors, la condition κ de f en $\boldsymbol{x}$ satisfait

$$\kappa = \limsup_{\varepsilon\to 0}\left\{\frac{\|f(\boldsymbol{y}) - f(\boldsymbol{x})\|}{\varepsilon\|f(\boldsymbol{x})\|} : \frac{\|\boldsymbol{y}-\boldsymbol{x}\|}{\|\boldsymbol{x}\|} \leq \varepsilon\right\} \tag{3.7}$$

$$= \frac{\|f'(\boldsymbol{x})\|}{\|f(\boldsymbol{x})\|/\|\boldsymbol{x}\|}. \tag{3.8}$$

Démonstration. L'égalité (3.7) est directe en utilisant la définition 3.1 :

$$\limsup_{\varepsilon\to 0}\left\{\frac{\|f(\boldsymbol{y})-f(\boldsymbol{x})\|}{\varepsilon\|f(\boldsymbol{x})\|}:\ \frac{\|\boldsymbol{y}-\boldsymbol{x}\|}{\|\boldsymbol{x}\|}\le\varepsilon\right\}=\lim_{\varepsilon\to 0}\frac{\kappa\varepsilon+o(\varepsilon)}{\varepsilon}=\kappa.$$

Ensuite, démontrons (3.8). En utilisant (3.2), on obtient pour $\|\boldsymbol{y}-\boldsymbol{x}\|\to 0$:

$$\|f(\boldsymbol{y})-f(\boldsymbol{x})\|=\|f'(\boldsymbol{x})(\boldsymbol{y}-\boldsymbol{x})\|+o(\|\boldsymbol{y}-\boldsymbol{x}\|).$$

D'après la définition (3.4) d'une norme d'opérateur, on a

$$\sup_{0<\|\boldsymbol{y}-\boldsymbol{x}\|\le\varepsilon\|\boldsymbol{x}\|}\|f'(\boldsymbol{x})(\boldsymbol{y}-\boldsymbol{x})\|=\sup_{\|\boldsymbol{y}-\boldsymbol{x}\|\le\varepsilon\|\boldsymbol{x}\|}\frac{\|f'(\boldsymbol{x})(\boldsymbol{y}-\boldsymbol{x})\|}{\|\boldsymbol{y}-\boldsymbol{x}\|}\|\boldsymbol{y}-\boldsymbol{x}\|=\|f'(\boldsymbol{x})\|\varepsilon\|\boldsymbol{x}\|.$$

Supposons $\varepsilon\to 0$. Alors, $\|\boldsymbol{y}-\boldsymbol{x}\|\to 0$ pour tout $\boldsymbol{y}$ tel que $\|\boldsymbol{y}-\boldsymbol{x}\|\le\varepsilon\|\boldsymbol{x}\|$, et on a ainsi

$$\lim_{\varepsilon\to 0}\frac{o(\|\boldsymbol{y}-\boldsymbol{x}\|)}{\varepsilon}\le\lim_{\|\boldsymbol{y}-\boldsymbol{x}\|\to 0}\|\boldsymbol{x}\|\frac{o(\|\boldsymbol{y}-\boldsymbol{x}\|)}{\|\boldsymbol{y}-\boldsymbol{x}\|}=0.$$

En utilisant les trois résultats ci-dessus, la formule (3.7) devient

$$\begin{aligned}\kappa&=\limsup_{\varepsilon\to 0}\left\{\frac{\|f'(\boldsymbol{x})(\boldsymbol{y}-\boldsymbol{x})\|+o(\|\boldsymbol{y}-\boldsymbol{x}\|)}{\varepsilon\|f(\boldsymbol{x})\|}:\ \frac{\|\boldsymbol{y}-\boldsymbol{x}\|}{\|\boldsymbol{x}\|}\le\varepsilon\right\}\\&=\lim_{\varepsilon\to 0}\frac{\|f'(\boldsymbol{x})\|\varepsilon\|\boldsymbol{x}\|}{\varepsilon\|f(\boldsymbol{x})\|}\\&=\frac{\|f'(\boldsymbol{x})\|}{\|f(\boldsymbol{x})/\|\boldsymbol{x}\|}.\end{aligned}$$

□

Appliquons ce théorème pour vérifier le résultat obtenu dans l'exemple 3.2. Soit $f(x_1,x_2)=x_1+x_2$, alors

$$f'(x_1,x_2)=\begin{bmatrix}\frac{\partial f}{\partial x_1}(x_1,x_2) & \frac{\partial f}{\partial x_2}(x_1,x_2)\end{bmatrix}=\begin{bmatrix}1 & 1\end{bmatrix}$$

et $\|f'(x_1,x_2)\|_1=1$ (pourquoi la norme 1 ?). On obtient ainsi

$$\kappa=\frac{\|f'(x_1,x_2)\|_1}{\|f(x_1,x_2)\|_1/\|(x_1,x_2)\|_1}=\frac{1}{|x_1+x_2|/(|x_1|+|x_2|)}=\frac{|x_1|+|x_2|}{|x_1+x_2|}.$$

Exemple 3.6

Rappelons de §2.1 que le calcul naïf de $f(x)=1-\sqrt{1-\sin^2(x)}$ était problématique en virgule flottante pour $0<x\ll 1$. Calculons donc sa condition afin de juger si cette fonction est vraiment difficile à évaluer numériquement.

Puisque la condition est une propriété mathématique, on peut la calculer également pour $f(x)=1-\cos(x)$. Alors,

$$\kappa(f)=\frac{|f'(x)||x|}{|f(x)|}=\frac{|\sin(x)||x|}{|1-\cos(x)|}=\frac{\sin(x)x}{1-\cos(x)}$$

car $0<x\ll 1$. Prenons la limite, on obtient en utilisant le théorème de Taylor

$$\lim_{x\to 0+}\kappa(f)=\lim_{x\to 0+}\frac{(x+O(x^3))\cdot x}{1-(1-\frac{1}{2}x^2+O(x^4))}=\lim_{x\to 0+}\frac{x^2+O(x^4)}{\frac{1}{2}x^2+O(x^4)}=2\lim_{x\to 0+}\frac{1+O(x^2)}{1+O(x^2)}=2$$

et alors $\kappa(f)\approx 2$ si $0<x\ll 1$. Ce problème est donc bien conditionné.

Un problème est souvent une suite d'autres problèmes. Dans ce cas, on peut borner la condition de leur composition.

Corollaire 3.7. Soient $f_1, f_2, \ldots, f_n$ différentiables. Alors, la condition de $f = f_n \circ f_{n-1} \circ \cdots \circ f_1$ satisfait

$$\kappa(f) \leq \kappa(f_n) \cdot \kappa(f_{n-1}) \cdots \kappa(f_1)$$

Démonstration. La fonction f est différentiable car elle est la composition de fonctions différentiables. On démontre le résultat pour une composition de deux fonctions $f(\boldsymbol{x}) = f_2(f_1(\boldsymbol{x}))$; le cas general en découle par récurrence. D'après Thm. 3.5, on a

$$\kappa(f) = \frac{\|f'(\boldsymbol{x})\|\|\boldsymbol{x}\|}{\|f(\boldsymbol{x})\|}.$$

En utilisant la règle de la chaîne pour $f = f_2 \circ f_1$,

$$f'(\boldsymbol{x}) = f_2'(f_1(\boldsymbol{x})) \cdot f_1'(\boldsymbol{x}),$$

et la propriété (f) dans Thm. 3.4

$$\|f'(\boldsymbol{x})\| \leq \|f_2'(f_1(\boldsymbol{x}))\| \cdot \|f_1'(\boldsymbol{x})\|,$$

la formule pour $\kappa(f)$ ci-dessus devient

$$\begin{aligned}\kappa(f) &\leq \frac{\|f_2'(f_1(\boldsymbol{x}))\|\|f_1'(\boldsymbol{x})\|\|\boldsymbol{x}\|}{\|f_2(f_1(\boldsymbol{x}))\|} \\ &= \frac{\|f_2'(f_1(\boldsymbol{x}))\|\|f_1(\boldsymbol{x})\|}{\|f_2(f_1(\boldsymbol{x}))\|} \cdot \frac{\|f_1'(\boldsymbol{x})\|\|\boldsymbol{x}\|}{\|f_1(\boldsymbol{x})\|} = \kappa(f_2) \cdot \kappa(f_1).\end{aligned}$$

□

3.3 Stabilité des algorithmes

On peut représenter un algorithme pour résoudre le problème f comme la nouvelle fonction $\tilde{f}$ qui est faite sur un ordinateur. Par exemple, $\tilde{f}$ est l'addition en arithmétique flottante mais elle est aussi une suite d'opérations élémentaires $\tilde{f}_i$ (addition, multiplication, soustraction, division, évaluation d'une racine, …) telle que $\tilde{f} = \tilde{f}_n \circ \tilde{f}_{n-1} \circ \cdots \circ \tilde{f}_1$.

À première vue, on pourrait penser que l'erreur d'un bon algorithme devrait être de la même grandeur que $\varepsilon_{\text{mach}}$:

$$\frac{\|\tilde{f}(\boldsymbol{x}) - f(\boldsymbol{x})\|}{\|f(\boldsymbol{x})\|} \leq C \cdot \varepsilon_{\text{mach}}, \quad \forall \boldsymbol{x},$$

où C est une constante petite. Malheureusement, une telle condition est beaucoup trop sévère pour les problèmes mal conditionnés (par exemple, la soustraction) car il faut tenir compte des perturbations dans la donnée $\boldsymbol{x}$. Alors, on a la condition suivante qui est plus réaliste :

Définition 3.8. Un algorithme $\tilde{f}$ pour un problème f est **forward stable** (stable au sens direct) si pour tout $\boldsymbol{x}$:

$$\frac{\|\tilde{f}(\boldsymbol{x}) - f(\boldsymbol{x})\|}{\|f(\boldsymbol{x})\|} \leq C \cdot \kappa_{f,\boldsymbol{x}} \cdot \varepsilon_{\text{mach}} + o(\varepsilon_{\text{mach}}),$$

où C ne dépend pas de $\boldsymbol{x}$.

La condition est une propriété du problème, c.-à-d, elle est un concept mathématique. En revanche, la stabilité est une propriété de l'algorithme, c.-à-d, la manière spécifique dont on choisit d'implémenter un problème par une suite d'opérations élémentaires. Il existe en général plusieurs algorithmes pour résoudre le même problème et leur stabilité n'est pas toujours la même :

Exemple 3.9

Continuons l'exemple 3.6. Calculons pour $0 < x \ll 1$ *(c.-à-d.* $x \to 0+$*) la stabilité de* $f(x) = 1-\sqrt{1-\sin^2(x)}$ *évaluée de la manière suivante :*

$$x \xrightarrow{f_1} \sin(x) \xrightarrow{f_2} \sin^2(x) \xrightarrow{f_3} \sqrt{1-\sin^2(x)} \xrightarrow{f_4} 1-\sqrt{1-\sin^2(x)},$$

où, par exemple, $f_3(y) = \sqrt{1-y}$*. En utilisant les approximations*

$$\sin x \approx x, \qquad \sqrt{1-x} \approx 1-\tfrac{1}{2}x, \qquad (1+m\varepsilon_1)(1+n\varepsilon_2) \approx (1+(m+n)\varepsilon_3), \quad |\varepsilon_i| \le \varepsilon_{mach}$$

on obtient que l'algorithme en arithmétique flottante calcule point par point

$$\begin{aligned}
x &\xrightarrow{\text{fl}} x(1+\varepsilon_1)\\
&\xrightarrow{\tilde{f}_1} \sin(x(1+\varepsilon_1))(1+\varepsilon_2) \approx x(1+\varepsilon_1)(1+\varepsilon_2) \approx x(1+2\varepsilon_3)\\
&\xrightarrow{\tilde{f}_2} x^2(1+2\varepsilon_3)^2(1+\varepsilon_4) \approx x^2(1+5\varepsilon_5)\\
&\xrightarrow{\tilde{f}_3} \sqrt{1-x^2(1+5\varepsilon_5)}(1+\varepsilon_6) \approx (1-\tfrac{1}{2}x^2(1+5\varepsilon_5))(1+\varepsilon_6) \approx 1+\varepsilon_6-\tfrac{1}{2}x^2(1+6\varepsilon_7)\\
&\xrightarrow{\tilde{f}_4} (1-(1+\varepsilon_6-\tfrac{1}{2}x^2(1+6\varepsilon_7)))(1+\varepsilon_8) \approx -\varepsilon_6+\tfrac{1}{2}x^2(1+7\varepsilon_9)
\end{aligned}$$

L'erreur relative de $\tilde{f} = \tilde{f}_4 \circ \cdots \circ \tilde{f}_1 \circ \text{fl}$ *est donc bornée comme*

$$\frac{|\tilde{f}(x)-f(x)|}{|f(x)|} \approx \frac{|-\varepsilon_6+\frac{1}{2}x^2(1+7\varepsilon_9)-\frac{1}{2}x^2|}{|\frac{1}{2}x^2|} \le \frac{\varepsilon_{mach}}{|\frac{1}{2}x^2|}+7\varepsilon_{mach}.$$

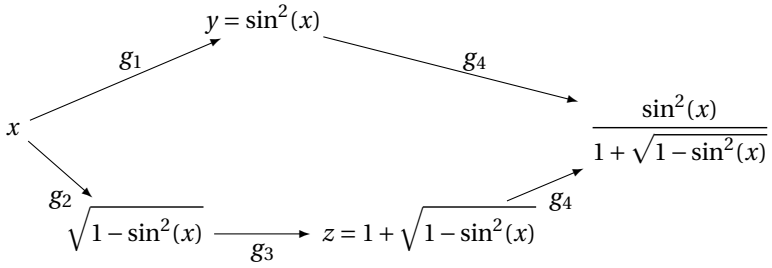
D'après l'exemple 3.6, $\kappa \approx 2$ *si* $x \to 0$*. Alors, cet algorithme n'est pas forward stable.*

Exemple 3.10

Maintenant étudions la stabilité de

$$f(x) = 1-\sqrt{1-\sin^2(x)} = \frac{\sin^2(x)}{1+\sqrt{1-\sin^2(x)}}$$

pour $0 < x \ll 1$ *évaluée comme*



où $g_1(x) = f_2(f_1(x))$ *et* $g_2(x) = f_3(f_2(f_1(x)))$ *dont les fonctions* f_i *sont définies dans l'exemple*

ci-dessus, et $g_4(y,z) = y/z$. D'après cet exemple, on obtient ainsi

$$\begin{aligned} x &\xrightarrow{\tilde{g}_1} x^2(1+5\varepsilon_1) \\ x &\xrightarrow{\tilde{g}_2} 1+\varepsilon_2-\tfrac{1}{2}x^2(1+6\varepsilon_3) \\ &\xrightarrow{\tilde{g}_3} 2+3\varepsilon_4-\tfrac{1}{2}x^2(1+7\varepsilon_5) \\ (y.z) &\xrightarrow{\tilde{g}_4} \frac{x^2(1+5\varepsilon_1)}{2+3\varepsilon_4-\frac{1}{2}x^2(1+7\varepsilon_5)}(1+\varepsilon_6) \approx \tfrac{1}{2}x^2(1+6\varepsilon_7)\left[1-\tfrac{1}{2}(3\varepsilon_4-\tfrac{1}{2}x^2(1+7\varepsilon_5))\right] \\ &\quad Ex/14/serie14.pdf \approx \tfrac{1}{2}x^2(1+7.5\varepsilon_8) \end{aligned}$$

L'erreur relative est donc bornée par $7.5\varepsilon_{mach}$ et cet algorithme est forward stable puisque $\kappa \approx 2$.

En pratique, la vérification de la forward stability est souvent difficile. Heureusement, il existe une autre analyse de stabilité basée sur le principe de Wilkinson.

Définition 3.11 (Principe de Wilkinson)**.** Un algorithme $\tilde{f}$ pour un problème f est **backward stable** si pour tout $\boldsymbol{x}$ il existe $\tilde{\boldsymbol{x}}$ tel que

$$\tilde{f}(\boldsymbol{x}) = f(\tilde{\boldsymbol{x}}) \qquad \text{et} \qquad \frac{\|\tilde{\boldsymbol{x}}-\boldsymbol{x}\|}{\|\boldsymbol{x}\|} \le C\cdot\varepsilon_{\text{mach}} + o(\varepsilon_{\text{mach}}),$$

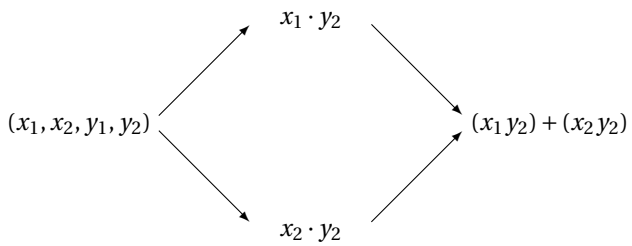
où C ne dépend pas de $\boldsymbol{x}$ (et elle n'est pas trop grande).

Autrement dit, le résultat d'un algorithme qui est backward stable correspond au résultat *exact* pour des données légèrement perturbées :

$$\tilde{f}(\boldsymbol{x}) = f(\boldsymbol{x}+\boldsymbol{\delta}), \qquad \|\boldsymbol{\delta}\| \lessapprox C\varepsilon_{\text{mach}}\|\boldsymbol{x}\|.$$

Exemple 3.12

Le produit scalaire $f(\boldsymbol{x},\boldsymbol{y}) = x_1y_1 + x_2y_2$ où $\boldsymbol{x} = (x_1,x_2)$ et $\boldsymbol{y} = (y_1,y_2)$:



Résultat numérique :

$$\tilde{f}(\boldsymbol{x},\boldsymbol{y}) = [x_1(1+\varepsilon_1)y_1(1+\varepsilon_2)(1+\varepsilon_3) + x_2(1+\varepsilon_4)y_2(1+\varepsilon_5)(1+\varepsilon_6)](1+\varepsilon_7), \qquad |\varepsilon_i| \le \varepsilon_{mach}.$$

Donc $\tilde{f}(\boldsymbol{x},\boldsymbol{y}) = f(\tilde{\boldsymbol{x}},\tilde{\boldsymbol{y}})$, où $\tilde{\boldsymbol{x}} = (\tilde{x}_1,\tilde{x}_2)$ et $\tilde{\boldsymbol{y}} = (\tilde{y}_1,\tilde{y}_2)$ et

$$\begin{aligned} \tilde{x}_1 &= x_1(1+\varepsilon_1)(1+\varepsilon_7), & \tilde{y}_1 &= y_2(1+\varepsilon_2)(1+\varepsilon_3), \\ \tilde{x}_2 &= x_2(1+\varepsilon_4)(1+\varepsilon_7), & \tilde{y}_2 &= y_2(1+\varepsilon_5)(1+\varepsilon_6). \end{aligned}$$

Puisque $(1+\varepsilon_1)(1+\varepsilon_2) \le 1+2\varepsilon_3+o(\varepsilon_{mach})$ où $|\varepsilon_i| \le \varepsilon_{mach}$, on obtient $C = 8$ en choisissant $\|\cdot\|_1$ comme norme. Donc, l'algorithme est backward stable.

Remarquons qu'il n'est pas nécessaire de connaître la condition du problème pour établir la backward stability d'un algorithme. En plus, un algorithme backward stable est aussi forward stable, mais pas nécessairement la réciproque :

Théorème 3.13. Soit $\tilde{f}$ un algorithme qui est backward stable pour le problème f. Alors, $\tilde{f}$ est aussi forward stable.

Démonstration. D'après la définition 3.11 de backward stable, il existe $\tilde{\boldsymbol{x}}$ tel que

$$\tilde{f}(\boldsymbol{x}) = f(\tilde{\boldsymbol{x}}) \qquad \text{et} \qquad \frac{\|\tilde{\boldsymbol{x}} - \boldsymbol{x}\|}{\|\boldsymbol{x}\|} \leq C \cdot \varepsilon_{\text{mach}} + o(\varepsilon_{\text{mach}}).$$

Soit κ la condition de f en $\boldsymbol{x}$. Alors, en utilisant (3.1), on a

$$\frac{\|f(\tilde{\boldsymbol{x}}) - f(\boldsymbol{x})\|}{\|f(\boldsymbol{x})\|} \leq \kappa \cdot C \cdot \varepsilon_{\text{mach}} + o(\varepsilon_{\text{mach}})$$

Puisque $\tilde{f}(\boldsymbol{x}) = f(\tilde{\boldsymbol{x}})$, on obtient que $\tilde{f}$ est forward stable d'après définition 3.8. □

Le résultat suivant est une conséquence directe du fait (vu en cours d'Algèbre) que toutes les normes sur un espace vectoriel de dimension finie sont équivalentes :

Théorème 3.14. Soit f un problème défini de $\mathbb{R}^n$ dans $\mathbb{R}^m$. Le fait que f soit bien ou mal conditionné ne dépend pas du choix des normes sur $\mathbb{R}^n$ et $\mathbb{R}^m$. C'est aussi vrai pour un algorithme qui est backward ou forward stable.

Chapitre 4

Intégration numérique

Etant donnée une fonction continue sur un intervalle borné $f\colon [a,b] \to \mathbb{R}$, on cherche à calculer l'intégrale $\int_a^b f(x)\,dx$.

La plupart des algorithmes numériques procèdent comme suit : on subdivise $[a,b]$ en plusieurs sous-intervalles ($a = x_0 < x_1 < x_2 < \ldots < x_N = b$) et on utilise le fait que

$$\int_a^b f(x)\,dx = \sum_{j=0}^{N-1} \int_{x_j}^{x_{j+1}} f(x)\,dx.$$

De cette manière, on est ramené au calcul de plusieurs intégrales pour lesquelles la longueur de l'intervalle d'intégration est relativement petite. Prenons une de ces intégrales et notons la longueur de l'intervalle par $h_j = x_{j+1} - x_j$. Un changement de variable nous donne alors

$$\int_{x_j}^{x_{j+1}} f(x)\,dx = h_j \int_0^1 f(x_j + t h_j)\,dt.$$

Notons enfin $g(t) = f(x_j + t h_j)$. Il reste alors à calculer une approximation de $\int_0^1 g(t)\,dt$.

4.1 Les formules de Newton–Cotes

L'intégration numérique est basée principalement sur la relation :

$$\int_0^1 g(t)\,dt = \int_0^1 p(t)\,dt + \int_0^1 E(t)\,dt$$

où p est un polynôme (d'interpolation) de g sur $[0,1]$, et $E = g - p$ est l'erreur. Soit p le polynôme de Lagrange qui passe par s couples $(c_i, g(c_i))$, $i = 1, \ldots, s$. En utilisant la formule de Lagrange (1.1), on obtient

$$\int_0^1 p(t)\,dt = \int_0^1 \sum_{i=1}^{s} g(c_i)\ell_i(t)\,dt = \sum_{i=1}^{s} g(c_i) b_i$$

où

$$b_i = \int_0^1 \ell_i(t)\,dt \qquad \text{et} \qquad \ell_i(t) = \prod_{\substack{j=1 \\ j\neq i}}^{s} \frac{t - c_j}{c_i - c_j}. \tag{4.1}$$

Observez que b_i ne dépend pas de g, la fonction qu'on veut intégrer. Ainsi, on a obtenu une construction générale pour approcher des intégrales :

s	ordre	poids b_i						nom
1	2	$\frac{1}{1}$						point milieu
2	2	$\frac{1}{2}$	$\frac{1}{2}$					trapèze
3	4	$\frac{1}{6}$	$\frac{4}{6}$	$\frac{1}{6}$				Simpson
4	4	$\frac{1}{8}$	$\frac{3}{8}$	$\frac{3}{8}$	$\frac{1}{8}$			Newton
5	6	$\frac{7}{90}$	$\frac{32}{90}$	$\frac{12}{90}$	$\frac{32}{90}$	$\frac{7}{90}$		Boole
6	6	$\frac{19}{288}$	$\frac{75}{288}$	$\frac{50}{288}$	$\frac{50}{288}$	$\frac{75}{288}$	$\frac{19}{288}$	—

TABLE 4.1 – Formules de Newton–Cotes

Définition 4.1. Une **formule de quadrature** à s étages est donnée par

$$\int_0^1 g(t)\,dt \approx \sum_{i=1}^{s} b_i \cdot g(c_i). \tag{4.2}$$

Les c_i, supposés distincts, s'appellent des **nœuds** et les b_i des **poids**.

Il y a plusieurs possibilités pour choisir les points d'interpolation (les nœuds) c_i. Les **formules de Newton–Cotes** à s étages s'accordent aux s points équidistants $c_i = (i-1)/(s-1)$, $i = 1,\dots,s$. Quelques-unes d'entre elles sont affichées dans le tableau 4.1. Leurs poids sont calculés d'une manière similaire à ceux de l'exemple suivant.

Exemple 4.2

La formule de Simpson

$$\int_0^1 g(t)\,dt \approx \tfrac{1}{6}(g(0)+4g(1/2)+g(1))$$

est obtenue si l'on fait passer une parabole par les trois points $(0,g(0)),(1/2,g(1/2)),(1,g(1))$. *Le premier poids est calculé par*

$$b_1 = \int_0^1 \ell_1(t)\,dt = \int_0^1 -2(t-\tfrac{1}{2})(-1)(t-1)\,dt = \int_0^1 (t-1)(2t-1)\,dt = \frac{1}{6}.$$

D'une façon similaire, on obtient $b_2 = \frac{4}{6}$ *et* $b_3 = b_1$.

Définition 4.3. La formule de quadrature est dite d'**ordre** au moins p si elle est exacte pour tous les polynômes g de degré $\le p-1$:

$$\int_0^1 g(t)\,dt = \sum_{i=1}^{s} b_i g(c_i), \qquad \forall g \in \mathbb{P}_{p-1}.$$

Rappelons que la formule de Newton–Cotes à s étages utilise implicitement un polynôme de degré $s-1$ pour interpoler l'intégrande g. Son ordre est donc au moins s car il n'y a pas

d'erreur d'interpolation si $g \in \mathbb{P}_{s-1}$. En fait, on gagne un ordre pour s impair grâce à la symétrie des poids, par exemple :

$$\begin{array}{lccccccccccc} \text{nœds :} & 0 \le & c_1 & < & c_2 & < & c_3 = 1/2 & < & 1-c_2 & < & 1-c_1 & \le 1 \\ \text{poids :} & & b_1 & & b_2 & & b_3 & & b_2 & & b_1 & \end{array}$$

Théorème 4.4. Soit $\sum_{i=1}^{s} b_i g(c_i)$ une formule de quadrature **symétrique**, c.-à-d.,

$$c_i = 1 - c_{s-i+1}, \quad b_i = b_{s-i+1},$$

qui est exacte pour les polynômes de degré $2m$. Alors, elle est automatiquement exacte pour les polynômes de degré $2m+1$.

Démonstration. Voir la série d'exercices. □

Remarquons que le théorème ci-dessus est valable pour toutes les formules symétriques, et pas seulement pour celles de Newton–Cotes.

4.2 Erreur des formules composées

Quelle est l'erreur induite par les formules de Newton–Cotes ? Soit p_{s-1} le polynôme d'interpolation qui passe par $(c_i, g(c_i))$, $i = 1, \dots, s$. Alors, l'erreur satisfait

$$|E_s(g,0,1)| = \left| \sum_{i=1}^{s} b_i g(c_i) - \int_0^1 g(t)\, dt \right| = \left| \int_0^1 p_{s-1}(t) - g(t)\, dt \right|.$$

On constate immédiatement que si l'erreur d'interpolation est petite, l'erreur de quadrature est petite aussi. En outre, si g est s fois dérivable, on obtient d'après Thm. 1.5

$$|E_s(g,0,1)| \le \left| \int_0^1 (t-c_1)\cdots(t-c_s) \frac{g^{(s)}(\xi(t))}{s!}\, dt \right| \le \frac{M_s}{s!} \left| \int_0^1 (t-c_1)\cdots(t-c_s)\, dt \right|. \tag{4.3}$$

où $M_s = \max_{\xi\in[0,1]} |g^{(s)}(\xi)|$.

Pour approcher numériquement l'intégrale $\int_a^b f(x)dx$, on peut faire tendre vers l'infini le nombre s de noeuds. Voir, par exemple, le tableau 4.2 à gauche : les formules de Newton–Cotes donnent de bons résultats pour la fonction $\cos(x)$. Malheureusement il existe des fonctions f telles que $\lim_{s\to\infty} E_s(f,a,b) \neq 0$. Prenons, par exemple, la fonction de Runge qui est C^∞. Ses polynômes d'interpolation aux points équidistants ne convergent pas (voir le phénomène de Runge de Chap. 1.1) et, en fait, les formules de Newton–Cotes divergent lorsque $s \to \infty$ (voir aussi le tableau 4.2 à gauche).

Par contre, si l'on procède en découpant l'intervalle $[a,b]$, comme expliqué au début de ce chapitre, on arrive à l'approximation

$$\int_a^b f(x)dx = \sum_{j=0}^{N-1} h_j \int_0^1 f(x_j + t h_j)\, dt \approx \sum_{j=0}^{N-1} h_j \sum_{i=1}^{s} b_i f(x_j + c_i h_j),$$

où $h_j = x_{j+1} - x_j$. On étudie ensuite l'erreur

$$\text{err} = \left| \int_a^b f(x)dx - \sum_{j=0}^{N-1} h_j \sum_{i=1}^{s} b_i f(x_j + c_i h_j) \right|. \tag{4.4}$$

s	I_s	erreur
3	1.920258141329866	$2.666\cdot10^{-1}$
4	1.765414544344723	$1.117\cdot10^{-1}$
5	1.646971707845383	$6.671\cdot10^{-3}$
6	1.649947747627470	$3.695\cdot10^{-3}$
7	1.653780378699500	$1.367\cdot10^{-4}$
8	1.653726754857197	$8.313\cdot10^{-5}$
9	1.653641552528667	$2.068\cdot10^{-6}$
10	1.653642303112869	$1.317\cdot10^{-6}$
11	1.653643644343505	$2.347\cdot10^{-8}$
exact	1.653643620863612	

s	I_s	erreur
3	6.794871794871795	$4.048\cdot10^{+0}$
4	2.081447963800905	$6.653\cdot10^{-1}$
5	2.374005305039788	$3.727\cdot10^{-1}$
6	2.307692307692307	$4.391\cdot10^{-1}$
7	3.870448673470801	$1.123\cdot10^{+0}$
8	2.898994409748379	$1.521\cdot10^{-1}$
9	1.500488907127911	$1.246\cdot10^{+0}$
10	2.398617897841834	$3.481\cdot10^{-1}$
11	4.673300555653497	$1.926\cdot10^{+0}$
exact	2.746801533890032	

TABLE 4.2 – Intégration numérique des integrals $\int_0^4 \cos(x)\,dx$ (à gauche) et $\int_{-5}^5 1/(1+x^2)\,dx$ (à droite) : I_s est le résultat de la formule de Newton–Cotes à s étages.

avec s fixé mais pour $N\to\infty$, ou également, $h=\max_{1\le j<N} h_j\to 0$.

Etudions d'abord l'erreur faite sur un sous-intervalle de longueur h

$$E_s(f,x_0,h)=h\left(\int_0^1 f(x_0+th)\,dt-\sum_{i=1}^{s} b_i f(x_0+c_i h)\right). \tag{4.5}$$

Observez que la borne (4.3) pour E_s montre que les formules de Newton–Cotes à s étages sont d'ordres s, mais pas $s+1$ pour s impair. On peut renforcer la borne de l'erreur de quadrature comme ceci :

Lemme 4.5. Considérons une formule de quadrature d'ordre p. Si $f\colon [x_0,x_0+h]\to\mathbb{R}$ est p fois continûment dérivable :

$$|E_s(f,x_0,h)|\le C\,h^{p+1}\max_{0\le t\le 1}|f^{(p)}(x_0+th)|,$$

où C ne dépend pas de f et de h.

Démonstration. Nous montrons $E_s(f,x_0,h)=Ch^{p+1}|f^{(p)}(x_0)|+O(h^{p+2})$ ce qui établit le théorème seulement pour $h\to 0$. La preuve rigoureuse est bien plus compliquée [1].

Si f est $p+1$ fois continûment dérivable, on peut alors développer $f(x_0+\alpha h)$ en série de Taylor ($0\le\alpha\le 1$) :

$$f(x_0+\alpha h)=\sum_{q=0}^{p}\frac{(\alpha h)^q}{q!}f^{(q)}(x_0)+\frac{(\alpha h)^{p+1}}{(p+1)!}f^{(p+1)}(\xi)$$

où $\xi\in(x_0,x_0+\alpha h)$. En remplaçant $f(x_0+th)$ et $f(x_0+c_ih)$ par de telles séries dans (4.5), on obtient

$$\begin{aligned}E_s(f,x_0,h)&=h\left(\int_0^1\sum_{q=0}^{p}\frac{(th)^q}{q!}f^{(q)}(x_0)\,dt-\sum_{i=1}^{s}b_i\sum_{q=0}^{p}\frac{(c_ih)^q}{q!}f^{(q)}(x_0)+O(h^{p+1})\right)\\&=\sum_{q=0}^{p}\frac{h^{q+1}}{q!}f^{(q)}(x_0)\underbrace{\left(\int_0^1 t^q\,dt-\sum_{i=1}^{s}b_ic_i^q\right)}_{E_s(t^q,0,1)}+O(h^{p+2}).\end{aligned}$$

1. Voir Thm. 2.1 dans la polycopié 2004–05 sur Chamilo.

Puisque la formule de quadrature est d'ordre p, on a pour $0 \le q \le p-1$ que $E_s(t^q, 0, 1)$ s'annule. On obtient ainsi

$$E_s(f, x_0, h) = \frac{h^{p+1}}{p!} f^{(p)}(x_0) \left(\frac{1}{p+1} - \sum_{i=1}^{s} b_i c_i^p \right) + O(h^{p+2})$$

et $C = \frac{1}{p!}\left(\frac{1}{p+1} - \sum_{i=1}^{s} b_i c_i^p\right)$. □

Grâce au résultat du théorème précédent, on peut facilement estimer l'erreur pour l'intervalle entier $[a, b]$.

Théorème 4.6. Soit $f : [a, b] \to \mathbb{R}$ p fois continûment dérivable et soit l'ordre de la formule de quadrature égal à p. Alors, l'erreur (4.4) satisfait

$$\text{err} \le C\, h^p\, (b-a) \max_{a \le x \le b} |f^{(p)}(x)|,$$

où $h = \max_{1 \le j \le N} h_j$, et C ne dépend pas de f et de h.

Démonstration. Comme l'erreur (4.4) est la somme des erreurs sur les sous-intervalles de la division, on obtient en utilisant le lemme 4.5

$$\begin{aligned} \text{err} &\le \sum_{j=0}^{N-1} h_j \left| \int_0^1 f(x_j + t h_j)\, dt - \sum_{i=1}^{s} b_i f(x_j + c_i h_j) \right| = \sum_{j=0}^{N-1} |E_s(f, x_j, h_j)| \\ &\le \sum_{j=0}^{N-1} C\, h_j^{p+1} \max_{0 \le t \le 1} |f^{(p)}(x_j + t h_j)| \\ &\le \sum_{j=0}^{N-1} h_j\, C\, h^p \max_{a \le x \le b} |f^{(p)}(x)| \end{aligned}$$

ce qui montre l'assertion du théorème, car $\sum_{j=0}^{N-1} h_j = b - a$. □

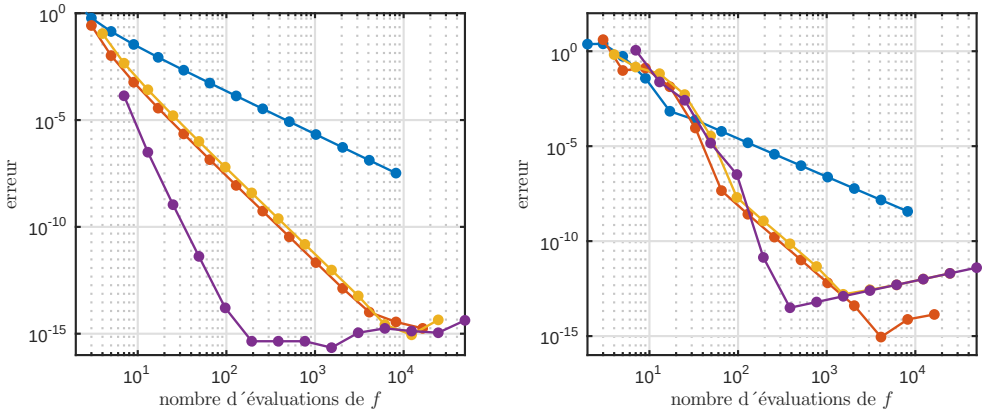
La figure 4.1 montre les résultats numériques pour $N = 1, 2, 4, 8, 16, 32, \ldots$ obtenus par les formules de Newton–Cotes composées à s étages pour les deux intégrales

$$\int_0^4 \cos(x)\, dx \quad \text{(à gauche)} \qquad \text{et} \qquad \int_{-5}^{5} \frac{1}{1+x^2}\, dx \quad \text{(à droite)}.$$

Les sous-intervalles sont équidistants $h_j = h = (b-a)/N$. Le nombre d'évaluations de l'intégrande f est égal à $\text{nf} = N(s-1)+1$ et il représente une mesure pour le travail (proportionnel au temps de calcul sur un ordinateur).

Les résultats de la figure (observez l'échelle logarithmique !) vérifient le théorème ci-dessus :

— le nombre de chiffres exacts, donné par $-\log_{10}(\text{err})$, dépend linéairement de $\log_{10}(\text{nf})$;
— la pente de chaque droite est $-p$ (où p est l'ordre de la formule) ;
— pour un travail équivalent, les formules avec un ordre élevé ont en général une meilleure précision.



Légende : $s = 2$ (ordre 2) $s = 3$ (ordre 4) $s = 4$ (ordre 4) $s = 7$ (ordre 8)

FIGURE 4.1 – L'erreur en fonction du travail pour les formules de Newton-Cotes composées avec le nombre d'étages fixé.

4.3 Formules d'ordre supérieur

Par la construction à l'aide des polynômes de Lagrange, il existe une formule de quadrature (b_i, c_i) à s étages, ayant un ordre $p \geq s$, pour tous les nœuds $c_1, \dots, c_s$ fixés et distincts. On a vu que l'ordre peut être $s+1$ si les nœuds sont bien choisis (par exemple, symétriques). Y a-t-il un choix des c_i permettant d'avoir un ordre supérieur ?

Lemme 4.7 (Jacobi 1826). Soit $(b_i, c_i)_{i=1}^s$ une formule d'ordre $p \geq s$ pour l'intervalle $[0,1]$. Alors, l'ordre est $\geq s+m$ si et seulement si

$$\int_0^1 M(t)q(t)\,dt = 0, \qquad \forall q \in \mathbb{P}_{m-1} \tag{4.6}$$

où $M(t) = (t-c_1)(t-c_2)\cdots(t-c_s)$.

Démonstration. Soit f un polynôme de degré $\leq s+m-1$. Alors on peut diviser f par le polynôme M de degré s pour obtenir

$$f(t) = M(t)q(t) + r(t), \qquad q \in \mathbb{P}_{m-1},\ r \in \mathbb{P}_{s-1}.$$

Comme la formule de quadrature est exacte pour r (l'ordre est $\geq s$ par hypothèse), on a alors

$$\int_0^1 f(t)\,dt = \int_0^1 M(t)q(t)\,dt + \int_0^1 r(t)\,dt = \int_0^1 M(t)q(t)\,dt + \sum_{i=1}^s b_i r(c_i).$$

D'autre part, on a

$$\sum_{i=1}^s b_i f(c_i) = \sum_{i=1}^s b_i \big(\underbrace{M(c_i)}_{=0} q(c_i) + r(c_i) \big) = \sum_{i=1}^s b_i r(c_i).$$

La formule est donc exacte pour f si et seulement si $\int_0^1 M(t)q(t)\,dt = 0$ où q est un polynôme arbitraire de degré $\leq m-1$. □

Quelle est la valeur maximale de m dans le lemme précédent?

Théorème 4.8. L'ordre d'une formule à s étages est $\leq 2s$.

Démonstration. Supposons par l'absurde que l'ordre soit $\geq 2s+1$. Alors, l'intégrale (4.6) s'annule pour tout polynôme $q \in \mathbb{P}_s$. En posant $q = M$, ceci contredit le fait que

$$\int_0^1 M(t)q(t)\,dt = \int_0^1 (M(t))^2\,dt > 0. \qquad \square$$

Nous verrons dans la section suivante qu'il existe, en effet, un polynôme M de degré s tel que (4.6) est vrai pour $m = s$. D'après le lemme 4.7, on obtient ainsi une formule d'ordre $2s$ en choisissant les nœuds comme les racines d'un tel $M(t)$.

4.4 Les polynômes orthogonaux de Legendre

Pour rendre les formules plus simples (et symétriques), nous faisons le changement de variable $\tau = 2t-1$ qui transforme l'intervalle $[0,1]$ pour tout t en l'intervalle $[-1,1]$ pour tout τ. Le but est de trouver des polynômes $P_k(\tau)$ de degré exactement k tel que

$$\int_{-1}^1 P_k(\tau)g(\tau)\,d\tau = 0, \qquad \forall g \in \mathbb{P}_{k-1}.$$

Ces polynômes sont appelés les **polynômes de Legendre** (1785) et ils sont définis à multiplication par un scalaire près, c-à-d, $\alpha P_k(\tau)$ est aussi un polynôme de Legendre pour tout $\alpha \neq 0$ pourvu que P_k le soit.

Les P_k représentent un exemple d'un système de **polynômes orthogonaux** car $\langle P_k, P_\ell\rangle = 0$ pour $k \neq \ell$, où

$$\langle f, g\rangle = \int_{-1}^1 f(\tau)g(\tau)\,d\tau \tag{4.7}$$

est un produit scalaire sur l'espace vectoriel des polynômes à coefficients réels. Comme pour les vecteurs dans $\mathbb{R}^n$, on peut construire les P_k à l'aide de la méthode de Gram–Schmidt.

Théorème 4.9. Le polynôme de Legendre P_k existe pour tout $k \geq 0$ et son degré est exactement égal à k. En plus, les $P_0, P_1, \ldots, P_k$ forment une base orthogonale pour $\mathbb{P}_k$ avec (4.7).

Démonstration. Le cas de base $P_0(\tau) = 1$ est trivial. Supposons par récurrence qu'on ait déjà construit les polynômes $P_0, P_1, \ldots, P_k$ qui satisfont le théorème. Pour construire P_{k+1}, on orthogonalise $Q_{k+1}(\tau) = \tau P_k(\tau)$ contre $P_0, \ldots, P_k$ (la méthode de Gram–Schmidt) :

$$P_{k+1} = Q_{k+1} - \sum_{j=0}^{k} \frac{\langle Q_{k+1}, P_j\rangle}{\langle P_j, P_j\rangle} P_j. \tag{4.8}$$

Etablissons que le théorème est vrai pour P_{k+1}.

a) La degré de Q_{k+1} est clairement $= k+1$. Puisque les degrés des P_j dans (4.8) sont $\leq k$, le coefficient dominant de Q_{k+1} n'a pas changé après l'orthogonalisation. Le degré de P_{k+1} est donc $= k+1$ et $\langle P_{k+1}, P_{k+1}\rangle = \int_{-1}^1 P_{k+1}^2(\tau)\,d\tau \neq 0$.

b) En élaborant le produit scalaire, on obtient pour tout $\ell \le k$

$$\langle P_{k+1}, P_\ell \rangle = \langle Q_{k+1}, P_\ell \rangle - \sum_{j=0}^{k} \frac{\langle Q_{k+1}, P_j \rangle}{\langle P_j, P_j \rangle} \langle P_j, P_\ell \rangle = \langle Q_{k+1}, P_\ell \rangle - \frac{\langle Q_{k+1}, P_\ell \rangle}{\langle P_\ell, P_\ell \rangle} \langle P_\ell, P_\ell \rangle = 0,$$

car $\langle P_j, P_\ell \rangle$ s'annule si $j < \ell$. On a donc que les $P_0, \dots, P_{k+1}$ forment une base de $\mathbb{P}_{k+1}$ car ils constituent $k+2$ "vecteurs" orthogonaux dans l'espace vectoriel $\mathbb{P}_{k+1}$ de dimension $k+2$.

c) En utilisant le fait que $P_0, \dots, P_k$ est une base orthogonale de $\mathbb{P}_k$, chaque $g \in \mathbb{P}_k$ s'écrit comme

$$g(\tau) = \sum_{j=0}^{k} \alpha_j P_j(\tau) \qquad \text{où } \alpha_j = \langle P_j, g \rangle / \langle P_j, P_j \rangle.$$

Alors, pour tout $g \in \mathbb{P}_k$ on obtient

$$\langle P_{k+1}, g \rangle = \sum_{j=0}^{k} \alpha_j \langle P_{k+1}, P_j \rangle = 0,$$

ce qui montre que P_{k+1} est un polynôme de Legendre. □

Si l'on calcule les P_k à l'aide de la formule (4.8), on peut bénéficier du fait que les P_k sont orthogonaux,

$$\langle \tau P_k, P_j \rangle = \int_{-1}^{1} P_k(\tau) \underbrace{\tau P_j(\tau)}_{\in \mathbb{P}_{j+1}} dt = 0 \qquad \text{si } j+1 < k$$

La formule (4.8) se simplifie ainsi à une récurrence avec seulement 3 termes :

$$P_{k+1} = \tau P_k - \frac{\langle \tau P_k, P_k \rangle}{\langle P_k, P_k \rangle} P_k - \frac{\langle \tau P_k, P_{k-1} \rangle}{\langle P_{k-1}, P_{k-1} \rangle} P_{k-1}. \tag{4.9}$$

Rappelons que les polynômes de Legendre peuvent être multipliés avec des constantes non-nulles. En choisissant ces constantes pour que $P_k(1) = 1$ pour tout $k \ge 0$, on peut démontrer le résultat suivant :

Théorème 4.10. Les polynômes de Legendre avec $P_k(1) = 1$ satisfont

$$P_0(\tau) = 1, \quad P_1(\tau) = \tau,$$
$$(k+1)P_{k+1}(\tau) = (2k+1)\,\tau\, P_k(\tau) - k\, P_{k-1}(\tau), \qquad k \ge 1.$$

Exemple 4.11

Les premiers de ces polynômes sont

$$P_0(\tau) = 1, \qquad P_1(\tau) = \tau, \qquad P_2(\tau) = \frac{3}{2}\tau^2 - \frac{1}{2}, \qquad P_3(x) = \frac{5}{2}\tau^3 - \frac{3}{2}\tau. \tag{4.10}$$

et ils sont dessinés dans la figure 4.2.

Puisque les racines de P_k sont utilisées comme nœuds dans une formule de quadrature, on aimerait connaître leur distribution.

Théorème 4.12. Toutes les racines de P_k sont réelles, simples et dans $(-1, 1)$.

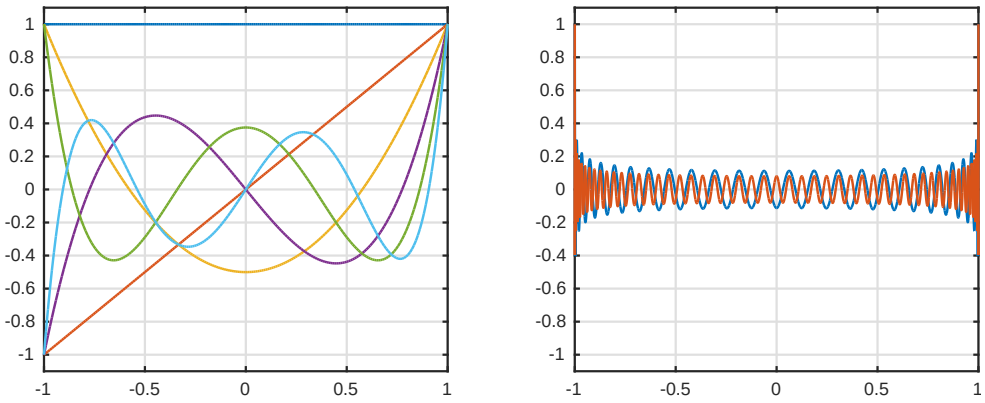


FIGURE 4.2 – Polynômes de Legendre : à gauche $P_0, \dots, P_5$ et á droite P_{50} et P_{100}.

Démonstration. Soient $\tau_1, \dots, \tau_r$ toutes les racines de $P_k(\tau)$ qui sont réelles, dans $(-1,1)$ et où $P_k(\tau)$ change de signe autour de τ_j. On veut montrer que $r = k$ (donc toutes les racines seront simples). Supposons par l'absurde que $r < k$. Alors, si on définit $q(\tau) = (\tau - \tau_1) \cdots (\tau - \tau_r)$, on obtient la contradiction

$$0 \neq \int_{-1}^{1} P_k(\tau) q(\tau)\, d\tau = \langle P_k, q \rangle = 0,$$

parce que $P_k(\tau) q(\tau)$ ne change pas de signe sur $(-1,1)$ et, en même temps, le degré de q est égal à $r < k$. □

Remarque 4.13

Les polynômes de Chebyshev T_k de la §1.3 sont aussi des polynômes orthogonaux, $\langle T_k, T_\ell \rangle = 0$ si $k \neq \ell$, où cette fois-ci le produit scalaire est donné par (voir la série d'exercices)

$$\langle f, g \rangle = \int_{-1}^{1} w(\tau) f(\tau) g(\tau)\, d\tau, \qquad w(\tau) = \frac{1}{\sqrt{1-\tau^2}}.$$

Observons, que les T_n satisfont aussi une récurrence avec 3 termes (théorème 1.8).

4.5 Formules de quadrature de Gauss

Dans ce paragraphe, nous construisons des formules de quadrature d'ordre $p = 2s$. Puisque le degré du polynôme de Legendre $P_s(\tau)$ est exactement s (voir le théorème 4.9), il existe $C_s \neq 0$ tel que

$$M(t) = (t - c_1)(t - c_2) \cdots (t - c_s) = C_s \cdot P_s(2t - 1),$$

En faisant le changement de variable $\tau = 2t - 1$, nous avons donc

$$\int_0^1 M(t) q(t)\, dt = \frac{C_s}{2} \int_{-1}^{1} P_s(\tau) q(\tau)\, d\tau = 0, \qquad \forall q \in \mathbb{P}_{s-1}.$$

D'après le théorème 4.12, toutes les racines de $P_s(\tau)$ sont réelles et situées dans l'intervalle ouvert $(0,1)$. Alors, le lemme 4.7 nous donne le résultat suivant.

s	I_s	erreur
3	1.660177678390006	$6.534 \cdot 10^{-03}$
4	1.653522218828409	$1.214 \cdot 10^{-04}$
5	1.653645005743247	$1.384 \cdot 10^{-06}$
6	1.653643610174168	$1.068 \cdot 10^{-08}$
7	1.653643620923169	$5.955 \cdot 10^{-11}$
8	1.653643620863360	$2.522 \cdot 10^{-13}$
9	1.653643620863617	$5.551 \cdot 10^{-15}$
10	1.653643620863612	$4.440 \cdot 10^{-16}$
11	1.653643620863613	$1.554 \cdot 10^{-15}$
exact	1.653643620863612	

s	I_s	erreur
3	4.791666666666670	$2.044 \cdot 10^{+00}$
4	1.854636591478698	$8.921 \cdot 10^{-01}$
5	3.534739601954471	$7.879 \cdot 10^{-01}$
6	2.308502792118851	$4.382 \cdot 10^{-01}$
7	3.080610401070965	$3.338 \cdot 10^{-01}$
8	2.540609588214999	$2.061 \cdot 10^{-01}$
9	2.893513485517553	$1.467 \cdot 10^{-01}$
10	2.651859424119450	$9.494 \cdot 10^{-02}$
11	2.812290560886774	$6.548 \cdot 10^{-02}$
exact	2.746801533890032	

TABLE 4.3 – Intégration numérique des integrales $\int_0^4 \cos(x)\,dx$ (à gauche) et $\int_{-5}^5 1/(1+x^2)\,dx$ (à droite) : I_s est le résultat de la formule de Gauss à s étages. A comparer avec le tableau 4.2.

Théorème 4.14 (Formules de quadrature de Gauss, 1814)**.** Pour chaque entier positif s, il existe une formule de quadrature à s étages d'ordre $p = 2s$. Elle est donnée par

$$\sum_{i=1}^{s} b_i\, g(c_i) \approx \int_0^1 g(s)\,ds$$

où

— les nœds $c_1, \ldots, c_s \in (0,1)$ sont les racines distinctes de $P_s(2t-1)$;
— les poids $b_1, \ldots, b_s$ sont donnés par (4.1).

Pour de petites valeurs de s, les formules de Gauss sont faciles à obtenir : il suffit de calculer les racines de (4.10) et de calculer les intégrales (4.1). Par exemple, pour $s = 5$, on obtient :

$$\int_0^1 g(t)\,dt \approx B_1\, g\big(\tfrac{1}{2} - C_1\big) + B_2\, g\big(\tfrac{1}{2} - C_2\big) + \tfrac{64}{225}\, g\big(\tfrac{1}{2}\big) + B_2\, g\big(\tfrac{1}{2} + C_2\big) + B_1\, g\big(\tfrac{1}{2} + C_1\big)$$

où

$$C_1 = \tfrac{1}{2}\sqrt{\tfrac{35+2\sqrt{70}}{63}},\ C_2 = \tfrac{1}{2}\sqrt{\tfrac{35-2\sqrt{70}}{63}},\ B_1 = \tfrac{322-13\sqrt{70}}{1800},\ B_2 = \tfrac{322+13\sqrt{70}}{1800}.$$

Après avoir trouvé des formules optimales, nous voulons refaire les calculs du tableau 4.2. Les résultats sont affichés dans le tableau 4.3 et ils montrent une claire amélioration. Même si la convergence pour la fonction de Runge n'est pas aussi vite, on peut démontrer que pour toute fonction continue les formules de Gauss convergent lorsque $s \to \infty$.

Si s est grand, le calcul exact des racines de $P_s(\tau)$ n'est pas toujours possible. Dans ce cas, on calcule les nœuds et les poids à l'aide de la décomposition spectrale d'une matrice spécifique (l'algorithme de Golub–Welsch, 1969 ; voir les travaux pratiques pour plus de détails).

Chapitre 5

Systèmes d'équations linéaires

Considérons un système d'équations linéaires : pour des coefficients a_{ij} et b_j donnés, on cherche les x_j tels que

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2 \\ \qquad\qquad\qquad\qquad\qquad\vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n = b_m, \end{cases} \tag{5.1}$$

En notation matricielle, on écrit $A\boldsymbol{x} = \boldsymbol{b}$, où

$$A = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix}, \qquad \boldsymbol{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}, \qquad \boldsymbol{b} = \begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix}.$$

La résolution des systèmes linéaires est un problème très bien étudié, car elle se présente naturellement dans plusieurs types de problèmes (interpolation, résolution des systèmes non-linéaires, intégration numérique, équations différentielles, optimisation, ...).

Dans ce chapitre nous supposons que A est inversible, alors A est carrée ($m = n$) et $\mathrm{Ker}(A) = \{\mathbf{0}\}$, ce que veut dire que le système (5.1) possède une solution unique (voir le cours d'Algèbre).

5.1 Condition d'une matrice et de résolution d'un système linéaire

Avant de présenter des algorithmes concrets pour résoudre un système linéaire, étudions d'abord la condition de résolution théorique. Autrement dit, soit $A\boldsymbol{x} = \boldsymbol{b}$ le système original, on aimerait estimer l'erreur $\|\boldsymbol{y} - \boldsymbol{x}\|$ où $\boldsymbol{y}$ est la solution exacte du système perturbé $\tilde{A}\boldsymbol{y} = \tilde{\boldsymbol{b}}$. On peut construire un tel système, par exemple, en perturbant tout les composants de A et $\boldsymbol{b}$:

$$\begin{aligned} \tilde{a}_{ij} &= a_{ij}(1+\varepsilon_{ij}), \quad |\varepsilon_{ij}| \le \varepsilon_A, \\ \tilde{b}_i &= b_i(1+\eta_i), \qquad |\eta_i| \le \varepsilon_b, \end{aligned} \tag{5.2}$$

où ε_A et ε_b spécifient la précision des données (par exemple, $\varepsilon_{\mathrm{mach}}$).

Rappelons les normes d'opérateurs de la Déf. 3.3. Puisque la quantité $\|A\|\|A^{-1}\|$ apparaît souvent dans les formules qui expriment la sensibilité aux perturbations dans des matrices, on l'appelle la condition d'une matrice.

Définition 5.1. Soit A inversible. On appelle $\kappa(A) = \|A\|\|A^{-1}\|$ la condition de A pour la norme d'operateur $\|\cdot\|$.

La condition de A dépend de la norme $\|\cdot\|$. Si l'on choisit $\|\cdot\|_p = \|\cdot\|_{p\to p}$, on peut la noter par $\kappa_p(A)$. Habituellement, on utilise $\kappa_2(A), \kappa_1(A)$ et $\kappa_\infty(A)$.

Théorème 5.2. Soit A inversible, alors pour toute norme d'opérateur $\|\cdot\|$:

(a) $\kappa(A) \geq 1$,

(b) $\kappa(\alpha A) = \kappa(A)$ pour $\alpha \neq 0$,

(c) $\kappa(A) = \max_{\|\boldsymbol{y}\|=1} \|A\boldsymbol{y}\| \,/\, \min_{\|\boldsymbol{z}\|=1} \|A\boldsymbol{z}\|$.

Démonstration. a) $1 = \|I\| = \|AA^{-1}\| \leq \|A\|\,\|A^{-1}\| = \kappa(A)$

b) Evident d'après Thm. 3.4 (d).

c) En utilisant $\|A^{-1}\| = \max_{\boldsymbol{x}\neq\boldsymbol{0}} \frac{\|A^{-1}\boldsymbol{x}\|}{\|\boldsymbol{x}\|} = \max_{\boldsymbol{z}\neq\boldsymbol{0}} \frac{\|\boldsymbol{z}\|}{\|A\boldsymbol{z}\|} = \left(\min_{\boldsymbol{z}\neq\boldsymbol{0}} \frac{\|A\boldsymbol{z}\|}{\|\boldsymbol{z}\|}\right)^{-1}$. □

La propriété (c) montre que pour $\kappa(A)$ grand, il existe $\boldsymbol{z}$ tel que $\|A\boldsymbol{z}\| \approx 0$, c.-à-d., A est presque une matrice singulière.

Exemple 5.3

Les matrices orthogonales sont bien conditionnées. Soit Q orthogonale, alors $Q^{-1} = Q^T$. Donc, la condition de Q pour $\|\cdot\|_2$ satisfait

$$\kappa_2(Q) = \|Q\|_2\|Q^{-1}\|_2 = \|Q\|_2\|Q^T\|_2 = 1$$

car $\|Q\|_2 = \sqrt{\lambda_{\max}(Q^TQ)} = \sqrt{\lambda_{\max}(I)} = 1$ où $\lambda_{\max}(\cdot)$ est la plus grande valeur propre.

Les matrices de Hilbert H_n et de Vandermonde V_n

$$H_n = \begin{bmatrix} 1 & \frac{1}{2} & \cdots & \frac{1}{n} \\ \frac{1}{2} & \frac{1}{3} & \cdots & \frac{1}{n+1} \\ \frac{1}{3} & \frac{1}{4} & \cdots & \frac{1}{n+2} \\ \vdots & \vdots & & \vdots \\ \frac{1}{n} & \frac{1}{n+1} & \cdots & \frac{1}{2n-1} \end{bmatrix}, \qquad V_n = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ c_1 & c_2 & \cdots & c_n \\ c_1^2 & c_2^2 & \cdots & c_n^2 \\ \vdots & \vdots & & \vdots \\ c_1^{n-1} & c_2^{n-1} & \cdots & c_n^{n-1} \end{bmatrix}, \quad c_j = j/n$$

sont très mal conditionnées, ce qui est facilement vérifié en Matlab :

```
for n = 1:10
  H = hilb(n);  v = (1:n)/n; V = vander(v);
  % Afficher les conditions de H et V
  fprintf('n = %2d, k_2(H) = %.0e, k_1(H) = %.0e, k_2(V) = %.0e, k_1(V) = %.0e \n', ...
      n, cond(H), cond(H,1), cond(V), cond(V,1))
end
```

avec le résultat :

```
n =  1, k_2(H) = 1e+00, k_1(H) = 1e+00, k_2(V) = 1e+00, k_1(V) = 1e+00
n =  2, k_2(H) = 2e+01, k_1(H) = 3e+01, k_2(V) = 6e+00, k_1(V) = 8e+00
n =  3, k_2(H) = 5e+02, k_1(H) = 7e+02, k_2(V) = 4e+01, k_1(V) = 7e+01
n =  4, k_2(H) = 2e+04, k_1(H) = 3e+04, k_2(V) = 3e+02, k_1(V) = 6e+02
n =  5, k_2(H) = 5e+05, k_1(H) = 9e+05, k_2(V) = 2e+03, k_1(V) = 5e+03
n =  6, k_2(H) = 1e+07, k_1(H) = 3e+07, k_2(V) = 2e+04, k_1(V) = 4e+04
n =  7, k_2(H) = 5e+08, k_1(H) = 1e+09, k_2(V) = 1e+05, k_1(V) = 3e+05
n =  8, k_2(H) = 2e+10, k_1(H) = 3e+10, k_2(V) = 1e+06, k_1(V) = 2e+06
```

```
n =  9, k_2(H) = 5e+11, k_1(H) = 1e+12, k_2(V) = 7e+06, k_1(V) = 2e+07
n = 10, k_2(H) = 2e+13, k_1(H) = 4e+13, k_2(V) = 6e+07, k_1(V) = 2e+08
```

Retournons à la question d'estimation de l'erreur $\|\boldsymbol{y}-\boldsymbol{x}\|$. Les hypothèses (5.2) impliquent pour $\|\cdot\|_1$ et $\|\cdot\|_\infty$ qu'il existe $\varepsilon_A, \varepsilon_b$ tels que

$$\|\tilde{A}-A\| \le \varepsilon_A \|A\| \qquad \text{et} \qquad \|\tilde{\boldsymbol{b}}-\boldsymbol{b}\| \le \varepsilon_b \|\boldsymbol{b}\|.$$

En supposant ces bornes sur les perturbations, nous avons le résultat suivant.

Théorème 5.4. Soient $A\boldsymbol{x}=\boldsymbol{b}$ et $(A+\delta A)\boldsymbol{y}=\boldsymbol{b}+\boldsymbol{\delta b}$ où A est une matrice inversible et $\|\delta A\| \le \varepsilon_A\|A\|$ et $\|\boldsymbol{\delta b}\| \le \varepsilon_b\|\boldsymbol{b}\|$. Si $\varepsilon_A\kappa(A)<1$, alors

$$\frac{\|\boldsymbol{y}-\boldsymbol{x}\|}{\|\boldsymbol{x}\|} \le \frac{1}{1-\varepsilon_A\kappa(A)}\left(\varepsilon_A\kappa(A)+\varepsilon_b\|A^{-1}\|\frac{\|\boldsymbol{b}\|}{\|\boldsymbol{x}\|}\right). \tag{5.3}$$

Démonstration. En utilisant l'identité

$$\begin{aligned} & A(\boldsymbol{y}-\boldsymbol{x}) = \boldsymbol{\delta b} - \delta A\boldsymbol{y} = \boldsymbol{\delta b} - \delta A(\boldsymbol{y}-\boldsymbol{x}) - \delta A\boldsymbol{x}, \\ \Longrightarrow \qquad & \boldsymbol{y}-\boldsymbol{x} = A^{-1}[\boldsymbol{\delta b} - \delta A(\boldsymbol{y}-\boldsymbol{x}) - \delta A\boldsymbol{x}] \end{aligned} \tag{5.4}$$

on obtient

$$\frac{\|\boldsymbol{y}-\boldsymbol{x}\|}{\|\boldsymbol{x}\|} \le \varepsilon_b\|A^{-1}\|\frac{\|\boldsymbol{b}\|}{\|\boldsymbol{x}\|} + \varepsilon_A\|A\|\|A^{-1}\|\frac{\|\boldsymbol{y}-\boldsymbol{x}\|}{\|\boldsymbol{x}\|} + \varepsilon_A\|A\|\|A^{-1}\|$$

et aussi

$$(1-\varepsilon_A\kappa(A))\frac{\|\boldsymbol{y}-\boldsymbol{x}\|}{\|\boldsymbol{x}\|} \le \varepsilon_b\|A^{-1}\|\frac{\|\boldsymbol{b}\|}{\|\boldsymbol{x}\|} + \varepsilon_A\kappa(A). \qquad \square$$

Si $\varepsilon_A=\varepsilon_b=\varepsilon$, on peut obtenir un résultat plus élégant. En utilisant

$$\|A\|\|\boldsymbol{x}\| \ge \|A\boldsymbol{x}\| = \|\boldsymbol{b}\| \qquad \Longrightarrow \qquad \frac{\|\boldsymbol{b}\|}{\|\boldsymbol{x}\|} \le \|A\|,$$

la borne (5.3) devient

$$\frac{\|\boldsymbol{y}-\boldsymbol{x}\|}{\|\boldsymbol{x}\|} \le \frac{2\varepsilon\kappa(A)}{1-\varepsilon\kappa(A)} = 2\varepsilon\kappa(A)+O(\varepsilon^2).$$

La sensibilité aux perturbations est donc environ $2\kappa(A)$ pour ε petit.

La formule ci-dessus montre que, pour des équations linéaires pour lesquelles $\kappa(A)$ est grand, la solution exacte $\boldsymbol{x}=A^{-1}\boldsymbol{b}$ peut être très sensible aux petites perturbations dans A et $\boldsymbol{b}$. Cette sensibilité est en fait une question de la condition (voir Chap. 3) de la résolution de $A\boldsymbol{x}=\boldsymbol{b}$ où A et $\boldsymbol{b}$ sont les données. Concrètement, on peut la mesurer comme

$$\kappa_{A,\boldsymbol{b}} = \lim_{\varepsilon\to 0}\sup\left\{\frac{\|\tilde{A}^{-1}\tilde{\boldsymbol{b}}-A^{-1}\boldsymbol{b}\|}{\varepsilon\|A^{-1}\boldsymbol{b}\|} : \frac{\|\tilde{A}-A\|}{\|A\|}\le\varepsilon,\ \frac{\|\tilde{\boldsymbol{b}}-\boldsymbol{b}\|}{\|\boldsymbol{b}\|}\le\varepsilon\right\}. \tag{5.5}$$

La quantité $\kappa_{A,\boldsymbol{b}}$ est presque la même que le κ définie dans Thm. 3.5 pour mesurer la condition de $f(A,\boldsymbol{b})=A^{-1}\boldsymbol{b}$. La seule différence est que les perturbations dans A et $\boldsymbol{b}$ sont mesurées séparément (ainsi, il est plus facile d'utiliser le Thm. 5.4). Cependant, on peut bien appeler $\kappa_{A,\boldsymbol{b}}$ la condition de résolution de $A\boldsymbol{x}=\boldsymbol{b}$.

Corollaire 5.5. La condition de résolution de $A\boldsymbol{x} = \boldsymbol{b}$ satisfait

$$\kappa_{A,\boldsymbol{b}} = \kappa(A) + \|A^{-1}\| \frac{\|\boldsymbol{b}\|}{\|A^{-1}\boldsymbol{b}\|} \leq 2\kappa(A).$$

Démonstration. Afin d'utiliser le Thm. 3.5, définissons

$$\boldsymbol{y} = \tilde{A}^{-1}\tilde{\boldsymbol{b}}, \quad \boldsymbol{x} = A^{-1}\boldsymbol{b}, \quad \tilde{A} = A + \delta A, \quad \tilde{\boldsymbol{b}} = \boldsymbol{b} + \boldsymbol{\delta b}.$$

La formule (5.5) devient

$$\kappa_{A,\boldsymbol{b}} = \lim_{\varepsilon \to 0} \sup \left\{ \frac{\|\boldsymbol{y} - \boldsymbol{x}\|}{\varepsilon \|\boldsymbol{x}\|} : \|\delta A\| \leq \varepsilon \|A\|, \ \|\boldsymbol{\delta b}\| \leq \varepsilon \|\boldsymbol{b}\| \right\}.$$

Pour obtenir la borne supérieure (c.-à-.d, le sup ou le plus petit des majorants), on peut d'abord établir que la borne du Thm. 3.5 pour $\kappa_{A,\boldsymbol{b}}$ est atteinte. Autrement, il existe δA et $\boldsymbol{\delta b}$ tels que $\leq$ dans (5.3) devient $=$. En fait, à l'aide des normes duales [1], on peut démontrer que les $\boldsymbol{x}$ et $\boldsymbol{y}$ ci-dessus satisfont

$$\|\boldsymbol{y} - \boldsymbol{x}\| = \varepsilon\kappa(A)\|\boldsymbol{x}\| + \varepsilon\|A^{-1}\|\|\boldsymbol{b}\| + O(\varepsilon^2),$$

où $\|\delta A\| \leq \varepsilon\|A\|$ et $\|\boldsymbol{\delta b}\| \leq \varepsilon\|\boldsymbol{b}\|$. En prenant la limite $\varepsilon \to 0$, on obtient directement le résultat du corollaire. □

5.2 Elimination de Gauss

Un algorithme bien connu pour résoudre le système linéaire (5.1) est l'élimination de Gauss (aussi appelée la méthode de Gauss). Si $a_{11} \neq 0$, on peut alors choisir a_{11} comme *pivot* et éliminer la variable x_1 dans les équations $2, 3, \ldots, n$ à l'aide de l'équation 1 : on calcule

$$\ell_{i1} = \frac{a_{i1}}{a_{11}}, \qquad i = 2, 3, \ldots, n,$$

et on remplace la ligne i par (ligne $i - \ell_{i1} \times$ ligne 1) :

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\ a_{22}^{(1)}x_2 + \cdots + a_{2n}^{(1)}x_n &= b_2^{(1)} \\ &\vdots \\ a_{n2}^{(1)}x_2 + \cdots + a_{nn}^{(1)}x_n &= b_n^{(1)}, \end{aligned}$$

où

$$a_{ij}^{(1)} = a_{ij} - \ell_{i1}a_{1j}, \qquad b_i^{(1)} = b_i - \ell_{i1}b_1 \qquad \text{pour } i, j = 2, \ldots, n.$$

On applique la même procédure au sous-système $(n-1) \times (n-1)$ défini par les (nouvelles) équations 2 à n, qui ne dépendent que des variables $x_2, \ldots, x_n$: si $a_{22}^{(1)} \neq 0$, on le choisit comme pivot pour éliminer x_2 dans les équations 3 à n, et ainsi de suite. L'étape k de l'algorithme s'écrit

$$\ell_{ik} = a_{ik}^{(k-1)} / a_{kk}^{(k-1)}, \qquad i = k+1, \ldots, n, \tag{5.6}$$

$$a_{ij}^{(k)} = a_{ij}^{(k-1)} - \ell_{ik}a_{kj}^{(k-1)}, \qquad i, j = k+1, \ldots, n, \tag{5.7}$$

$$b_i^{(k)} = b_i^{(k-1)} - \ell_{ik}b_k^{(k-1)}, \qquad i = k+1, \ldots, n. \tag{5.8}$$

1. Voir Thm. 7.3 dans Higham (2002), *Accuracy and stability of numerical algorithms*, 2nd edition, SIAM.

Après $n-1$ étapes, on obtient le *système triangulaire*

$$\begin{aligned} u_{11}x_1 + u_{12}x_2 + \cdots + u_{1n}x_n &= c_1 \\ u_{22}x_2 + \cdots + u_{2n}x_n &= c_2 \\ \ddots \qquad\quad &\vdots \\ u_{nn}x_n &= c_n, \end{aligned} \tag{5.9}$$

où $u_{ij} = a_{ij}^{(n-1)}$ et $c_i = b_j^{(n-1)}$. Ce système se résout facilement par substitution, en commençant par la dernière équation :

$$\begin{aligned} x_n &= c_n / u_{nn}, \\ x_{n-1} &= (c_{n-1} - u_{n-1,n}x_n)/u_{n-1,n-1}, \\ &\vdots \\ x_i &= (c_i - \sum_{j=i+1}^{n} u_{ij}x_j)/u_{ii}. \end{aligned}$$

Remarque 5.6

Si à l'étape i on a $a_{ii}^{(i-1)} = 0$, on ne peut pas le choisir comme pivot. Nous traiterons ce cas en détail les sections suivantes.

Le coût de l'élimination de Gauss est tabulé ci-dessous en calculant toutes les opérations nécessaires (addition, soustraction, multiplication, division).

Calcul	Coût d'étape k pour tout i,j					Total
	1	$\cdots$	k	$\cdots$	$n-1$	
ℓ_{ik}	$n-1$		$n-k$		1	$\approx \frac{1}{2}n^2$
$a_{ij}^{(k-1)} \to a_{ij}^{(k)}$	$2(n-1)^2$		$2(n-k)^2$		2	$\approx \frac{2}{3}n^3$
$b_i^{(k-1)} \to b_i^{(k)}$	$2(n-1)$		$2(n-k)$		2	$\approx n^2$
					Total :	$\approx \frac{2}{3}n^3$

Le total est fait approximativement pour n grand, par exemple,

$$(n-1)^2 + (n-2)^2 + \cdots + 2^2 + 1^2 \approx \int_0^n x^2\,dx = \tfrac{1}{3}n^3.$$

Par conséquent, transformer le système général (5.1) en système triangulaire (5.9) coûte environ $\frac{1}{3}n^3$ opérations. Similairement, la résolution du système triangulaire se fait en n^2 opérations.

5.3 Factorisation LU

Dans cette section, nous montrons que la méthode de Gauss est équivalente à la factorisation de la matrice A sous la forme $A = LU$ où les matrices L et U ne dépendent que de A. Une telle factorisation est souvent très utile comme nous le verrons plus tard.

Comme expliqué ci-dessus, le but est de transformer A, une colonne à la fois, en une matrice triangulaire U. Par exemple, supposons A de dimension 4×4, alors

$$\underset{A = A^{(0)}}{\begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \end{bmatrix}} \xrightarrow{L_1} \underset{A^{(1)}}{\begin{bmatrix} \times & \times & \times & \times \\ \mathbf{0} & \times & \times & \times \\ \mathbf{0} & \times & \times & \times \\ \mathbf{0} & \times & \times & \times \end{bmatrix}} \xrightarrow{L_2} \underset{A^{(2)}}{\begin{bmatrix} \times & \times & \times & \times \\ & \times & \times & \times \\ & \mathbf{0} & \times & \times \\ & \mathbf{0} & \times & \times \end{bmatrix}} \xrightarrow{L_3} \underset{A^{(3)} = U}{\begin{bmatrix} \times & \times & \times & \times \\ & \times & \times & \times \\ & & \times & \times \\ & & \mathbf{0} & \times \end{bmatrix}}$$

Ici, on a utilisé la convention habituelle que $\times$ est un coefficient quelconque et une place vide est un coefficient nul. Un caractère gras et rouge foncé indique que celui-ci est changé par l'algorithme.

L'opération L_k représente le calcul (5.9) qui introduit des zéros en dessous de la diagonale de la colonne k. On peut vérifier que le résultat de cette opération, $A^{(k)} = L_k A^{(k-1)}$, est en fait la multiplication de $A^{(k-1)}$ et

$$L_k = \begin{bmatrix} 1 & & & & & \\ & \ddots & & & & \\ & & 1 & & & \\ & & -\ell_{k+1,k} & 1 & & \\ & & \vdots & & \ddots & \\ & & -\ell_{n,k} & & & 1 \end{bmatrix}. \tag{5.10}$$

Par conséquent, la méthode de Gauss est équivalente à multiplier à gauche la matrice A par les matrices L_k

$$L_{n-1} \cdots L_2 L_1 A = U.$$

Si l'on pose $L^{-1} = L_{n-1} \cdots L_2 L_1$, on obtient la factorisation désirée : $A = LU$. L'observation cruciale (et peut-être stupéfiante) est que, malgré les matrices inverses, calculer $L = L_1^{-1} L_2^{-1} \cdots L_{n-1}^{-1}$ est très facile !

Lemme 5.7. Soit L_k la matrice (5.10). Alors,

$$L_k^{-1} = \begin{bmatrix} 1 & & & & & \\ & \ddots & & & & \\ & & 1 & & & \\ & & \ell_{k+1,k} & 1 & & \\ & & \vdots & & \ddots & \\ & & \ell_{n,k} & & & 1 \end{bmatrix}, \qquad L_1^{-1} L_2^{-1} \cdots L_{n-1}^{-1} = \begin{bmatrix} 1 & & & & \\ \ell_{2,1} & 1 & & & \\ \ell_{3,1} & \ell_{3,2} & 1 & & \\ \vdots & \vdots & & \ddots & \\ \ell_{n,1} & \ell_{n,2} & \cdots & \ell_{n,n-1} & 1 \end{bmatrix}$$

Démonstration. On peut écrire $L_k = I - \boldsymbol{\ell}_k \boldsymbol{e}_k^T$ en utilisant les vecteurs

$$\boldsymbol{e}_k = \begin{bmatrix} 0 & \vdots & 1 & 0 & \cdots & 0 \end{bmatrix}^T \qquad \boldsymbol{\ell}_k = \begin{bmatrix} 0 & \vdots & 0 & \ell_{k+1,k} & \vdots & \ell_{n,k} \end{bmatrix}^T.$$

Puisque $\boldsymbol{e}_k^T \boldsymbol{\ell}_k = 0$, on a directement

$$(I - \boldsymbol{\ell}_k \boldsymbol{e}_k^T)(I + \boldsymbol{\ell}_k \boldsymbol{e}_k^T) = I - \boldsymbol{\ell}_k \boldsymbol{e}_k^T \boldsymbol{\ell}_k \boldsymbol{e}_k^T = I,$$

ce qui montre la première formule du lemme. Afin de vérifier la deuxième formule, observons

$$L_k^{-1}L_{k+1}^{-1} = (I + \boldsymbol{\ell}_k \boldsymbol{e}_k^T)(I + \boldsymbol{\ell}_{k+1}\boldsymbol{e}_{k+1}^T) = I + \boldsymbol{\ell}_k \boldsymbol{e}_k^T + \boldsymbol{\ell}_{k+1}\boldsymbol{e}_{k+1}^T$$

car $\boldsymbol{\ell}_k \boldsymbol{e}_k^T \boldsymbol{\ell}_{k+1}\boldsymbol{e}_{k+1}^T = 0$. La matrice $L_k^{-1}L_{k+1}^{-1}$ est donc égale à I dont les colonnes k et $k+1$ sont remplacées par $\boldsymbol{\ell}_k$ et $\boldsymbol{\ell}_{k+1}$. Le produit total $L_1^{-1}L_2^{-1}\cdots L_{n-1}^{-1}$ a une propriété similaire. □

En pratique, il n'est pas nécessaire de stocker les L_k et on peut immédiatement remplir la matrice L avec les pivots ℓ_{ij} en faisant l'élimination de Gauss. Le coût de calcul de la factorisation $A = LU$ est donc $\approx \frac{2}{3}n^3$.

5.4 Changement de pivot

Considérons

$$A = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}.$$

Cette matrice est inversible,

$$A^{-1} = \begin{bmatrix} -1 & 1 \\ 1 & 0 \end{bmatrix},$$

et elle est bien conditionnée,

$$\kappa_1(A) = \|A\|_1 \|A^{-1}\|_1 = 2 \cdot 2 = 4.$$

Néanmoins, pour chaque système avec cette matrice, l'élimination de Gauss échoue car le premier pivot s'annule. Il est clair que simplement échanger les deux lignes de A nous permet de continuer l'élimination. Mais que se passe-t-il si un pivot non nul est petit?

Perturbons très légèrement la matrice ci-dessus,

$$\tilde{A} = \begin{bmatrix} 10^{-20} & 1 \\ 1 & 1 \end{bmatrix}.$$

La matrice perturbée est encore bien conditionnée avec $\kappa_1(\tilde{A}) \approx 4$ car

$$\tilde{A}^{-1} = \frac{1}{10^{-20}-1}\begin{bmatrix} 1 & -1 \\ -1 & 10^{-20} \end{bmatrix} \approx \begin{bmatrix} -1 & 1 \\ 1 & -10^{-20} \end{bmatrix}.$$

Puisque le premier pivot est non nul, on peut faire l'élimination de Gauss comme d'habitude : par exemple, le système

$$\begin{aligned} 10^{-20}x_1 + x_2 &= 1 \\ x_1 + x_2 &= 2 \end{aligned}$$

est transformé en

$$\begin{aligned} 10^{-20}x_1 + x_2 &= 1 \\ (1 - 10^{20})x_2 &= 2 - 10^{20} \end{aligned}$$

En supposant que le calcul est fait en double précision (c.-à-d., $\varepsilon_{\text{mach}} = 10^{-16}$), la résolution du système triangulaire devient

$$\begin{aligned} \tilde{x}_2 &= \text{fl}\,\frac{2 - 10^{20}}{1 - 10^{20}} = 1 \\ \tilde{x}_1 &= \text{fl}\,\frac{1 - \tilde{x}_2}{10^{-20}} = 0 \end{aligned}$$

En revanche, la solution exacte est $(x_1, x_2) \approx (1,1)$. L'élimination de Gauss n'a pas réussi à calculer une solution précise pour un problème qui est pourtant bien conditionné. Cet algorithme n'est donc pas stable!

Analysons cette perte de la précision. Si ℓ_{21} est très grand (ce qui est le cas dans l'exemple) alors,

$$\left.\begin{aligned} a_{22}^{(1)} &= a_{22} - \ell_{21} a_{12} \approx -\ell_{21} a_{12} \\ b_2^{(1)} &= b_2 - \ell_{21} b_1 \approx -\ell_{21} b_1 \end{aligned}\right\} \quad \Longrightarrow \quad x_2 = \frac{b_2^{(1)}}{a_{22}^{(1)}} \approx \frac{b_1}{a_{12}}.$$

Cette valeur, obtenue pour x_2, est en général assez correcte. Mais le calcul de x_1

$$x_1 = (b_1 - a_{12} x_2)/a_{11}$$

nécessite une soustraction qui est très mal conditionnée car $a_{12} x_2 \approx b_1$. A cause de cette extinction de chiffres, on perd de la précision. Il faut donc éviter des petits pivots, ou bien des ℓ_{ij} trop grands.

Au lieu de se contenter d'avoir $a_{11} \neq 0$, on cherche le *plus grand élément* $a_{i1} \neq 0$ en valeur absolue dans la première colonne, puis on échange les lignes 1 et i. Ceci implique $|\ell_{i1}| \leq 1$ *pour tout* i ; puis on fait la même chose pour les sous-systèmes qui apparaissent plus tard. Cette stratégie est appelée **l'élimination de Gauss avec changement de pivot partiel**.

Dans l'exemple précédent, cette permutation donne

$$\begin{aligned} x_1 + x_2 &= 2 \\ 10^{-20} x_1 + x_2 &= 1 \end{aligned}$$

et le pivot $l_{21} = 10^{-20}$ n'est pas du tout grand. Le système triangulaire

$$\begin{aligned} x_1 + x_2 &= 2 \\ (1 - 10^{-20}) x_2 &= 1 - 2 \cdot 10^{-20}. \end{aligned}$$

a une solution

$$\tilde{x}_2 = \mathrm{fl}\,\frac{1 - 2 \cdot 10^{-20}}{1 - 10^{-20}} = 1, \qquad \tilde{x}_1 = \mathrm{fl}(2 - \tilde{x}_2) = 1.$$

assez précise en double précision.

5.5 Factorisation LU avec changement de pivot

Analysons comment la stratégie de changer le pivot pendant la méthode de Gauss affecte la factorisation LU de la section précédente. Supposons que, à l'étape k, le pivot le plus grand en valeur absolue est $\alpha = a_{ik}^{(k-1)} \neq 0$ où $i \geq k$. Il faut donc échanger les lignes k et i, et puis faire l'élimination. Par exemple :

$$\underset{A^{(k-1)}}{\begin{bmatrix} \times & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & \alpha & \times & \times & \times \\ & \times & \times & \times & \times \end{bmatrix}} \xrightarrow{P_k} \underset{\tilde{A}^{(k)}}{\begin{bmatrix} \times & \times & \times & \times & \times \\ & \boldsymbol{\alpha} & \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} \\ & \times & \times & \times & \times \\ & \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} \\ & \times & \times & \times & \times \end{bmatrix}} \xrightarrow{L_k} \underset{A^{(k)}}{\begin{bmatrix} \times & \times & \times & \times & \times \\ & \alpha & \times & \times & \times \\ & \mathbf{0} & \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} \\ & \mathbf{0} & \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} \\ & \mathbf{0} & \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} \end{bmatrix}}$$

Echanger les deux lignes de $A^{(k-1)}$ est égal à $\tilde{A}^{(k)} = P_k A^{(k-1)}$ où P_k est une matrice de permutation. Pour l'exemple ci-dessus, on a

$$P_k = \begin{bmatrix} 1 & & & & \\ & & & 1 & \\ & & 1 & & \\ & 1 & & & \\ & & & & 1 \end{bmatrix}.$$

Rappelons que les colonnes et les lignes d'une matrice de permutation P sont des vecteurs unité ; en outre, $P^{-1} = P^T$ et $\det P = \pm 1$.

Si le pivot $\alpha = 0$, tous les éléments sous la diagonale sont nuls car α est le plus grand en valeur absolue. Dans ce cas, l'étape k est triviale et on peut simplement utiliser $P_k = L_k = I$:

$$\underset{A^{(k-1)}}{\begin{bmatrix} \times & \times & \times & \times & \times \\ & 0 & \times & \times & \times \\ & 0 & \times & \times & \times \\ & 0 & \times & \times & \times \\ & 0 & \times & \times & \times \end{bmatrix}} \xrightarrow{P_k} \underset{\tilde{A}^{(k)} = A^{(k-1)}}{\begin{bmatrix} \times & \times & \times & \times & \times \\ & 0 & \times & \times & \times \\ & 0 & \times & \times & \times \\ & 0 & \times & \times & \times \\ & 0 & \times & \times & \times \end{bmatrix}} \xrightarrow{L_k} \underset{A^{(k)} = A^{(k-1)}}{\begin{bmatrix} \times & \times & \times & \times & \times \\ & 0 & \times & \times & \times \\ & 0 & \times & \times & \times \\ & 0 & \times & \times & \times \\ & 0 & \times & \times & \times \end{bmatrix}}$$

Dans les deux cas, l'étape k est toujours possible en résolvant $A^{(k)} = L_k P_k A^{(k-1)}$. Ce processus est répété $n-1$ fois afin d'obtenir la matrice triangulaire sous la forme

$$L_{n-1} P_{n-1} \cdots L_2 P_2 L_1 P_1 A = U.$$

L'observation cruciale (et de nouveau un peu stupéfiante) est que cette décomposition est équivalente à $PA = LU$ où P est une matrice de permutation. Nous le montrons pour une matrice de dimension 4×4 :

$$L_3 P_3 L_2 P_2 L_1 P_1 A = U. \tag{5.11}$$

Définissons les matrices

$$\tilde{L}_3 = L_3, \quad \tilde{L}_2 = P_3 L_2 P_3^{-1}, \quad \tilde{L}_1 = P_3 P_2 L_1 P_2^{-1} P_3^{-1}.$$

Puisque les P_j dans ces formules sont appliqués *seulement à L_k pour $j > k$*, la forme de $\tilde{L}_k$ est encore égale à celle de (5.10), c.-à-d.,

$$\tilde{L}_k = P_{n-1} \cdots P_{k+1} (I - \boldsymbol{\ell}_k \boldsymbol{e}_k^T) P_{k+1}^{-1} \cdots P_{n-1}^{-1} = I - \tilde{\boldsymbol{\ell}}_k \boldsymbol{e}_k^T,$$

où $\tilde{\boldsymbol{\ell}}_k = P_{n-1} \cdots P_{k+1} \boldsymbol{\ell}_k$ est zéro dans les k premières composantes. La décomposition (5.11) devient alors

$$\tilde{L}_3 \tilde{L}_2 \tilde{L}_1 P_3 P_2 P_1 A = U \quad \Longrightarrow \quad (P_3 P_2 P_1) A = (\tilde{L}_1^{-1} \tilde{L}_2^{-1} \tilde{L}_3^{-1}) U \quad \Longrightarrow \quad PA = LU$$

car les P_k sont des matrices de permutation. De la même façon on peut démontrer le cas où A est de dimension quelconque. En résumé, on a démontré le résultat suivant :

Théorème 5.8. Soit $A \in \mathbb{R}^{n\times n}$. Alors l'élimination de Gauss avec changement de pivot partiel est équivalent à la **factorisation** LU de A,

$$PA = LU, \tag{5.12}$$

où P est une matrice de permutation et

$$L = \begin{bmatrix} 1 & & & & \\ \ell_{21} & 1 & & & \\ \ell_{31} & \ell_{32} & 1 & & \\ \vdots & \vdots & & \ddots & \\ \ell_{n1} & \ell_{n2} & \cdots & \ell_{n,n-1} & 1 \end{bmatrix}, \quad |\ell_{ij}| \le 1, \qquad U = \begin{bmatrix} u_{11} & u_{12} & \cdots & \cdots & u_{1n} \\ & u_{22} & \cdots & \cdots & u_{2n} \\ & & \ddots & & \vdots \\ & & & \ddots & \vdots \\ & & & & u_{nn} \end{bmatrix}.$$

La factorisation $PA = LU$ peut être interprétée comme la factorisation sans changement de pivot de la matrice PA. Il est important de noter que, en pratique, P n'est pas connue en avance car elle est construite au cours de la méthode de Gauss en choisissant les pivots. Grâce à l'interaction spécifique entre les P_j et L_k (expliquée ci-dessus) on peut les séparer en une seule matrice de permutation P et une seule matrice triangulaire L.

Une fois que la décomposition (5.12) est calculée avec un coût de $O(n^3)$, on peut résoudre $A\boldsymbol{x} = \boldsymbol{b}$ en observant que

$$A\boldsymbol{x} = \boldsymbol{b} \iff PA\boldsymbol{x} = P\boldsymbol{b} \iff LU\boldsymbol{x} = P\boldsymbol{b},$$

ce qui donne l'algorithme suivant :

1. Calculer $\boldsymbol{c} = P\boldsymbol{b}$ (permutation des éléments, coût = $O(n)$).
2. Résoudre $L\boldsymbol{y} = \boldsymbol{c}$ (système triangulaire, coût = $O(n^2)$).
3. Résoudre $U\boldsymbol{x} = \boldsymbol{y}$ (système triangulaire, coût = $O(n^2)$).

Des autres applications de la factorisation LU sont le calcul de $\det(A)$ et de A^{-1} (voir série d'exercices).

Remarque 5.9

L'élimination de Gauss avec changement de pivot partiel est l'algorithme standard pour la résolution des systèmes linéaires. Pour cette raison, quand on parle de la factorisation LU, il s'agit presque toujours de la factorisation avec changement de pivot partiel, $PA = LU$.

5.6 Implémentation

A l'aide du Théorème 5.8, on peut directement calculer la matrice L en faisant l'élimination de Gauss. L'algorithme formel est illustré ci-dessous. On a utilisé une notation typique de MATLAB pour denoter une partie d'un vecteur :

$$\boldsymbol{u}_{i,\boldsymbol{k}:\boldsymbol{m}} = \begin{bmatrix} u_{i,\boldsymbol{k}} & u_{i,\boldsymbol{k}+1} & \cdots & u_{i,\boldsymbol{m}} \end{bmatrix}^T,$$
$$\boldsymbol{u}_{\boldsymbol{k}:\boldsymbol{m},i} = \begin{bmatrix} u_{\boldsymbol{k},i} & u_{\boldsymbol{k}+1,i} & \cdots & u_{\boldsymbol{m},i} \end{bmatrix}^T.$$

On rappelle que par convention tout vecteur est un vecteur colonne.

Algorithme 5.10 (L'élimination de Gauss avec changement de pivot partiel).

1: $U \leftarrow A$, $L \leftarrow I$, $P \leftarrow I$
2: **for** $k = 1, \ldots, n-1$ **do**
3: Chercher $i \geq k$ tel que $|u_{ik}|$ soit maximal
4: **if** $i_{ik} \neq 0$ **then**
5: $\boldsymbol{u}_{k,k:m} \leftrightarrow \boldsymbol{u}_{i,k:m}$ (échanger deux lignes)
6: $\ell_{k,1:k-1} \leftrightarrow \ell_{i,1:k-1}$
7: $\boldsymbol{p}_{k,1:n} \leftrightarrow \boldsymbol{p}_{i,1:n}$
8: **for** $j = k+1, \ldots, n$ **do**
9: $\ell_{jk} \leftarrow u_{jk} / u_{kk}$
10: $\boldsymbol{u}_{j,k:m} \leftarrow \boldsymbol{u}_{j,k:m} - \ell_{j,k} \boldsymbol{u}_{k,k:m}$

Observez que la matrice de permutation P est construite en appliquant les permutations (pour les lignes de A) sur ses colonnes. Pour n grand, cet algorithme a le même coût que l'élimination de Gauss sans changement de pivot, c.-à-d., $\frac{2}{3}n^3$.

Ignorons maintenant le changement de pivot. Calculer la factorisation $A = LU$ est équivalent à résoudre les équations

$$a_{ij} = (LU)_{ij} = \sum_{r=1}^{\min(i,j)} \ell_{ir} u_{rj}, \qquad \text{car} \quad \ell_{ir} = u_{rj} = 0, \text{ pour } r > i, j.$$

Si les équations sont traitées dans un ordre convenable, on peut les résoudre facilement. L'élimination de Gauss (Alg. 5.10) est seulement une des multiples possibilités. Une autre méthode est de supposer que les premières $k-1$ colonnes de L et les premières $k-1$ lignes de U sont déjà connues. En posant $\ell_{kk} = 1$, on a

$$a_{kj} = \ell_{k1} u_{1j} + \cdots + \ell_{k,k-1} u_{k-1,j} + \boldsymbol{u_{kj}}, \qquad j = k, \ldots, n, \tag{5.13}$$

$$a_{ik} = \ell_{i1} u_{1k} + \cdots + \boldsymbol{\ell_{ik}} u_{kk}, \qquad i = k+1, \ldots, n. \tag{5.14}$$

On peut maintenant trouver les éléments en gras : d'abord la ligne k de U, puis la colonne k de L. Ainsi, on a la méthode de Doolittle :

Algorithme 5.11 (L'élimination de Gauss – version de Doolittle).

1: $U \leftarrow A$, $L \leftarrow I$
2: **for** $k = 1, \ldots, n$ **do**
3: **for** $j = k, \ldots, n$ **do**
4: $u_{kj} \leftarrow a_{kj} - \boldsymbol{\ell}^T_{k,1:k-1} \boldsymbol{u}_{1:k-1,j}$
5: **for** $i = k+1, \ldots, n$ **do**
6: $\ell_{ik} \leftarrow (a_{ik} - \boldsymbol{\ell}^T_{i,1:k-1} \boldsymbol{u}_{1:k-1,k}) / u_{kk}$

Il est aussi facile d'incorporer le changement de pivot partiel dans la méthode de Doolittle, et ainsi elle est équivalente à l'élimination de Gauss. Observez qu'il ne faut pas stocker des variables intermédiaires $a^{(k)}_{ij}$ dans la méthode de Doolittle, ce qui est souvent préférable.

5.7 Stabilité de la factorisation LU

Etudions maintenant la stabilité de l'élimination de Gauss, par exemple, en appliquant les algorithmes 5.10 ou 5.11 en virgule flottante. Typiquement pour les algorithmes d'algèbre linéaire numérique, on aimerait établir s'ils sont stables au sens de "backward analysis" (voir Chap. 3). Cela implique aussi leur stabilité au sens de "forward analysis".

Puisqu'il n'y a pas d'erreur en échangeant deux lignes numériquement, on suppose que les permutations nécessaires sont déjà effectuées sur la matrice donnée A ayant la décomposition exacte $A = LU$. Soient les matrices $\tilde{L}$ et $\tilde{U}$ le résultat de l'élimination de Gauss numériquement. D'après Déf. 3.11, on a besoin de trouver une estimation de la forme

$$\frac{\|\tilde{A} - A\|}{\|A\|} \leq C\varepsilon_{\text{mach}} + o(\varepsilon_{\text{mach}}). \tag{5.15}$$

où $\tilde{A} = \tilde{L}\tilde{U}$ est la matrice ayant des facteurs exacts qui coincident avec ceux obtenus numériquement pour A.

Théorème 5.12 (Wilkinson). Soit $A = LU$ une matrice ayant une décomposition LU. Soit $\tilde{A} = \tilde{L}\tilde{U}$, où $\tilde{L}$, $\tilde{U}$ sont les résultats numériques de l'élimination de Gauss (sans recherche de pivot) où $|\tilde{l}_{ij}| \leq 1$ pour tout i, j. Alors

$$|\tilde{a}_{ij} - a_{ij}| \leq 3 \cdot \alpha \cdot \min(i-1, j) \cdot \varepsilon_{\text{mach}} + O(\varepsilon_{\text{mach}}^2),$$

où $\alpha = \max_{ijk} |a_{ij}^{(k)}|$.

Démonstration. A l'étape k, on calcule (5.7) en prenant en considération les erreurs d'arrondi

$$\tilde{a}_{ij}^{(k)} = (\tilde{a}_{ij}^{(k-1)} - \tilde{\ell}_{ik}\tilde{a}_{kj}^{(k-1)}(1+\varepsilon_{ijk}))(1+\eta_{ijk}), \qquad i, j = k+1, \ldots, n,$$

où $|\varepsilon_{ijk}|, |\eta_{ijk}| \leq \varepsilon_{\text{mach}}$. Trouvons l'erreur de ce calcul :

$$\begin{aligned} |\mu_{ijk}| &= |\tilde{a}_{ij}^{(k)} - (\tilde{a}_{ij}^{(k-1)} - \tilde{\ell}_{ik}\tilde{a}_{kj}^{(k-1)})| \\ &\leq |\tilde{\ell}_{ik}||a_{kj}^{(k-1)}||\varepsilon_{ijk}| + |\tilde{a}_{ij}^{(k-1)} - \tilde{\ell}_{ik}\tilde{a}_{kj}^{(k-1)})||\eta_{ijk}| + |\tilde{\ell}_{ik}||\tilde{a}_{kj}^{(k-1)}||\varepsilon_{ijk}||\eta_{ijk}| \\ &\leq 3\alpha\varepsilon_{\text{mach}} + O(\varepsilon_{\text{mach}}^2) \end{aligned}$$

car $|\tilde{a}_{ij}^{(k-1)} - \tilde{\ell}_{ik}\tilde{a}_{kj}^{(k-1)}| \leq |\tilde{a}_{ij}^{(k-1)}| + |\tilde{\ell}_{ik}||\tilde{a}_{kj}^{(k-1)}| \leq 2\alpha$. En revanche, le calcul numérique de (5.6) pour les pivots

$$\tilde{\ell}_{ik} = \frac{\tilde{a}_{ik}^{(k-1)}}{\tilde{a}_{kk}^{(k-1)}}(1+\varepsilon_{ikk}), \qquad i = k+1, \ldots, n, \quad |\varepsilon_{ikk}| \leq \varepsilon_{\text{mach}}$$

donne des erreurs dans la colonne que est normalement nuls. Puisque l'on choisit $\tilde{a}_{ij}^{(k)} = 0$, on a

$$|\mu_{ikk}| = |\tilde{a}_{ik}^{(k)} - (\tilde{a}_{ij}^{(k-1)} - \tilde{\ell}_{ik}\tilde{a}_{kk}^{(k-1)})| \leq |\tilde{a}_{ik}^{(k-1)}\varepsilon_{ikk}| \leq \alpha \cdot \varepsilon_{\text{mach}}.$$

En résumé, on a pour $i = k+1, \ldots, n$ et $j = k, \ldots, n$

$$|\mu_{ijk}| = |\tilde{a}_{ij}^{(k)} - (\tilde{a}_{ij}^{(k-1)} - \tilde{\ell}_{ik}\tilde{a}_{kj}^{(k-1)})| \leq 3\alpha\varepsilon_{\text{mach}} + O(\varepsilon_{\text{mach}}^2) \tag{5.16}$$

La définition de $\tilde{A}$ implique que

$$\tilde{a}_{ij} = \sum_{k=1}^{\min(i,j)} \tilde{\ell}_{ik}\tilde{u}_{kj} = \sum_{k=1}^{\min(i,j)} \tilde{\ell}_{ik}\tilde{a}_{kj}^{(k-1)}$$

et (5.16) donne $\tilde{\ell}_{ik}\tilde{a}_{kj}^{(k-1)} = \tilde{a}_{ij}^{(k-1)} - \tilde{a}_{ij}^{(k)} + \mu_{ijk}$.

Premier cas : $i > j$ (partie strictement inférieure)

$$\tilde{a}_{ij} = \sum_{k=1}^{j} \tilde{\ell}_{ik}\tilde{a}_{kj}^{(k-1)} = \sum_{k=1}^{j} (\tilde{a}_{ij}^{(k-1)} - \tilde{a}_{ij}^{(k)} + \mu_{ijk}) = \underbrace{a_{ij}^{(0)}}_{=a_{ij}} - \underbrace{\tilde{a}_{ij}^{(j)}}_{=0} + \sum_{k=1}^{j} \mu_{ijk}.$$

Nous avons donc $|\tilde{a}_{ij} - a_{ij}| \leq 3\alpha \cdot j \cdot \varepsilon_{\text{mach}} + O(\varepsilon_{\text{mach}}^2)$.

Deuxième cas : $i \leq j$ (partie supérieure)

$$\tilde{a}_{ij} = \sum_{k=1}^{i-1} (\tilde{a}_{ij}^{(k-1)} - \tilde{a}_{ij}^{(k)} + \mu_{ijk}) + \underbrace{\tilde{\ell}_{ii}}_{=1}\tilde{a}_{ij}^{(i-1)} = a_{ij} - a_{ij}^{(i-1)} + a_{ij}^{(i-1)} + \sum_{k=1}^{i-1} \mu_{ijk},$$

ce qui implique $|\tilde{a}_{ij} - a_{ij}| \leq 3\alpha \cdot (i-1) \cdot \varepsilon_{\text{mach}} + O(\varepsilon_{\text{mach}}^2)$. Les deux cas ensemble donnent le résultat désiré. □

Pour la norme $\|\cdot\|_\infty$, ce théorème donne

$$\frac{\|\tilde{A} - A\|_\infty}{\|A\|_\infty} \leq \frac{\max_i \sum_j |\tilde{a}_{ij} - a_{ij}|}{\|A\|_\infty} \leq \frac{3\alpha n^2 \varepsilon_{\text{mach}}}{\|A\|_\infty} + O(\varepsilon_{\text{mach}}^2).$$

En comparant cette borne avec (5.15), la conclusion est que l'élimination de Gauss est *backward stable* si $|l_{ij}| \leq 1$ et si (en supposant le facteur n^2 est raisonnable)

$$\frac{\alpha}{\|A\|_\infty} \quad \text{n'est pas trop grand.}$$

La première condition est facile à satisfaire par changement de pivot partiel. Le théorème 5.12 reste aussi valable dans ce cas car on peut l'appliquer à la matrice PA au lieu de A. En revanche, la deuxième condition est trop compliquée. Habituellement, on introduit la constante (appelée le **facteur d'accroissement** du processus d'élimination, "growth factor" en anglais)

$$\gamma(A) := \frac{\alpha}{\max_{i,j} |a_{ij}|}.$$

Puisque $\max_{i,j} |a_{ij}| \leq \|A\|_\infty$, cela implique

$$\frac{\|\tilde{A} - A\|_\infty}{\|A\|_\infty} \leq 3\gamma(A) n^2 \varepsilon_{\text{mach}} + O(\varepsilon_{\text{mach}}^2).$$

Malheureusement, $\gamma(A)$ peut devenir très grand :

$$\begin{aligned} |a_{ij}^{(k)}| &= |a_{ij}^{(k-1)} - \ell_{ik} a_{kj}^{(k-1)}| \\ &\leq |a_{ij}^{(k-1)}| + |\ell_{ik}||a_{kj}^{(k-1)}| \\ &\leq 2 \max_{i,j} |a_{ij}^{(k-1)}| \leq 2^2 \max_{i,j} |a_{ij}^{(k-2)}| \leq \cdots \leq 2^{k-1} \max_{i,j} |a_{ij}|. \end{aligned}$$

Donc $\gamma \le 2^{n-1}$, et il y a en fait des matrices avec $\gamma = 2^{n-1}$, par exemple :

$$A = \begin{bmatrix} 1 & 0 & \cdots & 0 & 1 \\ -1 & 1 & & 0 & 1 \\ -1 & -1 & \ddots & \vdots & 1 \\ \vdots & & & 1 & 1 \\ -1 & \cdots & \cdots & -1 & 1 \end{bmatrix}$$

Pour une telle matrice, l'élimination de Gauss n'est pas un algorithme stable et les erreurs d'arrondi pourraient s'amplifier dangereusement. Mais heureusement γ est en général beaucoup plus petit (par exemple, pour les matrices aléatoires : $\gamma \approx O(n^{0.7})$), et l'élimination de Gauss avec recherche partielle de pivot est l'algorithme standard pour la résolution de $A\boldsymbol{x} = \boldsymbol{b}$.

Remarque 5.13

L'élimination de Gauss ***avec*** *changement de pivot partiel est backward stable en pratique, c.-à-d.,* $\gamma(A) \ll 2^{n-1}$. *Les matrices pour lesquelles* $\gamma(A) \gg 1$ *sont tellement exceptionnelles, que l'on peut les négliger. En revanche, l'élimination de Gauss* ***sans*** *changement de pivot n'est pas backward stable pour beaucoup de matrices. Par exemple, pour la matrice de la section 5.4, la décomposition exacte satisfait*

$$A = \begin{bmatrix} 10^{-20} & 1 \\ 1 & 1 \end{bmatrix} \quad \Longrightarrow \quad L = \begin{bmatrix} 1 & 0 \\ 10^{20} & 1 \end{bmatrix}, U = \begin{bmatrix} 10^{-20} & 1 \\ 0 & 1 - 10^{20} \end{bmatrix}$$

Même si l'algorithme donne un arrondissement très léger

$$\tilde{L} = \begin{bmatrix} 1 & 0 \\ 10^{20} & 1 \end{bmatrix}, \tilde{U} = \begin{bmatrix} 10^{-20} & 1 \\ 0 & -10^{-20} \end{bmatrix} \quad \Longrightarrow \quad \tilde{A} = \begin{bmatrix} 10^{-20} & 1 \\ 1 & 0 \end{bmatrix},$$

l'erreur au sens backward n'est pas petite :

$$\frac{\|\tilde{A} - A\|_\infty}{\|A\|_\infty} = \frac{1}{2} \gg \varepsilon_{mach} = 10^{-16}.$$

5.8 Factorisation de Cholesky

Etudions l'élimination de Gauss pour le cas important où A est une matrice symétrique définie positive :

Définition 5.14. Une matrice A est **symétrique définie positive (s.d.p.)** si $A = A^T$ et $\boldsymbol{x}^T A\boldsymbol{x} > 0$ pour tout $\boldsymbol{x} \neq \boldsymbol{0}$.

Par le théorème spectral, il existe $Q \in \mathbb{R}^{n\times n}$, $Q^{-1} = Q^T$ tel que $Q^T AQ = \Lambda$ est la matrice diagonale des valeurs propres réelles. Alors pour tout $\boldsymbol{x} \neq \boldsymbol{0}$ on a que $\boldsymbol{y} = Q^T\boldsymbol{x} \neq \boldsymbol{0}$ et

$$\boldsymbol{x}^T A\boldsymbol{x} = \boldsymbol{x}^T Q\Lambda Q^T\boldsymbol{x} = \boldsymbol{y}^T\Lambda\boldsymbol{y} = \lambda_1 y_1^2 + \cdots + \lambda_n y_n^2 \ge \lambda_{\min}\|\boldsymbol{y}\|_2^2 > 0 \iff 0 < \lambda_{\min} \le \lambda_i. \quad (5.17)$$

Alors une définition équivalente pour les matrices s.d.p. est qu'elles sont les matrices A telles que $A = A^T$ dont les valeur propres sont strictement positives.

Exemple 5.15

Soit $A = G^T G$ *où* $G \in \mathbb{R}^{m\times n}$ *et* $\operatorname{rang}(G) = n$. *Alors*

$$\boldsymbol{x}^T A\boldsymbol{x} = \boldsymbol{x}^T G^T G\boldsymbol{x} = \|G\boldsymbol{x}\|_2^2 \geq 0,$$

avec $\boldsymbol{x}^T A\boldsymbol{x} = 0 \iff G\boldsymbol{x} = 0 \iff \boldsymbol{x} = 0$ *car* $\operatorname{rang}(G) = n$.

Théorème 5.16. Propriétés des matrices $A \in \mathbb{R}^{n\times n}$ s.d.p. :

(a) Si $L \in \mathbb{R}^{m\times n}$ est de rang m, alors $LAL^T \in \mathbb{R}^{m\times m}$ est s.d.p.

(b) Les éléments de A sur la diagonale sont positifs.

(c) Toutes les sous-matrices principales de A sont s.d.p., c.-à-d., pour tout sous-ensemble non-vide $J \subset \{1,\dots,n\}$, $A_{J,J}$ est s.d.p.

(d) $\det(A) > 0$ (donc A est inversible)

(e) L'élément maximal en valeur absolue se trouve sur la diagonale.

Démonstration. (a) Puisque A est s.d.p., on a

$$(L^T\boldsymbol{y})^T A(L^T\boldsymbol{y}) > 0, \quad \forall L^T\boldsymbol{y} \neq \boldsymbol{0}.$$

Mais $\operatorname{rang}(L) = m \leq n$ et donc $\ker(L^T) = \{\boldsymbol{0}\}$. On obtient

$$\boldsymbol{y}^T LAL^T\boldsymbol{y} > 0, \quad \forall \boldsymbol{y} \neq \boldsymbol{0}.$$

(c) D'après (a) : $A_{J,J}$ peut s'écrire sous la forme LAL^T avec $L^T = [\boldsymbol{e}_{j_1},\dots,\boldsymbol{e}_{j_m}]$, $\operatorname{rang}(L^T) = m$.

(b) D'après (c) où $L = \boldsymbol{e}_i$.

(d) Par le théorème spectral on a $A = Q\Lambda Q^T$. Donc, d'après (5.17) :

$$\det(Q)\det(\Lambda)\det(Q^T) = (\det(Q))^2\det(\Lambda) = \lambda_1\cdot\lambda_2\cdots\lambda_n > 0.$$

(e) Par l'absurde, supposons que $|a_{ij}|$ est maximal avec $i \neq j$. Alors la sous-matrice $B = \begin{bmatrix} a_{ii} & a_{ij} \\ a_{ji} & a_{jj} \end{bmatrix}$ satisfait $\det(B) = a_{ii}a_{jj} - a_{ij}^2 \leq 0$, donc B n'est pas s.d.p. Cela est une contradiction d'après (c). □

Pour des matrices s.d.p., il existe une factorisation plus économique que LU :

Théorème 5.17 (Factorisation de Cholesky – l'élimination de Gauss symétrique)**.** Soit A une matrice s.d.p. Alors il existe une matrice triangulaire inférieure L tel que $A = LL^T$.

Démonstration. Soit

$$A = \left[\begin{array}{c|c} a_{11} & \boldsymbol{v}^T \\ \hline \boldsymbol{v} & B \end{array}\right], \qquad a_{11} \in \mathbb{R}, \boldsymbol{v} \in \mathbb{R}^{n-1}, B \in \mathbb{R}^{(n-1)\times(n-1)}.$$

Vu que $a_{11} > 0$ (voir Thm. 5.16(b)), on peut considérer les matrices par blocs :

$$L_1 = \left[\begin{array}{c|c} \frac{1}{\sqrt{a_{11}}} & 0 \\ \hline -\frac{\boldsymbol{v}}{\sqrt{a_{11}}} & I \end{array}\right] \qquad \Longrightarrow \qquad L_1 A L_1^T = \left[\begin{array}{c|c} 1 & 0 \\ \hline 0 & B - \frac{\boldsymbol{v}\boldsymbol{v}^T}{a_{11}} \end{array}\right].$$

Notons $A^{(1)} = B - \frac{\boldsymbol{v}\boldsymbol{v}^T}{a_{11}}$. Puisque $L_1 A L_1^T$ est s.d.p., sa sous-matrice principale $A^{(1)}$ est s.d.p. aussi (voir Thm. 5.16(c)). On peut donc répéter cette procédure pour $A^{(1)}$, $A^{(2)}$, ..., afin d'obtenir

$$\underbrace{L_n L_{n-1} \cdots L_1}_{L^{-1}} A \underbrace{L_1^T L_2^T \cdots L_n}_{L^{-T}} = I,$$

ce qui implique $A = LL^T$. □

Pour calculer L directement, on peut modifier l'élimination de Gauss en une forme symétrique. Une comparaison des coefficients dans l'identité $A = LL^T$ donne pour

$$\begin{aligned} i = k: &\qquad a_{kk} = \ell_{k1}^2 + \ell_{k2}^2 + \cdots + \ell_{kk}^2 \\ i > k: &\qquad a_{ik} = \ell_{i1}\ell_{k1} + \ell_{i2}\ell_{k2} + \cdots + \ell_{ik}\ell_{kk} \end{aligned}$$

et on en déduit l'algorithme suivant :

Algorithme 5.18 (L'élimination de Gauss symétrique – méthode de Cholesky).

1: **for** $k = 1, \ldots, n$ **do**
2: $\quad \ell_{kk} \leftarrow (a_{kk} - \boldsymbol{\ell}_{k,1:k-1}\boldsymbol{\ell}_{k,1:k-1}^T)^{1/2}$
3: $\quad$ **for** $i = k+1, \ldots, n$ **do**
4: $\quad\quad \ell_{ik} \leftarrow (a_{ik} - \boldsymbol{\ell}_{i,1:k-1}\boldsymbol{\ell}_{k,1:k-1}^T)/\ell_{kk}$

En négligeant les n racines, le nombre d'opérations nécessaires est

$$\sum_{k=1}^{n} 2(n-k)k \approx 2\int_0^n (n-x)x\,dx = \frac{n^3}{3}.$$

Ceci correspond à la moitié du coût de la factorisation LU.

Comme pour l'élimination de Gauss, on peut étudier la stabilité de l'algorithme de Cholesky.

Théorème 5.19. Soit A une matrice s.d.p. Alors l'élimination de Gauss (symétrique) sans recherche de pivot est stable au sens "backward".

Chapitre 6

Méthode des moindres carrés

Considérons un système d'équations où il y a plus d'équations que d'inconnues (on dit que le système est **surdéterminé**). Autrement dit, $m \geq n$ dans (5.1) et $A\boldsymbol{x} = \boldsymbol{b}$ où A est une matrice rectangulaire avec plus de lignes que de colonnes.

Un tel système ne possède, en général, pas de solution car $\boldsymbol{b} \notin \text{im}(A)$. L'idée est de le remplacer par

$$\text{trouver } \boldsymbol{x}^* \in \mathbb{R}^n \text{ tel que } \|A\boldsymbol{x}^* - \boldsymbol{b}\|_2 = \min_{x \in \mathbb{R}^n} \|A\boldsymbol{x} - \boldsymbol{b}\|_2. \tag{6.1}$$

Le nom **méthode des moindres carrés** indique le choix de la norme dans (6.1) : la somme des carrés des composantes du vecteur résidu $\boldsymbol{r} = A\boldsymbol{x} - \boldsymbol{b}$ doit être minimale.

6.1 Les équations normales

Théorème 6.1. Soient $A \in \mathbb{R}^{m \times n}$ et $\boldsymbol{b} \in \mathbb{R}^m$ où $m \geq n$. Un vecteur $\boldsymbol{x}^* \in \mathbb{R}^n$ minimise $\|A\boldsymbol{x} - \boldsymbol{b}\|_2$ si et seulement s'il satisfait **les équations normales** :

$$A^T A\boldsymbol{x}^* = A^T \boldsymbol{b}. \tag{6.2}$$

La solution $\boldsymbol{x}^*$ est unique si, et seulement si, $\text{rang}(A) = n$. Sinon il y a une infinité de solutions.

Démonstration. Montrons d'abord que $\boldsymbol{x}^*$ est un minimiseur si et seulement s'il est solution des équations normales.

$\Rightarrow$: Notons $f(\boldsymbol{x}) = \|A\boldsymbol{x} - \boldsymbol{b}\|_2^2$. Afin de calculer le gradient

$$\nabla f(\boldsymbol{x}) = \begin{bmatrix} \frac{\partial f}{\partial x_1} & \frac{\partial f}{\partial x_1} & \cdots & \frac{\partial f}{\partial x_n} \end{bmatrix}^T,$$

calculons $\frac{\partial f}{\partial x_k}$:

$$f(\boldsymbol{x}) = \sum_{i=1}^{m} \left(\sum_{j=1}^{n} a_{ij} x_j - b_i \right)^2$$

$$\frac{\partial f}{\partial x_k}(\boldsymbol{x}) = \sum_{i=1}^{m} \frac{\partial}{\partial x_k} \left(\sum_{j=1}^{n} a_{ij} x_j - b_i \right)^2 = \sum_{i=1}^{m} 2 \left(\sum_{j=1}^{n} a_{ij} x_j - b_i \right) a_{ik} = 2 \sum_{i=1}^{m} a_{ik} r_i$$

où $\boldsymbol{r} = A\boldsymbol{x} - \boldsymbol{b}$. On a

$$\nabla f(\boldsymbol{x}) = 2A^T \boldsymbol{r} = 2A^T (A\boldsymbol{x} - \boldsymbol{b}).$$

Puisque $\boldsymbol{x}^*$ est un minimiseur de $\|A\boldsymbol{x}-\boldsymbol{b}\|_2$, il l'est aussi de $f(\boldsymbol{x})$ car $\|\cdot\|^2$ est une fonction strictement croissante. Alors, $\nabla f(\boldsymbol{x}^*) = \boldsymbol{0} = 2A^T(A\boldsymbol{x}^*-\boldsymbol{b}) \implies A^TA\boldsymbol{x}^* = A^T\boldsymbol{b}$.

$\Leftarrow$: Soit $\boldsymbol{y}\in\mathbb{R}^n$. Alors

$$\begin{aligned} f(\boldsymbol{y}) &= f(\boldsymbol{y}-\boldsymbol{x}^*+\boldsymbol{x}^*) \\ &= \big(A(\boldsymbol{y}-\boldsymbol{x}^*)+A\boldsymbol{x}^*-\boldsymbol{b}\big)^T\big(A(\boldsymbol{y}-\boldsymbol{x}^*)+A\boldsymbol{x}^*-\boldsymbol{b}\big) \\ &= \|A(\boldsymbol{y}-\boldsymbol{x}^*)\|_2^2 + f(\boldsymbol{x}^*) + 2\big(A(\boldsymbol{y}-\boldsymbol{x}^*)\big)^T\big(A\boldsymbol{x}^*-\boldsymbol{b}\big) \\ &\geq f(\boldsymbol{x}^*) + 2\big(\boldsymbol{y}-\boldsymbol{x}^*\big)^T \underbrace{A^T\big(A\boldsymbol{x}^*-\boldsymbol{b}\big)}_{=\boldsymbol{0}} = f(\boldsymbol{x}^*) \end{aligned}$$

et $\boldsymbol{x}^*$ est bien un minimiseur car $\boldsymbol{y}$ est un vecteur quelconque.

Puis, cherchons le nombre de solutions de (6.2). En utilisant des résultats élémentaires d'algèbre linéaire [1], on peut montrer que pour tout $A\in\mathbb{R}^{m\times n}$:

$$\operatorname{im}(A^TA) = \operatorname{im}(A^T) \qquad \text{et} \qquad \ker(A^TA) = \ker(A).$$

Le système (6.2) est donc consistent (il y a au moins une solution) car son second membre $A^T\boldsymbol{b}\in\operatorname{im}(A^T)$ est dans l'image de A^TA. En plus, il n'y a qu'une solution ssi $\ker(A)=\{\boldsymbol{0}\} \iff \operatorname{rang}(A)=n$ car $m\geq n$. □

Supposons que $\operatorname{rang}(A)=n$. Alors, la matrice A^TA est symétrique définie positive (voir l'exemple 5.15) et on peut utiliser la factorisation de Cholesky pour la résolution des équations normales.

Algorithme 6.2 (Méthode des moindres carrés – les équations normales).
Soit $A\in\mathbb{R}^{m\times n}$ où $m\geq n$ et $\operatorname{rang}(A)=n$.

1: Construire la matrice $B=A^TA$ et le vecteur $\boldsymbol{c}=A^T\boldsymbol{b}$.
2: Calculer la factorisation de Cholesky $B=LL^T$.
3: Résolution du système triangulaire inférieur $L\boldsymbol{y}=\boldsymbol{c}$.
4: Résolution du système triangulaire supérieur $L^T\boldsymbol{x}=\boldsymbol{y}$.

Rappelons que A^TA est symétrique. Pour m,n grand, le coût de l'Algorithme 6.2 est donc environ :

$$\text{Résolution des moindres carrés par Cholesky : } \sim mn^2+\tfrac{1}{3}n^3$$

Exemple 6.3

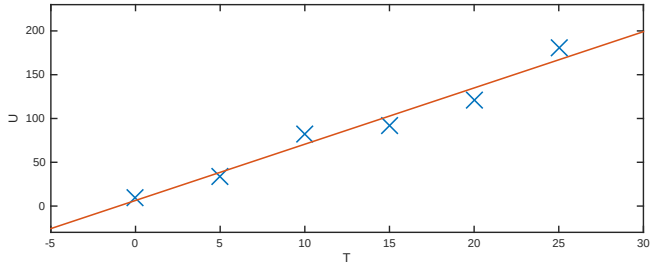
Les données suivantes décrivent des mesures du potentiel dans un câble électrique en fonction de la température du câble.

i	T_i $[^\circ C]$	U_i $[V]$
1	0	10.2
2	5	34.1
3	10	82.6
4	15	91.2
5	20	120.8
6	25	181.3

On suppose que le potentiel suit la loi affine $U=c_1T+c_2$. Comment trouver les paramètres c_1 et c_2 ? Etant donné que les données ne satisfont pas exactement la loi $U=aT+b$,

1. Par exemple, $\ker(A)=(\operatorname{im}(A^T))^\perp$

aucune droite ne passe en tous les points (U_i, T_i) dans la graphique ci-dessous.



Autrement dit, le problème $A\boldsymbol{x} = \boldsymbol{b}$ n'a aucune solution où

$$A = \begin{bmatrix} 0 & 1 \\ 5 & 1 \\ 10 & 1 \\ 15 & 1 \\ 20 & 1 \\ 25 & 1 \end{bmatrix}, \qquad \boldsymbol{x} = \begin{bmatrix} c_1 \\ c_2 \end{bmatrix}, \qquad \boldsymbol{b} = \begin{bmatrix} 10.2 \\ 34.1 \\ 82.6 \\ 91.2 \\ 120.8 \\ 181.3 \end{bmatrix}.$$

On peut estimer c_1 et c_2 au sens des moindres carrés : $\min_{\boldsymbol{x}} \|A\boldsymbol{x} - \boldsymbol{b}\|_2$. *La matrice $A^T A$ est de taille 2×2 et la résolution des équations normales donne $c_1 = 6.424$, $c_2 = 6.4$. La droite obtenue (voir graphique) correspond à une* ***régression linéaire****.*

L'algorithme (6.2) est simple, efficace et très souvent utilisé. Néanmoins, résoudre les équations normales n'est pas un bon algorithme quand A a des colonnes presque linéaires dépendantes. Par exemple, le rang de la matrice

$$A = \begin{bmatrix} 1 & 1 \\ \sqrt{\varepsilon_{\text{mach}}} & 0 \\ 0 & \sqrt{\varepsilon_{\text{mach}}} \end{bmatrix},$$

est maximal (la solution $\boldsymbol{x}^*$ est donc unique), mais les équations normales n'ont pas une solution unique en virgule flottante car la matrice

$$\text{fl}(A^T A) = \begin{bmatrix} \text{fl}(1+\varepsilon_{\text{mach}}) & 1 \\ 1 & \text{fl}(1+\varepsilon_{\text{mach}}) \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

n'est pas inversible.

Puisque l'analyse rigoureuse pour A rectangulaire est assez technique [2], analysons le cas de A carrée et inversible. D'après Cor. 5.5, on sait que la condition de résolution de $A\boldsymbol{x} = \boldsymbol{b}$ est au moins $\kappa(A)$. Ignorons que l'on peut appliquer l'élimination de Gauss ; on utilise l'Alg. 6.2 et il faut résoudre les équations normales $A^T A\boldsymbol{x} = A^T \boldsymbol{b}$. Calculons $\kappa_2(A^T A)$ qui nous donnera la condition de ce système pour la norme $\|\cdot\|_2$. D'après (3.5), on a

$$\|A^T A\|_2 = \left(\lambda_{\max}(A^T A)^T (A^T A)\right)^{1/2} = \left(\lambda_{\max}(A^T A)^2\right)^{1/2} = \lambda_{\max}(A^T A) = \|A\|_2^2,$$

où $\lambda_{\max}(\cdot)$ est la plus grande valeur propre. On a donc

$$\kappa_2(A^T A) = \|A^T A\|_2 \|(A^T A)^{-1}\|_2 = \|A\|_2^2 \|A^{-1}\|_2^2 = \kappa_2^2(A).$$

2. Voir Lecture 18 dans Trefethen (2013), *Approximation theory and approximation pratice*, SIAM.

On peut conclure que quelle que soit la méthode utilisée, les équations normales sont beaucoup plus sensibles aux erreurs d'arrondi que les équations originales. Par exemple, si $\kappa_2(A) = 10^{10}$, on risque de perdre tous les chiffres significatifs après avoir résolu les équations normales car $\kappa_2(A^TA) = 10^{20} > 1/\varepsilon_{\text{mach}}$. Pour A rectangulaire, cette conclusion reste vraie. Il faut donc construire des méthodes plus précises ...

6.2 Factorisation QR

Les matrices orthogonales sont très bien conditionnées. Pour limiter l'effet des erreurs d'arrondis, on cherche donc une méthode pour résoudre des problèmes de moindres carrés sans résoudre directement les équations normales mais en utilisant essentiellement des matrices orthogonales.

Définition 6.4. Soit $Q \in \mathbb{R}^{m\times n}$ où $m \geq n$. Si $Q^TQ = I_n$, la matrice Q est dite orthonormale.

Remarquons que les colonnes d'une d'une telle matrice Q forment un ensemble de vecteurs orthogonaux et unitaires. Si $m = n$, Q est une matrice orthogonale.

Théorème 6.5 (Factorisation QR)**.** Soit $A \in \mathbb{R}^{m\times n}$ où $m \geq n$. Il existe une matrice orthonormale $Q \in \mathbb{R}^{m\times n}$ et une matrice triangulaire supérieure $R \in \mathbb{R}^{n\times n}$ telles que $A = QR$. Cette décomposition s'appelle **la factorisation** QR **(réduite)**. Si $\text{rang}(A) = n$, elle est unique en choisissant la normalisation $r_{jj} > 0$ pour $j = 1,\ldots,n$.

Démonstration. Les matrices Q et R peuvent être calculées en utilisant la méthode de Gram–Schmidt pour les vecteurs colonnes $\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_n$ de A. Supposons d'abord que $\text{rang}\, A = n$ (donc les $\boldsymbol{a}_i$ sont linéairement indépendants) ; Gram–Schmidt construit un ensemble de vecteurs orthogonaux et unitaires $\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_n$ par

$$\begin{aligned}
\tilde{\boldsymbol{q}}_1 &= \boldsymbol{a}_1, & \boldsymbol{q}_1 &= \tilde{\boldsymbol{q}}_1/\|\tilde{\boldsymbol{q}}_1\|_2 \\
\tilde{\boldsymbol{q}}_2 &= \boldsymbol{a}_2 - (\boldsymbol{a}_2^T\boldsymbol{q}_1)\boldsymbol{q}_1, & \boldsymbol{q}_2 &= \tilde{\boldsymbol{q}}_2/\|\tilde{\boldsymbol{q}}_2\|_2 \\
&\vdots & & \\
\tilde{\boldsymbol{q}}_j &= \boldsymbol{a}_j - \sum_{i=1}^{j-1}(\boldsymbol{a}_j^T\boldsymbol{q}_i)\boldsymbol{q}_i, & \boldsymbol{q}_j &= \tilde{\boldsymbol{q}}_j/\|\tilde{\boldsymbol{q}}_j\|_2.
\end{aligned}$$

En écrivant les formules de la manière suivante

$$\begin{aligned}
\boldsymbol{a}_1 &= \tilde{\boldsymbol{q}}_1 = r_{11}\boldsymbol{q}_1, & r_{11} &= \|\tilde{\boldsymbol{q}}_1\|_2 \\
\boldsymbol{a}_2 &= (\boldsymbol{a}_2^T\boldsymbol{q}_1)\boldsymbol{q}_1 + \tilde{\boldsymbol{q}}_2 = r_{12}\boldsymbol{q}_1 + r_{22}\boldsymbol{q}_2, & r_{12} &= \boldsymbol{a}_2^T\boldsymbol{q}_1, \quad r_{22} = \|\tilde{\boldsymbol{q}}_2\|_2 \\
&\vdots & & \\
\boldsymbol{a}_j &= \sum_{i=1}^{j-1}(\boldsymbol{a}_j^T\boldsymbol{q}_i)\boldsymbol{q}_i + \tilde{\boldsymbol{q}}_j = \sum_{i=1}^{j-1} r_{ij}\boldsymbol{q}_i + r_{jj}\boldsymbol{q}_j, & r_{ij} &= \boldsymbol{a}_j^T\boldsymbol{q}_i, \quad r_{jj} = \|\tilde{\boldsymbol{q}}_j\|_2
\end{aligned}$$

on reconnaît la factorisation $A = QR$ avec $\boldsymbol{q}_j$ les vecteurs colonnes de Q et r_{ij} les composantes de R pour $i \leq j$.

Si $\operatorname{rang}(A) < n$, le processus ci-dessus échoue car il existe des étappes ℓ où $\tilde{\boldsymbol{q}}_\ell = \mathbf{0}$, c.-à-d,

$$\boldsymbol{a}_\ell = \sum_{i=1}^{\ell-1} (\boldsymbol{a}_\ell^T \boldsymbol{q}_i)\boldsymbol{q}_i.$$

En choisissant un vecteur unitaire $\boldsymbol{q}_\ell$ quelconque mais orthogonal aux $\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_{\ell-1}$, on obtient

$$\boldsymbol{a}_\ell = \sum_{i=1}^{\ell-1} r_{i\ell}\boldsymbol{q}_i + r_{\ell\ell}\boldsymbol{q}_\ell, \qquad r_{i\ell} = \boldsymbol{a}_\ell^T \boldsymbol{q}_i, \quad r_{\ell\ell} = 0,$$

et on peut continuer le processus. A la fin, on obtient de nouveau $A = QR$. Pour l'unicité voir la série d'exercices. □

La preuve nous donne immédiatement un algorithme pour calculer la factorisation QR.

Algorithme 6.6 (Factorisation QR – Gram–Schmidt classique (pas stable)).
Soit $A \in \mathbb{R}^{m\times n}$ où $m \geq n$ et $\operatorname{rang}(A) = n$.

1: **for** $j = 1, \ldots, n$ **do**
2: $\quad \tilde{\boldsymbol{q}}_j \leftarrow \boldsymbol{a}_j$
3: $\quad$ **for** $i = 1, \ldots, j-1$ **do**
4: $\quad\quad r_{ij} \leftarrow \boldsymbol{a}_j^T \boldsymbol{q}_i$
5: $\quad\quad \tilde{\boldsymbol{q}}_j \leftarrow \tilde{\boldsymbol{q}}_j - r_{ij}\boldsymbol{q}_i$
6: $\quad r_{jj} \leftarrow \|\tilde{\boldsymbol{q}}_j\|_2$
7: $\quad \boldsymbol{q}_j \leftarrow \tilde{\boldsymbol{q}}_j / r_{jj}$

L'algorithme ci-dessus n'est pas stable : numériquement, il produit des vecteurs $\boldsymbol{q}_j$ qui sont très loin d'être orthogonaux (c.-à-d., $\|\tilde{Q}^T\tilde{Q} - I\| \gg \varepsilon_{\text{mach}}$; voir les TP). Heureusement, on peut modifier le processus qui génère les vecteurs orthogonaux. Au lieu de calculer

$$\tilde{\boldsymbol{q}}_j = \boldsymbol{a}_j - \sum_{i=1}^{j-1} (\boldsymbol{q}_i^T \boldsymbol{a}_j)\boldsymbol{q}_i,$$

on fait

$$\begin{aligned}
\tilde{\boldsymbol{q}}_j^{(1)} &= \boldsymbol{a}_j - (\boldsymbol{q}_1^T \boldsymbol{a}_j)\boldsymbol{q}_1 \\
\tilde{\boldsymbol{q}}_j^{(2)} &= \tilde{\boldsymbol{q}}_j^{(1)} - (\boldsymbol{q}_2^T \tilde{\boldsymbol{q}}_j^{(1)})\boldsymbol{q}_2 \\
&\vdots \\
\tilde{\boldsymbol{q}}_j^{(i)} &= \tilde{\boldsymbol{q}}_j^{(i-1)} - (\boldsymbol{q}_i^T \tilde{\boldsymbol{q}}_j^{(i-1)})\boldsymbol{q}_i
\end{aligned}$$

Montrons que $\tilde{\boldsymbol{q}}_j^{(j-1)} = \tilde{\boldsymbol{q}}_j$ en arithmétique exacte :

$$\begin{aligned}
\tilde{\boldsymbol{q}}_j^{(j-1)} &= (I - \boldsymbol{q}_{j-1}\boldsymbol{q}_{j-1}^T)\tilde{\boldsymbol{q}}_j^{(j-2)} \\
&= (I - \boldsymbol{q}_{j-1}\boldsymbol{q}_{j-1}^T)(I - \boldsymbol{q}_{j-2}\boldsymbol{q}_{j-2}^T)\tilde{\boldsymbol{q}}_j^{(j-3)} \\
&= (I - \boldsymbol{q}_{j-1}\boldsymbol{q}_{j-1}^T - \boldsymbol{q}_{j-2}\boldsymbol{q}_{j-2}^T)\tilde{\boldsymbol{q}}_j^{(j-3)} \qquad (\text{car } \boldsymbol{q}_{j-1}^T\boldsymbol{q}_{j-2} = 0) \\
&= \cdots = (I - \boldsymbol{q}_{j-1}\boldsymbol{q}_{j-1}^T - \boldsymbol{q}_{j-2}\boldsymbol{q}_{j-2}^T - \cdots - \boldsymbol{q}_1\boldsymbol{q}_1^T)\boldsymbol{a}_j \\
&= \boldsymbol{a}_j - \sum_{i=1}^{j-1} (\boldsymbol{q}_i^T \boldsymbol{a}_j)\boldsymbol{q}_i = \tilde{\boldsymbol{q}}_j.
\end{aligned}$$

L'algorithme suivant est le processus de Gram-Schmidt que l'on utilise en pratique. Il donne des vecteurs $\tilde{\boldsymbol{q}}_j^{(j-1)}$ qui sont beaucoup plus précis, c.-à-d., orthogonaux.

Algorithme 6.7 (Factorisation QR – Gram–Schmidt modifiée (stable)).
Soit $A \in \mathbb{R}^{m\times n}$ où $m \geq n$ et $\operatorname{rang}(A) = n$.

1: **for** $i = 1,\dots,n$ **do**
2: $\quad \tilde{\boldsymbol{q}}_i \leftarrow \boldsymbol{a}_i$
3: **for** $i = 1,\dots,n$ **do**
4: $\quad r_{ii} \leftarrow \|\tilde{\boldsymbol{q}}_i\|_2$
5: $\quad \boldsymbol{q}_i \leftarrow \tilde{\boldsymbol{q}}_i / r_{ii}$
6: $\quad$ **for** $j = i+1,\dots,n$ **do**
7: $\quad\quad r_{ij} \leftarrow \boldsymbol{q}_i^T \tilde{\boldsymbol{q}}_j$
8: $\quad\quad \tilde{\boldsymbol{q}}_j \leftarrow \tilde{\boldsymbol{q}}_j - r_{ij}\boldsymbol{q}_i$

Le coût de cet algorithme est dominé par les calculs de $\boldsymbol{q}_i^T \tilde{\boldsymbol{q}}_j$ et $\tilde{\boldsymbol{q}}_j - r_{ij}\boldsymbol{q}_i$. On a donc pour m, n grand :

$$\text{Factorisation } QR \text{ réduite par Gram–Schmidt : } \sim \sum_{i=1}^{n}\sum_{j=i+1}^{n} 4m = 4m\sum_{i=1}^{n}(n-i) \sim 2mn^2.$$

Exemple 6.8

Pour illustrer l'amélioration de la méthode modifiée de Gram–Schmidt, calculons la factorisation QR de la matrice de Vandermonde A de taille 25×15 où

$$a_{ij} = \left[\left(\tfrac{i-1}{24}\right)^{j-1}\right], \qquad i = 1,\dots,25, \quad j = 1,\dots,15.$$

En MATLAB *on a le résultat suivant :*

$$\begin{aligned} &\textit{Alg. 6.6}: && \|\widetilde{Q}\widetilde{R} - A\|_\infty = 8.9\cdot 10^{-16}, && \|\widetilde{Q}^T\widetilde{Q} - I\|_\infty = 5.0 \\ &\textit{Alg. 6.7}: && \|\widetilde{Q}\widetilde{R} - A\|_\infty = 8.8\cdot 10^{-16}, && \|\widetilde{Q}^T\widetilde{Q} - I\|_\infty = 8.7\cdot 10^{-7} \end{aligned}$$

On constate que les deux algorithmes donnent une factorisation pour laquelle le produit $\widetilde{Q}\times\widetilde{R}$ est assez proche de la matrice A, mais que la matrice $\widetilde{Q}$ est très loin d'être orthonormale dans la méthode originale de Gram–Schmidt.

Il est parfois utile d'avoir une factorisation avec une matrice orthogonale. Ceci est facile à obtenir à partir de la factorisation réduite $A = QR$:

$$A = \underbrace{\begin{bmatrix} Q & Q_\perp \end{bmatrix}}_{=\widehat{Q}} \underbrace{\begin{bmatrix} R \\ 0 \end{bmatrix}}_{=\widehat{R}}, \qquad \text{où} \quad Q \in \mathbb{R}^{m\times n}, Q_\perp \in \mathbb{R}^{m\times(m-n)}, R \in \mathbb{R}^{n\times n}.$$

Ci-dessus, $Q_\perp$ est une matrice orthonormale dont les vecteurs sont aussi orthogonales à ceux de Q, ce qui implique $\widehat{Q}^T\widehat{Q} = I_n$. En pratique, on peut les obtenir en ajoutant $m-n$ vecteurs aléatoires supplémentaires à la fin de l'algoritme de Gram-Schmidt 6.7. On appelle la décomposition $A = \widehat{Q}\widehat{R}$ **la factorisation** QR **complète**.

Utilisons maintenant les factorisations $A = \widehat{Q}\widehat{R}$ pour résoudre (6.2). Comme le produit par une matrice orthogonale ne change pas la norme euclidienne du vecteur, on a

$$\|A\boldsymbol{x} - \boldsymbol{b}\|_2^2 = \|\widehat{Q}^T(\widehat{Q}\widehat{R}\boldsymbol{x} - \boldsymbol{b})\|_2^2 = \|\widehat{R}\boldsymbol{x} - \widehat{Q}^T\boldsymbol{b}\|_2^2 = \left\|\begin{bmatrix} R^T\boldsymbol{x} - Q^T\boldsymbol{b} \\ Q_\perp^T\boldsymbol{b} \end{bmatrix}\right\|_2^2 = \|R\boldsymbol{x} - Q^T\boldsymbol{b}\|_2^2 + \|Q_\perp^T\boldsymbol{b}\|_2^2.$$

Puisque $\|Q_\perp^T \boldsymbol{b}\|_2^2$ ne dépend pas de $\boldsymbol{x}$, les minimiseurs de $\min_{\boldsymbol{x}} \|A\boldsymbol{x}-\boldsymbol{b}\|_2$ sont les mêmes que ceux de $\min_{\boldsymbol{x}} \|R\boldsymbol{x}-Q^T\boldsymbol{b}\|_2$. On obtient alors la solution $\boldsymbol{x}^*$ de (6.2) en résolvant le système

$$R\boldsymbol{x}^* = Q^T\boldsymbol{b}.$$

Remarquons que ce système est triangulaire supérieur et inversible (car $\operatorname{rang}(A)=n$).

Algorithme 6.9 (Méthode des moindres carrés – factorisation QR).
Soit $A\in\mathbb{R}^{m\times n}$ où $m\geq n$ et $\operatorname{rang}(A)=n$.

1: Calculer la factorisation QR réduite $A=QR$.
2: Construire le vecteur $\boldsymbol{c}=Q^T\boldsymbol{b}$.
3: Résolution du système triangulaire supérieur $R\boldsymbol{x}=\boldsymbol{c}$.

En utilisant la méthode modifiée de Gram–Schmidt pour calculer $A=QR$, le coût de l'Algorithme 6.9 est pour m,n grands :

$$\text{Résolution des moindres carrés par Gram–Schmidt : } \sim (2m+1)n^2.$$

6.3 Réflexions de Householder

Dans l'Alg. 6.9, il est important que la matrice Q soit orthonormale pour le calcul de $Q^T\boldsymbol{b}$. Malgré l'amélioration de la méthode modifiée de Gram–Schmidt, l'exemple 6.8 montre que $\widetilde{Q}$ n'est pas très proche d'être orthonormale numériquement. En revanche, la méthode de Householder donne une matrice $\widetilde{Q}$ qui sera plus proche.

Rappelons que l'élimination de Gauss transforme une matrice carrée A en une matrice triangulaire supérieure U en multipliant par des matrices triangulaires inférieures L_k à gauche. On peut appliquer la même idée pour calculer la factorisation QR d'une matrice rectangulaire. Dans ce cas, on multiplie par des matrices orthogonales à gauche qui introduisent, colonne par colonne, des zéros pour que le résultat final soit la matrice R :

$$\underset{A}{\begin{bmatrix}\times&\times&\times\\ \times&\times&\times\\ \times&\times&\times\\ \times&\times&\times\\ \times&\times&\times\end{bmatrix}} \xrightarrow{Q_1} \underset{Q_1A}{\begin{bmatrix}\times&\times&\times\\ 0&\times&\times\\ 0&\times&\times\\ 0&\times&\times\\ 0&\times&\times\end{bmatrix}} \xrightarrow{Q_2} \underset{Q_2Q_1A}{\begin{bmatrix}\times&\times&\times\\ &\times&\times\\ &0&\times\\ &0&\times\\ &0&\times\end{bmatrix}} \xrightarrow{Q_3} \underset{Q_3Q_2Q_1A}{\begin{bmatrix}\times&\times&\times\\ &\times&\times\\ &&\times\\ &&0\\ &&0\end{bmatrix}} \tag{6.3}$$

En effet, à la dernière étappe on a obtenu une factorisation QR complète car

$$Q_3Q_2Q_1A=\begin{bmatrix}R\\0\end{bmatrix} \quad\Longleftrightarrow\quad A=\underbrace{Q_1^TQ_2^TQ_3^T}_{=\widehat{Q}}\widehat{R}$$

et $\widehat{Q}$ est bien orthogonale.

La question centrale est comment déterminer les matrices Q_k. Pour cela, on peut se servir des réflexions de Householder.

Définition 6.10. Soit $\boldsymbol{v}\in\mathbb{R}^n$ tel que $\|\boldsymbol{v}\|_2=1$. On appelle la matrice $H=I_n-2\boldsymbol{v}\boldsymbol{v}^T$ une **réflexion de Householder**.

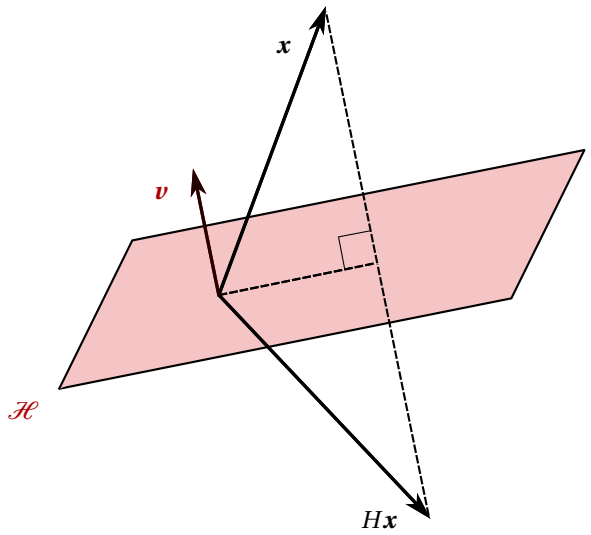


FIGURE 6.1 – Une réflexion de Householder.

Il est clair que H est une matrice symétrique. Vérifions qu'elle est également orthogonale :

$$HH^T = (I_n - 2\boldsymbol{v}\boldsymbol{v}^T)(I_n - 2\boldsymbol{v}\boldsymbol{v}^T) = I_n - 4\boldsymbol{v}\boldsymbol{v}^T + 4\boldsymbol{v}\underbrace{(\boldsymbol{v}^T\boldsymbol{v})}_{=1}\boldsymbol{v}^T = I_n.$$

De plus, la matrice H représente une réflexion géométriquement. Définissons l'hyper-plan $\mathscr{H} = \{\boldsymbol{x} \in \mathbb{R}^n : \boldsymbol{x} \perp \boldsymbol{v}\}$. On peut faire les observations suivantes :

(a) Pour tout $\boldsymbol{x} \in \mathbb{R}^n$, on a $H\boldsymbol{x} = \boldsymbol{x} - 2(\boldsymbol{v}^T\boldsymbol{x})\boldsymbol{v}$ ce qui montre que $\boldsymbol{x}$, $\boldsymbol{v}$ et $H\boldsymbol{x}$ se sont situés dans un même plan.

(b) Pour tout $\boldsymbol{x} \in \mathscr{H}$, on a $H\boldsymbol{x} = \boldsymbol{x}$ car $\boldsymbol{x}^T\boldsymbol{v} = 0$.

(c) Pour tout $\boldsymbol{x} \in \mathbb{R}^n$, on a $\boldsymbol{v}^T H\boldsymbol{x} = -\boldsymbol{v}^T\boldsymbol{x}$. Soit α l'angle entre $\boldsymbol{v}$ et $H\boldsymbol{x}$, alors l'angle entre $-\boldsymbol{v}$ et $\boldsymbol{x}$ est aussi α car

$$\cos\alpha = \frac{\boldsymbol{v}^T H\boldsymbol{x}}{\|\boldsymbol{v}\|_2\|H\boldsymbol{x}\|_2} = \frac{(-\boldsymbol{v})^T\boldsymbol{x}}{\|-\boldsymbol{v}\|_2\|\boldsymbol{x}\|_2}.$$

Donc, H est la réflexion à l'hyper-plan $\mathscr{H}$; voir aussi la figure 6.1.

Essayons maintenant d'introduire des zéros dans A en appliquant H. Il suffit de trouver $\boldsymbol{v}$ tel que le vecteur quelconque $\boldsymbol{x}$ est transformé de la manière suivante

$$\boldsymbol{x} = \begin{bmatrix} \times \\ \times \\ \times \\ \vdots \\ \times \end{bmatrix} \xrightarrow{H} H\boldsymbol{x} = \boldsymbol{x} - 2(\boldsymbol{v}^T\boldsymbol{x})\boldsymbol{v} = \begin{bmatrix} \pm\|\boldsymbol{x}\|_2 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \pm\|\boldsymbol{x}\|_2\boldsymbol{e}_1.$$

(Remarquons que $\|H\boldsymbol{x}\|_2$ est fixé car H est orthogonale.) Soit $\boldsymbol{x} \neq \boldsymbol{0}$. Puisque $\boldsymbol{x}, \boldsymbol{v}$ et $H\boldsymbol{x} = \pm\|\boldsymbol{x}\|_2\boldsymbol{e}_1$ sont coplanaires pour tout H, il existe un scalaire c tel que

$$\boldsymbol{x} + c\boldsymbol{v} = \pm\|\boldsymbol{x}\|_2\boldsymbol{e}_1 \qquad \Longrightarrow \qquad \boldsymbol{v} = \frac{\boldsymbol{x} \pm \|\boldsymbol{x}\|_2\boldsymbol{e}_1}{\|\boldsymbol{x} \pm \|\boldsymbol{x}\|_2\boldsymbol{e}_1\|_2}.$$

Pour éviter une soustraction mal conditionnée dans le calcul $H\boldsymbol{x}$, on prend habituellement

$$\boldsymbol{v} = \frac{\boldsymbol{x} + \operatorname{sgn}(x_1)\|\boldsymbol{x}\|_2 \boldsymbol{e}_1}{\|\boldsymbol{x} + \operatorname{sgn}(x_1)\|\boldsymbol{x}\|_2 \boldsymbol{e}_1\|_2}, \qquad \text{où } \operatorname{sgn}(x_1) = 0 \text{ si } x_1 = 0. \tag{6.4}$$

Si $\boldsymbol{x} = \boldsymbol{0}$, il ne faut pas introduire de zéros et donc on peut simplement choisir $H = I$, même si H n'est pas strictement une réflexion de Householder.

Retournons à l'exemple (6.3). D'après le résultat précédent, on a $Q_1 = H_1 = I_m - 2\boldsymbol{v}_1\boldsymbol{v}_1^T$ où $\boldsymbol{v}_1$ est déterminé pour introduire les zéros dans la première colonne de A (c.-à-d, $\boldsymbol{x}$ dans (6.4) est la première colonne de A). Pour la deuxième étape, on peut construire $H_2 = I_{m-1} - 2\boldsymbol{v}_2\boldsymbol{v}_2^T$ où $\boldsymbol{v}_2$ est maintenant choisi pour introduire les zéros en caractère gras dans la deuxième colonne de $Q_1 A$. La matrice Q_2 est ainsi

$$Q_2 = \begin{bmatrix} 1 & \\ & H_2 \end{bmatrix}.$$

Similairement, la dernière étape devient

$$Q_3 = \begin{bmatrix} 1 & & \\ & 1 & \\ & & H_3 \end{bmatrix}.$$

pour H_3 de taille 3×3.

Cet exemple se généralise facilement à $A \in \mathbb{R}^{m\times n}$ en utilisant

$$Q_k = \begin{bmatrix} I_{k-1} & \\ & H_k \end{bmatrix} \tag{6.5}$$

où $H_k \in \mathbb{R}^{m-k+1}$ est la réflexion qui introduit des zéros dans la $k$ième colonne de $Q_{k-1}\cdots Q_2 Q_1 A$. Après n étapes, ce processus donne la factorisation QR complète

$$A = \widehat{Q}\widehat{R} \qquad \text{où } \widehat{Q} = Q_1 Q_2 \cdots Q_n = \begin{bmatrix} Q & Q_\perp \end{bmatrix} \text{ et } \widehat{R} = \begin{bmatrix} R \\ 0 \end{bmatrix},$$

car les Q_i sont symétriques. Ainsi on a obtenu l'algorithme suivant qui calcule la factorisation QR (réduite).

Algorithme 6.11 (Factorisation QR – Householder).
Soit $A \in \mathbb{R}^{m\times n}$ où $m \geq n$ et $\operatorname{rang}(A) = n$.

1: **for** $k = 1, \ldots, n$ **do**
2: $\quad \boldsymbol{x} \leftarrow \boldsymbol{a}_{k:m,k}$
3: $\quad \boldsymbol{v}_k \leftarrow \boldsymbol{x} + \operatorname{sgn}(x_1)\|\boldsymbol{x}\|_2 \boldsymbol{e}_1$
4: $\quad$ **if** $\|\boldsymbol{v}_k\|_2 > 0$ **then**
5: $\quad\quad \boldsymbol{v}_k \leftarrow \boldsymbol{v}_k / \|\boldsymbol{v}_k\|_2$
6: $\quad$ **for** $i = k, \ldots, n$ **do**
7: $\quad\quad \boldsymbol{a}_{k:m,i} \leftarrow \boldsymbol{a}_{k:m,i} - 2(\boldsymbol{v}_k^T \boldsymbol{a}_{k:m,i})\boldsymbol{v}_k$
8: $R \leftarrow A_{1:n,1:n}$
9: La matrice Q est représentée par le produit $Q_1 Q_2 \cdots Q_n$ où Q_k est définie dans (6.5) et H_k est la réflection de Householder pour $\boldsymbol{v}_k$.

Remarquons que l'on a évité de construire les matrices H_k (et Q_k) explicitement car ceci nécessite $O((m-k)^2)$ opérations. En revanche, on les applique directement

$$H_k \boldsymbol{x} = \boldsymbol{x} - 2(\boldsymbol{x}^T \boldsymbol{v})\boldsymbol{v}$$

ce qui coûte seulement $4(m-k+1)$ opérations pour $\boldsymbol{x} \in \mathbb{R}^{m-k+1}$.

Pour m, n grands, le coût de l'Alg. 6.11 est environ $\sum_{k=1}^{n} 4(m-k+1)(n-k+1) \sim 4\int_1^n (m-x)(n-x)dx$, ce qui donne

$$\text{Factorisation } QR \text{ (réduite) par Householder : } \sim 2mn^2 - \frac{2}{3}n^3.$$

Remarquons que ce coût est sans calcul explicite de la matrice Q. Afin de l'obtenir, on garde tous les vecteurs $\boldsymbol{v}_k$ et puis on calcule

$$Q = \begin{bmatrix} \widehat{Q}\boldsymbol{e}_1 & \widehat{Q}\boldsymbol{e}_2 & \cdots & \widehat{Q}\boldsymbol{e}_n \end{bmatrix}$$

où $\widehat{Q}\boldsymbol{e}_k = Q_1(Q_2 \cdots (Q_n \boldsymbol{e}_k))$. En outre, souvent on n'a pas besoin de la matrice Q explicitement et il suffit de connaître le résultat après multiplication ; voir par exemple l'Alg. 6.9.

Exemple 6.12

Refaisons l'exemple 6.8 avec les réflexions de Householder. On obtient une matrice $\widetilde{Q}$ qui est plus proche d'être orthonormale :

$$\textit{Alg. 6.11}: \qquad \|\widetilde{Q}\widetilde{R} - A\|_\infty = 1.2 \cdot 10^{-14}, \qquad \|\widetilde{Q}^T\widetilde{Q} - I\|_\infty = 4.2 \cdot 10^{-14}$$

En comparaison, le résultat de la commande `[Q,R] = qr(A,0)` *en* MATLAB*, dont l'algorithme est basé sur les réflexions de Householder, donne*

$$\text{MATLAB}: \qquad \|\widetilde{Q}\widetilde{R} - A\|_\infty = 2.3 \cdot 10^{-15}, \qquad \|\widetilde{Q}^T\widetilde{Q} - I\|_\infty = 9.8 \cdot 10^{-15}$$

Chapitre 7

Equations non-linéaires

En pratique, on est souvent confronté à la résolution d'un système d'équations non linéaires. C'est-à-dire pour une fonction $f\colon \mathbb{R}^n \to \mathbb{R}^n$ donnée, on cherche un point $\boldsymbol{x}^* \in \mathbb{R}^n$, dite **une racine** de f, tel que

$$f(\boldsymbol{x}^*) = \mathbf{0}. \tag{7.1}$$

Remarquons que $\boldsymbol{x}^*$ n'est pas nécessairement unique. En général, il n'y a pas d'algorithme fini pour trouver les racines d'une fonction générale [1]. On est donc obligé d'utiliser des méthodes itératives.

7.1 Méthode de la bissection

Commençons à étudier le cas des fonctions scalaires $f\colon \mathbb{R} \to \mathbb{R}$. Supposons que f soit continue (trouver les racines d'une fonction qui n'est pas continue est un problème rare et très difficile numériquement). Notre première méthode repose sur le théorème suivant (voir Analyse I pour la preuve) :

Théorème 7.1 (Théorème des valeurs intermédiaires)**.** Pour une fonction continue $f\colon [a,b] \to \mathbb{R}$ avec $f(a)f(b) < 0$, il existe $x^* \in (a,b)$ tel que $f(x^*) = 0$.

L'idée est maintenant toute simple : étant données a, b telles que $f(a)f(b) < 0$, on divise l'intervalle $[a,b]$ en deux $m = (a+b)/2$, puis on vérifie le signe de $f(m)$ pour décider dans quel sous-intervalle il faut continuer à chercher. On obtient alors la **méthode de la bissection** :

Algorithme 7.2 (Méthode de la bissection)**.**
Soit $f\colon [a,b] \to \mathbb{R}$ continue telle que $f(a)f(b) < 0$.

$a^{(1)} \leftarrow a, \quad b^{(1)} \leftarrow b$
for $k = 1, 2, \ldots$ **do**
  $x^{(k)} \leftarrow (a^{(k)} + b^{(k)})/2$
  if $f(a^{(k)})f(x^{(k)}) < 0$ **then**
    $a^{(k+1)} \leftarrow a^{(k)}, \quad b^{(k+1)} \leftarrow x^{(k)}$
  else if $f(a^{(k)})f(x^{(k)}) > 0$ **then**
    $a^{(k+1)} \leftarrow x^{(k)}, \quad b^{(k+1)} \leftarrow b^{(k)}$
  else
    Arrêt : la racine est égal à $x^{(k)}$.

1. Par exemple, d'après Abel et Galois il n'existe pas de formule générale permettant de trouver les racines des polynômes de degré plus grand ou égal à 5.

Soit $f(x) = \cos(x)$. On aimerait calculer l'unique racine de $f(x)$ entre 0 et 2, c-à-d, $x^* = \pi/2$. Puisque $f(0) = 1$ et $f(2) = -0.41\ldots$, l'Alg. 7.2 donne le résultat suivant en MATLAB :

```
k =  1, xk = 1.500000000000000, xk - x* = -0.070796326794897
k =  2, xk = 1.750000000000000, xk - x* = +0.179203673205103
k =  3, xk = 1.625000000000000, xk - x* = +0.054203673205103
k =  4, xk = 1.562500000000000, xk - x* = -0.008296326794897
k =  5, xk = 1.593750000000000, xk - x* = +0.022953673205103
k =  6, xk = 1.578125000000000, xk - x* = +0.007328673205103
k =  7, xk = 1.570312500000000, xk - x* = -0.000483826794897
...
k = 41, xk = 1.570796326794607, xk - x* = -0.000000000000290
k = 42, xk = 1.570796326794834, xk - x* = -0.000000000000062
k = 43, xk = 1.570796326794948, xk - x* = +0.000000000000052
k = 44, xk = 1.570796326794891, xk - x* = -0.000000000000005
k = 45, xk = 1.570796326794920, xk - x* = +0.000000000000023
k = 46, xk = 1.570796326794905, xk - x* = +0.000000000000009
k = 47, xk = 1.570796326794898, xk - x* = +0.000000000000002
```

L'exemple ci-dessus montre que la méthode de la bissection continue indéfiniment car $x^{(k)} \in \mathbb{Q}$ et $x^* \in \mathbb{R}$. Par contre, elle donne une succession de points $x^{(k)}$, appelés les **itérés**, qui se rapprochent graduellement de la racine x^*. Cette méthode est donc appelée **itérative** car, en général, elle ne réussit pas à trouver la solution exacte pour un nombre fini d'étapes. Cela contraste avec les méthodes directes : en calcul exact, par exemple, l'élimination de Gauss trouve la solution en $\frac{2}{3}n^3$ opérations élémentaires.

Il est facile d'estimer l'erreur de la méthode de la bissection. Par construction, on sait qu'il existe une racine de f dans l'intervalle $[a^{(k)}, b^{(k)}]$. L'erreur à l'etape k est donc bornée par

$$|x^* - x^{(k)}| \leq \frac{1}{2}(b^{(k)} - a^{(k)}) = 2^{-k}(b-a), \qquad k \geq 1,$$

où x^* est une racine de f dans $[a, b]$. La méthode est donc convergente :

$$\lim_{k\to\infty} |x^* - x^{(k)}| = 0 \qquad \Longrightarrow \qquad \lim_{k\to\infty} x^{(k)} = x^*.$$

S'il y a plusieurs racines dans $[a, b]$, on ne sait pas vers laquelle l'algorithme converge.

Le résultat précédent permet de déterminer à l'avance le nombre d'étapes k nécessaires tel que l'erreur est plus petit que la tolérance $\epsilon > 0$. En particulier, pour que $|x^* - x^{(k)}| \leq \epsilon$, il faut que

$$k \geq \log_2(b-a) - \log_2(\epsilon) = \frac{\log((b-a)/\epsilon)}{\log(2)}.$$

Remarquons que, en virgule flottante, on ne peut pas choisir un trop petit ϵ ; par exemple, on peut choisir $\epsilon \approx (b-a)\varepsilon_{\text{mach}}$.

7.2 Méthode des points fixes

Soit $f\colon \mathbb{R}^n \to \mathbb{R}^n$ une fonction à plusieurs variables. Puisque la généralisation de la méthode de la bissection est assez difficile pour trouver les racines de f, on a besoin de nouveaux algorithmes. Une idée fructueuse est de considérer le problème du calcul d'un **point fixe** de l'application $\Phi\colon \mathbb{R}^n \to \mathbb{R}^n$; c.-à-d., on cherche $\boldsymbol{x}^* \in \mathbb{R}^n$ tel que

$$\Phi(\boldsymbol{x}^*) = \boldsymbol{x}^*. \tag{7.2}$$

Les problèmes (7.1) et (7.2) sont équivalents et il y a beaucoup de possibilités pour écrire (7.1) sous la forme (7.2). Par exemple, on peut définir Φ comme

$$\Phi(\boldsymbol{x}) = \boldsymbol{x} - f(\boldsymbol{x})$$

ou, plus généralement,

$$\Phi(\boldsymbol{x}) = \boldsymbol{x} - Bf(\boldsymbol{x}), \quad B \in \mathbb{R}^{n\times n} \text{ inversible.}$$

Pour résoudre (7.2), on se donne une approximation initiale $\boldsymbol{x}^{(0)}$ et on considère la méthode itérative

$$\boldsymbol{x}^{(k+1)} = \Phi(\boldsymbol{x}^{(k)}), \qquad k \geq 0. \tag{7.3}$$

Cet algorithme s'appelle la **méthode des points fixes**.

Habituellement, la fonction Φ est au moins continue. Dans ce cas, si la suite $(\boldsymbol{x}^{(k)})$ converge, sa limite, disons $\boldsymbol{x}^*$, est un point fixe de Φ :

$$\boldsymbol{x}^* = \lim_{k\to\infty} \boldsymbol{x}^{(k)} = \lim_{k\to\infty} \Phi(\boldsymbol{x}^{(k-1)}) = \Phi\left(\lim_{k\to\infty} \boldsymbol{x}^{(k-1)}\right) = \Phi(\boldsymbol{x}^*). \tag{7.4}$$

Même pour Φ continue, il y a une infinité de possibilités de transformer l'équation $f(\boldsymbol{x}) = \mathbf{0}$ en $\Phi(\boldsymbol{x}) = \boldsymbol{x}$. Certains choix donnent des algorithmes convergents et d'autres pas.

Exemple 7.3

Essayions de trouver la racine de $\cos x$ *sur* $[0,2]$ *par la méthode des point fixes pour les fonctions suivantes :*

$$\Phi_1(x) = x + \cos x, \quad \Phi_2(x) = x + \tfrac{1}{2}\cos x, \quad \Phi_3(x) = x + 10\cos x.$$

Observez que $\pi/2$ *est en fait un point fixe pour tous les* Φ_k*. Les itérés pour chaque* Φ_k *de partir à* $x^{(0)} = 1$ *sont mis dans le tableau ci-dessus. On constate que* Φ_1 *converge très rapidement vers* $\pi/2$*, tandis que* Φ_2 *plutôt lentement. Cependant,* Φ_3 *ne converge pas ; voir aussi la figure 7.1 qui montre les* $x^{(1)}, x^{(2)}, \ldots, x^{(1000)}$ *dans ce cas.*

k	$\Phi_1(x^{(k-1)})$	$\Phi_2(x^{(k-1)})$	$\Phi_3(x^{(k-1)})$
1	*+1.540302305868140*	*+1.270151152934070*	*+6.403023058681398*
2	*+1.570791601024261*	*+1.418219404556379*	*+16.331303517801356*
3	*+1.570796326794897*	*+1.494212214386053*	*+8.211972344440587*
4	*+1.570796326794897*	*+1.532466850275086*	*+4.708041987046139*
5	*+1.570796326794897*	*+1.551626896237748*	*+4.664572190564352*
6	*+1.570796326794897*	*+1.561211024515904*	*+4.186586488994912*
7	*+1.570796326794897*	*+1.566003602265855*	*-0.832486093236126*
8	*+1.570796326794897*	*+1.568399955356230*	*+5.897905008594261*
9	*+1.570796326794897*	*+1.569598139928781*	*+15.164836284312294*
10	*+1.570796326794897*	*+1.570197233218490*	*+6.603868326260775*
20	*+1.570796326794897*	*+1.570795741742529*	*+13.008129284555011*
50	*+1.570796326794897*	*+1.570796326794896*	*+45.516755237421769*
∞	$\pi/2$	$\pi/2$	—

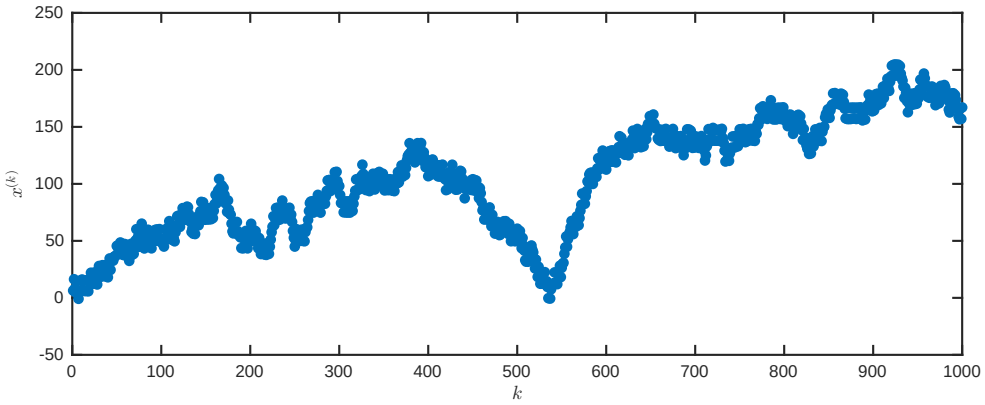


FIGURE 7.1 – Les itérés $x^{(k)} = \Phi_3(x^{(k-1)})$ pour $x^{(0)} = 1$ et $\Phi_3(x) = x + 10\cos x$.

7.3 Théorème du point fixe de Banach

L'exemple précédent montre que l'itération (7.3) n'est pas toujours convergente, même pour Φ continue. En outre, le taux de convergence varie fortement pour les méthodes convergentes. Cependant, pour des fonctions Φ particulières, la méthode des points fixes admet une analyse élégante.

Définition 7.4. Soit $f\colon A \to \mathbb{R}^m$ où $A \subseteq \mathbb{R}^n$. On dit que f est L-**lipschitzienne** si

$$\forall \boldsymbol{x}, \boldsymbol{y} \in A: \quad \|f(\boldsymbol{x}) - f(\boldsymbol{y})\| \leq L\|\boldsymbol{x} - \boldsymbol{y}\|,$$

où $0 \leq L < \infty$.

Remarquons que la constante L dans la définition dépend du choix des normes. De plus, il n'est pas difficile de démontrer à l'aide de la définition de la continuité, que toute fonction lipschitzienne est aussi continue.

Définition 7.5. L'application f est **une contraction** si elle est L-lipschitzienne avec $L < 1$.

Exemple 7.6

Si $f : \mathbb{R} \to \mathbb{R}$ de classe C^1 satisfait $|f'(x)| \leq L < 1$ pour tout $x \in \mathbb{R}$, alors f est une contraction : d'après le théorème des accroissements finis, on a

$$|f(x) - f(y)| \leq \max_{\xi \in (x,y)} |f'(\xi)|\,|x - y| \leq L|x - y|.$$

Théorème 7.7 (Théorème du point fixe (Banach, 1922)). Si $A \subseteq \mathbb{R}^n$ est un sous-ensemble fermé et $\Phi : A \to A$ une contraction, alors

(a) il existe un point fixe unique $\boldsymbol{x}^* = \Phi(\boldsymbol{x}^*)$ dans A,

(b) la suite $\boldsymbol{x}^{(k+1)} = \Phi(\boldsymbol{x}^{(k)})$ converge vers $\boldsymbol{x}^*$ pour tout $\boldsymbol{x}^{(0)} \in A$, en particulier,

$$\|\boldsymbol{x}^{(k)} - \boldsymbol{x}^*\| \leq L^k \|\boldsymbol{x}^{(0)} - \boldsymbol{x}^*\| \leq \frac{L^k}{1-L} \|\boldsymbol{x}^{(1)} - \boldsymbol{x}^{(0)}\|, \quad k \geq 0. \tag{7.5}$$

Démonstration. La démonstration est composée de 3 étapes.

L'existence d'un point fixe. Remarquons d'abord que $\boldsymbol{x}^{(k+1)} = \Phi(\boldsymbol{x}^{(k)}) \in A$ pour tout $k \geq 0$ car l'image de Φ est A par définition. Puis, on démontre que la suite $(\boldsymbol{x}^{(k)})$ est de Cauchy. Puisque Φ est L-lipschitzienne, on a pour $k \geq 0$

$$\|\boldsymbol{x}^{(k+1)} - \boldsymbol{x}^{(k)}\| = \|\Phi(\boldsymbol{x}^{(k)}) - \Phi(\boldsymbol{x}^{(k-1)})\| \leq L \|\boldsymbol{x}^{(k)} - \boldsymbol{x}^{(k-1)}\| \leq \cdots \leq L^k \|\boldsymbol{x}^{(1)} - \boldsymbol{x}^{(0)}\|.$$

Ainsi, pour $m \geq 0$

$$\begin{aligned}
\|\boldsymbol{x}^{(k+m)} - \boldsymbol{x}^{(k)}\| &\leq \|\boldsymbol{x}^{(k+m)} - \boldsymbol{x}^{(k+m-1)}\| + \cdots + \|\boldsymbol{x}^{(k+1)} - \boldsymbol{x}^{(k)}\| \\
&\leq (L^{k+m-1} + \cdots + L^{k+1} + L^k) \|\boldsymbol{x}^{(1)} - \boldsymbol{x}^{(0)}\| \\
&= (L^{m-1} + \cdots + L + 1) L^k \|\boldsymbol{x}^{(1)} - \boldsymbol{x}^{(0)}\| \\
&\leq L^k \sum_{m=0}^{\infty} L^m \|\boldsymbol{x}^{(1)} - \boldsymbol{x}^{(0)}\| \\
&\leq \frac{L^k}{1-L} \|\boldsymbol{x}^{(1)} - \boldsymbol{x}^{(0)}\|,
\end{aligned} \tag{7.6}$$

car $0 \leq L < 1$.

Soit $\varepsilon > 0$ donné. Puisque $\lim_{k\to\infty} L^k = 0$, la borne (7.6) montre qu'on peut choisir K tel que

$$\|\boldsymbol{x}^{(k+m)} - \boldsymbol{x}^{(k)}\| < \varepsilon, \qquad \forall k, m \geq K.$$

En particulier, la borne ci-dessus est satisfaite si

$$\frac{L^k}{1-L} \|\boldsymbol{x}^{(1)} - \boldsymbol{x}^{(0)}\| \leq \varepsilon.$$

En prenant le logarithme, on obtient que K est le plus petit entier positif tel que

$$K \geq \frac{\log[\|\boldsymbol{x}^{(1)} - \boldsymbol{x}^{(0)}\| / (\varepsilon(1-L))]}{\log(1/L)}. \tag{7.7}$$

La suite $(\boldsymbol{x}^{(k)})$ est donc de Cauchy ce qui implique qu'elle est convergente car $\mathbb{R}^n$ est complet. De plus, sa limite est un point fixe de Φ d'après (7.4) et elle est dans A car $A \subseteq \mathbb{R}^n$ est fermé.

L'unicité du point fixe. Soit $\boldsymbol{x}$ et $\boldsymbol{y}$ deux points fixes dans A. Alors

$$\|\boldsymbol{x} - \boldsymbol{y}\| = \|\Phi(\boldsymbol{x}) - \Phi(\boldsymbol{y})\| \leq L \|\boldsymbol{x} - \boldsymbol{y}\|$$

et donc

$$\underbrace{(1-L)}_{\in(0,1)} \|\boldsymbol{x} - \boldsymbol{y}\| \leq 0 \quad \Longrightarrow \quad \|\boldsymbol{x} - \boldsymbol{y}\| = 0 \quad \Longrightarrow \quad \boldsymbol{x} = \boldsymbol{y}$$

c.-à-d., le point fixe est unique.

L'erreur. Puisque $\boldsymbol{x}^{(k)} \to \boldsymbol{x}^*$, on a, en posant $k=0$ dans (7.6), que

$$\|\boldsymbol{x}^{(0)} - \boldsymbol{x}^*\| = \lim_{m\to\infty} \|\boldsymbol{x}^{(m)} - \boldsymbol{x}^{(0)}\| \le \frac{1}{1-L}\|\boldsymbol{x}^{(1)} - \boldsymbol{x}^{(0)}\|.$$

De plus, on a

$$\|\boldsymbol{x}^{(k)} - \boldsymbol{x}^*\| = \|\Phi(\boldsymbol{x}^{(k-1)}) - \Phi(\boldsymbol{x}^*)\| \le L\|\boldsymbol{x}^{(k-1)} - \boldsymbol{x}^*\| \le \cdots \le L^k\|\boldsymbol{x}^{(0)} - \boldsymbol{x}^*\|.$$

On obtient (7.5) en combinant ces deux bornes. □

7.4 Taux de convergence

Définition 7.8. Soit $\boldsymbol{x}^{(k)} \to \boldsymbol{x}^*$ pour $k \to \infty$. On dit que la *convergence de* $(\boldsymbol{x}^{(k)})$ *est d'ordre au moins* m s'il existe une constante C telle que

$$\|\boldsymbol{x}^{(k+1)} - \boldsymbol{x}^*\| \le C\|\boldsymbol{x}^{(k)} - \boldsymbol{x}^*\|^m$$

lorsque k est suffisamment grand.

On dit aussi que la convergence est
— linéaire si $m=1$ et $C<1$;
— superlinéaire si $m>1$;
— quadratique si $m=2$;
— cubique si $m=3$;
— ...

Définissons $\boldsymbol{e}^{(k)} = \boldsymbol{x}^{(k)} - \boldsymbol{x}^*$ et $e_k = \|\boldsymbol{e}^{(k)}\|$. Les suites suivantes

$$(a)\ e_{k+1} = 0.1\,e_k,\ e_0 = 0.05 \qquad (b)\ e_{k+1} = 0.1\,e_k + 10\,e_k^2,\ e_0 = 0.05 \qquad (c)\ e_{k+1} = 10\,e_k^2,\ e_0 = 0.09.$$

sont visibles dans la Fig. 7.2 utilisant une échelle logarithmique sur l'axe y. Remarquons que la convergence linéaire est clairement représentée par une droite dans un tel graphique car

$$e_{k+1} \approx Ce_k \quad\Longrightarrow\quad e_k \approx C^k e_0 \quad\Longrightarrow\quad \log e_k \approx k\log C + \log e_0.$$

Sous les conditions du Thm. 7.7, la méthode des points fixes $\boldsymbol{x}^{(k+1)} = \Phi(\boldsymbol{x}^{(k)})$ converge au moins linéairement. En général, on peut calculer l'ordre à l'aide d'un développement de Taylor pour Φ suffisamment dérivable :

$$\begin{aligned}\Phi(\boldsymbol{x}^{(k)}) &= \Phi(\boldsymbol{x}^* + \boldsymbol{e}^{(k)}) \\ &= \Phi(\boldsymbol{x}^*) + \sum_j \frac{\partial\Phi(\boldsymbol{x}^*)}{\partial x_j}(\boldsymbol{e}^{(k)})_j + \tfrac{1}{2}\sum_{i,j}\frac{\partial^2\Phi(\boldsymbol{x}^*)}{\partial x_i\partial x_j}(\boldsymbol{e}^{(k)})_i(\boldsymbol{e}^{(k)})_j + O(\|\boldsymbol{e}^{(k)}\|^3) \\ &= \Phi(\boldsymbol{x}^*) + \Phi'(\boldsymbol{x}^*)\boldsymbol{e}^{(k)} + \tfrac{1}{2}\Phi''(\boldsymbol{x}^*)(\boldsymbol{e}^{(k)},\boldsymbol{e}^{(k)}) + O(\|\boldsymbol{e}^{(k)}\|^3),\end{aligned}$$

où

$\Phi'(\boldsymbol{x}^*) : \mathbb{R}^n \to \mathbb{R}^n$ est une application linéaire (la matrice jacobienne (3.3)),
$\Phi''(\boldsymbol{x}^*) : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n$ est une application *bilinéaire*.

On a donc

$$\boldsymbol{e}^{(k+1)} = \boldsymbol{x}^{(k+1)} - \boldsymbol{x}^* = \Phi(\boldsymbol{x}^{(k)}) - \Phi(\boldsymbol{x}^*) = \Phi'(\boldsymbol{x}^*)\boldsymbol{e}^{(k)} + \tfrac{1}{2}\Phi''(\boldsymbol{x}^*)(\boldsymbol{e}^{(k)},\boldsymbol{e}^{(k)}) + O(\|\boldsymbol{e}^{(k)}\|^3)$$

et *en supposant que* $\boldsymbol{x}^{(k)} \to \boldsymbol{x}^*$ (alors $\|\boldsymbol{e}^{(k)}\| \to 0$) :

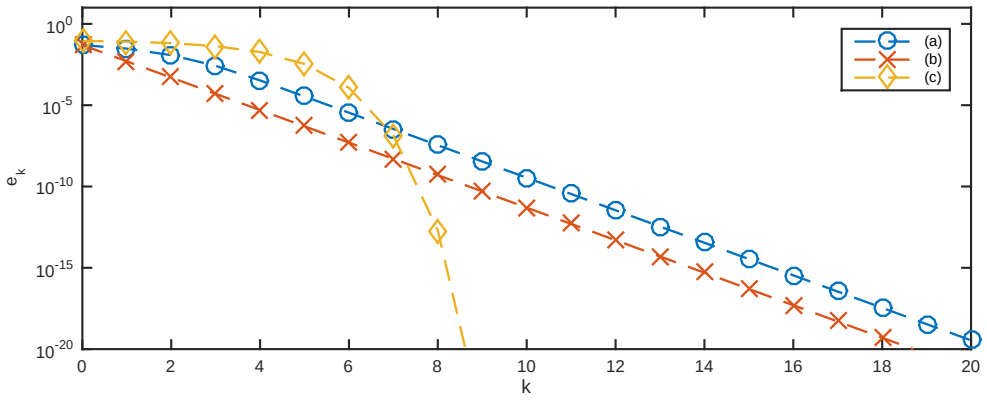


FIGURE 7.2 – Deux suites avec une convergence linéaire et une quadratique.

— Si $\Phi'(\boldsymbol{x}^*) \neq \mathbf{0}$ et $C_* = \|\Phi'(\boldsymbol{x}^*)\| < 1$, alors

$$\boldsymbol{e}^{(k+1)} = \Phi'(\boldsymbol{x}^*)\boldsymbol{e}^{(k)} + O(\|\boldsymbol{e}^{(k)}\|^2) \implies \|\boldsymbol{e}^{(k+1)}\| \leq (C_* + O(\|\boldsymbol{e}^{(k)}\|))\|\boldsymbol{e}^{(k)}\|.$$

La convergence est donc linéaire car pour $\|\boldsymbol{e}^{(k)}\|$ suffisamment petit, on a $C_* + O(\|\boldsymbol{e}^{(k)}\|) < 1$.

— Si $\Phi'(\boldsymbol{x}^*) = \mathbf{0}$ mais $\Phi''(\boldsymbol{x}^*) \neq \mathbf{0}$, alors

$$\boldsymbol{e}^{(k+1)} = \tfrac{1}{2}\Phi''(\boldsymbol{x}^*)(\boldsymbol{e}^{(k)}, \boldsymbol{e}^{(k)}) + O(\|\boldsymbol{e}^{(k)}\|^3) \implies \|\boldsymbol{e}^{(k+1)}\| \leq C\|\boldsymbol{e}^{(k)}\|^2,$$

pour k suffisamment grand. La convergence est donc quadratique.

Il est important de remarquer que cette analyse est asymptotique, c.-à-d, valable seulement pour $\boldsymbol{x}^{(k)}$ proche de $\boldsymbol{x}^*$. Si $\boldsymbol{x}^{(k)}$ est trop loin de la solution, la méthode peut diverger, osciller ou se comporter de façon chaotique.

Exemple 7.9

Pour résoudre $xe^x = 1$, dont la solution est $x^* = 0.567143\ldots$, on peut utiliser les méthodes des points fixes suivantes

1. *La formule $x = 1/e^x$ donne l'itération $x^{(k+1)} = e^{-x_k}$. On a*

$$|\Phi'(x^*)| = |-e^{-x^*}| = |-x^*| = 0.567143\ldots < 1, \qquad \Phi''(x^*) = e^{-x^*} = x^* \neq 0.$$

La convergence est donc linéaire si $x^{(0)}$ est suffisamment proche de x^*.

2. *Ecrire $xe^x + x = 1 + x \implies x = (1+x)/(e^x+1)$ donne $x^{(k+1)} = (1+x_k)/(1+e^{x_k})$. On a*

$$\Phi'(x^*) = \left.\frac{1 - xe^x}{(1+e^x)^2}\right|_{x=x^*} = 0, \qquad \Phi''(x^*) = \left.\frac{e^x(e^x x - e^x - x - 3)}{(1+e^x)^3}\right|_{x=x^*} = -0.361\ldots \neq 0.$$

La convergence est donc quadratique.

Pour les fonctions Φ_i de l'exemple 7.3, on a

$\Phi_1'(\frac{\pi}{2}) = \Phi_1''(\frac{\pi}{2}) = 0, \quad \Phi_1^{(3)}(\frac{\pi}{2}) = 1 \implies$ *convergence cubique ($m = 3$)*

$\Phi_2'(\frac{\pi}{2}) = \frac{1}{2} \implies$ *convergence linéaire ($m = 1$)*

$\Phi_3'(\frac{\pi}{2}) = -9 \implies$ *pas de convergence vers $x^* = \frac{\pi}{2}$*

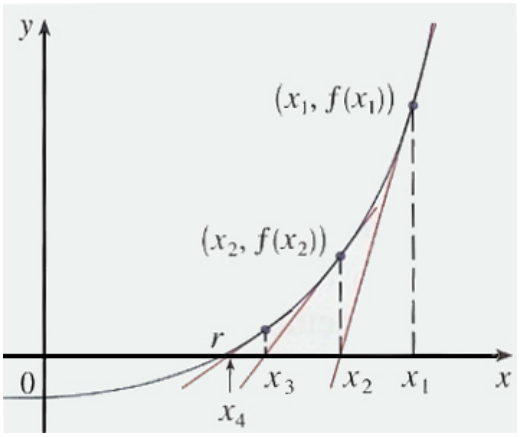


FIGURE 7.3 – La méthode de Newton scalaire.

7.5 Méthode de Newton scalaire

La méthode de Newton est l'une des méthodes les plus importantes pour la résolution des équations non-linéaires. Commençons d'abord par l'expliquer pour des fonctions scalaires d'une façon intuitive. Dans la section suivante, nous traiterons le cas des fonctions de plusieurs variables plus rigoureusement.

Rappelons que l'on souhaite trouver une racine de $f : \mathbb{R} \to \mathbb{R}$, disons $x^* \in \mathbb{R}$, tel que $f(x^*) = 0$. Supposons que f est assez régulière afin de la développer $f(x)$ en série de Taylor en $x = x^{(0)}$:

$$f(x) = f(x^{(0)}) + f'(x^{(0)})(x - x^{(0)}) + \frac{1}{2} f''(x^{(0)})(x - x^{(0)})^2 + \cdots$$

En négligeant les termes d'ordre 2 ou plus, on obtient la fonction *linéarisée* :

$$f(x) \approx \ell_0(x) = f(x^{(0)}) + f'(x^{(0)})(x - x^{(0)}) \tag{7.8}$$

En remplaçant f par ℓ_0, on peut utiliser la racine exacte de ℓ_0 comme nouvelle estimation de celle de f. En posant $\ell(x^{(1)}) = 0$ dans (7.8) et résolvant pour $x^{(1)}$, on obtient

$$x^{(1)} = x^{(0)} - \frac{f(x^{(0)})}{f'(x^{(0)})}.$$

Ce processus est facile à répéter pour $k = 1, 2, 3, 4, \ldots$: la fonctions linéaire en $x^{(k)}$ satisfait

$$f(x) \approx \ell_k(x) = f(x^{(k)}) + f'(x^{(k)})(x - x^{(k)}). \tag{7.9}$$

Sa racine donne la nouvelle approximation

$$x^{(k+1)} = x^{(k)} - \frac{f(x^{(k)})}{f'(x^{(k)})}. \tag{7.10}$$

On a obtenu ainsi la **méthode de Newton pour des fonctions scalaires**. Remarquons que la méthode n'est pas définie si f n'est pas dérivable, ainsi que si $f'(x^{(k)})$ s'annule.

Le principe de remplacer f par des fonctions linéaires ℓ_k pour lesquelles les racines sont faciles à trouver est visible dans la Figure 7.3. L'approximation de f par ℓ_k devient meilleure

lorsque $x^{(k)} \to 0$ ce qui suggère que la méthode de Newton peut converger très vite. Un petit exemple montre que cela est en fait le cas. Dans le tableau ci-dessous, on voit le résultat pour la fonction $f(x) = e^x - x + 2$. Le nombre de chiffres significatifs double après chaque étape ; cela est typiquement le cas pour la convergence quadratique.

k	$x^{(k)}$	$x^{(k)} - x^*$
1	1.000000000000000	-1.46×10^{-01}
2	1.163953413738653	$+1.77 \times 10^{-02}$
3	1.146421185043009	$+2.27 \times 10^{-04}$
4	1.146193258704499	$+3.80 \times 10^{-08}$
5	1.146193220620584	$+1.11 \times 10^{-15}$
6	1.146193220620582	-2.22×10^{-16}
∞	1.1461932206205825853…	

Montrons l'ordre de convergence par la théorie de la section précédente. La méthode de Newton est une méthode des points fixes pour la fonction

$$\Phi(x) = x - \frac{f(x)}{f'(x)}.$$

Supposons que $f \in C^2$ et $f'(x^*) \neq 0$. Alors, $f'(x) \neq 0$ pour x suffisamment proche de x^* car f' est continue en x^*. En prenant la dérivée de Φ, on obtient

$$\Phi'(x^*) = 1 - \frac{f'(x^*)f'(x^*) - f(x^*)f''(x^*)}{[f'(x^*)]^2} = \frac{f(x^*)f''(x^*)}{[f'(x^*)]^2}.$$

Puisque f'' est continue en x^*, on a aussi que $|f''(x^*)| < \infty$. On a donc $\Phi'(x^*) = 0$ et la convergence de la méthode de Newton est au moins quadratique si $x^{(k)}$ est assez proche de x^*.

7.6 Méthode de Newton vectorielle

Généralisons maintenant la méthode de Newton pour des fonctions de plusieurs variables $f\colon \mathbb{R}^n \to \mathbb{R}^n$. L'idée est de nouveau d'utiliser une approximation linéaire de f en $\boldsymbol{x}^{(k)}$. Pour cela, on a besoin d'une version multidimensionale de Taylor. Le résultat suivant sera suffisant pour nous :

Lemme 7.10. Soit $f\colon \mathbb{R}^n \to \mathbb{R}^m$ dérivable sur $D \subset \mathbb{R}^n$ convexe et ouvert. Supposons que la matrice jacobienne de f est L-lipschitzienne sur D :

$$\|f'(\boldsymbol{y}) - f'(\boldsymbol{x})\| \leq L\|\boldsymbol{y} - \boldsymbol{x}\| \qquad \forall \boldsymbol{x}, \boldsymbol{y} \in D.$$

Alors

$$\|f(\boldsymbol{x}) + f'(\boldsymbol{x})(\boldsymbol{y} - \boldsymbol{x}) - f(\boldsymbol{y})\| \leq \tfrac{L}{2}\|\boldsymbol{y} - \boldsymbol{x}\|^2.$$

Démonstration. Par hypothèse, la fonction

$$\varphi\colon [0,1] \to \mathbb{R}^n, \qquad t \mapsto f(\boldsymbol{x} + t(\boldsymbol{y} - \boldsymbol{x}))$$

est dérivable pour tout $\boldsymbol{x}, \boldsymbol{y} \in D$ car D est convexe. En particulier, la règle de la chaîne donne

$$\varphi'(t) = f'(\boldsymbol{x} + t(\boldsymbol{y} - \boldsymbol{x}))\,(\boldsymbol{y} - \boldsymbol{x}), \qquad 0 \leq t \leq 1.$$

En définissant,

$$w(\boldsymbol{x},\boldsymbol{y}) = f(\boldsymbol{x}) + f'(\boldsymbol{x})(\boldsymbol{y}-\boldsymbol{x}) - f(\boldsymbol{y})$$

on obtient

$$\|w(\boldsymbol{x},\boldsymbol{y})\| = \|\varphi(1) - \varphi'(0) - \varphi(0)\| = \|\int_0^1 \varphi'(t) - \varphi'(0)\,dt\| \le \int_0^1 \|\varphi'(t) - \varphi'(0)\|\,dt.$$

Maintenant utilisons le fait que f' est Lipschitzienne :

$$\|\varphi'(t) - \varphi'(0)\| = \|[f'(\boldsymbol{x} + t(\boldsymbol{y}-\boldsymbol{x})) - f'(\boldsymbol{x})](\boldsymbol{y}-\boldsymbol{x})\| \le tL\|\boldsymbol{y}-\boldsymbol{x}\|^2.$$

Finalement, on obtient le résultat du lemme :

$$\|w(\boldsymbol{x},\boldsymbol{y})\| \le \int_0^1 tL\|\boldsymbol{y}-\boldsymbol{x}\|^2\,dt = \tfrac{L}{2}\|\boldsymbol{y}-\boldsymbol{x}\|^2. \qquad \square$$

D'après le lemme précédent, on sait que l'approximation linéaire

$$f(\boldsymbol{x}) \approx \ell_k(\boldsymbol{x}) = f(\boldsymbol{x}^{(k)}) + f'(\boldsymbol{x}^{(k)})(\boldsymbol{x}-\boldsymbol{x}^{(k)})$$

est une bonne approximation de f dans un voisinage de $\boldsymbol{x}^{(k)}$. On remplace f par ℓ_k et on utilise la racine de ℓ_k comme nouvelle estimation de $\boldsymbol{x}^*$. En posant

$$\boldsymbol{0} = \ell_k(\boldsymbol{x}^{(k+1)}) = f(\boldsymbol{x}^{(k)}) + f'(\boldsymbol{x}^{(k)})(\boldsymbol{x}^{(k+1)} - \boldsymbol{x}^{(k)}),$$

on obtient l'itération

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} - [f'(\boldsymbol{x}^{(k)})]^{-1} f(\boldsymbol{x}^{(k)}).$$

En pratique, on ne calcule jamais la matrice inverse $[f'(\boldsymbol{x}^{(k)})]^{-1}$, mais on résout le système linéaire de taille $n \times n$ comme dans la formulation suivante :

$$\begin{aligned} &f'(\boldsymbol{x}^{(k)})\boldsymbol{p}_k = -f(\boldsymbol{x}^{(k)}) \qquad \text{(résoudre un système linéaire)} \\ &\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \boldsymbol{p}_k \end{aligned} \tag{7.11}$$

Cela est la **méthode de Newton pour des fonctions de plusieurs variables**. Remarquons qu'elle n'est pas définie lorsque $f'(\boldsymbol{x}^{(k)})$ n'est pas inversible.

Lemme 7.11. Soit $f\colon \mathbb{R}^n \to \mathbb{R}^m$ de classe C^1 sur $D \subset \mathbb{R}^n$. Soit $\boldsymbol{x}^* \in D$ et supposons que $f'(\boldsymbol{x}^*)$ est inversible. Alors, il existe un voisinage de $\boldsymbol{x}^*$ dans lequel $f'(\boldsymbol{x})$ est inversible et la fonction $\boldsymbol{x} \mapsto \|[f'(\boldsymbol{x})]^{-1}\|$ est bornée.

Démonstration. La fonction $\boldsymbol{x} \mapsto \det f'(\boldsymbol{x})$ est continue dans un voisinage de $\boldsymbol{x}^*$ car f' est continue et $\det X$ est un polynôme des composantes de la matrice X. Puisque $f'(\boldsymbol{x}^*)$ est inversible, on a $\det f'(\boldsymbol{x}^*) \neq 0$ et donc $\det f'(\boldsymbol{x}) \neq 0$ dans un voisinage de $\boldsymbol{x}^*$.

De plus, la formule de Cramer montre que sur un tel voisinage, les composantes de $[f'(\boldsymbol{x})]^{-1}$ sont des fonctions continues des composantes de $f'(\boldsymbol{x})$. La fonction $\boldsymbol{x} \mapsto \|[f'(\boldsymbol{x})]^{-1}\|_\infty$ est donc continue et bornée. Le résultat pour une norme quelconque est une conséquence du fait que toutes les normes sur $\mathbb{R}^n$ sont équivalentes. $\square$

La méthode de Newton est donc bien définie si $f(\boldsymbol{x}^*)$ est inversible et si $\boldsymbol{x}^{(k)}$ est proche de $\boldsymbol{x}^*$. Etablissons maintenant sa convergence.

Théorème 7.12. Supposons $f\colon \mathbb{R}^n \to \mathbb{R}^n$ de classe C^1 sur $D \subset \mathbb{R}^n$ convexe et ouvert, et

$$\|f'(\boldsymbol{y}) - f'(\boldsymbol{x})\| \le L\|\boldsymbol{y} - \boldsymbol{x}\|, \quad \forall \boldsymbol{x}, \boldsymbol{y} \in D.$$

Soit $f(\boldsymbol{x}^*) = \boldsymbol{0}$ où $\boldsymbol{x}^* \in D$ et supposons $f'(\boldsymbol{x}^*)$ est inversible. Alors, pour $\boldsymbol{x}^{(0)}$ suffisamment proche de $\boldsymbol{x}^*$, la méthode de Newton converge vers $\boldsymbol{x}^*$ et elle satisfait

$$\|\boldsymbol{x}^{(k+1)} - \boldsymbol{x}^*\| \le C\|\boldsymbol{x}^{(k)} - \boldsymbol{x}^*\|^2, \qquad k = 0, 1, 2, \ldots,$$

pour une constante C qui ne dépend pas de k. Sa convergence est donc au moins *quadratique*.

Démonstration. On peut supposer que $\boldsymbol{x}^{(k)} \in D$. Comme $f(\boldsymbol{x}^*) = \boldsymbol{0}$, on a d'après le Lemme 7.10 que

$$f(\boldsymbol{x}^{(k)}) = f(\boldsymbol{x}^{(k)}) - f(\boldsymbol{x}^*) = f'(\boldsymbol{x}^{(k)})(\boldsymbol{x}^{(k)} - \boldsymbol{x}^*) + w(\boldsymbol{x}^{(k)}, \boldsymbol{x}^*), \qquad \|w(\boldsymbol{x}^{(k)}, \boldsymbol{x}^*)\| \le \tfrac{L}{2}\|\boldsymbol{x}^{(k)} - \boldsymbol{x}^*\|^2.$$

Par définition (7.11) de la méthode de Newton, on obtient

$$\boldsymbol{x}^{(k+1)} - \boldsymbol{x}^* = \boldsymbol{x}^{(k)} - \boldsymbol{x}^* - [f'(\boldsymbol{x}^{(k)})]^{-1} f(\boldsymbol{x}^{(k)}).$$

Puis, en substituant la formule pour $f(\boldsymbol{x}^{(k)})$ ci-dessus, l'erreur satisfait

$$\boldsymbol{x}^{(k+1)} - \boldsymbol{x}^* = -[f'(\boldsymbol{x}^{(k)})]^{-1} w(\boldsymbol{x}^{(k)}, \boldsymbol{x}^*)$$

D'après le Lemme 7.11, il existe $\widetilde{\beta}$ et un voisinage $\widetilde{D} \subset D$ de $\boldsymbol{x}^*$ tels que $\|[f'(\boldsymbol{x}^{(k)})]^{-1}\| \le \widetilde{\beta}$ pour tout $\boldsymbol{x}^{(k)} \in \widetilde{D}$. Donc,

$$\|\boldsymbol{x}^{(k+1)} - \boldsymbol{x}^*\| \le \|[f'(\boldsymbol{x}^{(k)})]^{-1}\| \|w(\boldsymbol{x}^{(k)}, \boldsymbol{x}^*)\| \le \tfrac{1}{2}\widetilde{\beta} L \|\boldsymbol{x}^{(k)} - \boldsymbol{x}^*\|^2.$$

Il nous reste à montrer la convergence. En choisissant $\boldsymbol{x}^{(0)}$ tel que $\|\boldsymbol{x}^{(0)} - \boldsymbol{x}^*\| \le 1/(\widetilde{\beta} L)$ (c.-à-d., $\boldsymbol{x}^{(0)}$ est suffisamment proche de $\boldsymbol{x}^*$), on obtient

$$\|\boldsymbol{x}^{(1)} - \boldsymbol{x}^*\| \le \tfrac{1}{2}\|\boldsymbol{x}^{(0)} - \boldsymbol{x}^*\|$$

et la suite $\boldsymbol{x}^{(0)}, \boldsymbol{x}^{(1)}, \ldots$ converge vers $\boldsymbol{x}^*$ par induction. □

On peut montrer que si $f''(\boldsymbol{x}^*)$ ne s'annule pas, la convergence de la méthode de Newton dans le théorème ci-dessus est exactement quadratique. En outre, si f est C^1 mais f' n'est pas Lipschitzienne, la convergence est seulement superlinéaire (Dans ce cas, on a $\|f(\boldsymbol{x}) + f'(\boldsymbol{x})(\boldsymbol{y} - \boldsymbol{x}) - f(\boldsymbol{y})\| = o(\|\boldsymbol{y} - \boldsymbol{x}\|)$ au lieu du résultat du Lemme 7.10).

Le théorème 7.12 montre la convergence *locale* de la méthode de Newton, c.-à-d., si $\boldsymbol{x}^{(0)}$ est assez proche d'une solution $\boldsymbol{x}^*$ de (7.1), la suite $(\boldsymbol{x}^{(k)})$ converge vers $\boldsymbol{x}^*$. Concernant la convergence *globale*, on en sait très peu et on ne sait analyser que quelques cas de fonctions simples. L'exemple le plus connu est de résoudre

$$f(z) = z^3 - 1 = 0, \qquad z \in \mathbb{C}. \tag{7.12}$$

En utilisant $\boldsymbol{x} = (x_1, x_2) \in \mathbb{R}^2$ et $z = x_1 + \imath x_2$, on a aussi

$$f(\boldsymbol{x}) = \begin{bmatrix} x_1^3 - 3x_1x_2^2 - 1 \\ 3x_1^2x_2 - x_2^3 \end{bmatrix}, \qquad f'(\boldsymbol{x}) = \begin{bmatrix} 3x_1^2 - 3x_2^2 & -6x_1x_2 \\ 6x_1x_2 & 3x_1^2 - 3x_2^2 \end{bmatrix}.$$

On peut vérifier numériquement que si $\boldsymbol{x}^{(0)} = (1,1)^T$ la méthode de Newton converge vers $\boldsymbol{x}^* = (1,0)$, c.-à-d, $z^* = 1$.

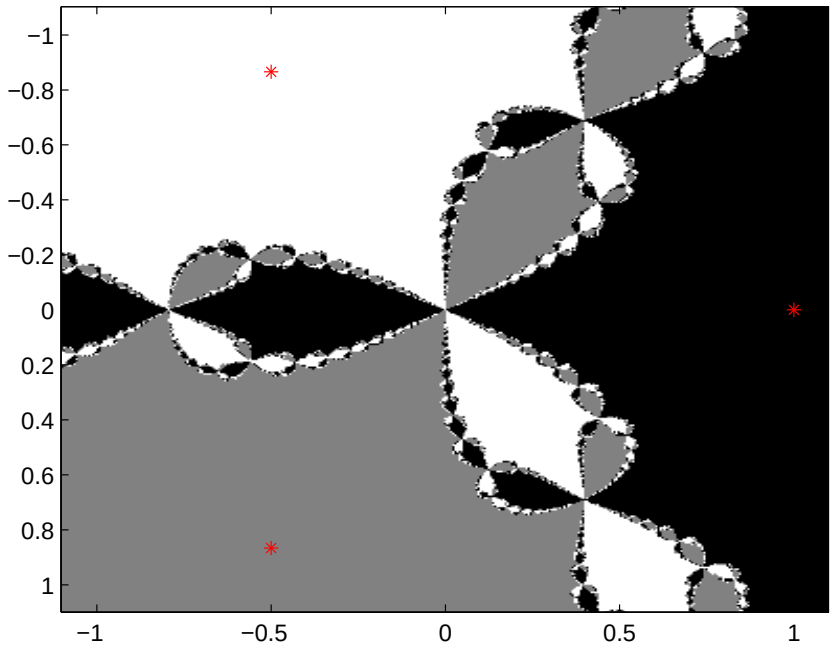


FIGURE 7.4 – Bassins d'attraction de la méthode de Newton pour (7.12).

En réalité, le système (7.12) possède trois solutions distinctes $1, (-1 \pm \imath\sqrt{3})/2$. En changeant l'approximation initiale $\boldsymbol{x}^{(0)}$, on obtient des suites qui convergent vers les différentes solutions. Un calcul par ordinateur donne la Fig. 7.4 qui montre les *bassins d'attraction* de la méthode de Newton :

$$A(\boldsymbol{x}^*) = \{\boldsymbol{x}^{(\mathbf{0})} \in \mathbb{R}^2 : \ (\boldsymbol{x}^{(k)}) \text{ converge vers } \boldsymbol{x}^*\}.$$

Les $\boldsymbol{x}^{(\mathbf{0})}$ du domaine noir entraînent une convergence vers $\boldsymbol{x}^* = (1,0)$, ceux du domaine gris vers $\boldsymbol{x}^* = (-1-\imath\sqrt{3})/2$ et ceux du domaine blanc vers $\boldsymbol{x}^* = (-1+\imath\sqrt{3})/2$. On observe que la suite $\boldsymbol{x}^{(k)}$ ne converge pas nécessairement vers la solution la plus proche de $\boldsymbol{x}^*$; les domaines sont en fait fractals.

7.7 Modifications de la méthode de Newton

En général, l'erreur de la méthode de Newton diminue bien plus vite que celle des méthodes linéaires. En revanche, chaque étape peut être très coûteuse car il faut résoudre un système d'équations linéaires de taille $n \times n$. En utilisant l'élimination de Gauss, chaque étape coûtera $O(n^3)$ opérations. Pour réduire le coût de l'évaluation/factorisation de $f'(\boldsymbol{x}^{(k)})$, on peut réutiliser la matrice jacobienne et sa décomposition LU pendant m itérations avant de la mettre à jour :

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} - [f'(\boldsymbol{x}_0)]^{-1} f(\boldsymbol{x}^{(k)}), \qquad k = 0, \ldots, m. \tag{7.13}$$

Cette méthode est appelée la *méthode de Newton modifiée* et, sous certaines hypothèses génériques, elle a une convergence linéaire (voir les TP). En effet, pour $m = \infty$, l'itération (7.13) n'est rien d'autre qu'une méthode des points fixes avec

$$\Phi(\boldsymbol{x}) = \boldsymbol{x} - [f'(\boldsymbol{x}_0)]^{-1} f(\boldsymbol{x})$$

et $\Phi'(\boldsymbol{x}^*)$ ne s'annule pas en général.

Si l'on dispose d'une formule explicite pour $f(\boldsymbol{x})$, on peut calculer $f'(\boldsymbol{x})$ analytiquement. Mais en pratique, il arrive souvent que la forme analytique de la matrice $f'(\boldsymbol{x})$ soit inconnue. Dans cette situation, on approche les dérivées partielles à l'aide de la formule

$$\frac{\partial f_i}{\partial x_j}(\boldsymbol{x}^{(k)}) = \lim_{h\to 0} \frac{f_i(\boldsymbol{x}^{(k)} + h\boldsymbol{e}_j) - f_i(\boldsymbol{x}^{(k)})}{h},$$

où $f(\boldsymbol{x}) = (f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_n(\boldsymbol{x}))$ et $\boldsymbol{e}_j = (0,\ldots,0,1,0,\ldots,0)$ tel que $(\boldsymbol{e}_j)_j = 1$. Puisqu'il est impossible de prendre une limite dans un programme comme MATLAB, il faut plutôt prendre un $h > 0$ petit :

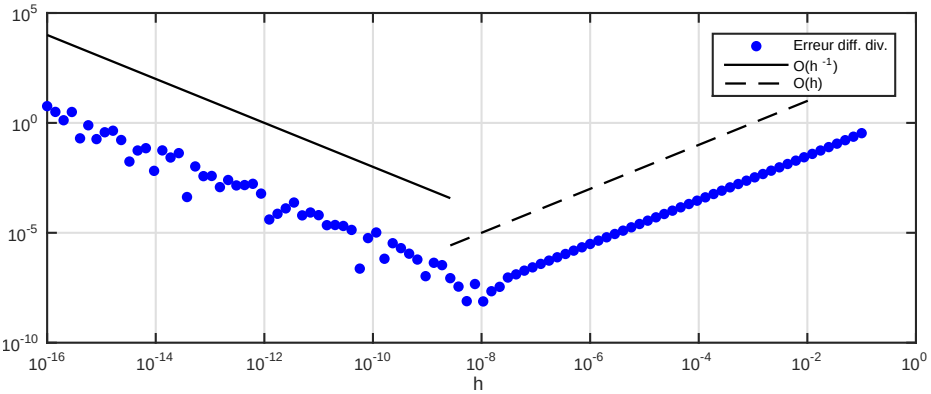
$$\frac{\partial f_i}{\partial x_j}(\boldsymbol{x}^{(k)}) \approx \frac{f_i(\boldsymbol{x}^{(k)} + h\boldsymbol{e}_j) - f_i(\boldsymbol{x}^{(k)})}{h} = \delta f_i[\boldsymbol{x}^{(k)}, \boldsymbol{x}^{(k)} + h\boldsymbol{e}_j]$$

où δf_i est une différence divisée (voir la Déf. 1.2). Une telle approximation est aussi appelée une *différence finie* et elle coûte n évaluations de f en plus pour chaque $\boldsymbol{x}^{(k)}$. Mais comment doit-on choisir le h ?

Pour simplifier la notation, considérons une fonction scalaire à une variable et notons-la $g(x)$. Un développement en série de Taylor montre que

$$D = \frac{g(x+h) - g(x)}{h} = g'(x) + \underbrace{\frac{h}{2} \cdot g''(x)}_{\substack{\text{erreur de} \\ \text{discretisation}}} + O(h^2). \tag{7.14}$$

Par conséquent, h doit être petit pour que l'erreur de la discrétisation ne soit pas trop grande. D'autre part, la soustraction dans (7.14) est très mal conditionnée car $g(x+h) \to g(x)$ pour $h \to 0$. Cela est clairement visible dans la figure ci-dessous où D est calculé en MATLAB pour la fonction $g(x) = e^x/(\cos^3(x) + \sin^3(x))$ en $x = \pi/4$.



Essayons d'expliquer le comportement de cette erreur en étudiant les erreurs d'arrondi de la formule (7.14). D'après la Sec. 2.3, on obtient en virgule flottante [2] :

$$\begin{aligned}\hat{D} &= \frac{g((x+h)(1+\varepsilon_1))(1+\varepsilon_2) - g(x)(1+\varepsilon_3)}{h}(1+\varepsilon_4), \qquad |\varepsilon_i| \le \varepsilon_{\text{mach}} \\ &\approx \frac{g(x+h) - g(x)}{h} + \underbrace{\frac{1}{h}\Big(g'(x+h)(x+h)\varepsilon_1 + g(x+h)(\varepsilon_2+\varepsilon_4) - g(x)(\varepsilon_3+\varepsilon_4)\Big)}_{\text{erreurs d'arrondi}}, \qquad \varepsilon_i \text{ petit},\end{aligned}$$

2. On a supposé que l'évaluation de g est bien conditionnée car l'erreur relative est bornée par ε_2 et ε_3.

où on a utilisé $g((x+h)(1+\varepsilon_1)) = g(x+h+(x+h)\varepsilon_1) \approx g(x+h) + g'(x+h)(x+h)\varepsilon_1$. En substituant (7.14), il faut donc choisir $h > 0$ pour minimiser

$$|\hat{D} - g'(x)| \approx \left| \frac{1}{h}(g'(x+h)(x+h)\varepsilon_1 + g(x+h)(\varepsilon_2+\varepsilon_4) - g(x)(\varepsilon_3+\varepsilon_4)) + \frac{h}{2}g''(x) + O(h^2) \right|.$$

Au lieu de minimiser cette formule exactement (ce qui nécessite de connaître g' et g''), on voudrait plutôt déterminer la grandeur de h par rapport à $\varepsilon_{\text{mach}}$. Le "minimum" est ainsi obtenu par

$$h^{-1}(\cdots)\,\varepsilon_{\text{mach}} \approx h(\cdots) \implies h = O(\sqrt{\varepsilon_{\text{mach}}}).$$

On choisit souvent $h = \sqrt{\varepsilon_{\text{mach}}}$ ce qui donne $h \approx 10^{-8}$ en MATLAB.

7.8 Minimisation et la méthode de Gauss–Newton

Nous sommes intéressés maintenant par des systèmes non linéaires surdéterminés. On considère une fonction $r\colon \mathbb{R}^n \to \mathbb{R}^m$ où $m > n$ et on cherche une solution de $r(\boldsymbol{x}) = \mathbf{0}$. Evidemment, comme on a plus de conditions que d'inconnues, ce problème ne possède en général pas de solution. On se contente donc de trouver un vecteur $\boldsymbol{x}^* \in \mathbb{R}^n$ qui s'accorde le mieux aux équations, par exemple, le vecteur qui satisfait

$$\|r(\boldsymbol{x})\|_2 = \sqrt{\sum_{i=1}^{m} r_i^2(\boldsymbol{x})} \to \min, \tag{7.15}$$

où $r(\boldsymbol{x}) = (r_1(\boldsymbol{x}), r_2(\boldsymbol{x}), \ldots, r_m(\boldsymbol{x}))$.

Si r est de classe C^2, la *fonction scalaire de plusieurs variables*

$$f\colon \mathbb{R}^n \to \mathbb{R},\ \boldsymbol{x} \mapsto \tfrac{1}{2}\|r(\boldsymbol{x})\|_2^2 = \tfrac{1}{2}\sum_{i=1}^{m} r_i^2(\boldsymbol{x}) \tag{7.16}$$

est aussi de classe C^2 car $\boldsymbol{x} \mapsto \|\boldsymbol{x}\|_2^2$ est de classe C^∞. Rappelons (voir (3.3)) que

$$f'(\boldsymbol{x})\colon \mathbb{R}^n \to \mathbb{R},\ \boldsymbol{p} \mapsto f'(\boldsymbol{x})\boldsymbol{p} = \left[\frac{\partial f}{\partial x_1}(\boldsymbol{x}) \quad \cdots \quad \frac{\partial f}{\partial x_n}(\boldsymbol{x}) \right] \boldsymbol{p}.$$

En utilisant le **gradient** de f,

$$\nabla f\colon \mathbb{R}^n \to \mathbb{R}^n,\ \boldsymbol{x} \mapsto \begin{bmatrix} \frac{\partial f}{\partial x_1}(\boldsymbol{x}) \\ \vdots \\ \frac{\partial f}{\partial x_n}(\boldsymbol{x}) \end{bmatrix},$$

on a $f'(\boldsymbol{x})\boldsymbol{p} = [\nabla f(\boldsymbol{x})]^T \boldsymbol{p}$. Une condition *nécessaire*[3] pour que $\boldsymbol{x}$ soit un minimum local est donc d'avoir $f'(\boldsymbol{x})\boldsymbol{p} = 0$ pour tout $\boldsymbol{p} \in \mathbb{R}^n$, c.-à-d., $\nabla f(\boldsymbol{x}) = \mathbf{0}$. Par conséquent, une possibilité pour résoudre (7.15) est d'appliquer la méthode de Newton pour ∇f, c.-à-d., on fait l'itération

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} - [\nabla^2 f(\boldsymbol{x}^{(k)})]^{-1} \nabla f(\boldsymbol{x}^{(k)}) \tag{7.17}$$

où $\nabla^2 f\colon \mathbb{R}^n \to \mathbb{R}^n$ est la **matrice hessienne** de f qui représente la dérivée de ∇f (voir ci-dessous pour une formule). Si $\boldsymbol{x}^{(k)}$ est proche d'une solution de (7.15), cette itération converge sous certaines hypothèses sur f.

3. Clairement, elle n'est pas suffisante : prenons $f(x) = x^3$.

Corollaire 7.13. Supposons $f\colon \mathbb{R}^n \to \mathbb{R}$ de classe C^2 sur $D \subset \mathbb{R}^n$ convexe et ouvert, et

$$\|\nabla^2 f(\boldsymbol{y}) - \nabla^2 f(\boldsymbol{x})\| \le L\|\boldsymbol{y} - \boldsymbol{x}\|, \quad \forall \boldsymbol{x}, \boldsymbol{y} \in D.$$

Soit $\nabla f(\boldsymbol{x}^*) = \boldsymbol{0}$ où $\boldsymbol{x}^* \in D$ et $\nabla^2 f(\boldsymbol{x}^*)$ inversible. Alors, pour $\boldsymbol{x}^{(k)}$ suffisamment proche de $\boldsymbol{x}^*$, la méthode (7.17) satisfait

$$\|\boldsymbol{x}^{(k+1)} - \boldsymbol{x}^*\| \le C\|\boldsymbol{x}^{(k)} - \boldsymbol{x}^*\|^2$$

pour une constante C.

Démonstration. D'après Thm. 7.12 pour $\nabla f(\boldsymbol{x}) = \boldsymbol{0}$ où $\nabla f\colon \mathbb{R}^n \to \mathbb{R}^n$. □

L'itération (7.17) nécessite ∇f et $\nabla^2 f$. Calculons d'abord $\nabla f(\boldsymbol{x}) \in \mathbb{R}^n$. En prenant la dérivée partielle de (7.16), on obtient

$$[\nabla f(\boldsymbol{x})]_i = \frac{\partial f}{\partial x_i}(\boldsymbol{x}) = \frac{1}{2}\frac{\partial}{\partial x_i}\sum_{k=1}^{m} r_k^2(\boldsymbol{x}) = \sum_{k=1}^{m} r_k(\boldsymbol{x})\frac{\partial r_k}{\partial x_i}(\boldsymbol{x}).$$

Le gradient s'écrit plus facilement en notation matricielle :

$$\nabla f(\boldsymbol{x}) = \begin{bmatrix} \frac{\partial r_1}{\partial x_1}(\boldsymbol{x}) & \frac{\partial r_2}{\partial x_1}(\boldsymbol{x}) & \cdots & \frac{\partial r_m}{\partial x_1}(\boldsymbol{x}) \\ \frac{\partial r_1}{\partial x_2}(\boldsymbol{x}) & \frac{\partial r_2}{\partial x_2}(\boldsymbol{x}) & \cdots & \frac{\partial r_m}{\partial x_2}(\boldsymbol{x}) \\ \vdots & \vdots & & \vdots \\ \frac{\partial r_1}{\partial x_n}(\boldsymbol{x}) & \frac{\partial r_2}{\partial x_n}(\boldsymbol{x}) & \cdots & \frac{\partial r_m}{\partial x_n}(\boldsymbol{x}) \end{bmatrix} \begin{bmatrix} r_1(\boldsymbol{x}) \\ r_2(\boldsymbol{x}) \\ \vdots \\ r_m(\boldsymbol{x}) \end{bmatrix} = (r'(\boldsymbol{x}))^T r(\boldsymbol{x})$$

où $r'_{ij} = \partial r_i / \partial x_j$ est la matrice jacobienne de r (remarquons la transposée ci-dessus par rapport à la définition (3.3)).

Puis, calculons $\nabla^2 f(\boldsymbol{x}) \in \mathbb{R}^{n\times n}$ comme la matrice jacobienne de $\nabla f(\boldsymbol{x})$:

$$[\nabla^2 f]_{ij} = \frac{\partial}{\partial x_j}[\nabla f]_i = \frac{\partial}{\partial x_j}\frac{\partial f}{\partial x_i} = \frac{\partial}{\partial x_j}\Big(\sum_{k=1}^{m} r_k \frac{\partial r_k}{\partial x_i}\Big) = \sum_{k=1}^{m}\Big(\frac{\partial r_k}{\partial x_j}\frac{\partial r_k}{\partial x_i} + r_k \frac{\partial^2 r_k}{\partial x_j \partial x_i}\Big)$$

où on a supprimé l'évaluation en $\boldsymbol{x}$ dans la notation. Donc

$$\nabla^2 f(\boldsymbol{x}) = (r'(\boldsymbol{x}))^T r'(\boldsymbol{x}) + \sum_{k=1}^{m} r_k(\boldsymbol{x})\nabla^2 r_k(\boldsymbol{x}) = B(\boldsymbol{x}) + H(\boldsymbol{x}), \tag{7.18}$$

où $\nabla^2 r_k(\boldsymbol{x})$ est la matrice hessienne de r_k en $\boldsymbol{x}$:

$$\nabla^2 r_k(\boldsymbol{x}) = \begin{bmatrix} \frac{\partial^2 r_k}{\partial x_1^2}(\boldsymbol{x}) & \frac{\partial^2 r_k}{\partial x_1 \partial x_2}(\boldsymbol{x}) & \cdots & \frac{\partial^2 r_k}{\partial x_1 \partial x_n}(\boldsymbol{x}) \\ \frac{\partial^2 r_k}{\partial x_2 \partial x_1}(\boldsymbol{x}) & \frac{\partial^2 r_k}{\partial x_2^2}(\boldsymbol{x}) & \cdots & \frac{\partial^2 r_k}{\partial x_2 \partial x_n}(\boldsymbol{x}) \\ \vdots & \vdots & & \vdots \\ \frac{\partial^2 r_k}{\partial x_n \partial x_1}(\boldsymbol{x}) & \frac{\partial^2 r_k}{\partial x_n \partial x_2}(\boldsymbol{x}) & \cdots & \frac{\partial^2 r_k}{\partial x_n^2}(\boldsymbol{x}) \end{bmatrix}.$$

Rappelons du cours d'analyse que la matrice hessienne d'une fonction est symétrique si celle-ci est de classe C^2.

Le calcul de $\nabla^2 f$ peut être coûteux parce qu'on a besoin de toutes les dérivées secondes de r_k. Afin de l'éviter, une autre possibilité est d'approcher $r(\boldsymbol{x})$ dans (7.15) autour d'une approximation $\boldsymbol{x}^{(k)}$ par une fonction linéaire et de prendre $\boldsymbol{x}^{(k+1)}$ comme la solution de

$$\|r(\boldsymbol{x}^{(k)}) + r'(\boldsymbol{x}^{(k)})\,(\boldsymbol{x}^{(k+1)} - \boldsymbol{x}^{(k)})\|_2 \to \min. \tag{7.19}$$

Puisque $r'(\boldsymbol{x}^{(k)})$ est une matrice de taille $m \times n$ où $m > n$, la solution de ce problème est en fait les moindres carrés du *système d'équations linéaires surdéterminé*

$$r'(\boldsymbol{x}^{(k)})\,(\boldsymbol{x}^{(k+1)} - \boldsymbol{x}^{(k)}) \approx -r(\boldsymbol{x}^{(k)}).$$

Leur résolution est, par exemple, faite à l'aide des équations normales

$$B(\boldsymbol{x}^{(k)})\,(\boldsymbol{x}^{(k+1)} - \boldsymbol{x}^{(k)}) = -\nabla f(\boldsymbol{x}^{(k)}). \tag{7.20}$$

Puisque la matrice $B(\boldsymbol{x}) = (r'(\boldsymbol{x}))^T r'(\boldsymbol{x})$ est symétrique et définie positive, on peut utiliser la décomposition de Cholesky. Cependant, une autre méthode plus stable et souvent plus efficace est par une factorisation QR de la matrice $r'(\boldsymbol{x}^{(k)})$ (voir Chap. 6). On a ainsi obtenu la **méthode de Gauss-Newton** :

$$\begin{aligned} &\|r'(\boldsymbol{x}^{(k)})\boldsymbol{p}_k + r(\boldsymbol{x}^{(k)})\|_2 \to \min \qquad \text{(résoudre par la méthode des moindres carrés)} \\ &\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \boldsymbol{p}_k \end{aligned} \tag{7.21}$$

On peut donc construire et résoudre les étapes dans la méthode de Gauss–Newton d'une manière efficace. Qu'est-ce qu'on peut dire de son taux de convergence ? La formule (7.18) montre que (7.20) est comme la méthode de Newton (7.17) après avoir négligé la matrice H. Donc, sa convergence n'est plus quadratique en général mais il y a des cas où la convergence est quand même assez rapide.

Lemme 7.14. Supposons $r\colon \mathbb{R}^n \to \mathbb{R}^m$ de classe C^3. Soit $\nabla f(\boldsymbol{x}^*) = (r'(\boldsymbol{x}^*))^T r(\boldsymbol{x}^*) = \boldsymbol{0}$ où $\boldsymbol{x}^* \in D$ et $\operatorname{rang} r'(\boldsymbol{x}^*) = n$. Alors, pour $\boldsymbol{x}^{(k)}$ suffisamment proche de $\boldsymbol{x}^*$, l'erreur de la méthode de Gauss–Newton satisfait

$$\|\boldsymbol{x}^{(k+1)} - \boldsymbol{x}^*\| \le \left(\|[B(\boldsymbol{x}^{(k)})]^{-1} H(\boldsymbol{x}^{(k)})\| + C\|\boldsymbol{x}^{(k)} - \boldsymbol{x}^*\|\right) \|(\boldsymbol{x}^{(k)} - \boldsymbol{x}^*)\|$$

pour une constante C et

$$B(\boldsymbol{x}) = (r'(\boldsymbol{x}))^T r'(\boldsymbol{x}), \qquad H(\boldsymbol{x}) = \sum_{k=1}^{m} r_k(\boldsymbol{x})\,\nabla^2 r_k(\boldsymbol{x}).$$

Démonstration. On procède comme dans la preuve du Thm. 7.12 mais pour ∇f et l'itération (7.20).

Remarquons d'abord que comme r est C^3, $\nabla f = (r')^T r$ est C^2 et $\nabla^2 f$ est L-lipschitzienne dans un voisinage de $\boldsymbol{x}^*$. Puis, comme $\operatorname{rang} r'(\boldsymbol{x}^*) = n \le m$, la matrice $B(\boldsymbol{x}^*)$ de taille $n \times n$ est inversible. D'apres le Lemme 7.11, $B(\boldsymbol{x})$ reste inversible dans un voisinage de $\boldsymbol{x}^*$.

Comme $\nabla f(\boldsymbol{x}^*) = \boldsymbol{0}$, on a d'après le Lemme 7.10 que

$$\nabla f(\boldsymbol{x}^{(k)}) = \nabla f(\boldsymbol{x}^{(k)}) - \nabla f(\boldsymbol{x}^*) = [B(\boldsymbol{x}^{(k)}) + H(\boldsymbol{x}^{(k)})]\,(\boldsymbol{x}^{(k)} - \boldsymbol{x}^*) + O(\|\boldsymbol{x}^{(k)} - \boldsymbol{x}^*\|^2).$$

Par définition de la méthode de Gauss–Newton, on obtient

$$\begin{aligned}\boldsymbol{x}^{(k+1)} - \boldsymbol{x}^* &= \boldsymbol{x}^{(k)} - \boldsymbol{x}^* - [B(\boldsymbol{x}^{(k)})]^{-1}\nabla f(\boldsymbol{x}^{(k)}) \\ &= [B(\boldsymbol{x}^{(k)})]^{-1}[B(\boldsymbol{x}^{(k)})(\boldsymbol{x}^{(k)} - \boldsymbol{x}^*) - \nabla f(\boldsymbol{x}^{(k)})].\end{aligned}$$

Puis, en substituant la formule pour $\nabla f(\boldsymbol{x}^{(k)})$ ci-dessus, l'erreur satisfait

$$\boldsymbol{x}^{(k+1)} - \boldsymbol{x}^* = [B(\boldsymbol{x}^{(k)})]^{-1}[H(\boldsymbol{x}^{(k)})(\boldsymbol{x}^{(k)} - \boldsymbol{x}^*) + O(\|\boldsymbol{x}^{(k)} - \boldsymbol{x}^*\|^2)]$$

et donc il existe une constante C telle que

$$\|\boldsymbol{x}^{(k+1)} - \boldsymbol{x}^*\| \leq \|[B(\boldsymbol{x}^{(k)})]^{-1}H(\boldsymbol{x}^{(k)})\|\|(\boldsymbol{x}^{(k)} - \boldsymbol{x}^*)\| + C\|\boldsymbol{x}^{(k)} - \boldsymbol{x}^*\|^2. \qquad \square$$

Une conséquence de ce lemme est que la convergence est *superlinéaire*

— si $r(\boldsymbol{x}^*) = \mathbf{0}$, c.-à-d., si le système surdéterminé possède une solution exacte ;

— si $r''(\boldsymbol{x}^*) = \mathbf{0}$, c.-à-d., si les fonctions $r(\boldsymbol{x})$ n'ont pas de courbure en $\boldsymbol{x}^*$;

car dans ce cas on a pour $\boldsymbol{x}^{(k)} \to \boldsymbol{x}^*$

$$\|\boldsymbol{x}^{(k+1)} - \boldsymbol{x}^*\| \leq \Big(\underbrace{\|[B(\boldsymbol{x}^{(k)})]^{-1}H(\boldsymbol{x}^{(k)})\|}_{\to 0} + C\|\boldsymbol{x}^{(k)} - \boldsymbol{x}^*\|\Big)\|(\boldsymbol{x}^{(k)} - \boldsymbol{x}^*)\|.$$

En revanche, on a

— convergence *linéaire* si $r(\boldsymbol{x}^*)$ est assez petit, c.-à-d., il existe une approximation proche des données ;

— *divergence* si $r(\boldsymbol{x}^*)$ est trop grand,

car

$$\|\boldsymbol{x}^{(k+1)} - \boldsymbol{x}^*\| \leq \Big(\underbrace{\|[B(\boldsymbol{x}^{(k)})]^{-1}H(\boldsymbol{x}^{(k)})\|}_{<1 \text{ ou } \geq 1?} + \underbrace{C\|\boldsymbol{x}^{(k)} - \boldsymbol{x}^*\|}_{\to 0}\Big)\|(\boldsymbol{x}^{(k)} - \boldsymbol{x}^*)\|.$$

Chapitre 8

Valeurs et vecteurs propres

Rappelons la définition des valeurs et vecteur propres :

Définition 8.1. Soit $A \in \mathbb{R}^{n\times n}$ (ou $\mathbb{C}^{n\times n}$). Le scalaire $\lambda \in \mathbb{C}$ est une **valeur propre** de A s'il existe un vecteur $\boldsymbol{v} \in \mathbb{C}^n$ tel que

$$A\boldsymbol{v} = \lambda \boldsymbol{v}, \qquad \boldsymbol{v} \neq \boldsymbol{0}. \tag{8.1}$$

Le vecteur $\boldsymbol{v}$ est appelé un **vecteur propre** (à droite) de valeur propre λ.

L'équation (8.1) équivaut à $\ker(A-\lambda I) \neq \{\boldsymbol{0}\}$, ou encore

$$p_A(\lambda) = \det(A-\lambda I) = (-1)^n\lambda^n + c_{n-1}\lambda^{n-1} + \cdots + c_0 = 0,$$

où $p_A(\lambda)$ est le **polynôme caractéristique** de A. Les valeurs propres de A sont alors les racines du polynôme caractéristique et, en comptant leur multiplicité, il y en a n.

Rappelons des TP qu'il est déconseillé en général de calculer les racines du polynôme caractéristique à cause des erreurs d'arrondi pendant le calcul. On a donc besoin des méthodes spécialisées pour calculer λ et $\boldsymbol{v}$. Mais d'abord, analysons la condition de leur calcul.

8.1 Condition du calcul des valeurs et vecteurs propres

Sur l'ordinateur, il est en général impossible de calculer les valeurs propres de la matrice exacte A à cause des erreurs d'arrondi ; on travaille plutôt avec une matrice perturbée $\tilde{A}$ avec $\tilde{a}_{ij} = a_{ij}(1+\varepsilon_{ij})$, $|\varepsilon_{ij}| \leq \varepsilon_{\text{mach}}$. Il est donc important d'étudier l'influence de ces perturbations sur les valeurs propres et sur les vecteurs propres de la matrice. Pour montrer ceci, considérons la famille de matrices

$$A(\varepsilon) = A + \varepsilon E, \qquad |\varepsilon| \leq \varepsilon_{\text{mach}}, \quad \|E\| \leq \|A\|.$$

Théorème 8.2 (Gerschgorin 1931). Soit $A \in \mathbb{R}^{n\times n}$ (ou $\mathbb{C}^{n\times n}$).

(a) Si λ est une valeur propre de A, alors il existe un indice i tel que

$$|\lambda - a_{ii}| \le R_i, \qquad R_i = \sum_{\substack{j=1 \\ j\neq i}}^{n} |a_{ij}|,$$

c.-à-d., toutes les valeurs propres de A se trouvent dans l'union des disques de Gerschgorin

$$D_i = \{z \in \mathbb{C} : |z - a_{ii}| \le R_i\}.$$

(b) Si une composante connexe de $\cup_{i=1}^{n} D_i$ consiste de k disques, elle contient exactement k valeurs propres de A.

Démonstration. Soit $\boldsymbol{v} \neq \mathbf{0}$ un vecteur propre et choisissons i tel que $\|\boldsymbol{v}\|_\infty = |v_i| > 0$. La ligne i de l'équation $A\boldsymbol{v} = \lambda\boldsymbol{v}$ donne

$$\sum_j a_{ij} v_j = \lambda v_i \quad \Longrightarrow \quad \sum_{j\neq i} a_{ij} v_j = (\lambda - a_{ii}) v_i.$$

En divisant par v_i, on a démontré (a) :

$$|\lambda - a_{ii}| = \Big|\sum_{j\neq i} a_{ij} \frac{v_j}{v_i}\Big| \le \sum_{j\neq i} \big|a_{ij}\big|\Big|\frac{v_j}{v_i}\Big| \le R_i.$$

L'affirmation (b) est obtenue par un argument de continuité. Considérons la fonction continue

$$A(t) = D + tB, \qquad D = \operatorname{diag}(a_{11}, a_{22}, \ldots, a_{nn}),$$

pour laquelle $A(0) = D$, $A(1) = A$. Puisque les racines d'un polynôme sont des fonctions continues de ses coefficients (voir le cours "Analyse II"), les valeurs propres de $A(t)$ varient continuement en t. Remarquons que les disques de Gerschgorin $D_i(t)$ de la matrice $A(t)$ ont des rayons croissants en $t \ge 0$:

$$R_i(t) = \sum_{\substack{j=1 \\ j\neq i}}^{n} |tb_{ij}| = t \sum_{\substack{j=1 \\ j\neq i}}^{n} |a_{ij}| = tR_i(1).$$

Donc, pour $0 \le t \le 1$, les $D_i(t)$ grandissent monotonement et continûment jusqu'aux disques associés de $A(1)$. Supposons maintenant qu'une composante connexe de $\cup_{i=1}^{n} D_i(1)$ consiste de k disques, disons $D_1(1), \ldots, D_k(1)$. En prenant $t \to 0$, les disques $D_1(t), \ldots, D_k(t)$ deviennent les points $D_1(0) = a_{11}, \ldots, D_k(0) = a_{kk}$ qui contiennent k valeurs propres de D. Par continuité et d'après (a), les valeurs propres de $A(t)$ ne peuvent pas échapper aux $D_i(t)$, donc, les $D_1(1), \ldots, D_k(1)$ contiennent exactement k valeurs propres de $A(1) = A$. □

Utilisons maintenant le théorème de Gerschgorin pour démontrer la condition des valeurs propres d'une matrice A qui est **diagonalisable**, c.-à-d., pour laquelle il existe une matrice T telle que

$$T^{-1} A T = \operatorname{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n).$$

Donc, les colonnes de T se composent de n vecteurs propres qui sont linéairement indépendants. Rappelons d'après le théorème spectral qu'une matrice symétrique réelle est diagonalisable par une matrice T orthogonale car on peut toujours trouver n vecteurs propres orthogonaux.

Théorème 8.3 (Bauer–Fike 1960). Soit $A \in \mathbb{R}^{n\times n}$ (ou $\mathbb{C}^{n\times n}$) diagonalisable et soit $\tilde{A} = A + E$. Alors, pour chaque valeur propre $\tilde{\lambda}$ de $\tilde{A}$, il existe une valeur propre λ_i de A tel que

$$|\tilde{\lambda} - \lambda_i| \le \kappa_\infty(T)\, \|E\|_\infty$$

où $\kappa_\infty(T) = \|T\|_\infty \|T^{-1}\|_\infty$ est la condition de T pour la norme $\|\cdot\|_\infty$ (voir (3.6)).

Démonstration. Transformons la matrice $\tilde{A}$ par la matrice T qui diagonalise A :

$$T^{-1}\tilde{A}T = T^{-1}(A+E)T = \operatorname{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n) + T^{-1}ET.$$

Observons que $T^{-1}\tilde{A}T$ a les mêmes valeurs propres que $\tilde{A}$ car

$$\tilde{A}\tilde{\boldsymbol{v}} = \tilde{\lambda}\tilde{\boldsymbol{v}} \iff T^{-1}\tilde{A}\tilde{\boldsymbol{v}} = \tilde{\lambda}T^{-1}\tilde{\boldsymbol{v}} \iff T^{-1}\tilde{A}T\tilde{\boldsymbol{w}} = \tilde{\lambda}\tilde{\boldsymbol{w}}, \quad \tilde{\boldsymbol{w}} = T^{-1}\tilde{\boldsymbol{v}}.$$

Posons $F = T^{-1}ET$. Le Thm. 8.2 de Gerschgorin pour $T^{-1}\tilde{A}T$ implique l'existence d'un indice i tel que

$$|\tilde{\lambda} - (\lambda_i + f_{ii})| \le \sum_{j\neq i} |f_{ij}|.$$

L'inégalité triangulaire et la définition (3.6) donnent

$$|\tilde{\lambda} - \lambda_i| = |f_{ii} + (\tilde{\lambda} - \lambda_i - f_{ii})| \le |f_{ii}| + \sum_{j\neq i} |f_{ij}| = \sum_j |f_{ij}| \le \|T^{-1}ET\|_\infty \le \kappa_\infty(T)\, \|E\|_\infty. \qquad \square$$

Similairement à la définition 3.1 de la condition relative, on peut définir la condition absolue du calcul de la valeur propre λ_i de A : la plus petite constante κ_i telle que

$$\|\tilde{A} - A\| \le \delta \implies |\tilde{\lambda}_i - \lambda_i| \le \kappa_i\, \delta + o(\delta), \qquad \delta \to 0. \tag{8.2}$$

D'après Thm. 8.3 avec $\|E\|_\infty = \|\tilde{A} - A\|_\infty \le \delta$, on a

$$|\tilde{\lambda}_i - \lambda_i| \le \kappa_\infty(T)\, \delta \qquad \implies \qquad \kappa_i \le \kappa_\infty(T).$$

La condition est donc bien pour des matrices symétriques car T est orthogonale et $\kappa_\infty(T) \le n$.

Toutefois, observons qu'on obtient seulement une estimation pour l'erreur absolue et non pour l'erreur relative. En outre, le théorème de Bauer–Fike donne la même borne $\kappa_\infty(T)$ comme condition de toutes les valeurs propres. Une analyse plus précise pour une seule valeur propre est possible si elle est simple.

Théorème 8.4 (Différentiabilité des valeurs propres). Soit $A \in \mathbb{R}^{n\times n}$ (où $\mathbb{C}^{n\times n}$). Supposons que λ_* est une racine simple de $p_A(\lambda)$ avec le vecteur propre à droite $\boldsymbol{v}_*$ et le **vecteur propre à gauche** $\boldsymbol{u}_*$, c.-à-d, $\boldsymbol{u}_*^H A = \bar{\lambda}\boldsymbol{u}_*^H$. Alors, pour ε suffisamment petit, la matrice $A(\varepsilon) = A + \varepsilon E$ possède une valeur propre simple $\lambda(\varepsilon)$ près de λ_*. De plus, la fonction $\lambda(\varepsilon)$ est différentiable dans un voisinage de 0 et on a

$$\lambda(\varepsilon) = \lambda_* + \varepsilon \frac{\boldsymbol{u}_*^H E \boldsymbol{v}_*}{\boldsymbol{u}_*^H \boldsymbol{v}_*} + O(\varepsilon^2).$$

Démonstration. Montrons le théorème pour $\mathbb{C}^{n\times n}$ car le cas pour $\mathbb{R}^{n\times n}$ est pareil. Définissons la fonction

$$f\colon \mathbb{C}\times\mathbb{R} \to \mathbb{C},\ (\lambda, \varepsilon) \to p_{A(\varepsilon)}(\lambda) = \det(A + \varepsilon E - \lambda I)$$

et observons qu'elle est un polynôme en λ (et donc analytique). Comme λ_* est une racine simple de p_A, on a

$$f(\lambda_*, 0) = p_A(\lambda_*) = 0, \qquad \frac{\partial f}{\partial \lambda}(\lambda_*, 0) = p'_A(\lambda_*) \neq 0.$$

Le théorème des fonctions implicites (voir le cours "Analyse II") garanti l'existence de la fonction $\lambda : \mathbb{R} \to \mathbb{C}$ du théorème : elle est différentiable dans un voisinage de 0 et $\lambda(\varepsilon)$ est une valeur propre de $A(\varepsilon)$ car

$$\lambda(0) = \lambda_*, \qquad f(\lambda(\varepsilon), \varepsilon) = p_{A(\varepsilon)}(\lambda(\varepsilon)) = 0.$$

Puis, démontrer que le vecteur propre $\boldsymbol{v}(\varepsilon)$ de $\lambda(\varepsilon)$ est différentiable. Les relations ci-dessus impliquent qu'il existe $\boldsymbol{v}(\varepsilon) \neq \mathbf{0}$ tel que

$$\boldsymbol{v}(0) = \boldsymbol{v}_*, \qquad (A(\varepsilon) - \lambda(\varepsilon) I)\boldsymbol{v}(\varepsilon) = \mathbf{0}. \tag{8.3}$$

Puisque $\lambda(\varepsilon)$ est une racine simple de $p_{A(\varepsilon)}$ dans un voisinage de 0, sa multiplicité géométrique vaut 1, c.-à-d., $\dim\ker(A(\varepsilon) - \lambda(\varepsilon)) = 1$. On peut donc fixer une des composantes de $\boldsymbol{v}(\varepsilon)$ à 1 et utiliser la règle de Cramer pour calculer le reste. On voit alors que les autres composantes de $\boldsymbol{v}(\varepsilon)$ sont des fonctions rationnelles des coefficients de la matrice $A(\varepsilon) - \lambda(\varepsilon) I$ et donc différentiables.

Il nous reste à montrer que $\lambda'(0) = \boldsymbol{u}_*^H E \boldsymbol{v}_* / \boldsymbol{u}_*^H \boldsymbol{v}_*$. Pour cela, on dérive (8.3) par rapport à ε et on pose $\varepsilon = 0$:

$$(A - \lambda_* I)\boldsymbol{v}'(0) + (E - \lambda'(0) I)\boldsymbol{v}_* = \mathbf{0}.$$

En multipliant à gauche par $\boldsymbol{u}_*^H$, le premier terme s'annule et on obtient

$$\boldsymbol{u}_*^H E \boldsymbol{v}_* - \lambda'(0) \boldsymbol{u}_*^H \boldsymbol{v}_* = 0. \qquad \square$$

Analysons de nouveau la condition du calcul de λ_* en utilisant le Thm. 8.4. D'après (8.2) avec $\|\tilde{A} - A\|_2 = \varepsilon \|E\|_2 \leq \delta$, on obtient

$$|\tilde{\lambda}_* - \lambda_*| = \varepsilon \frac{|\boldsymbol{u}_*^H E \boldsymbol{v}_*|}{|\boldsymbol{u}_*^H \boldsymbol{v}_*|} + O(\varepsilon^2) \leq \varepsilon \frac{\|E\|_2}{|\boldsymbol{u}_*^H \boldsymbol{v}_*|} + O(\varepsilon^2) \leq \frac{\delta}{|\boldsymbol{u}_*^H \boldsymbol{v}_*|} + O(\delta^2),$$

où on a utilisé $|\boldsymbol{u}_*^H E \boldsymbol{v}_*| \leq \|\boldsymbol{u}_*\|_2 \|E\|_2 \|\boldsymbol{v}_*\|_2 = \|E\|_2$. Ainsi, la **condition d'une valeur propre simple** λ_* de A est définie par

$$\kappa_* = \frac{1}{|\boldsymbol{u}_*^H \boldsymbol{v}_*|},$$

où $\boldsymbol{u}_*$ et $\boldsymbol{v}_*$ sont des vecteurs propres unitaires à gauche et à droite de A de la valeur propre λ_*.

Exemple 8.5

1. *Si A est normale (c.-à-d. $A^H A = A A^H$), d'après le théorème spectral, il existe une matrice U t.q. $U^H U = I$ (unitaire) et $U^H A U = \Lambda$. On a donc $\boldsymbol{u}_i = \boldsymbol{v}_i$ pour tout i, c.-à-d. $\kappa_i = 1$. Les valeurs propres sont donc bien conditionnées.*

2. *Si A n'est pas normale, le calcul de λ_i peut être mal conditionné :*

$$A = \begin{bmatrix} 1 & \alpha \\ 0 & 2 \end{bmatrix}, \quad \lambda_1 = 1, \quad \boldsymbol{v}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad \boldsymbol{u}_1 = \frac{1}{\sqrt{1+\alpha^2}} \begin{bmatrix} 1 \\ -\alpha \end{bmatrix}.$$

Donc $\kappa_1 = \sqrt{1+\alpha^2} \implies$ mal conditionné si α est grand.

3. *Un exemple à l'extreme :*

$$A = \begin{bmatrix} \lambda_1 & 1 \\ 0 & \lambda_1 \end{bmatrix}, \qquad E = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}.$$

Le polynôme caractéristique devient

$$p_{A+\varepsilon E}(\lambda) = \det(A + \varepsilon E - \lambda I) = (\lambda_1 - \lambda)^2 - \varepsilon.$$

Les valeurs propres sont donc

$$\lambda_{1,2}(\varepsilon) = \lambda_1 \pm \sqrt{\varepsilon},$$

c.-à-d., la perturbation est $O(\sqrt{\varepsilon}) \gg O(\varepsilon)$. Il n'y a donc pas de différentiabilité pour des valeurs propres multiples, et le calcul de ces dernières est très mal conditionné!

8.2 Méthode de la puissance

Un algorithme simple pour calculer les valeurs propres d'une matrice A est basé sur l'itération

$$\boldsymbol{y}^{(k+1)} = A\boldsymbol{y}^{(k)}, \qquad \boldsymbol{y}^{(0)} \in \mathbb{C}^n \text{ donné.}$$

Cette itération est appelée la **méthode de la puissance** car on a $\boldsymbol{y}^{(k)} = A^k \boldsymbol{y}^{(0)}$.

Dans le théorème ci-dessous, on démontre que $\boldsymbol{y}^{(k)}$ tend vers un vecteur propre de A. Rappelons que les vecteurs propres ne sont pas uniques : soit $\boldsymbol{v}$ un vecteur propre, alors $\alpha\boldsymbol{v}$ est aussi un vecteur propre pour tout $\alpha \neq 0$. Il est donc difficile d'étudier la convergence en analysant $\|\boldsymbol{y}^{(k)} - \boldsymbol{v}\|$. En revanche, on va utiliser leur *angle* :

$$\sin\theta(\boldsymbol{y}^{(k)}, \boldsymbol{v}) = \frac{\|\Pi_{\boldsymbol{v}}^{\perp} \boldsymbol{y}^{(k)}\|_2}{\|\boldsymbol{y}^{(k)}\|_2}, \tag{8.4}$$

où $\Pi_{\boldsymbol{v}}^{\perp} \boldsymbol{y}^{(k)}$ est la projection orthogonale de $\boldsymbol{y}^{(k)}$ sur le sous-espace normal à $\boldsymbol{v}$. On dit que $\boldsymbol{y}^{(k)}$ tend vers le vecteur propre $\boldsymbol{v}$ si $\sin\theta(\boldsymbol{y}^{(k)}, \boldsymbol{v}) \to 0$ pour $k \to \infty$.

D'abord analysons l'itération pour $A = \Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$ *diagonale* où

$$|\lambda_1| \geq |\lambda_2| \geq \cdots |\lambda_n|.$$

Supposons que $\lambda_1 \neq 0$ car sinon $\Lambda = 0$ et $\boldsymbol{y}^{(k)} = \boldsymbol{0}$. Utilisons les partitions

$$\boldsymbol{y}^{(k)} = \begin{bmatrix} y_1^{(k)} \\ \boldsymbol{y}_2^{(k)} \end{bmatrix}, \qquad \Lambda = \begin{bmatrix} \lambda_1 & \\ & \Lambda_2 \end{bmatrix}$$

où $y_1^{(k)}$ et λ_1 sont des scalaires. Alors, la méthode de la puissance donne

$$\boldsymbol{y}^{(k)} = \begin{bmatrix} y_1^{(k)} \\ \boldsymbol{y}_2^{(k)} \end{bmatrix} = \Lambda^k \boldsymbol{y}^{(0)} = \begin{bmatrix} \lambda_1^k y_1^{(0)} \\ \Lambda_2^k \boldsymbol{y}_2^{(0)} \end{bmatrix} = \lambda_1^k \begin{bmatrix} y_1^{(0)} \\ (\lambda_1^{-1}\Lambda_2)^k \boldsymbol{y}_2^{(0)} \end{bmatrix} \tag{8.5}$$

Observons que $\boldsymbol{e}_1 = (1, 0, \ldots, 0)$ est un vecteur propre de λ_1. Donc,

$$\Pi_{\boldsymbol{e}_1}^{\perp} \boldsymbol{y}^{(k)} = \boldsymbol{y}^{(k)} - (\boldsymbol{e}_1^H \boldsymbol{y}^{(k)})\boldsymbol{e}_1 = \begin{bmatrix} 0 \\ \boldsymbol{y}_2^{(k)} \end{bmatrix}.$$

En utilisant (8.4), on obtient

$$\sin^2\theta(\boldsymbol{y}^{(k)},\boldsymbol{e}_1)=\frac{\|\boldsymbol{y}_2^{(k)}\|_2^2}{\|\boldsymbol{y}^{(k)}\|_2^2}=\frac{\|\boldsymbol{y}_2^{(k)}\|_2^2}{|y_1^{(k)}|^2+\|\boldsymbol{y}_2^{(k)}\|_2^2}\leq\frac{\|\boldsymbol{y}_2^{(k)}\|_2^2}{|y_1^{(k)}|^2},$$

où on a supposé que $y_1^{(k)}=\lambda_1^k y_1^{(0)}\neq 0$. Donc, (8.5) donne

$$\sin\theta(\boldsymbol{y}^{(k)},\boldsymbol{e}_1)\leq\|(\lambda_1^{-1}\Lambda_2)^k\|_2\frac{\|\boldsymbol{y}_2^{(0)}\|_2}{|y_1^{(0)}|}=\left|\frac{\lambda_2}{\lambda_1}\right|^k\frac{\|\boldsymbol{y}_2^{(0)}\|_2}{|y_1^{(0)}|},$$

car Λ_2 est une matrice diagonale. On voit que si $|\lambda_1|>|\lambda_2|$, l'itération converge vers $\boldsymbol{e}_1$.

On peut maintenant obtenir le résultat pour des matrices diagonalisables :

Lemme 8.6. Soit $B\in\mathbb{C}^{n\times n}$ inversible. Alors, pour tout $\boldsymbol{x},\boldsymbol{y}\in\mathbb{C}^n$:

$$\kappa_2^{-1}(B)\sin\theta(\boldsymbol{x},\boldsymbol{y})\leq\sin\theta(B\boldsymbol{x},B\boldsymbol{y})\leq\kappa_2(B)\sin\theta(\boldsymbol{x},\boldsymbol{y}).$$

Démonstration. Voir la série d'exercices. □

Théorème 8.7. Soit $A=T\Lambda T^{-1}$ une matrice diagonalisable où $\Lambda=\operatorname{diag}(\lambda_1,\ldots,\lambda_n)$ et les colonnes de T sont des vecteurs propres à droite $\boldsymbol{v}_1,\ldots,\boldsymbol{v}_n$. Si $|\lambda_1|>|\lambda_2|\geq|\lambda_3|\geq\cdots\geq|\lambda_n|$, la méthode de la puissance satisfait

$$\sin\theta(\boldsymbol{y}^{(k)},\boldsymbol{v}_1)\leq C\kappa_2(T)\left|\frac{\lambda_2}{\lambda_1}\right|^k,$$

où on a supposé que $\boldsymbol{u}_1^H\boldsymbol{y}^{(0)}\neq 0$ avec $\boldsymbol{u}_1$ un vecteur propre à gauche de λ_1. La constante C ne dépend pas de k.

Démonstration. Le vecteur propre à droite $\boldsymbol{v}_1$ peut être choisi comme $\boldsymbol{v}_1=T\boldsymbol{e}_1$ car

$$A\boldsymbol{v}_1=\lambda_1\boldsymbol{v}_1\quad\Longleftrightarrow\quad(T^{-1}AT)T^{-1}\boldsymbol{v}_1=\lambda_1T^{-1}\boldsymbol{v}_1\quad\Longleftrightarrow\quad\Lambda\boldsymbol{e}_1=\lambda_1\boldsymbol{e}_1.$$

Similairement, celui à gauche devient $\boldsymbol{u}_1=T^{-H}\boldsymbol{e}_1$:

$$\boldsymbol{u}_1^HA=\lambda_1\boldsymbol{u}_1^H\quad\Longleftrightarrow\quad\boldsymbol{u}_1^HT(T^{-1}AT)=\lambda_1\boldsymbol{u}_1^HT\quad\Longleftrightarrow\quad\boldsymbol{e}_1^T\Lambda=\lambda_1\boldsymbol{e}_1^T.$$

La méthode de la puissance donne

$$\boldsymbol{y}^{(k)}=A^k\boldsymbol{y}^{(0)}\quad\Longleftrightarrow\quad T^{-1}\boldsymbol{y}^{(k)}=(T^{-1}AT)^kT^{-1}\boldsymbol{y}^{(0)}.$$

Donc, en notant $\boldsymbol{z}^{(k)}=T^{-1}\boldsymbol{y}^{(k)}$, on a aussi $\boldsymbol{z}^{(k)}=\Lambda^k\boldsymbol{z}^{(0)}$. Puisque $\boldsymbol{v}_1=T\boldsymbol{e}_1$, le résultat ci-dessus pour des matrices diagonales donne

$$\sin\theta(T^{-1}\boldsymbol{y}^{(k)},T^{-1}\boldsymbol{v}_1)=\sin\theta(\boldsymbol{z}^{(k)},\boldsymbol{e}_1)\leq\left|\frac{\lambda_2}{\lambda_1}\right|^k\frac{\|\boldsymbol{z}_2^{(0)}\|_2}{|z_1^{(0)}|}.$$

La condition $z_1^{(0)}\neq 0$ équivaut $0\neq\boldsymbol{e}_1^T\boldsymbol{z}^{(0)}=\boldsymbol{u}_1^H(TT^{-1})\boldsymbol{y}^{(0)}$ car $T^H\boldsymbol{u}_1=\boldsymbol{e}_1$. Appliquer le lemme 8.6, on obtient la borne pour $\sin\theta(\boldsymbol{y}^{(k)},\boldsymbol{v}_1)$. □

Le théorème montre que l'angle entre $\boldsymbol{z}^{(k)} = T^{-1}\boldsymbol{y}^{(k)}$ et $\boldsymbol{e}_1 = T^{-1}\boldsymbol{v}_1$ tend vers zéro. Afin d'interpréter ce résultat du point de vue de $\boldsymbol{y}^{(k)}$, observons que $\boldsymbol{z}^{(k)}$ sont les composantes de $\boldsymbol{y}^{(k)}$ dans la base des vecteur propres :

$$\boldsymbol{y}^{(k)} = T\boldsymbol{z}^{(k)} = \sum_{i=1}^{k} z_i^{(k)} \boldsymbol{v}_i.$$

D'après (8.5), on sait que

$$\boldsymbol{z}^{(k)} = \lambda_1^k \begin{bmatrix} z_1^{(0)} \\ (\lambda_2/\lambda_1)^k z_2^{(0)} \\ \vdots \\ (\lambda_n/\lambda_1)^k z_n^{(0)} \end{bmatrix} = \lambda_1^k z_1^{(0)} \begin{bmatrix} 1 \\ O(|\lambda_2/\lambda_1|)^k \\ \vdots \\ O(|\lambda_2/\lambda_1|)^k \end{bmatrix} \to (\lambda_1^k z_1^{(0)})\, \boldsymbol{e}_1, \qquad k \to \infty,$$

car $z_1^{(0)} \neq 0$ et $0 \neq |\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq \cdots \geq |\lambda_n|$. On obtient alors,

$$\boldsymbol{y}^{(k)} = \sum_{i=1}^{k} z_i^{(k)} \boldsymbol{v}_i = \lambda_1^k z_1^{(0)} \left(\boldsymbol{v}_1 + \sum_{i=2}^{n} O\big(\Big|\frac{\lambda_2}{\lambda_1}\Big|^k\big) \boldsymbol{v}_i \right), \tag{8.6}$$

ce qui clairement montre que la distance entre $\boldsymbol{v}_1$ et un multiple de $\boldsymbol{y}^{(k)}$ tend vers zéro comme

$$\|\boldsymbol{v}_1 - \alpha_k \boldsymbol{y}^{(k)}\|_2 = O\big(\Big|\frac{\lambda_2}{\lambda_1}\Big|^k\big), \qquad \alpha_k = 1/(z_1^{(0)}\lambda_1^k).$$

Remarque 8.8

La condition $|\lambda_1| > |\lambda_2|$ est nécessaire (c.-à-d., il faut que λ_1 soit une valeur propre simple). Par exemple, pour la matrice

$$A = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \qquad \lambda_1 = -1, \quad \lambda_2 = 1,$$

la méthode de la puissance ne converge pas en général car les itérés satisfont

$$\boldsymbol{y}^{(0)} = \boldsymbol{y}^{(2)} = \boldsymbol{y}^{(4)} = \cdots = \begin{bmatrix} \alpha \\ \beta \end{bmatrix}, \quad \boldsymbol{y}^{(1)} = \boldsymbol{y}^{(3)} = \boldsymbol{y}^{(5)} = \cdots = \begin{bmatrix} \beta \\ \alpha \end{bmatrix}.$$

En revanche, malgré le fait que la condition $\boldsymbol{u}_1^H \boldsymbol{y}^{(0)} \neq 0$ soit aussi nécessaire, elle est toujours satisfaite en pratique si l'on prend $\boldsymbol{y}^{(0)}$ aléatoire.

On a donc obtenu la convergence de $\boldsymbol{y}^{(k)}$ vers le vecteur propre dominant $\boldsymbol{v}_1$. Comment peut-on maintenant trouver une approximation de la valeur propre dominante λ_1 ? Puisqu'une valeur propre et son vecteur propre satisfont l'équation (8.1), l'idée est de trouver l'approximation $\lambda^{(k)}$ en utilisant $\boldsymbol{y}^{(k)}$ telle que

$$A\boldsymbol{y}^{(k)} \approx \lambda^{(k)} \boldsymbol{y}^{(k)}.$$

En appliquant la méthode des moindres carrés pour la résolution de ce système surdéterminé,

$$\|A\boldsymbol{y}^{(k)} - \lambda^{(k)} \boldsymbol{y}^{(k)}\|_2 = \|\boldsymbol{y}^{(k)}\lambda^{(k)} - A\boldsymbol{y}^{(k)}\|_2 \to \min,$$

les équations normales (6.2) donnent [1]

$$\lambda^{(k)} = \frac{(\boldsymbol{y}^{(k)})^H A \boldsymbol{y}^{(k)}}{(\boldsymbol{y}^{(k)})^H \boldsymbol{y}^{(k)}}.$$

On appelle cette approximation le **quotient de Rayleigh**.

1. On a utilisé $\cdot^H$ au lieu de $\cdot^T$ car on se trouve, en général, dans $\mathbb{C}$.

Corollaire 8.9. Sous conditions du Thm. 8.7, le quotient de Rayleigh satisfait

$$|\lambda^{(k)} - \lambda_1| = O\Big(\Big|\frac{\lambda_2}{\lambda_1}\Big|^k\Big).$$

Soit A une matrice normale (e.g., $A = A^H$ si $A \in \mathbb{C}^{n\times n}$ ou $A = A^T$ si $A \in \mathbb{R}^{n\times n}$) ayant des vecteurs propres orthogonaux, alors

$$|\lambda^{(k)} - \lambda_1| = O\Big(\Big|\frac{\lambda_2}{\lambda_1}\Big|^{2k}\Big).$$

Démonstration. D'après (8.6) et en utilisant l'identité $\overline{z}\,z = |z|^2$ pour tout $z \in \mathbb{C}$, on a

$$\begin{aligned}
(\boldsymbol{y}^{(k)})^H A\boldsymbol{y}^{(k)} &= (\boldsymbol{y}^{(k)})^H \boldsymbol{y}^{(k+1)} \\
&= \frac{\overline{\lambda}_1^k}{\overline{z}_1^{(0)}}\left(\boldsymbol{v}_1 + \sum_{i=2}^n O\Big(\Big|\frac{\lambda_2}{\lambda_1}\Big|^k\Big)\boldsymbol{v}_i\right)^H \frac{\lambda_1^{k+1}}{z_1^{(0)}}\left(\boldsymbol{v}_1 + \sum_{j=2}^n O\Big(\Big|\frac{\lambda_2}{\lambda_1}\Big|^{k+1}\Big)\boldsymbol{v}_i\right) \\
&= \frac{|\lambda_1|^{2k}\lambda_1}{|z_1^{(0)}|^2}\left(\|\boldsymbol{v}_1\|_2^2 + \sum_{j=2}^n O\Big(\Big|\frac{\lambda_2}{\lambda_1}\Big|^{k+1}\Big)\boldsymbol{v}_1^H\boldsymbol{v}_i + \sum_{i=2}^n O\Big(\Big|\frac{\lambda_2}{\lambda_1}\Big|^k\Big)\boldsymbol{v}_i^H\boldsymbol{v}_1\right. \\
&\qquad\qquad \left. + \sum_{i=2}^n\sum_{j=2}^n O\Big(\Big|\frac{\lambda_2}{\lambda_1}\Big|^{2k+1}\Big)\boldsymbol{v}_i^H\boldsymbol{v}_j\right) \\
&= \frac{|\lambda_1|^{2k}\lambda_1}{|z_1^{(0)}|^2}\left(\|\boldsymbol{v}_1\|_2^2 + O\Big(\Big|\frac{\lambda_2}{\lambda_1}\Big|^k\Big)\right)
\end{aligned}$$

Similairement, on obtient

$$(\boldsymbol{y}^{(k)})^H\boldsymbol{y}^{(k)} = \frac{|\lambda_1|^{2k}}{|z_1^{(0)}|^2}\left(\|\boldsymbol{v}_1\|_2^2 + O\Big(\Big|\frac{\lambda_2}{\lambda_1}\Big|^k\Big)\right).$$

Puisque $(a + O(\varepsilon))^{-1} = a^{-1} + O(\varepsilon)$ pour $a > 0$ et $\varepsilon \to 0$, on a alors

$$\lambda^{(k)} = \frac{(\boldsymbol{y}^{(k)})^H A\boldsymbol{y}^{(k)}}{(\boldsymbol{y}^{(k)})^H\boldsymbol{y}^{(k)}} = \lambda_1\frac{\|\boldsymbol{v}_1\|_2^2 + O\big(\big|\frac{\lambda_2}{\lambda_1}\big|^k\big)}{\|\boldsymbol{v}_1\|_2^2 + O\big(\big|\frac{\lambda_2}{\lambda_1}\big|^k\big)} = \lambda_1\left(1 + O\Big(\Big|\frac{\lambda_2}{\lambda_1}\Big|^k\Big)\right).$$

Pour une matrice normale, on a $\boldsymbol{v}_i^H\boldsymbol{v}_j = 0$ si $i \neq j$. Les termes ci-dessus qui contiennent $O\big(\big|\frac{\lambda_2}{\lambda_1}\big|^k\big)$ et $O\big(\big|\frac{\lambda_2}{\lambda_1}\big|^{k+1}\big)$ s'annulent et il reste seulement ceux avec $O\big(\big|\frac{\lambda_2}{\lambda_1}\big|^{2k}\big)$ et $O\big(\big|\frac{\lambda_2}{\lambda_1}\big|^{2k+1}\big)$. □

Les composantes de $\boldsymbol{y}^{(k)}$ croissent ou décroissent exponentiellement (voir (8.6)). Pour éviter des overflows et underflows, il faut normaliser à chaque étape comme dans l'Alg. 8.10.

Algorithme 8.10 (Méthode de la puissance).

1: Soit donné $\boldsymbol{y}^{(0)} \in \mathbb{R}^n$ (ou $\mathbb{C}^n$) avec $\|\boldsymbol{y}^{(0)}\|_2 = 1$
2: **for** $k = 0, 1, \ldots$ **do**
3: $\quad \boldsymbol{w} = A\boldsymbol{y}^{(k)}$
4: $\quad \lambda^{(k)} = \boldsymbol{w}^H\boldsymbol{y}^{(k)}$
5: $\quad \boldsymbol{y}^{(k+1)} = \boldsymbol{w}/\|\boldsymbol{w}\|_2$

Si $|\lambda_1| \approx |\lambda_2|$, la convergence de la méthode sera très lente. En outre, on ne peut trouver que la valeur propre dominante, et non pas des autres comme la plus petite en valeur absolue par exemple.

8.3 Méthode de la puissance inverse

Supposons qu'on connaisse une approximation σ de la valeur propre cherchée λ_i (il n'est pas nécessaire de supposer que λ_i soit la plus grande valeur propre de A). L'idée est d'appliquer la méthode de la puissance à la matrice $(A-\sigma I)^{-1}$. En effet, les valeurs propres de cette matrice sont $(\lambda_i-\sigma)^{-1}$:

$$A\boldsymbol{v}=\lambda\boldsymbol{v} \quad\Longleftrightarrow\quad (A-\sigma I)\boldsymbol{v}=(\lambda-\sigma)\boldsymbol{v} \quad\Longleftrightarrow\quad (A-\sigma I)^{-1}\boldsymbol{v}=\frac{1}{\lambda-\sigma}\boldsymbol{v}.$$

Supposons que λ_i est la valeur propre de A la plus proche de σ, et λ_I la seconde,

$$|\sigma-\lambda_i|<|\sigma-\lambda_I|\le|\sigma-\lambda_j| \qquad \forall\, j\neq i,$$

alors

$$\frac{1}{|\sigma-\lambda_i|}>\frac{1}{|\sigma-\lambda_I|}\ge\frac{1}{|\sigma-\lambda_j|}, \qquad j\neq i.$$

Si $\sigma\approx\lambda_i$, on a

$$\frac{1}{|\sigma-\lambda_i|}\gg\frac{1}{|\sigma-\lambda_I|}$$

et la convergence sera très rapide car elle est déterminée par le facteur (voir Thm. 8.7)

$$\frac{|\sigma-\lambda_I|^{-1}}{|\sigma-\lambda_i|^{-1}}=\frac{|\sigma-\lambda_i|}{|\sigma-\lambda_I|}\ll 1.$$

On obtient ainsi l'Alg. 8.11. Remarquons que le quotient de Rayleigh est calculé pour la matrice originale A pour des raisons numériques. En outre, après avoir calculé la décomposition LU de la matrice $A-\sigma I$, résoudre les systèmes $(A-\sigma I)\boldsymbol{w}=\boldsymbol{y}^{(k)}$ n'est pas cher.

Algorithme 8.11 (Méthode de la puissance inverse).

1: Soit donné $\boldsymbol{y}^{(0)}\in\mathbb{R}^n$ (ou $\mathbb{C}^n$) avec $\|\boldsymbol{y}^{(0)}\|_2=1$ et $\sigma\approx\lambda_i$
2: **for** $k=0,1,\ldots$ **do**
3: Résoudre $(A-\sigma I)\boldsymbol{w}=\boldsymbol{y}^{(k)}$ (appliquer $(A-\sigma I)^{-1}$)
4: $\boldsymbol{y}^{(k+1)}=\boldsymbol{w}/\|\boldsymbol{w}\|_2$
5: $\lambda^{(k+1)}=(\boldsymbol{y}^{(k+1)})^H A\boldsymbol{y}^{(k+1)}$

8.4 Décomposition en valeurs singulières (SVD)

Dans la section 8.6, nous étudierons des méthodes pour calculer plusieurs valeurs propres. Leur convergence utilisera l'angle entre deux sous-espaces ce qui utilise la SVD.

Théorème 8.12 (SVD). Soit $A\in\mathbb{R}^{m\times n}$ ($\mathbb{C}^{m\times n}$). Il existe des matrices orthogonales (unitaires) U et V telles que

$$A=U\Sigma V^H, \qquad \Sigma=\left[\begin{array}{ccc|c} \sigma_1 & & & \\ & \ddots & & \mathbf{0} \\ & & \sigma_{\min(m,n)} & \\ \hline & \mathbf{0} & & 0 \end{array}\right]\in\mathbb{R}^{m\times n} \tag{8.7}$$

où $\sigma_1\ge\sigma_2\ge\cdots\ge\sigma_{\min(m,n)}$. Les σ_i sont appelées les **valeurs singulières**, les colonnes $\boldsymbol{v}_i$ de V les **vecteurs singuliers à droite**, les colonnes $\boldsymbol{u}_i$ de U les **vecteurs singuliers à gauche**, et (8.7) la décomposition en valeurs singulières ou (en anglais) "singular value decomposition" (**SVD**).

Démonstration. Démontrons le résultat dans $\mathbb{C}$ et pour $n \leq m$ car le cas pour $\mathbb{R}$ est pareil et si $m \geq n$ on peut démontrer le résultat pour A^H.

On procède par récurrence sur n. Pour $n = 1$, la matrice A est un vecteur $\boldsymbol{a} \in \mathbb{C}^m$. Si $\boldsymbol{a} = \mathbf{0}$, prenons $U = I$ et $V = 1$. Sinon, définissons $\boldsymbol{u}_1 = \boldsymbol{a}/\|\boldsymbol{a}\|_2$ et prenons les matrices unitaires

$$U = \begin{bmatrix} \boldsymbol{u}_1 & U_\perp \end{bmatrix} \in \mathbb{C}^{m\times m}, \qquad V = \begin{bmatrix} 1 \end{bmatrix} \in \mathbb{C}^{1\times 1},$$

c.-à-d., les colonnes de $U_\perp \in \mathbb{C}^{m\times(m-1)}$ sont mutuellement orthogonales et aussi orthogonales à $\boldsymbol{u}_1$. Ainsi on a

$$U^H A V = \left[\begin{array}{c} \boldsymbol{u}_1^H \boldsymbol{a} \\ \hline U_\perp^H \boldsymbol{a} \end{array}\right] = \left[\begin{array}{c} \sigma_1 \\ \hline \mathbf{0} \end{array}\right], \qquad \sigma_1 = \|\boldsymbol{a}\|_2.$$

Pour $n > 1$, prenons $\boldsymbol{v}_1 \in \mathbb{C}^n$ tel que $\|\boldsymbol{v}_1\|_2 = 1$ et $\|A\|_2 = \|A\boldsymbol{v}_1\|_2$ (voir (3.5) pour la norme 2 d'une matrice). En définissant $\sigma_1 = \|A\|_2$ et $\boldsymbol{u}_1 = A\boldsymbol{v}_1/\sigma_1$ (si $\sigma_1 = 0$, $\boldsymbol{u}_1$ est quelconque), on a

$$A\boldsymbol{v}_1 = \sigma_1 \boldsymbol{u}_1.$$

Posons les matrices unitaires

$$U = \begin{bmatrix} \boldsymbol{u}_1 & U_\perp \end{bmatrix} \in \mathbb{C}^{m\times m}, \qquad V = \begin{bmatrix} \boldsymbol{v}_1 & V_\perp \end{bmatrix} \in \mathbb{C}^{n\times n},$$

on obtient

$$U^H A V = \left[\begin{array}{c|c} \boldsymbol{u}_1^H A \boldsymbol{v}_1 & \boldsymbol{u}_1^H A V_\perp \\ \hline U_\perp^H A \boldsymbol{v}_1 & U_\perp^H A V_\perp \end{array}\right] = \left[\begin{array}{c|c} \sigma_1 & \boldsymbol{u}_1^H A V_\perp \\ \hline \mathbf{0} & U_\perp^H A V_\perp \end{array}\right].$$

Nous montrons $\boldsymbol{u}_1^H A V_\perp = \mathbf{0}$. Puisque $\|A\|_2 = \|A^H\|_2$ et $\|A\|_2 = \|U^H A V\|_2$ (car la norme 2 est invariante par transformation unitaire), on a

$$\begin{aligned} \sigma_1^2 &= \|A^H\|_2^2 = \|V^H A^H U\|_2^2 \\ &= \left\| \left[\begin{array}{c|c} \sigma_1 & \mathbf{0} \\ \hline V_\perp^H A^H \boldsymbol{u}_1 & V_\perp^H A^H U_\perp \end{array}\right] \right\|_2^2 \geq \left\| \left[\begin{array}{c|c} \sigma_1 & \mathbf{0} \\ \hline V_\perp^H A^H \boldsymbol{u}_1 & V_\perp^H A^H U_\perp \end{array}\right] \boldsymbol{e}_1 \right\|_2^2 \\ &= \left\| \left[\begin{array}{c} \sigma_1 \\ \hline V_\perp^H A^H \boldsymbol{u}_1 \end{array}\right] \right\|_2^2 = \sigma_1^2 + \|V_\perp^H A^H \boldsymbol{u}_1\|_2^2. \end{aligned}$$

On obtient ainsi $\|V_\perp^H A^H \boldsymbol{u}_1\|_2 = 0 \implies V_\perp^H A^H \boldsymbol{u}_1 = \mathbf{0}$ et donc

$$U^H A V = \left[\begin{array}{c|c} \sigma_1 & \mathbf{0} \\ \hline \mathbf{0} & U_\perp^H A V_\perp \end{array}\right].$$

Par récurrence, il existe des matrices unitaires $\widehat{U} \in \mathbb{C}^{(m-1)\times(m-1)}$ et $\widehat{V} \in \mathbb{C}^{(n-1)\times(n-1)}$ telles que

$$\widehat{U}^H (U_\perp^H A V_\perp) \widehat{V} = \begin{bmatrix} \widehat{\Sigma} \\ \mathbf{0} \end{bmatrix}$$

et $\widehat{\Sigma} \in \mathbb{R}^{(n-1)\times(n-1)}$ diagonale. En choisissant les matrices unitaires

$$\overline{U} = \begin{bmatrix} \boldsymbol{u}_1 & U_\perp \widehat{U}^H \end{bmatrix} \in \mathbb{C}^{m\times m}, \qquad \overline{V} = \begin{bmatrix} \boldsymbol{v}_1 & V_\perp \widehat{V} \end{bmatrix} \in \mathbb{C}^{n\times n},$$

on obtient ainsi le résultat du théorème

$$\overline{U}^H A \overline{V} = \left[\begin{array}{c|c} \sigma_1 & \mathbf{0} \\ \hline \mathbf{0} & \widehat{U}^H U_\perp^H A V_\perp \widehat{V} \end{array}\right] = \left[\begin{array}{c|c} \sigma_1 & \mathbf{0} \\ \hline \mathbf{0} & \widehat{\Sigma} \\ \hline \mathbf{0} & \mathbf{0} \end{array}\right]. \qquad \square$$

Il est possible de calculer la SVD à l'aide de deux décompositions spectrales : Soit $A = U\Sigma V^H$ et $m \geq n$, alors

$$A^H A = V\Sigma^H \Sigma V^H, \qquad AA^H = U\Sigma\Sigma^H U^H$$

où

$$\Sigma^H\Sigma = \operatorname{diag}(\sigma_1^2,\dots,\sigma_n^2) \qquad \Sigma\Sigma^H = \operatorname{diag}(\sigma_1^2,\dots,\sigma_n^2,\underbrace{0,\dots,0}_{m-n}).$$

Pour des raisons numériques (voir les séries d'exercices), il est préférable de calculer la SVD en utilisant une décomposition spectrale de la matrice hermitienne

$$\begin{bmatrix} 0 & A^H \\ A & 0 \end{bmatrix}.$$

En fait, il existe des algorithmes encore plus performants et plus stables numériquement.[2]

La SVD donne beaucoup de propriétés de l'algèbre linéaire d'une matrice :

Corollaire 8.13. Soit $A = U\Sigma V^H \in \mathbb{R}^{m\times n}$ une SVD. Alors

(a) $A\boldsymbol{v}_i = \sigma_i \boldsymbol{u}_i$ et $\boldsymbol{u}_i^H A = \sigma_i \boldsymbol{v}_i^H$ pour $1 \leq i \leq \min(m,n)$

(b) $\|A\|_2 = \sigma_1$

(c) $\operatorname{rang}(A)$ = le nombre de valeurs singulières non-nulles

(d) Soit $r = \operatorname{rang}(A)$, alors

$$\operatorname{im}(A) = \operatorname{Vect}\{\boldsymbol{u}_1,\dots,\boldsymbol{u}_r\}$$
$$\ker(A) = \operatorname{Vect}\{\boldsymbol{v}_{r+1},\dots,\boldsymbol{v}_n\}$$

(e) Soit $r = \operatorname{rang}(A)$, alors A est une combinaison linéaire de r matrices de rang 1 :

$$A = \sum_{i=1}^{r} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^H.$$

Démonstration. (a) En comparant les colonnes de $AV = U\Sigma$ et les lignes de $U^H A = \Sigma V^H$.

(b) $\sigma_1 = \|\Sigma\|_2 = \|U\Sigma V^H\|_2$ car la norme 2 est invariante par transformation unitaire.

(c) Le rang d'une matrice diagonale vaut le nombre de ses composantes non-nulles. Puisque U et V sont unitaires, on a $\operatorname{rang}(U\Sigma V^H) = \operatorname{rang}(\Sigma) = r$ où $\sigma_1 \geq \cdots \geq \sigma_{r-1} \geq \sigma_r > \sigma_{r+1} = 0$.

(d) Soit $\boldsymbol{x} \in \mathbb{C}^n$ quelconque. Puisque V est unitaire, on a $\boldsymbol{x} = \sum_{i=1}^n \alpha_i \boldsymbol{v}_i$ et donc d'après (a)

$$A\boldsymbol{x} = \sum_{i=1}^{n} \alpha_i A\boldsymbol{v}_i = \sum_{i=1}^{r} (\alpha_i\sigma_i)\boldsymbol{u}_i \in \operatorname{Vect}\{\boldsymbol{u}_1,\dots,\boldsymbol{u}_r\}.$$

On a ainsi $\operatorname{im}(A) = \{A\boldsymbol{x} : \boldsymbol{x} \in \mathbb{C}^n\} = \operatorname{Vect}\{\boldsymbol{u}_1,\dots,\boldsymbol{u}_r\}$ car $\dim\operatorname{rang}(A) = r$.

D'après (a), on a $A\boldsymbol{v}_i = \boldsymbol{0}$ si $i > r$. Le théorème du rang donne $\dim\ker(A) = n - r$ et donc $\ker(A) = \operatorname{Vect}\{\boldsymbol{v}_{r+1},\dots,\boldsymbol{v}_n\}$.

(e) Une exercice de multiplication par partitions :

$$A = (U\Sigma)V^H = \begin{bmatrix}\sigma_1\boldsymbol{u}_1 & \cdots & \sigma_r\boldsymbol{u}_r & \boldsymbol{0} & \cdots & \boldsymbol{0}\end{bmatrix}\begin{bmatrix}\boldsymbol{v}_1^H \\ \vdots \\ \boldsymbol{v}_n^H\end{bmatrix} = \sum_{i=1}^{r}\sigma_i\boldsymbol{u}_i\boldsymbol{v}_i^H. \qquad \square$$

2. Par exemple, l'algorithme Golub–Kahan ; voir Sect. 8.6 dans Golub & Van Loan (1996), *Matrix Computations* Johns Hopkins University Press.

Soit $A \in \mathbb{R}^{n\times n}$ inversible. Alors $A^{-1} = V\Sigma^{-1}U^T$ et on a donc

$$\|A^{-1}\|_2 = 1/\sigma_n(A), \qquad \kappa_2(A) = \frac{\sigma_1(A)}{\sigma_n(A)},$$

où $\kappa_2(A)$ est la condition de la matrice A (voir la Déf. 5.1). En généralisant les formules ci-dessus pour des matrices $A = U\Sigma V^T \in \mathbb{R}^{m\times n}$ non-carrées et singulières, on définit la condition d'une telle matrice comme

$$\kappa_2(A) = \frac{\sigma_1(A)}{\sigma_{\min(m,n)}(A)}$$

et la **pseudo-inverse de Moore–Penrose** comme

$$A^+ = V\Sigma^+U^T, \qquad \Sigma^+ = \left[\begin{array}{ccc|c} \sigma_1^{-1} & & & \\ & \ddots & & \mathbf{0} \\ & & \sigma_r^{-1} & \\ \hline & \mathbf{0} & & \mathbf{0} \end{array}\right], \qquad r = \operatorname{rang}(A).$$

Soient $A \in \mathbb{R}^{m\times n}$ où $m \geq n$ et $\boldsymbol{b} \in \mathbb{R}^m$. Rappelons que le système $A\boldsymbol{x} = \boldsymbol{b}$ surdéterminé n'a pas de solution unique en général. On peut trouver quand-même une solution utile à l'aide de la méthode des moindres carrés :

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} \|A\boldsymbol{x} - \boldsymbol{b}\|_2 = \|A\boldsymbol{x}^* - \boldsymbol{b}\|_2^2$$

où $\boldsymbol{x}^*$ est déterminé par les équations normales ou par une décomposition QR (voir le Chap. 6). En utilisant la formule convenable

$$\boldsymbol{x}^* = A^+\boldsymbol{b},$$

la pseudo-inverse donne une méthode additionnelle qui est aussi applicable aux systèmes carrés et sous-déterminés ($m < n$) :

(a) Si l'équation $A\boldsymbol{x} = \boldsymbol{b}$ n'a pas de solution, alors $\boldsymbol{x}^* = A^+\boldsymbol{b}$ est un vecteur qui minimise $\|A\boldsymbol{x} - \boldsymbol{b}\|_2$.

(b) Si $A \in \mathbb{R}^{n\times n}$ est inversible, alors $A^+ = A^{-1}$ et $\boldsymbol{x}^*$ est la solution unique.

(c) Si l'équation $A\boldsymbol{x} = \boldsymbol{b}$ a une infinité de solutions, alors $\boldsymbol{x}^* = A^+\boldsymbol{b}$ est la solution qui minimise $\|\boldsymbol{x}\|_2$.

Voir les séries d'exercices pour la preuve.

La SVD a aussi une interprétation géométrique par rapport l'application linéaire $\boldsymbol{x} \mapsto A\boldsymbol{x}$. Par exemple, pour une matrice de taille 2×2, la SVD décompose l'action de A en trois parties (voir aussi la Fig. 8.1) : une rotation V^T, suivie d'une dilatation Σ et finalement une autre rotation U. Observez que la condition de la matrice $\kappa_2(A)$ est une mesure de la déformation de l'ellipse dans la Fig. 8.1.

Rappelons que la norme 2 d'une matrice satisfait

$$\begin{aligned} \|A\|_2 &= \max_{\|\boldsymbol{x}\|_2=1} \|A\boldsymbol{x}\|_2 = \max_{\|\boldsymbol{x}\|_2=1} (\boldsymbol{x}^H A^H A\boldsymbol{x})^{1/2} \\ &= \max_{\|\boldsymbol{y}\|_2=1} \|A^H\boldsymbol{y}\|_2 = \max_{\|\boldsymbol{y}\|_2=1} (\boldsymbol{y}^H AA^H\boldsymbol{y})^{1/2} \end{aligned}$$

En utilisant la SVD, on peut montrer une autre propriété similaire.

Théorème 8.14. Soit $A \in \mathbb{R}^{m\times n}$ (ou $\in \mathbb{C}^{m\times n}$). Alors

$$\|A\|_2 = \max_{\|\boldsymbol{x}\|_2=1} \max_{\|\boldsymbol{y}\|_2=1} |\boldsymbol{y}^H A\boldsymbol{x}|$$

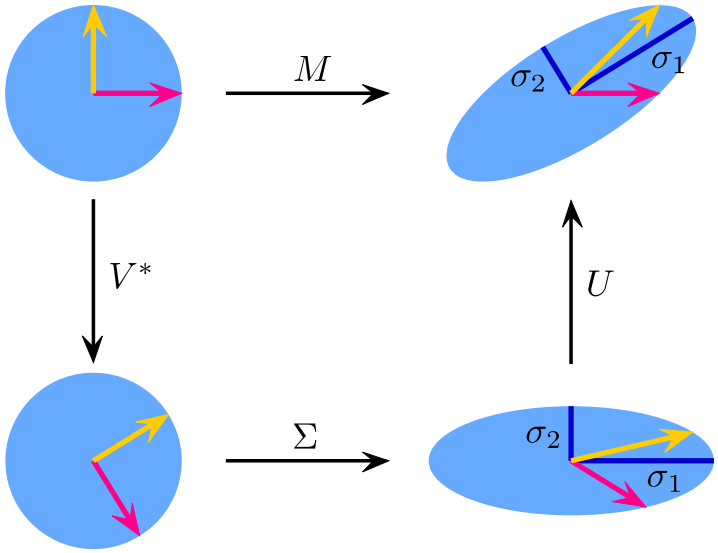


FIGURE 8.1 – La SVD de la matrice $M = U\Sigma V^T$. (Figure de `https://fr.wikipedia.org/wiki/Décomposition_en_valeurs_singulières`)

Démonstration. D'après Cauchy–Schwarz et la définition d'une norme d'opérateur, on a

$$|\boldsymbol{y}^T A\boldsymbol{x}| \leq \|\boldsymbol{y}\|_2 \|A\boldsymbol{x}\|_2 \leq \|\boldsymbol{y}\|_2 \|A\|_2 \|\boldsymbol{x}\|_2.$$

Donc

$$\max_{\|\boldsymbol{x}\|_2=1} \max_{\|\boldsymbol{y}\|_2=1} |\boldsymbol{y}^T A\boldsymbol{x}| \leq \sigma_1.$$

Mais les vecteurs singuliers $\boldsymbol{v}_1$ et $\boldsymbol{u}_1$ satisfont $\|\boldsymbol{u}_1\|_2 = \|\boldsymbol{v}_1\|_2 = 1$ et

$$|\boldsymbol{u}_1^H A\boldsymbol{v}_1| = |\boldsymbol{u}_1^H(\sigma_1\boldsymbol{u}_1)| = \sigma_1. \qquad \square$$

8.5 Angle entre des sous-espaces

Soit $\mathscr{X} = \operatorname{im} X$ le sous-espace vectoriel engendré par les colonnes de la matrice $X \in \mathbb{R}^{n\times p}$. Si $\operatorname{rang} X = p$, la projection orthogonale sur $\mathscr{X}$ satisfait

$$\Pi_{\mathscr{X}} = X(X^T X)^{-1} X^T.$$

En effet, on vérifie que $\Pi_{\mathscr{X}}$ est une projection (car $\Pi_{\mathscr{X}}^2 = \Pi_{\mathscr{X}}$) orthogonale (car $\Pi_{\mathscr{X}}^T = \Pi_{\mathscr{X}}$) sur $\mathscr{X}$ (car $\operatorname{im}\Pi_{\mathscr{X}} = \operatorname{im} X$ et le rang de X est maximale). En outre, la projection orthogonale sur

$$\mathscr{X}^\perp = \{\boldsymbol{y} \in \mathbb{R}^n : \boldsymbol{x}^T\boldsymbol{y} = 0, \forall \boldsymbol{x} \in \mathscr{X}\}$$

satisfait

$$\Pi_{\mathscr{X}}^\perp = I - \Pi_{\mathscr{X}} = I - X(X^T X)^{-1} X^T.$$

Définition 8.15. Soient $\mathscr{X}$ et $\mathscr{Y}$ deux sous-espaces de $\mathbb{R}^n$ de même dimension. L'angle $\theta(\mathscr{X},\mathscr{Y})$ est défini comme

$$\sin\theta(\mathscr{X},\mathscr{Y}) = \max\{\|\Pi_{\mathscr{X}}^{\perp}\boldsymbol{y}\|_2 : \boldsymbol{y}\in\mathscr{Y},\ \|\boldsymbol{y}\|_2 = 1\}.$$

Observez que pour $\mathscr{X} = \operatorname{Vect}\boldsymbol{x}$ et $\mathscr{Y} = \operatorname{Vect}\boldsymbol{y}$, on obtient l'angle entre les deux vecteurs $\boldsymbol{x}$ et $\boldsymbol{y}$, voir (8.4). On peut montrer que la formule ci-dessus est symétrique,

$$\sin\theta(\mathscr{X},\mathscr{Y}) = \max\left\{\|\Pi_{\mathscr{Y}}^{\perp}\boldsymbol{x}\|_2 : \boldsymbol{x}\in\mathscr{X},\ \|\boldsymbol{x}\|_2 = 1\right\}.$$

On peut calculer $\theta(\mathscr{X},\mathscr{Y})$ facilement à l'aide des projections.

Théorème 8.16. Soient $\mathscr{X}$ et $\mathscr{Y}$ deux sous-espaces de $\mathbb{R}^n$ de même dimension, alors

$$\sin^2\theta(\mathscr{X},\mathscr{Y}) = \|\Pi_{\mathscr{X}}^{\perp}\Pi_{\mathscr{Y}}\Pi_{\mathscr{X}}^{\perp}\|_2.$$

Démonstration. Il existe une matrice $Y\in\mathbb{R}^{n\times p}$ telle que $\mathscr{Y} = \operatorname{im} Y$. Soit $U_0 = \begin{bmatrix}\boldsymbol{u}_1 & \cdots & \boldsymbol{u}_r\end{bmatrix}$ la matrice qui est composée des r premiers vecteurs singuliers de Y. D'après Cor. 8.13(d) et puisque $U_0^T U_0 = I_r$, on a

$$\{\boldsymbol{y}\in\mathscr{Y} : \|\boldsymbol{y}\|_2 = 1\} \;=\; \{U_0\boldsymbol{a} : \boldsymbol{a}\in\mathbb{R}^r,\ \|\boldsymbol{a}\|_2 = 1\}.$$

En utilisant le Thm. 8.15, on obtient donc

$$\begin{aligned}\sin\theta(\mathscr{X},\mathscr{Y}) &= \max\{\|\Pi_{\mathscr{X}}^{\perp}\boldsymbol{y}\|_2 : \boldsymbol{y}\in\mathscr{Y},\ \|\boldsymbol{y}\|_2 = 1\}\\ &= \max\{\|\Pi_{\mathscr{X}}^{\perp}U_0\boldsymbol{a}\|_2 : \boldsymbol{a}\in\mathbb{R}^r,\ \|\boldsymbol{a}\|_2 = 1\}\\ &= \|\Pi_{\mathscr{X}}^{\perp}U_0\|_2,\end{aligned}$$

où on a utilisé la définition de la norme 2 d'une matrice.

Puis, observez que $\Pi_{\mathscr{Y}} = U_0U_0^T$. En utilisant l'identité $\|A\|_2 = \|AA^T\|_2^{1/2}$ pour A une matrice quelconque, on a aussi

$$\|\Pi_{\mathscr{X}}^{\perp}U_0\|_2^2 = \|\Pi_{\mathscr{X}}^{\perp}U_0U_0^T\Pi_{\mathscr{X}}^{\perp}\|_2 = \|\Pi_{\mathscr{X}}^{\perp}\Pi_{\mathscr{Y}}\Pi_{\mathscr{X}}^{\perp}\|_2$$

car $\Pi_{\mathscr{X}}^{\perp}$ est une matrice symétrique. □

Pour certains sous-espaces, on peut calculer leur angle explicitement :

Lemme 8.17. Etant donnés les deux sous-espaces de $\mathbb{R}^n$:

$$\mathscr{X} = \operatorname{im}\begin{bmatrix}I_p\\0\end{bmatrix},\quad \mathscr{Y} = \operatorname{im}\begin{bmatrix}I_p\\M\end{bmatrix}$$

où $M\in\mathbb{R}^{(n-p)\times p}$ avec $\|M\|_2 < 1$. Alors, $\sin\theta(\mathscr{X},\mathscr{Y}) = \|M\|_2/(1+\|M\|_2)$.

Démonstration. Notons $\theta = \theta(\mathscr{X},\mathscr{Y})$. Afin d'utiliser le Thm. 8.16, on calcule

$$\Pi_{\mathscr{X}}^{\perp} = \begin{bmatrix}0 & \\ & I_{n-p}\end{bmatrix},\qquad \Pi_{\mathscr{Y}} = \begin{bmatrix}I_p\\M\end{bmatrix}(I_p + M^TM)^{-1}\begin{bmatrix}I_p & M^T\end{bmatrix}$$

et ainsi

$$\sin^2\theta = \|\Pi_{\mathscr{Y}}^{\perp}\Pi_{\mathscr{Y}}\Pi_{\mathscr{X}}^{\perp}\|_2 = \|M(I_p + M^TM)^{-1}M^T\|_2.$$

Soit $U\Sigma V^T = M$ une SVD, alors

$$M(I_p + M^T M)^{-1} M^T = U\Sigma(I_p + \Sigma^T\Sigma)^{-1}\Sigma^T U^T.$$

En utilisant $0 \le \sigma_i(M) \le \|M\|_2 < 1$, on a

$$\sin^2\theta = \|\Sigma(I_p + \Sigma^T\Sigma)^{-1}\Sigma^T\|_2 = \max_{1\le i\le r} \frac{\sigma_i^2(M)}{1+\sigma_i^2(M)} = \frac{\|M\|_2^2}{1+\|M\|_2^2},$$

car $x/(1+x)$ est une fonction croissante sur $[0,1]$. (Ceci montre aussi que $I_p + M^T M$ est inversible). □

8.6 L'itération orthogonale

Généralisons maintenant la méthode de la puissance pour calculer plusieurs valeurs propres en même temps. Pour simplifier, considérons seulement le cas dans $\mathbb{R}$. En principe, il suffit de faire l'itération

$$X^{(k+1)} = AX^{(k)}, \qquad X^{(0)} \in \mathbb{R}^{n\times p} \text{ donné.}$$

On espère que les colonnes de $X^{(k)} = A^k X^{(0)}$ convergent vers p vecteurs propres de A.

En appliquant la théorie pour la méthode de la puissance à chaque colonne de $X^{(k)}$, on constate que (voir (8.6))

$$\lambda_1^{-k} X^{(k)} \to \begin{bmatrix} c_1\boldsymbol{v}_1 & c_2\boldsymbol{v}_1 & \cdots & c_p\boldsymbol{v}_1 \end{bmatrix}.$$

Donc, toutes les colonnes de $X^{(k)}$ convergent vers le même vecteur propre dominant de A. La méthode ne semble pas très utile!

Rappelons que les vecteurs propres recherchés par notre méthode ne sont pas uniques et on s'intéresse plutôt au sous-espace qu'ils engendrent. La solution est maintenant d'imposer que les colonnes de $X^{(k)}$ restent mutuellement orthogonales, c.-à-d., $(X^{(k)})^T X^{(k)} = I_p$. Par exemple, on peut simplement calculer la décomposition QR de $X^{(k)}R^{(k)} = AX^{(k-1)}$. Si $R^{(k)}$ est inversible, $\operatorname{im} X^{(k)} = \operatorname{im} AX^{(k-1)}$ et on n'a pas changé le sous-espace engendré par les colonnes de $X^{(k)}$. On obtient comme ci l'itération suivante.

Algorithme 8.18 (L'itération orthogonale).

1: Soit donné $X^{(0)} \in \mathbb{R}^{n\times p}$ avec $(X^{(0)})^T X^{(0)} = I_p$
2: **for** $k = 0, 1, \ldots$ **do**
3: $\quad Z^{(k+1)} = AX^{(k)}$
4: $\quad X^{(k+1)}R^{(k+1)} = Z^{(k+1)}$ (calculer la décomposition QR)
5: $\quad T^{(k+1)} = (X^{(k+1)})^T AX^{(k+1)}$

La décomposition QR dans la ligne 4 a aussi l'avantage d'éviter les overflows et underflow, comme la ligne 4 dans Alg. 8.11 pour la méthode de la puissance originale. Dans la ligne 5 on calcule l'estimation suivante

$$AX^{(k+1)} \approx X^{(k+1)}T^{(k+1)}$$

au sens des moindre carrés car les valeurs propres satisfont cette équation si $X^{(k+1)}$ se compose de vecteurs propres. L'approximation pour les valeurs propres est la diagonale de $T^{(k)}$. Elle satisfait $t_{ii}^{(k)} = (\boldsymbol{x}_i^{(k)})^T A\boldsymbol{x}_i^{(k)}$, c.-à-d., le quotient de Rayleigh pour la $i$ième colonne $\boldsymbol{x}_i^{(k)}$ de $X^{(k)}$.

Etudiions maintenant la convergence de l'itération orthogonale en calculant l'angle entre les sous-espaces vectoriels engendrés par les vecteurs propres recherchées et par les colonnes de $X^{(k)}$, c.-à-d.,

$$\mathscr{E}_1 = \operatorname{Vect}\{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_p\} \qquad \text{et} \qquad \mathscr{X}^{(k)} = \operatorname{im} X^{(k)} = \operatorname{im} Z^{(k)} = \operatorname{im} A^k X^{(0)}$$

où on a supposé que toutes les matrices $R^{(k)}$ dans les décompositions QR sont inversibles.

Comme dans la méthode de la puissance originale, considérons la convergence pour $A = \Lambda = \operatorname{diag}(\lambda_1, \ldots, \lambda_n)$ diagonale où

$$|\lambda_1| \geq |\lambda_2| \geq \cdots |\lambda_n|.$$

Supposons que $\lambda_p \neq 0$. En utilisant, les partitions

$$A = \begin{bmatrix} \Lambda_1 & \\ & \Lambda_2 \end{bmatrix}, \quad \Lambda_1 \in \mathbb{R}^{p\times p}, \qquad X^{(k)} = \begin{bmatrix} X_1^{(k)} \\ X_2^{(k)} \end{bmatrix}, \quad X_1^{(k)} \in \mathbb{R}^{p\times p},$$

l'itération orthogonale calcule

$$X^{(k)} = A^k X^{(0)} = \begin{bmatrix} \Lambda_1^k X_1^{(0)} \\ \Lambda_2^k X_2^{(0)} \end{bmatrix} = \begin{bmatrix} I_p \\ M^{(k)} \end{bmatrix} \Lambda_1^k X_1^{(0)}, \qquad M^{(k)} = \Lambda_2^k X_2^{(0)} (\Lambda_1^k X_1^{(0)})^{-1}.$$

On a supposé $X_1^{(0)}$ inversible, ce qui est toujours vrai en pratique. (Puisque $\lambda_p \neq 0$, on sait déjà que Λ_1 est inversible.)

Le sous-espace des vecteurs propres qui correspondent aux valeurs propres de $\lambda_1, \ldots, \lambda_p$ est $\mathscr{E}_1 = \operatorname{im}\begin{bmatrix} I_p \\ 0 \end{bmatrix}$. Puisque $\Lambda_1^k X_1^{(0)}$ est inversible, on a d'après le Lemme 8.17

$$\sin\theta(\mathscr{X}^{(k)}, \mathscr{E}_1) = \frac{\|M^{(k)}\|_2}{1 + \|M^{(k)}\|_2} \leq \|M^{(k)}\|_2 \leq \left|\frac{\lambda_{p+1}}{\lambda_p}\right|^k \|X_2^{(0)} (X_1^{(0)})^{-1}\|_2.$$

Donc, si $|\lambda_p| > |\lambda_{p+1}|$, les colonnes de $X^{(k)}$, comme sous-espace, convergent vers $\mathscr{E}_1$. Un résultat similaire à celui du Thm. 8.7 est vrai pour une matrice quelconque diagonalisable si $|\lambda_p| > |\lambda_{p+1}|$.

Observez que l'on peut appliquer l'analyse ci-dessus pour les i premières colonnes de $X^{(k)}$. Donc, si $|\lambda_i| > |\lambda_{i+1}|$, ces colonnes de $X^{(k)}$, comme sous-espace, convergent vers les i premiers vecteurs propres dominants de A. On peut également montrer la convergence de la diagonale de $T^{(k)}$ vers les valeurs propres dominantes, de manière similaire au Cor. 8.9 (sans démonstration).

8.7 L'algorithme QR

La version simple de l'algorithme QR n'est rien d'autre que la méthode de la section précédente commencée avec $X^{(0)} = I_n$. Donc, on aimerais calculer toutes les valeurs propres. L'itération orthogonale en ce cas est définie par les équations suivantes :

$$X^{(0)} = I_n \tag{8.8}$$

$$Z^{(k)} = AX^{(k-1)} \qquad \text{(calculer la puissance)} \tag{8.9}$$

$$X^{(k)} R^{(k)} = Z^{(k)} \qquad \text{(calculer la décomposition QR)} \tag{8.10}$$

En revanche, l'**algorithme QR** fait l'itération suivante

$$\widetilde{A}^{(0)} = A \tag{8.11}$$

$$\widetilde{Q}^{(k)} \widetilde{R}^{(k)} = \widetilde{A}^{(k-1)} \qquad \text{(calculer la décomposition QR)} \tag{8.12}$$

$$\widetilde{A}^{(k)} = \widetilde{R}^{(k)} \widetilde{Q}^{(k)} \qquad \text{(calculer la multiplication)} \tag{8.13}$$

On a bien observé que l'algorithme QR se compose simplement de deux étapes : décomposer la matrice itérée en Q et R, et la composer après comme RQ.

On voudrait montrer maintenant que les deux itérations ci-dessus sont équivalentes. Définissons d'abord les matrices

$$\widetilde{Q}_k = \widetilde{Q}^{(1)}\widetilde{Q}^{(2)}\cdots\widetilde{Q}^{(k)}, \qquad \widetilde{R}_k = \widetilde{R}^{(k)}\widetilde{R}^{(k-1)}\cdots\widetilde{R}^{(1)},$$

qui groupent les matrices Q et R pendant les itérations.

Lemme 8.19. Les matrices dans l'algorithme QR (8.11)–(8.13) satisfont

$$\widetilde{A}^{(k)} = (\widetilde{Q}_k)^T A\widetilde{Q}_k, \qquad A^k = \widetilde{Q}_k\widetilde{R}_k.$$

Démonstration. Puisque $\widetilde{Q}^{(k)}$ est orthogonale, (8.12) implique $\widetilde{R}^{(k)} = (\widetilde{Q}^{(k)})^T\widetilde{A}^{(k-1)}$. Donc, d'après (8.13), on obtient la première formule :

$$\begin{aligned}\widetilde{A}^{(k)} &= (\widetilde{Q}^{(k)})^T\widetilde{A}^{(k-1)}\widetilde{Q}^{(k)} = (\widetilde{Q}^{(k)})^T(\widetilde{Q}^{(k-1)})^T\widetilde{A}^{(k-2)}\widetilde{Q}^{(k-1)}\widetilde{Q}^{(k)}\\ &= (\widetilde{Q}^{(k)})^T(\widetilde{Q}^{(k-1)})^T\cdots(\widetilde{Q}^{(1)})^T\widetilde{A}^{(0)}\widetilde{Q}^{(1)}\cdots\widetilde{Q}^{(k-1)}\widetilde{Q}^{(k)} = (\widetilde{Q}_k)^T A\widetilde{Q}_k.\end{aligned}$$

Puis, (8.13) donne

$$\begin{aligned}\widetilde{R}^{(k)} &= \widetilde{A}^{(k)}(\widetilde{Q}^{(k)})^T = (\widetilde{Q}_k)^T A\widetilde{Q}_k(\widetilde{Q}^{(k)})^T\\ &= (\widetilde{Q}_k)^T A\widetilde{Q}_{k-1}.\end{aligned}$$

En multipliant par $\widetilde{R}_{k-1}$ à droite, on obtient

$$\widetilde{R}_k = \widetilde{R}^{(k)}\widetilde{R}_{k-1} = (\widetilde{Q}_k)^T A\widetilde{Q}_{k-1}\widetilde{R}_{k-1}$$

et par récurrence

$$\widetilde{Q}_k\widetilde{R}_k = A\widetilde{Q}_{k-1}\widetilde{R}_{k-1} = \cdots = A^{k-1}\widetilde{Q}_1\widetilde{R}_1 = A^k. \qquad \square$$

L'itération orthogonale avec $X^{(0)} = I_n$ construit implicitement $A^k = X^{(k)}R_k$ où

$$R_k = R^{(k)}R^{(k-1)}\cdots R^{(1)}.$$

En effet, $X^{(1)}R_1 = Z^{(1)} = A$ et donc, par récurrence, on a d'après (8.9) et (8.10) que

$$A^k = AA^{k-1} = AX^{(k-1)}R_{k-1} = Z^{(k-1)}R_{k-1} = X^{(k)}R^{(k)}R_{k-1} = X^{(k)}R_k.$$

D'après la théorie de la section précédente, les colonnes de $X^{(k)}$ convergent vers les vecteurs propres de A (mesurés par leurs angles) en supposant que $|\lambda_1| > |\lambda_2| > \cdots > |\lambda_n|$. Cette conclusion reste vraie pour l'algorithme QR car $\widetilde{Q}_k$ est le même facteur Q dans la décomposition QR de A^k. Par conséquent, $\widetilde{A}^{(k)} = T^{(k)} = (X^{(k)})^T AX^{(k)}$.

Que peut-on dire des matrices $T^{(k)}, \widetilde{A}^{(k)}$? Pour une matrice symétrique $A = A^T$, les vecteurs propres $\boldsymbol{v}_i$ sont orthogonaux. Puisque $\boldsymbol{x}_i^{(k)} \to \pm\boldsymbol{v}_i$, on a

$$t_{ij}^{(k)} = [(X^{(k)})^T AX^{(k)}]_{ij} = (\boldsymbol{x}_i^{(k)})^T A\boldsymbol{x}_j^{(k)} \to \pm\boldsymbol{v}_j^T(\lambda_j\boldsymbol{v}_j) = \begin{cases}\pm\lambda_i, & i = j,\\ 0, & i \neq j.\end{cases}$$

Donc les matrices $T^{(k)}, \widetilde{A}^{(k)}$ convergent vers une matrice *diagonale* qui contient les valeurs propres de A. Pour une matrice A quelconque, on peut montrer que si $|\lambda_1| > |\lambda_2| > \cdots > |\lambda_n|$, ces matrices convergent vers une matrice *triangulaire* (voir les TP).

Chapitre 9

Equations différentielles ordinaires

Ce chapitre est consacré à la résolution numérique d'un système d'équations différentielles ordinaires (EDO)

$$\begin{aligned} y_1' &= f_1(t, y_1, \ldots, y_m), \quad & y_1(t_0) &= y_{10} \\ y_2' &= f_2(t, y_1, \ldots, y_m), \quad & y_2(t_0) &= y_{20} \\ &\vdots \\ y_m' &= f_m(t, y_1, \ldots, y_m), \quad & y_m(t_0) &= y_{m0} \end{aligned} \tag{9.1}$$

En notation vectorielle, ce système s'écrit

$$\boldsymbol{y}' = \boldsymbol{f}(t, \boldsymbol{y}), \qquad \boldsymbol{y}(t_0) = \boldsymbol{y}_0, \tag{9.2}$$

où $\boldsymbol{y} = (y_1, \ldots, y_m)^T$ et $\boldsymbol{f} : \mathbb{R} \times \mathbb{R}^m \to \mathbb{R}^m$.

Les équations différentielles sont utiles pour la modélisation, mais souvent difficiles (ou impossibles) à résoudre analytiquement. On a donc besoin des méthodes numériques pour leur résolution.

9.1 Existence et unicité

Le problème avec conditions initiales (9.2) s'appelle un problème de Cauchy. Même si $\boldsymbol{f}$ est continue, on n'a pas la garantie que la solution d'un problème de Cauchy soit unique : par exemple, on peut vérifier directement que l'EDO $y' = \sqrt{|y|}$ avec $y(0) = 0$ a une infinité de solutions :

$$y_c(t) = \begin{cases} \frac{1}{4}(t-c)^2, & 0, \le t \le c, \\ 0 & t > c. \end{cases}$$

Pour garantir l'unicité, on rappelle le théorème suivant (voir le cours d'Analyse).

Théorème 9.1 (Picard–Lindelöf (1894))**.** Dans un voisinage de $(t_0, \boldsymbol{y}_0)$, soit $\boldsymbol{f}(t, \boldsymbol{y})$ continue et, par rapport à $\boldsymbol{y}$, lipschitzienne,

$$\|\boldsymbol{f}(t, \boldsymbol{x}) - \boldsymbol{f}(t, \boldsymbol{z})\| \le L\|\boldsymbol{x} - \boldsymbol{z}\| \qquad \forall (t, \boldsymbol{x}), (t, \boldsymbol{x}) \text{ suff. proche de } (t_0, \boldsymbol{y}_0).$$

Alors, le problème de Cauchy possède une unique solution maximale définie sur un intervalle ouvert $(t_-, t_+) \ni t_0$. Si de plus la fonction est lipschitzienne par rapport à $\boldsymbol{y}$ dans $\mathbb{R} \times \mathbb{R}^m$, alors $(t_-, t_+) = \mathbb{R}$ et on dit que la solution est globale.

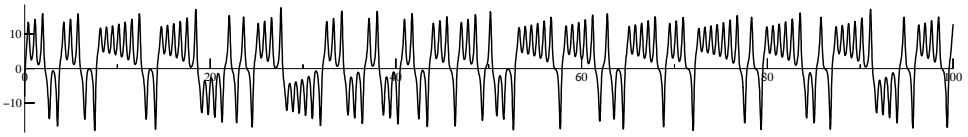
Par conséquent, pour une fonction $\boldsymbol{f} \in C^1$ (continûment différentiable) dans un voisinage de $(t_0, \boldsymbol{y}_0)$, il existe $\alpha > 0$ tel que (9.2) possède exactement une solution $\boldsymbol{y}(t)$ sur $(t_0 - \alpha, t_0 + \alpha)$.

Exemple 9.2

Une équation très célèbre est celle de Lorenz (1979)

$$\begin{aligned} y_1' &= -\sigma y_1 + \sigma y_2, & y_1(0) &= -8, \\ y_2' &= -y_1 y_3 + r y_1 - y_2, & y_2(0) &= 8, \\ y_3' &= y_1 y_2 - b y_3, & y_3(0) &= r - 1, \end{aligned}$$

où $\sigma = 10, r = -28, b = 2/3$. *La solution est chaotique et ne devient jamais périodique. Voici la composante* $y_1(t)$ *comme fonction de* t *sur l'intervalle* $[0, 100]$.



Les méthodes classiques comme les méthodes de Runge–Kutta (voir la Sec. 9.2) ou les méthodes à plusieurs pas nous permettent de trouver sans difficultés des bonnes approximations.

9.2 Méthodes de Runge–Kutta

Une grande famille de méthodes pour la résolution de (9.2) est basée sur l'idée suivante : Si on souhaite calculer $\boldsymbol{y}(t)$ sur l'intervalle $[t_0, T]$, on subdivise l'intervalle en plusieurs pas (ou sous-intervalles) $t_0 < t_1 < t_2 < \cdots < t_N = T$. Ensuite, on calcule des approximations $\boldsymbol{y}_n \approx \boldsymbol{y}(t_n)$ par [1]

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + h_n \boldsymbol{\Phi}(h_n, t_n, \boldsymbol{y}_n), \tag{9.3}$$

où $h_n = t_{n+1} - t_n$ et $\boldsymbol{\Phi}$ est la fonction qui représentera la méthode numérique. Observons que l'on a seulement besoin de $\boldsymbol{y}_n$ pour évaluer $\boldsymbol{\Phi}$, et pas de toute l'historique $\boldsymbol{y}_{n-1}, \boldsymbol{y}_{n-2}, \ldots$. On appelle (9.3) une **méthode à un pas**, contrairement aux méthodes à plusieurs pas qui s'appuient sur plusieurs $\boldsymbol{y}_i$ antérieurement calculés.

Pour simplifier la notation, nous considérons seulement le premier pas ($n = 0$ dans (9.3)) avec $h_0 = h$:

$$\boldsymbol{y}' = \boldsymbol{f}(t, \boldsymbol{y}),\ \boldsymbol{y}(t_0) = \boldsymbol{y}_0 \quad \Longrightarrow \quad \boldsymbol{y}(t_0 + h) - \boldsymbol{y}(t_0) = \int_{t_0}^{t_0+h} \boldsymbol{f}(\tau, \boldsymbol{y}(\tau))\, d\tau.$$

On voit que $\boldsymbol{\Phi}$ devrait satisfaire

$$h\boldsymbol{\Phi}(h, t_0, \boldsymbol{y}_0) \approx \int_{t_0}^{t_0+h} \boldsymbol{f}(\tau, \boldsymbol{y}(\tau))\, d\tau.$$

On peut maintenant approcher chaque composante de cette intégrale par une formule de quadrature (voir Chap. 4). On obtient ainsi des fonctions $\boldsymbol{\Phi}$ et donc aussi plusieurs méthodes numériques pour la résolution de (9.2).

L'exemple le plus simple est la **méthode d'Euler** (1768). En approchant l'intégrale avec un seul point à gauche,

$$\int_{t_0}^{t_0+h} \boldsymbol{f}(\tau, \boldsymbol{y}(\tau))\, d\tau \approx h\boldsymbol{f}(t_0, \boldsymbol{y}_0), \tag{9.4}$$

1. Faites attention : $\boldsymbol{y}_n$ n'est pas la $n$ième composante du vecteur $\boldsymbol{y}$.

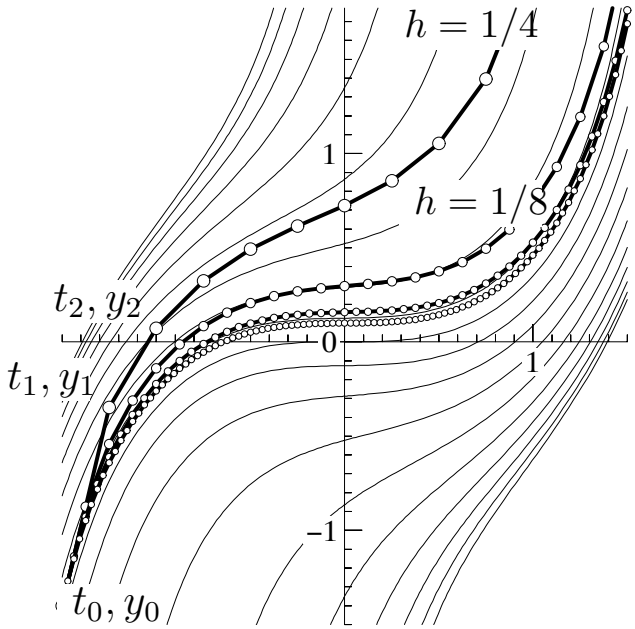


FIGURE 9.1 – Solutions obtenues par la méthode d'Euler pour (9.5).

on obtient l'itération

$$\boldsymbol{y}_1 = \boldsymbol{y}_0 + h\boldsymbol{f}(t_0, \boldsymbol{y}_0).$$

L'approximation est donc obtenu en remplaçant chaque composante de la solution $\boldsymbol{y}(t)$ par sa tangente au point (t_0, y_0). La Fig. 9.1 montre la solution numérique pour le problème

$$y' = t^2 + y^2, \quad y(-1.5) = -1.4 \tag{9.5}$$

et pour $h = 1/4, 1/8$, etc. On peut observer la convergence vers une fonction qui, comme on le verra dans la Sec. 9.3, est la solution du problème.

La méthode d'Euler utilise une formule de quadrature d'ordre 1. L'idée évidente est d'utiliser des formules ayant un ordre plus élevé :

$$h\boldsymbol{\Phi}(h, t_0, \boldsymbol{y}_0) = \int_{t_0}^{t_0+h} \boldsymbol{f}(\tau, \boldsymbol{y}(\tau))\, d\tau \approx h \sum_{i=1}^{s} b_i \boldsymbol{f}(t_0 + hc_i, \boldsymbol{y}(t_0 + hc_i)) \tag{9.6}$$

où $(b_i, c_i)_{i=1}^{s}$ est une formule de quadrature pour l'intervalle $[0, 1]$. Il semble qu'il faille connaître $\boldsymbol{y}(\tau_i)$ en des points $\tau_i \geq t_0$ pour définir $\boldsymbol{\Phi}$, mais dans une méthode à un pas, la fonction $\boldsymbol{\Phi}$ ne peut utiliser que $\boldsymbol{y}(\tau_0) = \boldsymbol{y}_0$. Heureusement, il y a une astuce …

La **méthode de Runge** (1895) s'est inspirée de la formule du point milieu,

$$\int_{t_0}^{t_0+h} \boldsymbol{f}(\tau, \boldsymbol{y}(\tau))\, d\tau \approx h\boldsymbol{f}(t_0 + \tfrac{h}{2}, \underbrace{\boldsymbol{y}(t_0 + \tfrac{h}{2})}_{\text{inconnu}}), \tag{9.7}$$

et on remplace la valeur inconnue $\boldsymbol{y}(t_0 + \frac{h}{2})$ par la méthode d'Euler. Ceci nous donne

$$\boldsymbol{y}_1 = \boldsymbol{y}_0 + h\boldsymbol{f}(t_0 + \tfrac{h}{2}, \boldsymbol{y}_0 + \tfrac{h}{2}\boldsymbol{f}(t_0, \boldsymbol{y}_0)). \tag{9.8}$$

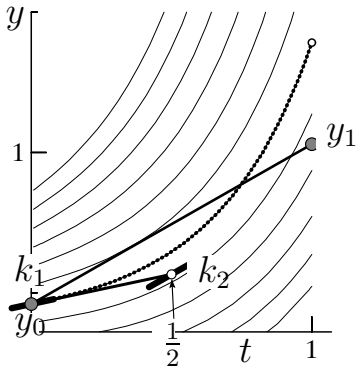


FIGURE 9.2 – La méthode de Runge.

En utilisant les deux **étages** $\boldsymbol{k}_1, \boldsymbol{k}_2$, on peut également écrire (voir aussi la Fig. 9.2)

$$\boldsymbol{y}_1 = \boldsymbol{y}_0 + h\boldsymbol{k}_2, \qquad \boldsymbol{k}_1 = \boldsymbol{f}(t_0, \boldsymbol{y}_0), \quad \boldsymbol{k_2} = \boldsymbol{f}(t_0 + \tfrac{h}{2}, \boldsymbol{y}_0 + \tfrac{h}{2}\boldsymbol{k}_1).$$

On peut généraliser cette idée à la formule (9.6) : pour approcher les $\boldsymbol{y}(t_0 + hc_i)$ dans la définition de l'i-ème étage,

$$\boldsymbol{k}_i \approx \boldsymbol{f}(t_0 + hc_i, \boldsymbol{y}(t_0 + hc_i)),$$

on utilise les étages $\boldsymbol{k}_1, \dots, \boldsymbol{k}_{i-1}$ antérieurement calculés :

$$\boldsymbol{y}(t_0 + hc_i) = \boldsymbol{y}_0 + \int_{t_0}^{t_0+hc_i} \boldsymbol{f}(\tau, \boldsymbol{y}(\tau))\, d\tau \approx \boldsymbol{y}_0 + h\sum_{j=1}^{i-1} a_{ij}\boldsymbol{k}_j. \tag{9.9}$$

On est conduit ainsi à la définition suivante (Kutta 1901).

Définition 9.3. Une **méthode de Runge–Kutta explicite à s étages** est donnée par

$$\begin{aligned}
\boldsymbol{k}_1 &= \boldsymbol{f}(t_0 + c_1 h, \boldsymbol{y}_0) \\
\boldsymbol{k}_2 &= \boldsymbol{f}(t_0 + c_2 h, \boldsymbol{y}_0 + h a_{21}\boldsymbol{k}_1) \\
&\vdots \\
\boldsymbol{k}_s &= \boldsymbol{f}(t_0 + c_s h, \boldsymbol{y}_0 + h(a_{s1}\boldsymbol{k}_1 + a_{s2}\boldsymbol{k}_2 + \cdots + a_{s,s-1}\boldsymbol{k}_{s-1})) \\
\boldsymbol{y}_1 &= \boldsymbol{y}_0 + h(b_1\boldsymbol{k}_1 + b_2\boldsymbol{k}_2 + \cdots + b_s\boldsymbol{k}_s),
\end{aligned} \tag{9.10}$$

où c_i, a_{ij} et b_j sont des coefficients.

Par la suite nous supposerons toujours que les c_i satisfont

$$c_1 = 0, \qquad c_i = \sum_{j=1}^{i-1} a_{ij}, \quad (i = 2, \dots, 2).$$

Ceci est une condition naturelle car elle signifie que les intégrales dans (9.9) sont calculées exactement pour $\boldsymbol{f} \equiv \boldsymbol{1}$.

Les **tableaux de Butcher** donnent une écriture compacte :

$$\begin{array}{c|cccc} c_1 & & & & \\ c_2 & a_{21} & & & \\ c_3 & a_{31} & a_{32} & & \\ \vdots & \vdots & & \ddots & \\ c_s & a_{s1} & a_{s2} & \cdots & a_{s,s-1} \\ \hline & b_1 & b_2 & \cdots & b_{s-1} \quad b_s \end{array}$$

Par exemple :

Euler : $\begin{array}{c|c} 0 & \\ \hline & 1 \end{array}$ Runge : $\begin{array}{c|cc} 0 & & \\ \frac{1}{2} & \frac{1}{2} & \\ \hline & 0 & 1 \end{array}$ Heun : $\begin{array}{c|ccc} 0 & & & \\ \frac{1}{3} & \frac{1}{3} & & \\ \frac{2}{3} & 0 & \frac{2}{3} & \\ \hline & \frac{1}{4} & 0 & \frac{3}{4} \end{array}$

Similaire à la Def. 4.3 pour les formules de quadrature, on a la notion de l'ordre d'une méthode à un pas.

Définition 9.4. Une méthode de Runge–Kutta (9.10) est d'**ordre** p si la solution de $\boldsymbol{y}' = \boldsymbol{f}(t, \boldsymbol{y})$, $\boldsymbol{y}(t_0) = \boldsymbol{y}_0$ satisfait

$$\|\boldsymbol{y}_1 - \boldsymbol{y}(t_0 + h)\| = O(h^{p+1}), \qquad h \to 0, \tag{9.11}$$

pour toute fonction $\boldsymbol{f}(t, \boldsymbol{y})$ suffisamment différentiable. La quantité $\boldsymbol{y}_1 - \boldsymbol{y}(t_0 + h)$ s'appelle l'**erreur locale** de la méthode.

Il suffit de montrer l'ordre pour l'EDO scalaire

$$y' = f(t, y), \qquad y(0) = y_0,$$

car l'ordre pour l'EDO vectorielle (9.2) suit en appliquant le résultat scalaire à chacune des composantes de $\boldsymbol{f}$.

La méthode d'Euler est d'ordre 1, car

$$\begin{aligned} y_1 - y(t_0 + h) &= y_1 - (y(t_0) + h y'(t_0) + O(h^2)) \\ &= y_1 - (\underbrace{y_0 + h f(t_0, y_0)}_{= y_1} + O(h^2)) = O(h^2). \end{aligned}$$

La méthode de Runge (9.8) est basée sur la formule du point milieu qui est une formule de quadrature d'ordre 2 (voir la tableau 4.1 et le Lemme 4.5) :

$$y(t_0 + h) = y_0 + \int_{t_0}^{t_0+h} f(\tau, y(\tau))\, d\tau = y_0 + h f(t_0 + \tfrac{h}{2}, y(t_0 + \tfrac{h}{2})) + O(h^3)$$

En remplaçant $y(t_0 + \frac{h}{2})$ par la valeur $y_0 + \frac{h}{2} f(t_0, y_0)$ de la méthode d'Euler, on ajoute une terme $O(h^2)$ (voir (9.11)) :

$$y(t_0 + h) = y_0 + h f\left(t_0 + \tfrac{h}{2}, y_0 + \tfrac{h}{2} f(t_0, y_0) + O(h^2)\right) + O(h^3))$$

En utilisant un développement de Taylor,

$$f(t_0, y_0 + h) = f(t_0, y_0) + h f_y(t_0, y_0) + O(h^2)$$

où $f_y = \partial f / \partial y$, on obtient

$$y(t_0 + h) = y_0 + f(t_0 + \tfrac{h}{2}, y_0 + \tfrac{h}{2} f(t_0, y_0)) + O(h^3)) = y_1 + O(h^3).$$

La méthode de Runge est donc d'ordre 2.

Pour construire des méthodes d'ordre plus élevé, il faut développer la solution exacte $y(t_0 + h)$ et la solution numérique $y_1 = y_1(h)$ en séries de Taylor autour de $h = 0$. On a, par exemple,

$$y''(t) = (y'(t))' = \frac{d}{dt}(f(t, y(t)) = f_t(t, y(t)) + f_y(t, y(t)) y' = f_t(\cdot) + f_y(\cdot) f(\cdot),$$

où $f_t = \partial f / \partial t$ et $f_y = \partial f / \partial y$. Une comparaison des coefficients de h^i pour $i = 1, \ldots, p$ donne des conditions pour les paramètres a_{ij} et b_i. L'idée est simple, mais l'exécution de ce plan est loin d'être facile (8 conditions pour l'ordre $p = 4$, 200 pour $p = 8$ et 1205 pour $p = 10$). Voir la série d'exercices pour le calcul de $p = 2$.

Exemple 9.5

Quelques méthodes d'ordre 4 sont inspirées des règles de Simpson 1/3 *et* 3/8.

0				
$\frac{1}{2}$	$\frac{1}{2}$			
$\frac{1}{2}$	0	$\frac{1}{2}$		
1	0	0	1	
	$\frac{1}{6}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{6}$

« La » méthode de Runge–Kutta

0				
$\frac{1}{3}$	$\frac{1}{3}$			
$\frac{2}{3}$	$-\frac{1}{3}$	1		
1	1	−1	1	
	$\frac{1}{8}$	$\frac{3}{8}$	$\frac{3}{8}$	$\frac{1}{8}$

La règle 3/8

9.3 Convergence des méthodes de Runge–Kutta

En commençant en $t = t_0$, nous appliquons n fois une méthode à un pas (9.3) à l'EDO $\boldsymbol{y}' = \boldsymbol{f}(t, \boldsymbol{y}), \boldsymbol{y}(t_0) = \boldsymbol{y}_0$ et nous cherchons à estimer l'**erreur globale** $\|\boldsymbol{y}(t_n) - \boldsymbol{y}_n\|$ en $t = t_n = n h_n$.

Théorème 9.6. Soit $\boldsymbol{y}(t)$ la solution de $\boldsymbol{y}' = \boldsymbol{f}(t,\boldsymbol{y}), \boldsymbol{y}(t_0) = \boldsymbol{y}_0$ sur l'intervalle $[t_0, T]$. Supposons que

(a) l'erreur locale satisfasse pour $t \in [t_0, T]$, $h \le h_{\max}$,

$$\|\boldsymbol{y}(t+h) - \boldsymbol{y}(t) - h\boldsymbol{\Phi}(t,\boldsymbol{y}(t),h)\| \le Ch^{p+1};$$

(b) $\boldsymbol{\Phi}(t,\boldsymbol{y},h)$ soit lipschitzienne en $\boldsymbol{y}$ dans un voisinage de la solution $(t,\boldsymbol{y}(t))$, pour tout $h \le h_{\max}$ et $t \in [t_0, T]$:

$$\|\boldsymbol{\Phi}(t,\boldsymbol{x},h) - \boldsymbol{\Phi}(t,\boldsymbol{z},h)\| \le L\|\boldsymbol{x}-\boldsymbol{z}\| \qquad \forall \boldsymbol{x},\boldsymbol{z} \in U = \{\boldsymbol{z} : \|\boldsymbol{z} - \boldsymbol{y}(t)\| \le R\}.$$

Alors, si $h = \max_i h_i$ est suffisamment petit, l'erreur globale satisfait

$$\|\boldsymbol{y}(t_n) - \boldsymbol{y}_n\| \le \widetilde{C} \cdot h^p = O(h^p), \qquad \text{pour tout } t_n \in [t_0, T],$$

où $\widetilde{C}$ dépend de L, T mais pas de h.

Nous remarquons qu'il ne suffit pas d'additionner les erreurs commises à chaque pas de la méthode. En effet, en faisant un pas de la méthode Runge–Kutta, on a déjà pris une autre trajectoire que la solution exacte ; on résout plutôt

$$\tilde{\boldsymbol{y}}' = \boldsymbol{f}(t,\tilde{\boldsymbol{y}}), \qquad \tilde{\boldsymbol{y}}(t_1) = \boldsymbol{y}_1 \neq \boldsymbol{y}(t_1).$$

Ceci est clairement visible dans la Fig. 9.1. Donc, l'erreur peut être amplifiée lors de sa propagation, car

$$\underbrace{\boldsymbol{y}(t_n) - \tilde{\boldsymbol{y}}(t_n)}_{\substack{\text{erreur globale} \\ \text{due au 1er pas}}} \neq \boldsymbol{y}(t_1) - \tilde{\boldsymbol{y}}(t_1) = \underbrace{\boldsymbol{y}(t_1) - \boldsymbol{y}_1}_{\text{erreur locale}}.$$

Démonstration. En utilisant les propriétés (a) et (b) de la méthode, on calcule

$$\begin{aligned}
\|\boldsymbol{y}(t_{n+1}) - \boldsymbol{y}_{n+1}\| &= \|\boldsymbol{y}(t_{n+1}) - \boldsymbol{y}_n - h_n\boldsymbol{\Phi}(t_n,\boldsymbol{y}_n,h_n)\| \\
&\le \|\boldsymbol{y}(t_{n+1}) - \boldsymbol{y}(t_n) - h_n\boldsymbol{\Phi}(t_n,\boldsymbol{y}(t_n),h_n)\| \\
&\quad + h_n\|\boldsymbol{\Phi}(t_n,\boldsymbol{y}(t_n),h_n) - \boldsymbol{\Phi}(t_n,\boldsymbol{y}_n,h_n)\| + \|\boldsymbol{y}(t_n) - \boldsymbol{y}_n\| \\
&\le Ch_n^{p+1} + h_nL\|\boldsymbol{y}(t_n) - \boldsymbol{y}_n\| + \|\boldsymbol{y}(t_n) - \boldsymbol{y}_n\| \\
&\le Ch_n^{p+1} + e^{h_nL}\|\boldsymbol{y}(t_n) - \boldsymbol{y}_n\|,
\end{aligned}$$

car $1 + x \le e^x$ pour tout x. En écrivant $E_n = \|\boldsymbol{y}(t_n) - \boldsymbol{y}_n\|$, on a pour tout n

$$\begin{aligned}
E_n &\le Ch_{n-1}^{p+1} + e^{h_{n-1}L}E_{n-1} \\
&\le Ch_{n-1}^{p+1} + e^{h_{n-1}L}(Ch_{n-2}^{p+1} + e^{h_{n-2}L}E_{n-2}) \\
&\le \cdots \le Ch_{n-1}^{p+1} + \left(\sum_{j=1}^{n-1} e^{h_{n-1}L}\cdots e^{h_jL}Ch_{j-1}^{p+1}\right) + e^{h_{n-1}L}\cdots e^{h_0L}E_0 \\
&\le Ch^p \sum_{j=1}^{n} e^{(t_n - t_j)L}h_{j-1}
\end{aligned}$$

car $h_j \le h$ et $e^{h_{n-1}L}\cdots e^{h_jL} = e^{(t_n-t_j)L}$ pour tout $j = 0, \ldots, n$. Cette dernière somme est bornée par l'intégrale suivante :

$$E_n \le Ch^p \sum_{j=1}^{n} e^{(t_n-t_j)L}h_{j-1} \le Ch^p \int_{t_0}^{t_n} e^{L(t_n-t)}\,dt = Ch^p\,\frac{e^{L(t_n-t_0)} - 1}{L}.$$

Pour justifier l'utilisation de (b), il faut prendre h suffisamment petit pour que la solution numérique reste dans l'ensemble U, ce qui est le cas lorsque

$$\frac{Ch^p}{L}(e^{L(T-t_0)}-1) < R. \qquad \square$$

Supposons maintenant que (9.3) représente une méthode de Runge–Kutta et vérifions les hypothèses du théorème précédent. La condition (a) est satisfaite pour une méthode d'ordre p par Def. 9.4. Il reste à vérifier la condition (b). D'après (9.10), on a

$$\boldsymbol{\Phi}(t,\boldsymbol{y},h) = \sum_{i=1}^{s} b_i \boldsymbol{k}_i, \qquad \boldsymbol{k}_i = \boldsymbol{f}(t+c_i h, \boldsymbol{y} + h\sum_{j=1}^{i-1} a_{ij}\boldsymbol{k}_j).$$

Observez que les étages dépendent de $\boldsymbol{y}$.

Lemme 9.7. Si $\boldsymbol{f}$ satisfait une condition de Lipschitz $\|\boldsymbol{f}(t,\boldsymbol{y})-\boldsymbol{f}(t,\boldsymbol{z})\| \le \widetilde{L}\|\boldsymbol{y}-\boldsymbol{z}\|$, alors dans un voisinage de la solution $\boldsymbol{y}(t)$, la méthode de Runge–Kutta satisfait la condition (b) du Thm. 9.6 avec constante

$$L = 1 + h\widetilde{L}\Big(\sum_i |b_i| + h_{\max}\widetilde{L}\sum_{i,j}|b_i a_{ij}| + (h_{\max}\widetilde{L})^2 \sum_{i,j,k}|b_i a_{ij} a_{jk}| + \cdots\Big).$$

Démonstration. Dénotons les étages par $\boldsymbol{k}_i(\boldsymbol{y})$ pour souligner la dépendance de $\boldsymbol{y}$. La condition de Lipschitz pour $\boldsymbol{f}(t,\boldsymbol{y})$ appliquée à $\boldsymbol{k}_i(\boldsymbol{y})-\boldsymbol{k}_i(\boldsymbol{z})$ nous donne

$$\begin{aligned}
\|\boldsymbol{k}_1(y)-\boldsymbol{k}_1(z)\| &= \|\boldsymbol{f}(t,\boldsymbol{y})-\boldsymbol{f}(t,\boldsymbol{z})\| \le \widetilde{L}\|\boldsymbol{y}-\boldsymbol{z}\|,\\
\|\boldsymbol{k}_2(y)-\boldsymbol{k}_2(z)\| &\le \widetilde{L}\|\boldsymbol{y}-\boldsymbol{z}+ha_{21}(\boldsymbol{k}_1(\boldsymbol{y})-\boldsymbol{k}_1(\boldsymbol{z}))\| \le \widetilde{L}(1+h_{max}\widetilde{L}|a_{21}|)\|\boldsymbol{y}-\boldsymbol{z}\|,
\end{aligned}$$

et ainsi de suite. Ces estimations insérées dans

$$\|\boldsymbol{\Phi}(t,\boldsymbol{y},h)-\boldsymbol{\Phi}(t,\boldsymbol{z},h)\| \le \|\boldsymbol{y}-\boldsymbol{z}\| + h\sum_{i=1}^{s}|b_i|\cdot\|\boldsymbol{k}_i(\boldsymbol{y})-\boldsymbol{k}_i(\boldsymbol{z})\|$$

donnent le résultat. $\square$

9.4 Méthodes implicites

Il existe aussi d'autres méthodes dites **implicites**. Par exemple, on peut modifier la méthode d'Euler en utilisant un seul point à droite dans (9.4). On obtient ainsi la **méthode d'Euler implicite**

$$\boldsymbol{y}_1 = \boldsymbol{y}_0 + h\boldsymbol{f}(t_1,\boldsymbol{y}_1).$$

Cette méthode nécessite la résolution d'un système non linéaire à chaque pas car $\boldsymbol{y}_1$ n'est pas obtenu directement en évaluant la fonction à droite ci-dessus. En effet, en supposant que $\boldsymbol{f}$ est lipschitzienne de constante L, l'application $\boldsymbol{x} \mapsto \boldsymbol{y}_0 + h\boldsymbol{f}(t_1,\boldsymbol{x})$ est lipschitzienne de constante hL. Si $hL<1$, cette application devient contractante et possède donc un unique point fixe par le théorème du point fixe pour tout h assez petit.

En approchant $\boldsymbol{y}(t_0+\frac{h}{2})$ par $\frac{1}{2}(\boldsymbol{y}_0+\boldsymbol{y}_1)$ dans le formule du point milieu (9.7), on obtient la **méthode implicite du point milieu**

$$\boldsymbol{y}_1 = \boldsymbol{y}_0 + h\boldsymbol{f}\left(t_0+\frac{h}{2}, \frac{\boldsymbol{y}_0+\boldsymbol{y}_1}{2}\right).$$

On peut également généraliser les méthodes de Runge–Kutta :

Définition 9.8. Une **méthode de Runge–Kutta implicite à s étages** est donnée par

$$\begin{aligned} \boldsymbol{k}_i &= \boldsymbol{f}(t_0 + c_i h, \boldsymbol{y}_0 + h \sum_{j=1}^{s} a_{ij} \boldsymbol{k}_j) \\ \boldsymbol{y}_1 &= \boldsymbol{y}_0 + h \sum_{i=1}^{s} b_i \boldsymbol{k}_i, \end{aligned} \tag{9.12}$$

où c_i, a_{ij} et b_j sont des coefficients et $a_{ij} \neq 0$ pour tout $j \geq i$.

Euler implicite : $\begin{array}{c|c} 1 & 1 \\ \hline & 1 \end{array}$ point milieu : $\begin{array}{c|c} \frac{1}{2} & \frac{1}{2} \\ \hline & 1 \end{array}$ trapèze (Crank–Nicolson) : $\begin{array}{c|cc} 0 & 0 & 0 \\ 1 & \frac{1}{2} & \frac{1}{2} \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}$

Dans toutes ces méthodes implicites, pour calculer $\boldsymbol{y}_{n+1}$ à partir de $\boldsymbol{y}_n$, on peut appliquer une méthode de résolution de système non-linéaire (e.g. Newton ou point fixe). La question évidente est : à quoi servent les méthodes implicites sachant qu'elles sont bien plus compliquées que les méthodes explicites. La réponse est donnée dans la section suivante.

9.5 Equations différentielles raides (stiff)

Pour certaines EDOs, dites **raides**, les méthodes explicites ont besoin d'un pas de temps h dramatiquement petit à tel point qu'elles deviennent inutilisables. Considérons le problème assez simple

$$\varepsilon y' = -y + \cos t, \qquad 0 < \varepsilon \ll 1. \tag{9.13}$$

Cette équation est linéaire inhomogène. Cherchons une solution particulière de la forme $y(t) = A\cos t + B\sin t$. En introduisant cette fonction dans (9.13), on obtient

$$-\varepsilon A \sin t + \varepsilon B \cos t = -A\cos t - B\sin t + \cos t,$$

et une comparaison des coefficients donne $A = 1/(1+\varepsilon^2)$ et $B = \varepsilon/(1+\varepsilon^2)$. Comme la solution générale de (9.13) est la somme de la solution générale de l'équation homogène et d'une solution particulière, on obtient

$$y(t) = e^{-t/\varepsilon} C + \cos t + \varepsilon \sin t + O(\varepsilon^2).$$

La méthode d'Euler (explicite) appliquée au problème (9.13) avec des pas constants, donne avec $t_n = nh$

$$y_{n+1} = \left(1 - \frac{h}{\varepsilon}\right) y_n + \frac{h}{\varepsilon} \cos t_n.$$

Ceci est une équation aux différences finies qui est linéaire et inhomogène. On peut montrer que la solution générale satisfait

$$y_n = \left(1 - \frac{h}{\varepsilon}\right)^n C + \cos t_n + \varepsilon \sin t + O(h\varepsilon).$$

On voit que la solution numérique y_n est proche de la solution exacte seulement si $|1 - h/\varepsilon| < 1$, c.-à-d., si $0 < h < 2\varepsilon$. Si ε est très petit, par exemple $\varepsilon = 10^{-6}$, une telle restriction est inacceptable. Voir aussi le dessin à droite dans la Fig. 9.3.

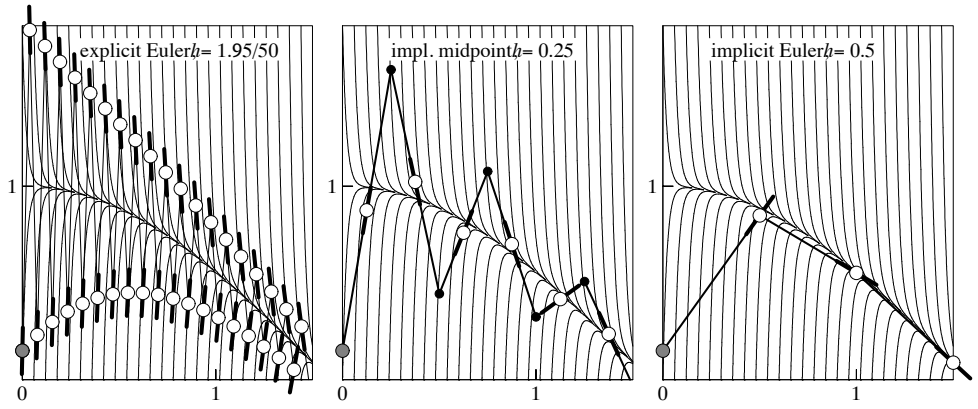


FIGURE 9.3 – Solutions exactes et numériques pour le problème (9.13) et $\varepsilon = 1/50$.

En revanche, pour la méthode d'Euler implicite, le même calcul donne

$$\left(1+\frac{h}{\varepsilon}\right) y_{n+1} = y_n + \frac{h}{\varepsilon}\cos t_n.$$

dont la solution peut être écrite sous la forme

$$y_n = \left(1+\frac{h}{\varepsilon}\right)^{-n} C + \cos t_n + \varepsilon \sin t + O(h\varepsilon).$$

Cette fois-ci nous n'avons pas de restriction sur la longueur du pas, car $|1+h/\varepsilon|^{-1} < 1$ pour tout $h > 0$. Le dessin à droite de la Fig. 9.3 illustre bien la bonne approximation même si h est très grand.

Le calcul précédent a montré que ce n'est pas la solution particulière qui cause des difficultés à la méthode explicite, mais c'est l'approximation de la solution de l'équation homogène $\varepsilon y' = -y$. Nous considérons donc le problème un peu plus général

$$y'(t) = \lambda y(t), \qquad y(0) = 1, \tag{9.14}$$

comme l'**équation de test** (Dahlquist). Sa solution exacte est

$$y(t) = e^{\lambda t} = e^{\mathrm{Re}\,\lambda t}(\cos(\mathrm{Im}\,\lambda t) + \imath \sin(\mathrm{Im}\,\lambda t))$$

et elle reste bornée pour tout $t \geq 0$ si $\mathrm{Re}\,\lambda \leq 0$. Il est donc intéressant d'étudier sous quelles conditions la solution numérique reste bornée aussi.

Pour Euler explicite, on trouve $y_{n+1} = y_n + h\lambda y_n = R(h\lambda) y_n$ où

$$R(z) = 1 + z \qquad \text{(Euler explicite)}.$$

Pour Euler implicite, on trouve $y_{n+1} = y_n + h\lambda y_{n+1} = R(h\lambda) y_n$ où

$$R(z) = \frac{1}{1-z} \qquad \text{(Euler implicite)}.$$

En général, pour une méthode de Runge–Kutta appliquée au (9.14) où $\lambda \in \mathbb{C}$ est fixé, on obtient une récurrence du type

$$y_{n+1} = R(h\lambda) y_n$$

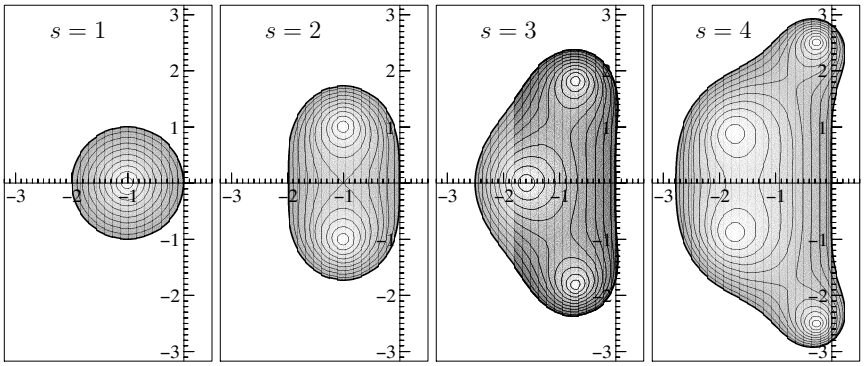


FIGURE 9.4 – Domaines de stabilité pour les méthodes de Runge–Kutta explicites avec s étages et d'ordre $p = s$.

où $R(z)$ est une fraction rationnelle appelée la **fonction de stabilité** qui ne dépend pas du produit $h\lambda$.

On voit que la solution numérique reste bornée si et seulement si $|R(h\lambda)| \leq 1$. On définit donc naturellement le domaine de stabilité correspondant.

Définition 9.9. Le **domaine de stabilité** d'une méthode de Runge-Kutta avec fonction de stabilité $R(z)$ est défini par

$$\mathscr{S} = \{z \in \mathbb{C} : |R(z)| \leq 1\}.$$

Pour Euler explicite, $\mathscr{S}$ est un disque fermé de centre -1 et de rayon 1. Ainsi, si $\lambda < 0$, la condition de stabilité s'écrit $h \leq |2/\lambda|$. Pour Euler implicite, $\mathscr{S}$ est le complémentaire d'un disque ouvert de centre 1 et de rayon 1. Ainsi, les solutions numériques restent bornées pour tout pas de temps h (c'est une méthode *inconditionnellement stable*).

On définie maintenant la A-stabilité.

Définition 9.10. Une méthode de Runge-Kutta est dite ***A*-stable** si son domaine de stabilité vérifie

$$\mathbb{C}_- = \{z \in \mathbb{C} : \operatorname{Re} z \leq 0\} \subset \mathscr{S}$$

ou de manière équivalente, sa fonction de stabilité vérifie

$$|R(z)| \leq 1, \quad \text{pour tout } z \in \mathbb{C} \text{ avec } \operatorname{Re} z \leq 0.$$

La méthode d'Euler implicite est donc A-stable. Pour quelques méthodes de Runge–Kutta explicites (Euler explicite incluse), les domaines de stabilité sont dessinés dans la Fig. 9.4. On constate que la condition de stabilité $h\lambda \in \mathscr{S}$ impose une restriction sévère à h. Ces méthodes ne sont donc pas recommandées pour la résolution des équations différentielles raides.

9.6 Application : l'équation de la chaleur

Considérons l'équation de la chaleur en dimension 1 d'espace et de temps

$$\frac{\partial u}{\partial t} - \frac{\partial^2 u}{\partial x^2} = 0, \qquad x \in [0,1], \quad t > 0.$$

avec conditions aux limites $u(0,t) = u(1,t) = 0$ et une condition initiale $u(x,0) = g(x)$.

Pour la résolution numérique, on commence par discrétiser l'espace $[0,1]$ avec une grille uniforme de $N+2$ points

$$x_j = j\Delta x, \qquad j = 0,\dots N+1,$$

où $\Delta x = 1/(N+1)$ est le pas en espace. On approche $u(x_j,t)$ par $u_j(t)$ en considérant l'approximation par une différence finie (voir la Sec. 7.7)

$$\frac{\partial^2 u}{\partial x^2}(x_j,t) \approx \frac{u(x_{j-1},t) - 2u(x_j,t) + u(x_{j+1},t)}{\Delta x^2} \approx \frac{u_{j-1} - 2u_j + u_{j+1}}{\Delta x^2}.$$

On obtient le système d'EDO suivant de dimension N,

$$\frac{\partial u_j}{\partial t} = \frac{u_{j-1} - 2u_j + u_{j+1}}{\Delta x^2}, \qquad j = 1,\dots N.$$

en tenant compte des conditions aux limites $u_0 = u_{N+1} \equiv 0$. Sous forme matricielle, en posant $\boldsymbol{u} = (u_1,\dots,u_N)^T$, on obtient

$$\boldsymbol{u}'(t) = A\boldsymbol{u}(t), \qquad \text{avec} \qquad A = \frac{1}{\Delta x^2}\begin{bmatrix} -2 & 1 & & & \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ & & & 1 & -2 \end{bmatrix}.$$

La condition initiale $u(x,0) = g(x)$ devient

$$\boldsymbol{u}(0) = \begin{bmatrix} g(x_1) & g(x_2) & \cdots & g(x_N) \end{bmatrix}^T.$$

La matrice A est symétrique réelle donc diagonalisable en base orthonormée, $A = Q\Lambda Q^T$, avec des valeurs propres réelles. De plus, on sait (voir la série d'exercices) que les valeurs propres de la matrice A se situent dans l'intervalle ouvert $(-4/\Delta x^2, 0)$. Avec le changement de variable $\boldsymbol{u}(t) = Q\boldsymbol{v}(t)$, le système ci-dessus s'écrit comme système de n équations scalaires *découplées* :

$$\boldsymbol{v}'(t) = \Lambda\boldsymbol{v}(t), \qquad \boldsymbol{v}(0) = Q^T\boldsymbol{u}(0). \tag{9.15}$$

La solution est donc

$$v_i(t) = e^{\lambda_i t}\cdot v_i(0).$$

Puisque toutes valeurs propres sont réelles et inférieures à 0, on a aussi

$$\|\boldsymbol{u}(t)\|_2 = \|\boldsymbol{v}(t)\|_2 \to 0, \quad t \to \infty.$$

Considérons maintenant les méthodes d'Euler explicite et implicite appliquées à ce système avec un pas de temps h. Pour la méthode d'Euler explicite, on obtient

$$\boldsymbol{u}_{n+1} = (I + hA)\boldsymbol{u}_n, \qquad \boldsymbol{v}_{n+1} = (I + h\Lambda)\boldsymbol{v}_n.$$

où $\boldsymbol{v}_n = Q^T\boldsymbol{u}_n$ pour tout n (c'est aussi la méthode d'Euler appliquée à (9.15)). Ainsi, la solution numérique reste bornée si $|R(h\lambda_i)| = |1 + h\lambda_i| \le 1$, pour toutes les valeurs propres λ_i avec $\lambda_i \in (-4/\Delta x^2, 0)$. Ceci donne la condition de stabilité

$$h < \frac{1}{2}\Delta x^2.$$

On remarque que cette condition est très restrictive, car elle impose $h \ll \Delta x$ pour que la solution numérique reste bornée, ce qui rend la méthode très coûteuse. Le problème est donc raide.

Pour la méthode d'Euler implicite, on a

$$\boldsymbol{u}_{n+1} = \boldsymbol{u}_n + hA\boldsymbol{u}_{n+1} = (I - hA)^{-1}\boldsymbol{u}_n, \qquad \boldsymbol{v}_{n+1} = (1 - h\Lambda)^{-1}\boldsymbol{v}_n.$$

Les valeurs propres de $I - hA$ et $I - h\Lambda$ sont $1 - h\lambda_i > 1$ donc ces matrices sont bien inversibles. De plus, les coefficients $R(h\lambda_i)$ de $(1 - h\Lambda)^{-1}$ vérifient la condition $|R(h\lambda_i)| \leq 1$ pour tout $h > 0$. La solution numérique reste donc bornée et cette méthode implicite est bien adaptée pour résoudre ce problème raide. Remarquons que l'on obtient directement la même conclusion en utilisant la A-stabilité de la méthode d'Euler implicite.