

# **Introduction à l'Analyse Numérique**

**Ernst Hairer et Gerhard Wanner**

**Travaux Pratiques**  
**en collaboration avec**  
**Assyr Abdulle**

Université de Genève  
Section de mathématiques  
Case postale 240  
CH-1211 Genève 24

Juin 2005

# Table de Matière

<b>I</b>	<b>Intégration Numérique</b>	1
I.1	Formules de quadrature et leur ordre	1
I.2	Etude de l'erreur	5
I.3	Formules d'un ordre supérieur	9
I.4	Polynômes orthogonaux de Legendre	10
I.5	Formules de quadrature de Gauss	13
I.6	Un programme adaptatif – TEGRAL	15
I.7	L'epsilon-algorithme	18
I.8	Exercices	20
<b>II</b>	<b>Interpolation et Approximation</b>	23
II.1	Différences divisées et formule de Newton	23
II.2	Erreur de l'interpolation	26
II.3	Polynômes de Chebyshev	28
II.4	Convergence de l'interpolation	30
II.5	Influence des erreurs d'arrondi sur l'interpolation	34
II.6	Transformée de Fourier discrète (DFT)	37
II.7	Transformée cosinus discrète (DCT) et JPEG	40
II.8	Transformée de Fourier rapide (FFT)	43
II.9	Interpolation trigonométrique	44
II.10	Interpolation par fonctions spline	46
II.11	L'erreur du spline	50
II.12	Exercices	53
<b>III</b>	<b>Équations Différentielles Ordinaires</b>	57
III.1	Quelques exemples typiques	58
III.2	Méthodes de Runge-Kutta	60
III.3	Convergence des méthodes de Runge-Kutta	63
III.4	Un programme à pas variables	65
III.5	Méthodes multipas (multistep methods)	67
III.6	Étude de l'erreur locale	71
III.7	Stabilité	72
III.8	Convergence des méthodes multipas	73
III.9	Équations différentielles raides (stiff)	75
III.10	Intégration géométrique	80
III.11	Exercices	84
<b>IV</b>	<b>Systèmes d'Équations Linéaires</b>	87
IV.1	Elimination de Gauss	88
IV.2	Le choix du pivot	90
IV.3	La condition d'une matrice	93
IV.4	La stabilité d'un algorithme	95

IV.5	L'algorithme de Cholesky . . . . .	98
IV.6	Systèmes surdéterminés – méthode des moindres carrés . . . . .	100
IV.7	Décomposition QR d'une matrice . . . . .	101
IV.8	Etude de l'erreur de la méthode des moindres carrés . . . . .	104
IV.9	Exercices . . . . .	109
<b>V</b>	<b>Valeurs et Vecteurs Propres . . . . .</b>	<b>113</b>
V.1	La condition du calcul des valeurs propres . . . . .	114
V.2	La méthode de la puissance . . . . .	118
V.3	Transformation sous forme de Hessenberg (ou tridiagonale) . . . . .	119
V.4	Méthode de bisection pour des matrices tridiagonales . . . . .	121
V.5	L'itération orthogonale . . . . .	124
V.6	L' algorithme QR . . . . .	126
V.7	Exercices . . . . .	130
<b>VI</b>	<b>Méthodes Itératives – Equations Non Linéaires . . . . .</b>	<b>134</b>
VI.1	Méthode des approximations successives . . . . .	134
VI.2	Méthodes itératives pour systèmes linéaires . . . . .	136
VI.3	Méthode de Newton . . . . .	138
VI.4	Méthode de Gauss-Newton . . . . .	141
VI.5	Exercices . . . . .	146
<b>VII</b>	<b>Travaux Pratiques . . . . .</b>	<b>147</b>
VII.1	Introduction au FORTRAN 90 . . . . .	147
VII.2	Intégration par la règle du Trapèze et Simpson . . . . .	148
VII.3	Calcul de racines par la méthode de la bisection . . . . .	149
VII.4	Wegral: programme d'intégration d'ordre 8 . . . . .	149
VII.5	Interpolation . . . . .	151
VII.6	Interpolation et erreur . . . . .	152
VII.7	Résolution d'équations différentielles et calcul de la trajectoire des planètes . . . . .	154
VII.8	Décomposition LR . . . . .	156
VII.9	Décomposition QR et trajectoire d'un astéroïde . . . . .	157
VII.10	FORTRAN 90/95 . . . . .	160

## Avant-propos

Cette brochure est une révision des polycopiés distribués pendant le cours “Analyse Numérique” (2 heures par semaine) donné en 2004/2005. Elle est une version corrigée et modifiée des polycopiés distribués en 1991/92 et en 2000/2001. Durant des années, ce cours a été donné alternativement par les auteurs.

En ce lieu, nous aimerais remercier Laurent Jay, Luc Rey-Bellet, Stéphane Cirilli, Pierre Leone, Assyr Abdulle, Michael Hauth, Martin Hairer et des nombreux assistants et étudiants soit pour leur aide dans la préparation des exercices soit pour la correction des erreurs (typographiques et mathématiques).

## **Ouvres générales sur l'analyse numérique**

Il y a un grand assortiment de livres qui introduisent le sujet d'analyse numérique (voir le rayon 65 à la bibliothèque de la section de mathématiques avec plus de 400 livres). En voici quelques références. Les numéros entre crochets (p. ex. [MA 65/403]) vous permettent de trouver le livre facilement à la bibliothèque.

- K.E. Atkinson (1978): *An Introduction to Numerical Analysis*. John Wiley & Sons. [MA 65/142]
- W. Boehm & H. Pautzsch (1993): *Numerical Methods*. AK Peters. [MA 65/332]
- G. Dahlquist & A. Björck (1969): *Numeriska Metoder*. CWK Gleerup, Bokfoerlag, Lund, Sweden. Traduction anglaise: *Numerical Methods*. Prentice-Hall, 1974. [MA 65/84]
- P. Deuflhard & A. Hohmann (1993): *Numerische Mathematik I. Eine algorithmisch orientierte Einführung*. Walter de Gruyter & Co. Traduction anglaise: *Numerical analysis. A first course in scientific computation*. Walter de Gruyter & Co., 1995. [MA 65/301, MA 65/309]
- G. Evans (1995): *Practical Numerical Analysis*. John Wiley & Sons. [MA 65/374]
- W. Gautschi (1997): *Numerical Analysis. An Introduction*. Birkhäuser. [MA 65/393]
- G. Hämmerlin & K.-H. Hoffmann (1991): *Numerische Mathematik*. Zweite Auflage. Springer. [MA 65/303]
- M.J. Maron (1982): *Numerical analysis: a practical approach*. Macmillan.
- W.H. Press, B.P. Flannery, S.A. Teukolsky & W.T. Vetterling (1986): *Numerical Recipes (FORTRAN). The Art of Scientific Computing*. Cambridge Univ. Press. [MA 65/239]
- J. Rappaz & M. Picasso (1998): *Introduction à l'analyse numérique*. Presses polytechniques et universitaires romandes. [MA 65/404]
- A. Quarteroni, R. Sacco & F. Saleri (2000): *Numerical Mathematics*. Springer. [MA 65/432]
- M. Schatzman (1991): *Analyse numérique : cours et exercices pour la licence*. Paris : InterEditions. [MA E188]
- H.R. Schwarz (1986): *Numerische Mathematik*. B.G. Teubner. [MA 65/249]
- G.W. Stewart (1996): *Afternotes on Numerical Analysis*. SIAM. [MA 65/389]
- J. Stoer & R. Bulirsch (1972): *Einführung in die Numerische Mathematik*. Heidelberger Taschenbücher, Springer. Traduction anglaise: *Introduction to Numerical Analysis*. Springer, 1980. [MA 65/48]
- Ch. Überhuber (1995): *Computer-Numerik 1 und 2*. Springer.

# Chapitre I

## Intégration Numérique

Pour ses calculs en physique et en astronomie, Newton est le premier à utiliser des formules de quadrature, suivi en cela par ses successeurs anglais (Cotes 1711, Simpson 1743). Euler, dans son gigantesque traité (*Inst. Calculi Integralis* 1768, 1769, 1770, Opera XI-XIii), met toute son ingéniosité à rechercher des primitives analytiques. Cependant, de nombreuses intégrales résistent encore et toujours à l'envahisseur (exemples  $\int \frac{e^x}{x} dx$ ,  $\int \frac{dx}{\log x}$ ); de nombreux calculs en astronomie (perturbations des orbites planétaires) contraignent Gauss (1814) à intensifier la théorie des formules de quadrature. Les programmes qui ont tourné sur les premiers ordinateurs furent en grande partie les calculs d'intégrales: ces problèmes sont les plus faciles à programmer. Pour cette même raison, nous commençons par ce sujet.

**Problème.** Etant donné une fonction continue sur un intervalle borné

$$f : [a, b] \rightarrow \mathbb{R}, \quad (0.1)$$

on cherche à calculer l'intégrale

$$\int_a^b f(x) dx. \quad (0.2)$$

### Bibliographie sur ce chapitre

- P.J. Davis & P. Rabinowitz (1975): *Methods of Numerical Integration*. Academic Press, New York.  
G. Evans (1993): *Practical Numerical Integration*. John Wiley & Sons. [MA 65/336]  
V.I. Krylov (1959): *Priblizhennoe Vychislenie Integralov*. Goz. Izd. Fiz.-Mat. Lit., Moscow. Traduction anglaise: *Approximate calculation of integrals*. Macmillan, 1962. [MA 65/185]  
R. Piessens, E. de Doncker-Kapenga, C.W. Überhuber & D.K. Kahaner (1983): QUADPACK. *A Subroutine Package for Automatic Integration*. Springer Series in Comput. Math., vol. 1. [MA 65/210]  
A.H. Stroud (1974): *Numerical quadrature and Solution of Ordinary Differential Equations*. Springer. [MA 65/89]

### I.1 Formules de quadrature et leur ordre

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

$$\int_a^b f(x) dx = \sum_{j=0}^{N-1} \int_{x_j}^{x_{j+1}} f(x) dx. \quad (1.1)$$

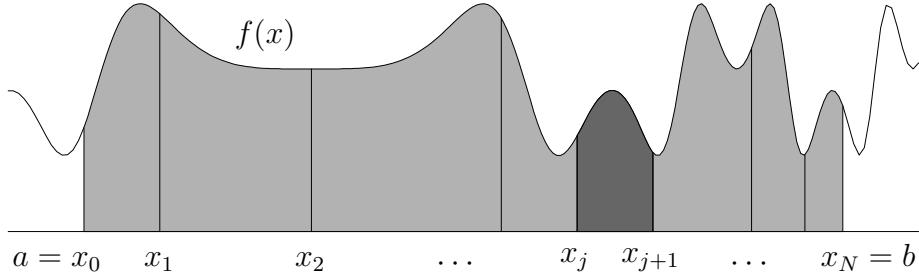


FIG. I.1: Une division d'un intervalle en sous-intervalles

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

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

Notons enfin  $g(t) := f(x_j + th_j)$ . Il reste alors à calculer une approximation de

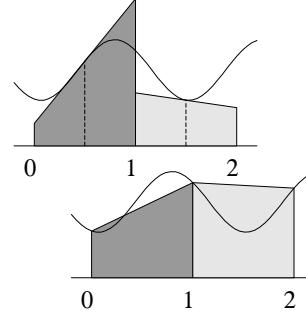
$$\int_0^1 g(t) dt. \quad (1.2)$$

**Exemples.** 1. La *formule du point milieu*

$$\int_0^1 g(t) dt \approx g(1/2).$$

2. La *formule du trapèze*

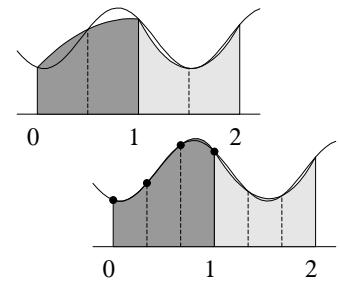
$$\int_0^1 g(t) dt \approx \frac{1}{2}(g(0) + g(1)).$$



Ces deux formules (point milieu et trapèze) sont exactes si  $g(t)$  est un polynôme de degré  $\leq 1$ .

3. On obtient la *formule de Simpson* si l'on passe une parabole (polynôme de degré 2) par les trois points  $(0, g(0)), (1/2, g(1/2)), (1, g(1))$  et si l'on approche l'intégrale (1.2) par l'aire sous la parabole:

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



4. La “pulcherrima et utilissima regula” de Newton (degré 3) :

$$\int_0^1 g(t) dt \approx \frac{1}{8}(g(0) + 3g(1/3) + 3g(2/3) + g(1)).$$

5. En généralisant cette idée (passer un polynôme de degré  $s - 1$  par les  $s$  points équidistants  $(i/(s-1), g(i/(s-1))), i = 0, \dots, s-1$ ), on obtient les *formules de Newton-Cotes* (Newton 1676, Cotes 1711). Pour  $s \leq 7$  les coefficients de ces formules sont données dans le tableau I.1. Leur dessin en figure I.2 montre que les poids “explosent” au-delà de  $s = 10$ . Si on veut augmenter la précision, il vaut mieux raffiner les subdivisions en (1.1) qu'augmenter le degré  $s$ .

**Définition 1.1** Une *formule de quadrature à  $s$  étages* est donnée par

$$\int_0^1 g(t) dt \approx \sum_{i=1}^s b_i g(c_i). \quad (1.3)$$

Les  $c_i$  sont les nœuds de la formule de quadrature et les  $b_i$  en sont les poids.

TAB. I.1: Formules de Newton-Cotes

$s$	ordre	poids $b_i$						nom
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	$\frac{19}{288}$	$\frac{75}{288}$	$\frac{50}{288}$	$\frac{50}{288}$	$\frac{75}{288}$	$\frac{19}{288}$	—
7	8	$\frac{41}{840}$	$\frac{216}{840}$	$\frac{27}{840}$	$\frac{272}{840}$	$\frac{27}{840}$	$\frac{216}{840}$	$\frac{41}{840}$
								Weddle

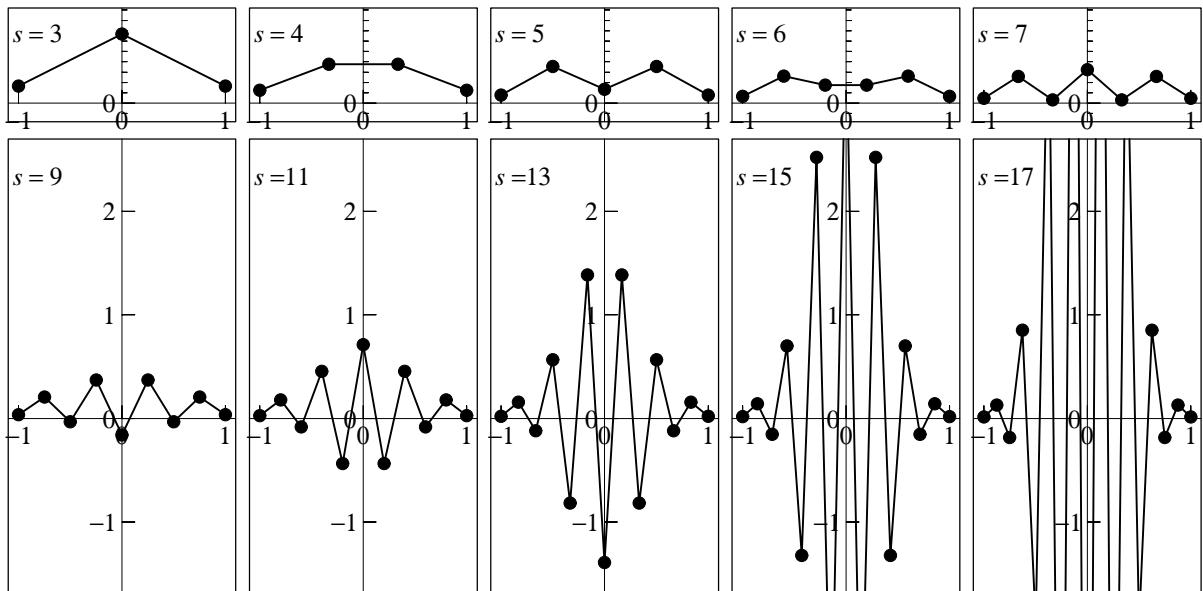


FIG. I.2: Dessin des poids des formules de Newton-Cotes

If there are four ordinates at equal intervals, let  $A$  be the sum of the first and the fourth,  $B$  the sum of the second and third, and  $R$  the interval between the first and the fourth ; then ... the area between the first and the fourth ordinates will be  $(A + 3B)R/8$ .

(I. Newton, *Methodus*, publ. 1711, cité d'après H.H. Goldstine, p. 76)

**Définition 1.2** On dit que l'ordre de la formule de quadrature (1.3) est  $p$ , si la formule est exacte pour tous les polynômes de degré  $\leq p - 1$ , c.-à-d.,

$$\int_0^1 g(t) dt = \sum_{i=1}^s b_i g(c_i) \quad \text{pour} \quad \deg g \leq p - 1. \quad (1.4)$$

On voit que les formules du point milieu et du trapèze sont d'ordre 2. La formule de Newton-Cotes à  $s$  étages a un ordre  $p \geq s$  (par définition).

**Théorème 1.3** La formule de quadrature (1.3) a un ordre  $p$  si et seulement si

$$\sum_{i=1}^s b_i c_i^{q-1} = \frac{1}{q} \quad \text{pour } q = 1, 2, \dots, p. \quad (1.5)$$

*Démonstration.* La nécessité de (1.5) est une conséquence de (1.4) si l'on pose  $g(t) = t^{q-1}$ . Pour en montrer la suffisance, on utilise le fait qu'un polynôme de degré  $p - 1$  est une combinaison linéaire de  $1, t, \dots, t^{p-1}$  et que l'intégrale  $\int_0^1 g(t) dt$  ainsi que l'expression  $\sum_{i=1}^s b_i g(c_i)$  sont linéaires en  $g$ .  $\square$

En fixant les nœuds  $c_1, \dots, c_s$  (distincts), la condition (1.5) avec  $p = s$  est un système linéaire pour  $b_1, \dots, b_s$

$$\begin{pmatrix} 1 & 1 & \dots & 1 \\ c_1 & c_2 & \dots & c_s \\ \vdots & \vdots & & \vdots \\ c_1^{s-1} & c_2^{s-1} & \dots & c_s^{s-1} \end{pmatrix} \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_s \end{pmatrix} = \begin{pmatrix} 1 \\ 1/2 \\ \vdots \\ 1/s \end{pmatrix}. \quad (1.6)$$

Comme la matrice dans (1.6) est inversible (matrice de Vandermonde), la résolution de ce système nous donne une formule de quadrature d'ordre  $p \geq s$ .

Si l'on vérifie les conditions (1.5) pour la formule de Simpson, on fait une observation intéressante. Par définition, il est évident que la condition (1.5) est satisfaite pour  $q = 1, 2, 3$ , mais on remarque qu'elle satisfait aussi (1.5) pour  $q = 4$ :

$$\begin{aligned} \frac{1}{6} \cdot 0^3 + \frac{4}{6} \cdot \left(\frac{1}{2}\right)^3 + \frac{1}{6} \cdot 1^3 &= \frac{1}{4} \\ \frac{1}{6} \cdot 0^4 + \frac{4}{6} \cdot \left(\frac{1}{2}\right)^4 + \frac{1}{6} \cdot 1^4 &= \frac{5}{24} \neq \frac{1}{5}. \end{aligned}$$

Elle est donc d'ordre 4. Par conséquent, elle n'est pas seulement exacte pour des polynômes de degré 2 mais aussi pour des polynômes de degré 3. Ceci est une propriété générale d'une formule symétrique.

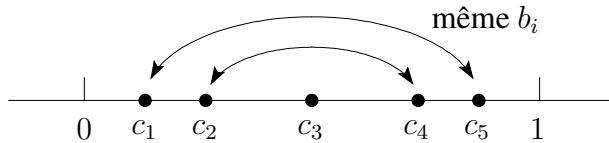


FIG. I.3: Coefficients et nœuds d'une formule de quadrature symétrique

**Théorème 1.4** Une formule de quadrature symétrique (c.-à-d.  $c_i = 1 - c_{s+1-i}$ ,  $b_i = b_{s+1-i}$  pour tout  $i$ ; voir la fig. I.3) a toujours un ordre pair. C.-à-d., si elle est exacte pour les polynômes de degré  $\leq 2m - 2$ , elle est automatiquement exacte pour les polynômes de degré  $2m - 1$ .

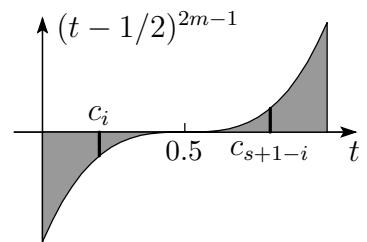
*Démonstration.* Chaque polynôme de degré  $2m - 1$  peut être écrit sous la forme

$$g(t) = C \cdot (t - 1/2)^{2m-1} + g_1(t)$$

où  $g_1(t)$  est de degré  $\leq 2m - 2$ . Il suffit alors de montrer qu'une formule symétrique est exacte pour  $(t - 1/2)^{2m-1}$ . À cause de la symétrie de cette fonction, la valeur exacte vaut

$$\int_0^1 (t - 1/2)^{2m-1} dt = 0.$$

Pour une formule de quadrature symétrique on a  $b_i(c_i - 1/2)^{2m-1} + b_{s+1-i}(c_{s+1-i} - 1/2)^{2m-1} = 0$ . Donc, l'approximation numérique de  $\int_0^1 (t - 1/2)^{2m-1} dt$  est également zéro.  $\square$



## I.2 Etude de l'erreur

Afin d'étudier l'erreur commise en approchant l'intégrale par une formule de quadrature, commençons par une *expérience numérique* :

Prenons une fonction  $f(x)$ , définie sur  $[a, b]$ , divisons l'intervalle en plusieurs sous-intervalles équidistants ( $h = (b - a)/N$ ) et appliquons une formule de quadrature du paragraphe précédent. Ensuite, étudions l'erreur (en échelle logarithmique)

$$err = \int_a^b f(x) dx - \sum_{j=0}^{N-1} h \sum_{i=1}^s b_i f(x_j + c_i h) \quad (2.1)$$

en fonction de  $fe$  (nombre d'évaluations de la fonction  $f(x)$ ; on a  $fe = N \cdot (s - 1) + 1$  pour Newton-Cotes). Le nombre  $fe$  représente une mesure pour le travail (proportionnel au temps de calcul sur un ordinateur). La fig. I.4 montre les résultats (pour  $N = 1, 2, 4, 8, 16, 32, \dots$ ) obtenus par les formules de Newton-Cotes pour les deux intégrales :

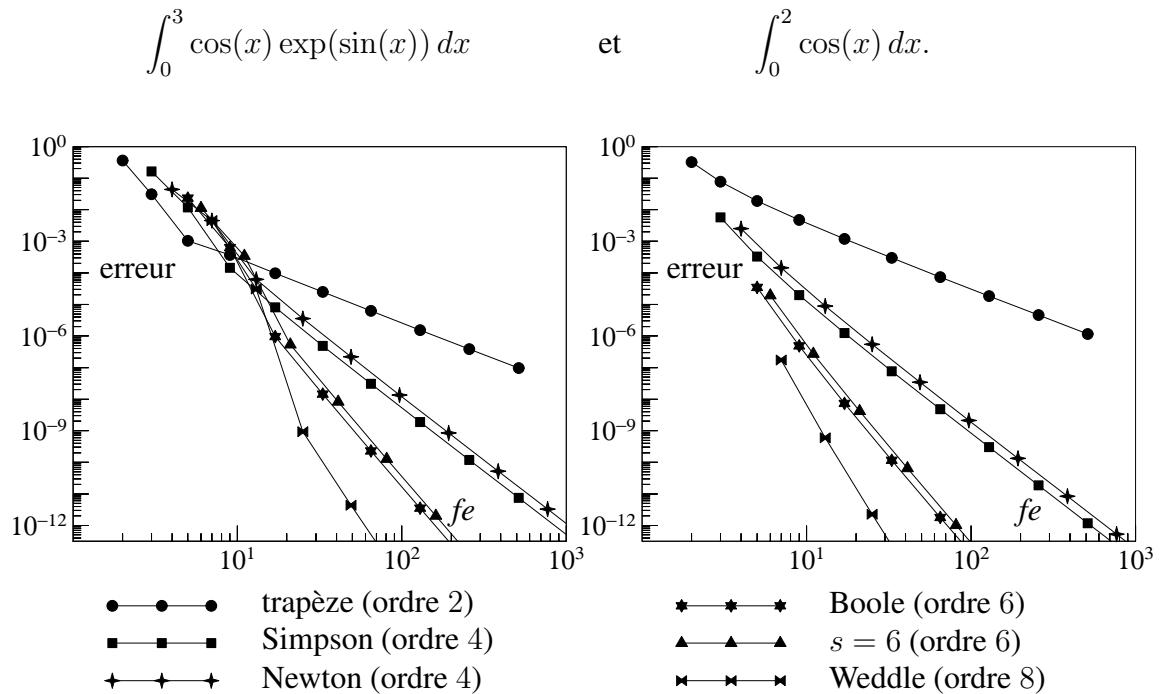


FIG. I.4: L'erreur en fonction du travail  $fe$  pour les formules de Newton-Cotes

En étudiant les résultats de la fig. I.4, nous constatons que :

- le nombre de chiffres exacts, donné par  $-\log_{10}(err)$ , dépend linéairement de  $\log_{10}(fe)$ ;
- 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 une meilleure précision.

### Explication des résultats de la fig. I.4.

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

$$\begin{aligned} E(f, x_0, h) &= \int_{x_0}^{x_0+h} f(x) dx - h \sum_{i=1}^s b_i f(x_0 + c_i h) \\ &= h \left( \int_0^1 f(x_0 + th) dt - \sum_{i=1}^s b_i f(x_0 + c_i h) \right). \end{aligned} \quad (2.2)$$

En supposant  $f$  suffisamment différentiable, on peut remplacer  $f(x_0 + th)$  et  $f(x_0 + c_i h)$  par les séries de Taylor (développées autour de  $x_0$ ), et on obtient ainsi

$$\begin{aligned} E(f, x_0, h) &= \sum_{q \geq 0} \frac{h^{q+1}}{q!} \left( \int_0^1 t^q dt - \sum_{i=1}^s b_i c_i^q \right) f^{(q)}(x_0) \\ &= \frac{h^{p+1}}{p!} \left( \frac{1}{p+1} - \sum_{i=1}^s b_i c_i^p \right) f^{(p)}(x_0) + \mathcal{O}(h^{p+2}) \end{aligned} \quad (2.3)$$

(ici, on a bien supposé que la formule de quadrature ait l'ordre  $p$  mais pas l'ordre  $p+1$ ). La constante

$$C = \frac{1}{p!} \left( \frac{1}{p+1} - \sum_{i=1}^s b_i c_i^p \right) \quad (2.4)$$

s'appelle *constante de l'erreur*. Supposons que  $h$  soit petit de manière à ce que le terme  $\mathcal{O}(h^{p+2})$  dans (2.3) soit négligeable par rapport au terme  $Ch^{p+1}f^{(p)}(x_0)$ , alors on obtient

$$err = \sum_{j=0}^{N-1} E(f, x_j, h) \approx Ch^p \sum_{j=0}^{N-1} h f^{(p)}(x_j) \approx Ch^p \int_a^b f^{(p)}(x) dx = Ch^p \left( f^{(p-1)}(b) - f^{(p-1)}(a) \right).$$

Cette formule nous permet de mieux comprendre les résultats de la fig. I.4. Comme  $err \approx C_1 \cdot h^p$  et  $fe \approx C_2/h$ , nous avons

$$\log_{10}(err) \approx \log_{10}(C_1) + p \cdot \log_{10}(h) \approx Const - p \cdot \log_{10}(fe).$$

Ceci montre la dépendance linéaire entre les quantités  $\log_{10}(err)$  et  $\log_{10}(fe)$ , et aussi le fait que la pente soit de  $-p$ .

### Estimation rigoureuse de l'erreur.

Notre but suivant est de trouver une formule exacte de l'erreur d'une formule de quadrature. Une telle estimation nous permettra de démontrer des théorèmes de convergence et assurera une certaine précision du résultat numérique.

**Théorème 2.1** *Considérons une formule de quadrature d'ordre  $p$  et un entier  $k$  satisfaisant  $k \leq p$ . Si  $f : [x_0, x_0 + h] \rightarrow \mathbb{R}$  est  $k$  fois continûment différentiable, l'erreur (2.2) vérifie*

$$E(f, x_0, h) = h^{k+1} \int_0^1 N_k(\tau) f^{(k)}(x_0 + \tau h) d\tau \quad (2.5)$$

où  $N_k(\tau)$ , le noyau de Peano, est donné par

$$N_k(\tau) = \frac{(1-\tau)^k}{k!} - \sum_{i=1}^s b_i \frac{(c_i - \tau)_+^{k-1}}{(k-1)!} \quad \text{où} \quad (\sigma)_+^{k-1} := \begin{cases} (\sigma)^{k-1} & \text{si } \sigma > 0, \\ 0 & \text{si } \sigma \leq 0. \end{cases}$$

*Démonstration.* Nous introduisons la série de Taylor avec reste<sup>1</sup>

$$f(x_0 + th) = \sum_{j=0}^{k-1} \frac{(th)^j}{j!} f^{(j)}(x_0) + h^k \int_0^t \frac{(t-\tau)^{k-1}}{(k-1)!} f^{(k)}(x_0 + \tau h) d\tau \quad (2.6)$$

dans la formule (2.2) pour  $E(f, x_0, h)$ . En utilisant

$$\int_0^t (t-\tau)^{k-1} g(\tau) d\tau = \int_0^1 (t-\tau)_+^{k-1} g(\tau) d\tau$$

<sup>1</sup>voir le paragraphe III.7 du livre de E. Hairer & G. Wanner (1995), *Analysis by Its History*. Undergraduate Texts in Mathematics, Springer.

et le fait que la partie polynomiale de (2.6) ne donne pas de contribution à l'erreur (à cause de  $p \geq k$ ), nous obtenons

$$E(f, x_0, h) = h^{k+1} \int_0^1 \left( \int_0^1 \frac{(t-\tau)_+^{k-1}}{(k-1)!} dt - \sum_{i=1}^s b_i \frac{(c_i-\tau)_+^{k-1}}{(k-1)!} \right) f^{(k)}(x_0 + \tau h) d\tau.$$

Une évaluation de l'intégrale intérieure donne le résultat.  $\square$

**Théorème 2.2 (Propriétés du noyau de Peano)** *Considérons une formule de quadrature d'ordre  $p$  et un nombre  $k$  satisfaisant  $1 \leq k \leq p$ . Alors, on a:*

- a)  $N'_k(\tau) = -N_{k-1}(\tau)$  pour  $k \geq 2$  (pour  $\tau \neq c_i$  si  $k = 2$ );
- b)  $N_k(1) = 0$  pour  $k \geq 1$  si  $c_i \leq 1$  ( $i = 1, \dots, s$ );
- c)  $N_k(0) = 0$  pour  $k \geq 2$  si  $c_i \geq 0$  ( $i = 1, \dots, s$ );
- d)  $\int_0^1 N_p(\tau) d\tau = \frac{1}{p!} \left( \frac{1}{p+1} - \sum_{i=1}^s b_i c_i^p \right) = C$  (constante de l'erreur (2.4));
- e)  $N_1(\tau)$  est linéaire par morceaux, de pente  $-1$  et avec des sauts de hauteur  $b_i$  aux points  $c_i$  ( $i = 1, \dots, s$ ).  $\square$

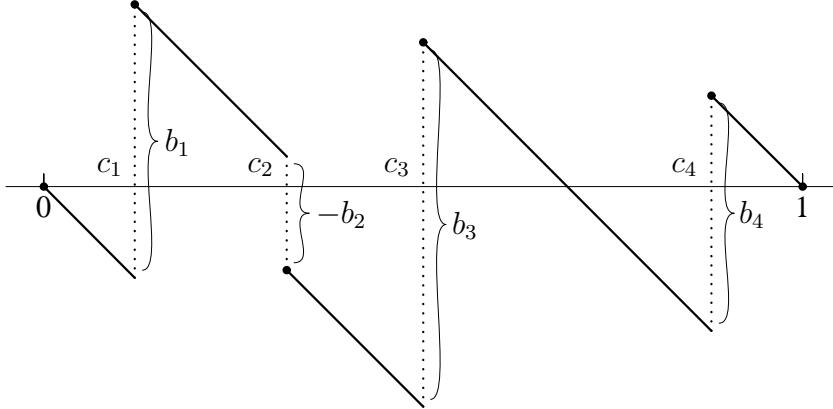


FIG. I.5: Le noyau de Peano  $N_1(\tau)$  d'une formule de quadrature

Les noyaux de Peano pour la formule du point milieu sont

$$N_1(\tau) = \begin{cases} -\tau & \text{si } \tau < 1/2 \\ 1 - \tau & \text{si } \tau \geq 1/2 \end{cases} \quad N_2(\tau) = \begin{cases} \tau^2/2 & \text{si } \tau \leq 1/2 \\ (1 - \tau)^2/2 & \text{si } \tau \geq 1/2 \end{cases}$$

(voir la fig. I.6). Pour la formule de Newton-Cotes ( $s = 5$ ), ils sont dessinés dans la fig. I.7.

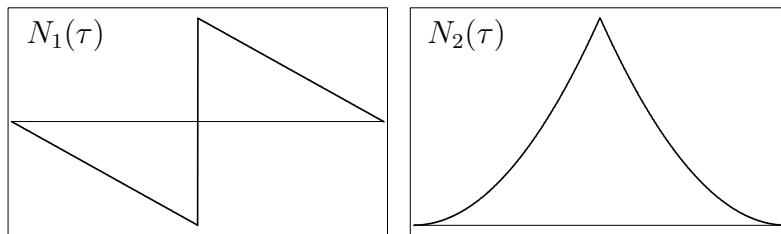
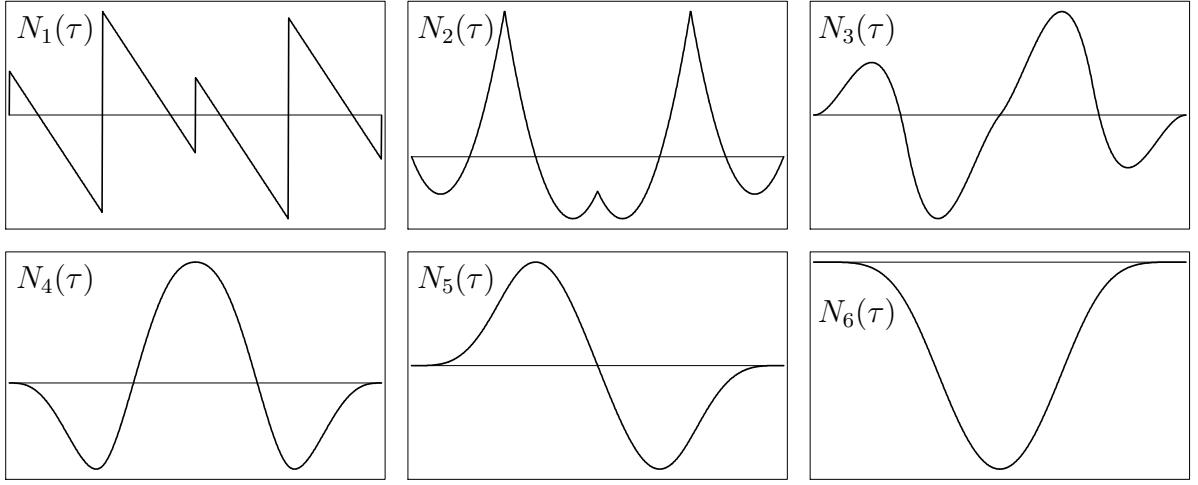


FIG. I.6: Noyaux de Peano pour la formule du point milieu

Grâce au résultat du théorème précédent, on peut facilement estimer l'erreur pour l'intervalle entier  $[a, b]$ . Pour une division arbitraire (équidistante ou non;  $h_j = x_{j+1} - x_j$ ), notons l'erreur par

$$err = \int_a^b f(x) dx - \sum_{j=0}^{N-1} h_j \sum_{i=1}^s b_i f(x_j + c_i h_j). \quad (2.7)$$

FIG. I.7: Noyaux de Peano pour la formule de Newton-Cotes avec  $s = 5$ 

**Théorème 2.3** Soit  $f : [a, b] \rightarrow \mathbb{R}$   $k$  fois continûment différentiable et soit l'ordre de la formule de quadrature égal à  $p$  ( $p \geq k$ ). Alors, l'erreur (2.7) admet l'estimation

$$|err| \leq h^k \cdot (b - a) \cdot \int_0^1 |N_k(\tau)| d\tau \cdot \max_{x \in [a, b]} |f^{(k)}(x)| \quad (2.8)$$

où  $h = \max_j h_j$ .

*Démonstration.* La formule (2.5) donne

$$\begin{aligned} |E(f, x_0, h)| &\leq h^{k+1} \int_0^1 |N_k(\tau)| \cdot |f^{(k)}(x_0 + \tau h)| d\tau \\ &\leq h^{k+1} \int_0^1 |N_k(\tau)| d\tau \cdot \max_{x \in [x_0, x_0 + h]} |f^{(k)}(x)|. \end{aligned}$$

Comme l'erreur (2.7) est la somme des erreurs sur les sous-intervalles de la division, on obtient

$$\begin{aligned} |err| &\leq \sum_{j=0}^{N-1} |E(f, x_j, h_j)| \leq \sum_{j=0}^{N-1} \underbrace{h_j^{k+1}}_{\leq h^k \cdot h_j} \cdot \int_0^1 |N_k(\tau)| d\tau \cdot \underbrace{\max_{x \in [x_j, x_{j+1}]} |f^{(k)}(x)|}_{\leq \max_{x \in [a, b]} |f^{(k)}(x)|} \end{aligned}$$

ce qui montre l'assertion (2.8), car  $\sum_{j=0}^{N-1} h_j = b - a$ . □

**Exemples.** Pour la formule du point milieu, on a

$$|err| \leq h^2 \cdot (b - a) \cdot \frac{1}{24} \cdot \max_{x \in [a, b]} |f''(x)|;$$

pour la formule du trapèze

$$|err| \leq h^2 \cdot (b - a) \cdot \frac{1}{12} \cdot \max_{x \in [a, b]} |f''(x)|;$$

pour la formule de Simpson

$$|err| \leq h^4 \cdot (b - a) \cdot \frac{1}{2880} \cdot \max_{x \in [a, b]} |f^{(4)}(x)|;$$

pour la formule de Newton-Cotes ( $s = 5$ )

$$|err| \leq h^6 \cdot (b - a) \cdot \frac{1}{1935360} \cdot \max_{x \in [a,b]} |f^{(6)}(x)|.$$

Le calcul de  $\int_0^1 |N_p(\tau)| d\tau$  pour ces formules n'est pas difficile. Considérons par exemple la formule de Newton-Cotes ( $s = 5$ ). On constate que  $N_6(\tau)$  ne change pas de signe sur  $[0, 1]$  (voir fig. I.7) et on utilise la propriété (d) du théorème 2.2. Ceci donne

$$\int_0^1 |N_6(\tau)| d\tau = \left| \int_0^1 N_6(\tau) d\tau \right| = \frac{1}{6!} \left| \frac{1}{7} - \left( \frac{32}{90} \left( \frac{1}{4} \right)^6 + \frac{12}{90} \left( \frac{1}{2} \right)^6 + \frac{32}{90} \left( \frac{3}{4} \right)^6 + \frac{7}{90} 1^6 \right) \right| = \frac{1}{1935360}.$$

### I.3 Formules d'un ordre supérieur

Aber Gauss hat in den Göttinger Commentarien gezeigt, dass man durch schickliche Wahl der Abscissen, für welche die Ordinaten berechnet werden, den Grad der Näherung auf das Doppelte treiben kann ; ... Die grosse Einfachheit und Eleganz der Gaussischen Resultate, lässt einen einfachen Weg vermuten. (Jacobi 1826, *Crelle J.* 1, p. 302)

Si l'on fixe les nœuds  $c_1, \dots, c_s$  (distincts), il existe une formule de quadrature  $(b_i, c_i)$  unique, ayant un ordre  $p \geq s$ . On obtient les poids  $b_i$  soit par la résolution du système linéaire (1.6), soit par la formule de l'exercice 1.

*Question.* Y a-t-il un choix des  $c_i$  permettant d'avoir un ordre supérieur?

**Théorème 3.1** Soit  $(b_i, c_i)_{i=1}^s$  une formule de quadrature d'ordre  $p \geq s$  et soit

$$M(t) = (t - c_1) \cdot \dots \cdot (t - c_s). \quad (3.1)$$

Alors, l'ordre est  $\geq s + m$  si et seulement si

$$\int_0^1 M(t)g(t) dt = 0 \quad \text{pour tout polynôme } g(t) \text{ de degré } \leq m - 1. \quad (3.2)$$

*Démonstration.* Soit  $f(t)$  un polynôme de degré  $\leq s + m - 1$ . L'idée, due à Jacobi (1826), est de diviser  $f(t)$  par  $M(t)$  et d'écrire  $f(t)$  sous la forme

$$f(t) = M(t)g(t) + r(t)$$

où  $\deg r \leq s - 1$  et  $\deg g \leq m - 1$ . Alors, l'intégrale exacte et l'approximation numérique satisfont

$$\begin{aligned} \int_0^1 f(t) dt &= \int_0^1 M(t)g(t) dt + \int_0^1 r(t) dt \\ \sum_{i=1}^s b_i f(c_i) &= \sum_{i=1}^s b_i \underbrace{M(c_i)}_0 g(c_i) + \sum_{i=1}^s b_i r(c_i). \end{aligned}$$

Comme la formule de quadrature est exacte pour  $r(t)$  (l'ordre est  $\geq s$  par hypothèse), elle est exacte pour  $f(t)$  si et seulement si  $\int_0^1 M(t)g(t) dt = 0$ . □

**Exemple 3.2** Pour qu'une formule de quadrature à  $s = 3$  étages ait un ordre  $\geq 4$ , il faut que

$$0 = \int_0^1 (t - c_1)(t - c_2)(t - c_3) dt = \frac{1}{4} - (c_1 + c_2 + c_3)\frac{1}{3} + (c_1c_2 + c_1c_3 + c_2c_3)\frac{1}{2} - c_1c_2c_3,$$

ce qui est équivalent à

$$c_3 = \frac{1/4 - (c_1 + c_2)/3 + c_1c_2/2}{1/3 - (c_1 + c_2)/2 + c_1c_2}.$$

**Exemple 3.3** Continuons l'étude de formules de quadrature à  $s = 3$  étages et essayons de déterminer les  $c_1, c_2, c_3$  pour que l'ordre soit  $p = 6$ . Par le théorème 3.1, il faut que

$$\begin{aligned} c_1c_2c_3 - \frac{1}{2}(c_1c_2 + c_1c_3 + c_2c_3) + \frac{1}{3}(c_1 + c_2 + c_3) &= \frac{1}{4} \\ \frac{1}{2}c_1c_2c_3 - \frac{1}{3}(c_1c_2 + c_1c_3 + c_2c_3) + \frac{1}{4}(c_1 + c_2 + c_3) &= \frac{1}{5} \\ \frac{1}{3}c_1c_2c_3 - \frac{1}{4}(c_1c_2 + c_1c_3 + c_2c_3) + \frac{1}{5}(c_1 + c_2 + c_3) &= \frac{1}{6} \end{aligned} \quad (3.3)$$

Ce système est non linéaire en  $c_1, c_2, c_3$  et paraît difficile à résoudre. Par contre, il est linéaire en  $\sigma_1 = c_1 + c_2 + c_3$ ,  $\sigma_2 = c_1c_2 + c_1c_3 + c_2c_3$  et  $\sigma_3 = c_1c_2c_3$ , qui sont les coefficients du polynôme

$$M(t) = (t - c_1)(t - c_2)(t - c_3) = t^3 - \sigma_1t^2 + \sigma_2t - \sigma_3.$$

En résolvant le système (3.3) pour  $\sigma_1, \sigma_2, \sigma_3$ , on obtient  $\sigma_1 = 3/2$ ,  $\sigma_2 = 3/5$  et  $\sigma_3 = 1/20$ , et donc

$$M(t) = t^3 - \frac{3}{2}t^2 + \frac{3}{5}t - \frac{1}{20} = \left(t - \frac{1}{2}\right)\left(t - \frac{5 - \sqrt{15}}{10}\right)\left(t - \frac{5 + \sqrt{15}}{10}\right).$$

Par chance, le polynôme  $M(t)$  ne possède que des racines réelles. Elles sont toutes dans l'intervalle  $(0, 1)$ , ce que nous convient. Avec les poids  $b_i$ , obtenues par le système linéaire (1.6), nous avons donc trouvé une formule de quadrature d'ordre  $p = 6$  avec seulement  $s = 3$  étages:

$$\int_0^1 g(t) dt \approx \frac{5}{18}g\left(\frac{5 - \sqrt{15}}{10}\right) + \frac{8}{18}g\left(\frac{1}{2}\right) + \frac{5}{18}g\left(\frac{5 + \sqrt{15}}{10}\right).$$

**Théorème 3.4** Si  $p$  est l'ordre d'une formule de quadrature à  $s$  étages, alors

$$p \leq 2s. \quad (3.4)$$

*Démonstration.* Supposons, par l'absurde, que l'ordre satisfasse  $p \geq 2s + 1$ . Alors, l'intégrale dans (3.2) est nulle pour tout polynôme  $g(t)$  de degré  $\leq s$ . Ceci contredit le fait que

$$\int_0^1 M(t)M(t) dt = \int_0^1 (t - c_1)^2 \cdot \dots \cdot (t - c_s)^2 dt > 0. \quad \square$$

## I.4 Polynômes orthogonaux de Legendre

Pour construire une formule de quadrature d'ordre  $2s$  avec  $s = 4, 5, \dots$ , on peut en principe faire le même calcul que dans l'exemple 3.3. Toutefois, l'approche avec les polynômes de Legendre simplifie les calculs, fournit des formules simples pour  $M(t)$  et donne beaucoup de compréhension pour les formules de quadrature d'un ordre optimal.

Pour rendre les formules plus simples (et symétriques), nous faisons le changement de variable  $\tau = 2t - 1$  qui transforme l'intervalle  $[0, 1]$  pour  $t$  en l'intervalle  $[-1, 1]$  pour  $\tau$ .

**Problème.** Trouver, pour chaque entier positif  $k$ , un polynôme  $P_k(\tau)$  de degré  $k$  tel que

$$\int_{-1}^1 P_k(\tau)g(\tau) d\tau = 0 \quad \text{si } \deg g \leq k-1. \quad (4.1)$$

On sait que les fonctions  $X_n$  et  $Y_n$ , introduites dans l'analyse par Legendre, sont d'un très-grand secours dans plusieurs théories importantes, en particulier dans la théorie de l'attraction des sphéroïdes et dans celle de la figure des planètes ; ...

(O. Bonnet, *J.d.math.* vol. 17, 1852, p. 265)

Ces polynômes ont été introduits en 1785 par Legendre et sont d'une importance dépassant largement les formules de quadrature (voir citation). On les appelle *polynômes orthogonaux* car  $\langle P_k, P_j \rangle = 0$  pour  $k \neq j$ , où

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

est un produit scalaire sur l'espace vectoriel des polynômes à coefficients réels. Le polynôme  $P_s(2t - 1)$  jouera le rôle de  $M(t)$  dans les théorème 3.1 pour les formules de quadrature d'ordre  $p = 2s$ .

**Théorème 4.1 (formule de Rodrigues)** *Le polynôme  $P_k(\tau)$ , défini par*

$$P_k(\tau) = \frac{1}{2^k \cdot k!} \frac{d^k}{d\tau^k} ((\tau^2 - 1)^k), \quad (4.2)$$

satisfait la condition (4.1). La constante (de mormalisation) est choisie pour avoir  $P_k(1) = 1$ .

*Démonstration.* Soit  $g(\tau)$  un polynôme de degré  $\leq k-1$ . Il suffit de montrer que le polynôme, défini par (4.2), satisfait  $\langle P_k, g \rangle = 0$ . Plusieurs intégrations par parties donnent

$$\begin{aligned} \langle P_k, g \rangle &= C_k \int_{-1}^1 \frac{d^k}{d\tau^k} ((\tau^2 - 1)^k) \cdot g(\tau) d\tau \\ &= C_k \underbrace{\frac{d^{k-1}}{d\tau^{k-1}} ((\tau^2 - 1)^k) \cdot g(\tau) \Big|_{-1}^1}_{= 0} - C_k \int_{-1}^1 \frac{d^{k-1}}{d\tau^{k-1}} ((\tau^2 - 1)^k) \cdot g'(\tau) d\tau \\ &= \dots = (-1)^k \cdot C_k \int_{-1}^1 (\tau^2 - 1)^k \cdot g^{(k)}(\tau) d\tau = 0, \end{aligned}$$

car  $g^{(k)}(\tau) = 0$ , ce qui démontre l'affirmation (4.1).

En considérant la série de Taylor de  $f(\tau) = (\tau - 1)^k g(\tau)$  autour de  $\tau = 1$ , on voit que  $f^{(k)}(1)/k! = g(1)$ . Ceci nous permet d'évaluer la dérivée dans (4.2) au point  $\tau = 1$ .  $\square$

Les premiers de ces polynômes sont

$$\begin{aligned} P_0(\tau) &= 1, & P_1(\tau) &= \tau, & P_2(\tau) &= \frac{3}{2}\tau^2 - \frac{1}{2}, & P_3(\tau) &= \frac{5}{2}\tau^3 - \frac{3}{2}\tau \\ P_4(\tau) &= \frac{35}{8}\tau^4 - \frac{30}{8}\tau^2 + \frac{3}{8}, & P_5(\tau) &= \frac{63}{8}\tau^5 - \frac{70}{8}\tau^3 + \frac{15}{8}\tau. \end{aligned} \quad (4.3)$$

Ils sont dessinés dans la fig. I.8 et sont alternativement des fonctions paires et impaires :

$$\begin{aligned} P_k(\tau) &= P_k(-\tau) & \text{si } k \text{ est pair} \\ P_k(\tau) &= -P_k(-\tau) & \text{si } k \text{ est impair.} \end{aligned} \quad (4.4)$$

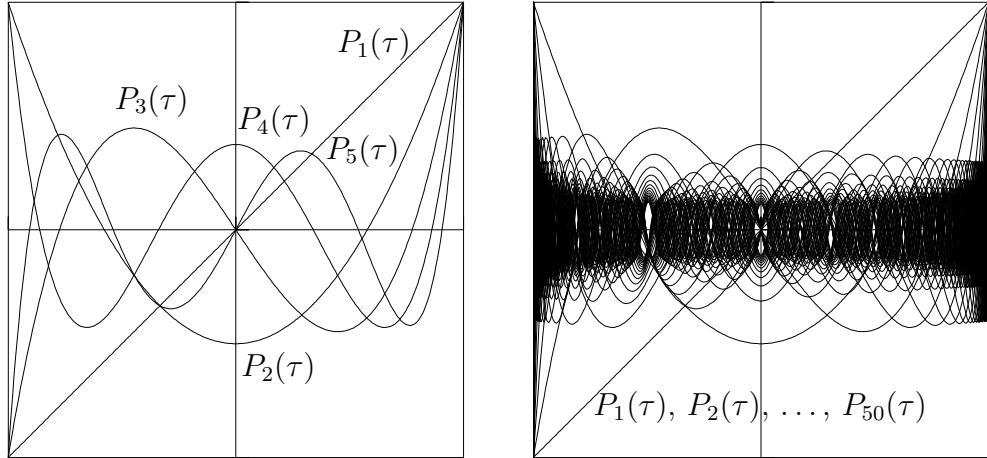


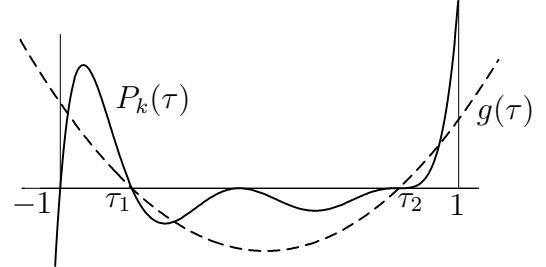
FIG. I.8: Polynômes de Legendre

**Théorème 4.2** *Toutes les racines de \$P\_k(\tau)\$ sont réelles, simples et dans l'intervalle ouvert \$(-1, 1)\$.*

*Démonstration.* Notons par \$\tau\_1, \dots, \tau\_r\$ les racines de \$P\_k(\tau)\$ qui sont réelles, dans \$(-1, 1)\$ et où \$P\_k(\tau)\$ change de signe. Le but est de montrer \$r = k\$. Supposons, par l'absurde, que \$r < k\$. Avec le polynôme \$g(\tau) = (\tau - \tau\_1) \cdot \dots \cdot (\tau - \tau\_r)\$ de degré \$r < k\$, on a

$$0 \neq \int_{-1}^1 \underbrace{P_k(\tau)}_{\text{ne change pas}} g(\tau) d\tau = \langle P_k, g \rangle = 0,$$

de signe sur \$(-1, 1)\$



d'où la contradiction. □

**Théorème 4.3 (formule de récurrence)** *Les polynômes de Legendre satisfont pour \$k \geq 1\$*

$$(k+1) P_{k+1}(\tau) = (2k+1) \tau P_k(\tau) - k P_{k-1}(\tau). \quad (4.5)$$

*Démonstration.* Si on multiplie \$P\_k(\tau)\$ par \$\tau\$, on obtient un polynôme de degré \$k+1\$ qui possède les mêmes puissances de \$\tau\$ que \$P\_{k+1}(\tau)\$. On peut donc soustraire, avec un facteur \$a\$ bien choisi, pour faire disparaître le terme \$\tau^{k+1}\$. Le prochain terme, \$\tau^{k-1}\$, est éliminé par un multiple de \$P\_{k-1}(\tau)\$; le terme suivant \$\tau^{k-3}\$ par un multiple de \$P\_{k-3}(\tau)\$ etc. Donc on peut écrire

$$\tau P_k(\tau) = a \cdot P_{k+1}(\tau) + b \cdot P_{k-1}(\tau) + c \cdot P_{k-3}(\tau) + d \cdot P_{k-5}(\tau) + \dots \quad (4.6)$$

Grande surprise: tous les coefficients \$c, d, \dots\$ sont nuls ! Pour voir cela, multiplions l'équation (4.6) par \$P\_{k-3}(\tau)\$ et intégrons le tout de \$-1\$ à \$1\$. Par orthogonalité, tous les termes vont s'annuler, p.ex.,

$$\int_{-1}^1 \tau P_k(\tau) \cdot P_{k-3}(\tau) d\tau = \int_{-1}^1 P_k(\tau) \cdot \underbrace{\tau \cdot P_{k-3}(\tau)}_{g(\tau)} d\tau = 0,$$

sauf le terme \$\int\_{-1}^1 (P\_{k-3}(\tau))^2 d\tau > 0\$, et \$c\$ doit être zéro. La même chose s'applique à \$d, e\$ etc.

En comparant dans (4.6) le coefficient de \$\tau^{k+1}\$ avec le terme dominant de (4.2), on trouve

$$a = \frac{k+1}{2k+1} \quad \text{et en posant } t = 1, \text{ on trouve} \quad b = 1 - a = \frac{k}{2k+1}.$$

□

## I.5 Formules de quadrature de Gauss

Dans ce paragraphe, nous construisons des formules de quadrature ayant un ordre  $p = 2s$ . Avec

$$M(t) = C \cdot P_s(2t - 1), \quad (5.1)$$

où  $P_s(\tau)$  est le polynôme de Legendre de degré  $s$ , nous avons

$$\int_0^1 P_s(2t - 1)g(2t - 1) dt = \frac{1}{2} \int_{-1}^1 P_s(\tau)g(\tau) d\tau = 0 \quad \text{si } \deg g \leq s - 1.$$

Toutes les racines de  $P_s(2t - 1)$  sont réelles et situées dans l'intervalle ouvert  $(0, 1)$  (théorème 4.2). Alors, le théorème 3.1 nous donne le résultat suivant.

**Théorème 5.1 (Gauss 1814)** *Pour chaque entier positif  $s$ , il existe une formule de quadrature unique à  $s$  étages d'ordre  $p = 2s$ . Elle est donnée par:*

$c_1, \dots, c_s$  sont les racines de  $P_s(2t - 1)$ ;

$b_1, \dots, b_s$  sont donnés par (1.6). □

Pour de petites valeurs de  $s$ , les *formules de Gauss* sont faciles à obtenir: il suffit de calculer les racines de (4.3) et de résoudre le système (1.6) tout en exploitant la symétrie de la formule. Pour  $s \leq 5$ , on obtient ainsi:

$$\begin{aligned} s = 1 : \quad & \int_0^1 g(t) dt \approx g\left(\frac{1}{2}\right) \quad (\text{formule du point milieu}) \\ s = 2 : \quad & \int_0^1 g(t) dt \approx \frac{1}{2}g\left(\frac{1}{2} - \frac{\sqrt{3}}{6}\right) + \frac{1}{2}g\left(\frac{1}{2} + \frac{\sqrt{3}}{6}\right) \\ s = 3 : \quad & \int_0^1 g(t) dt \approx \frac{5}{18}g\left(\frac{1}{2} - \frac{\sqrt{15}}{10}\right) + \frac{8}{18}g\left(\frac{1}{2}\right) + \frac{5}{18}g\left(\frac{1}{2} + \frac{\sqrt{15}}{10}\right) \\ s = 4 : \quad & \int_0^1 g(t) dt \approx \mu g\left(\frac{1}{2} - \delta\right) + \mu' g\left(\frac{1}{2} - \delta'\right) + \mu' g\left(\frac{1}{2} + \delta'\right) + \mu g\left(\frac{1}{2} + \delta\right) \\ s = 5 : \quad & \int_0^1 g(t) dt \approx \nu g\left(\frac{1}{2} - \epsilon\right) + \nu' g\left(\frac{1}{2} - \epsilon'\right) + \frac{64}{225}g\left(\frac{1}{2}\right) + \nu' g\left(\frac{1}{2} + \epsilon'\right) + \nu g\left(\frac{1}{2} + \epsilon\right) \\ \delta = \frac{1}{2} \sqrt{\frac{15 + 2\sqrt{30}}{35}}, \quad & \delta' = \frac{1}{2} \sqrt{\frac{15 - 2\sqrt{30}}{35}}, \quad \mu = \frac{1}{4} - \frac{\sqrt{30}}{72}, \quad \mu' = \frac{1}{4} + \frac{\sqrt{30}}{72}, \\ \epsilon = \frac{1}{2} \sqrt{\frac{35 + 2\sqrt{70}}{63}}, \quad & \epsilon' = \frac{1}{2} \sqrt{\frac{35 - 2\sqrt{70}}{63}}, \quad \nu = \frac{322 - 13\sqrt{70}}{1800}, \quad \nu' = \frac{322 + 13\sqrt{70}}{1800}. \end{aligned}$$

**Expérience numérique.** Après avoir trouvé de formules optimales, nous sommes intéressés à refaire les calculs pour les intégrales  $\int_0^3 \cos(x) \exp(\sin(x)) dx$  et  $\int_0^2 \cos(x) dx$  de la figure I.4. Les résultats correspondants peuvent être admirés en figure I.9 ; ils montrent une claire amélioration.

### Calcul des coefficients pour $s$ grand

Si  $s$  est grand (disons  $s \geq 10$ ), le calcul exact des racines de  $P_s(\tau)$  n'est pas toujours possible et la résolution exacte du système (1.6) peut poser des problèmes. Décrivons alors leur calcul pratique.

*Calcul de nœuds.* En utilisant la formule de récurrence (4.5), on peut facilement calculer la valeur de  $P_s(\tau)$  pour un  $\tau$  donné. Le calcul des racines  $\gamma_1, \dots, \gamma_s$  du polynôme  $P_s(\tau)$  peut alors être fait par bisection (voir exercice 12), et on obtient les nœuds de la formule de Gauss à l'aide de  $c_i = (1 + \gamma_i)/2$ .

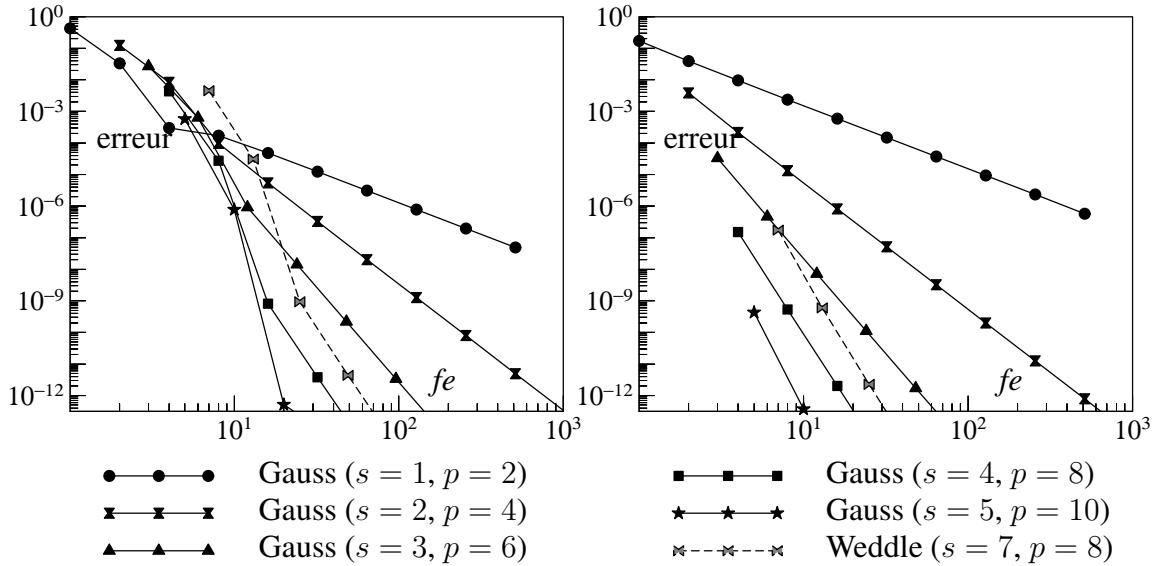


FIG. I.9: L'erreur en fonction du travail  $fe$  pour les formules de Gauss (en traitillés sont répétés les résultats pour Weddle de la figure I.4)

*Calcul de poids.* Au lieu de résoudre le système (1.6), on peut aussi utiliser la formule explicite

$$b_i = \frac{1}{(1 - \gamma_i^2)(P_s'(\gamma_i))^2} = \frac{1 - \gamma_i^2}{s^2(P_{s-1}(\gamma_i))^2}, \quad (5.2)$$

qui est donnée sans démonstration (voir M. Abramowitz & I.A. Stegun, Handbook of Mathematical Functions, page 887). La deuxième identité de (5.2) est une conséquence de l'exercice 11.

### Formules de Lobatto

Un avantage de la formule de Simpson est le fait que  $c_1 = 0$  et  $c_s = 1$ . Le noeud pour  $c_s = 1$  coïncide avec le noeud pour  $c_1 = 0$  du *pas suivant*. En mettant ces valeurs ensemble, on peut ainsi faire une petite économie.

**Problème.** Trouver des formules de quadrature d'ordre maximal à condition que  $c_1 = 0$  et  $c_s = 1$ . La réponse, due à R. Lobatto (1852), et redécouverte par R. Radau (1880), est : on pose

$$M(t) = P_s(2t - 1) - P_{s-2}(2t - 1), \quad (5.3)$$

et on prend pour  $c_1, \dots, c_s$  les racines de ce polynôme. Comme  $M(t)$  est orthogonal à tout polynôme de degré  $s - 3$ , nous obtenons une formule de quadrature d'ordre  $2s - 2$ . À cause de  $P_s(1) = 1$  et de  $P_s(-1) = (-1)^s$ , on aura toujours  $M(0) = M(1) = 0$ . Les racines restantes de  $M(t)$  sont réelles, simples et à l'intérieur de  $(0, 1)$ . Pour prouver ceci, on adapte la démonstration du théorème 4.2. Nous avons alors :

**Théorème 5.2** *Pour chaque entier  $s \geq 2$  il existe une formule de quadrature à  $s$  noeuds d'ordre  $2s - 2$  satisfaisant  $c_1 = 0$  et  $c_s = 1$ .*  $\square$

Des cas particuliers sont la formule du trapèze ( $s = 2$ ) et la formule de Simpson ( $s = 3$ ). Pour  $s = 4$  et  $s = 5$ , on obtient

$$\int_0^1 g(t) dt \approx \frac{1}{12}g(0) + \frac{5}{12}g\left(\frac{1}{2} - \frac{\sqrt{5}}{10}\right) + \frac{5}{12}g\left(\frac{1}{2} + \frac{\sqrt{5}}{10}\right) + \frac{1}{12}g(1) \text{ (ordre 6)}$$

$$\int_0^1 g(t) dt \approx \frac{1}{20}g(0) + \frac{49}{180}g\left(\frac{1}{2} - \frac{\sqrt{21}}{14}\right) + \frac{16}{45}g\left(\frac{1}{2}\right) + \frac{49}{180}g\left(\frac{1}{2} + \frac{\sqrt{21}}{14}\right) + \frac{1}{20}g(1) \text{ (ordre 8).}$$

Une comparaison des formules de Gauss et de Lobatto (du même ordre) montre une légère supériorité des formules de Gauss.

## I.6 Un programme adaptatif – TEGRAL

Posons-nous le problème d'écrire un programme

FUNCTION TEGRAL ( FCN, A, B, TOL )

qui, pour une fonction  $f : [a, b] \rightarrow \mathbb{R}$  donnée, calcule la valeur de  $\int_a^b f(x) dx$  à une précision relative de TOL. Si l'on fixe la formule de quadrature (par exemple la formule de Gauss d'ordre 30 avec  $s = 15$ ), il faut trouver une division  $\Delta = \{a = x_0 < x_1 < \dots < x_N = b\}$  de l'intervalle  $(a, b)$  afin que l'approximation numérique  $I_\Delta$  satisfasse

$$\left| I_\Delta - \int_a^b f(x) dx \right| \leq \text{TOL} \cdot \int_a^b |f(x)| dx. \quad (6.1)$$

Pour une fonction  $f$  ne changeant pas de signe sur  $(a, b)$ , la condition (6.1) signifie que *l'erreur relative* est bornée par TOL. On a mis la valeur absolue sous l'intégrale de droite pour éviter des ennuis dans le cas où  $\int_a^b f(x) dx$  est très petit ou nul.

Pour écrire un tel programme, on est confronté aux deux problèmes suivants:

- choix de la division pour que (6.1) soit satisfait;
- estimation de l'erreur  $I_\Delta - \int_a^b f(x) dx$ .

### Détermination de la division

Pour un sous-intervalle  $(x_0, x_0 + h)$  de  $(a, b)$ , on sait calculer les valeurs

$$\text{res}(x_0, x_0 + h) = h \sum_{i=1}^s b_i f(x_0 + c_i h), \quad (6.2)$$

$$\text{resabs}(x_0, x_0 + h) = h \sum_{i=1}^s b_i |f(x_0 + c_i h)|. \quad (6.3)$$

Supposons, pour le moment, qu'on connaisse aussi une estimation de l'erreur

$$\text{err}(x_0, x_0 + h) \approx \text{res}(x_0, x_0 + h) - \int_{x_0}^{x_0+h} f(x) dx. \quad (6.4)$$

L'algorithme pour trouver une division convenable est le suivant:

- on calcule  $\text{res}(a, b)$ ,  $\text{resabs}(a, b)$  et  $\text{err}(a, b)$ . Si

$$|\text{err}(a, b)| \leq \text{TOL} \cdot \text{resabs}(a, b),$$

on accepte  $\text{res}(a, b)$  comme approximation de  $\int_a^b f(x) dx$  et on arrête le calcul; sinon

- on subdivise  $(a, b)$  en deux sous-intervalles  $I_1 = (a, (a+b)/2)$  et  $I_2 = ((a+b)/2, b)$  et on calcule  $\text{res}(I_1)$ ,  $\text{resabs}(I_1)$ ,  $\text{err}(I_1)$  et  $\text{res}(I_2)$ ,  $\text{resabs}(I_2)$ ,  $\text{err}(I_2)$ . On pose  $N = 2$  et on regarde si

$$\sum_{j=1}^N |\text{err}(I_j)| \leq \text{TOL} \cdot \left( \sum_{j=1}^N \text{resabs}(I_j) \right). \quad (6.5)$$

Si (6.5) est vérifié, on accepte  $\text{res}(I_1) + \text{res}(I_2)$  comme approximation de  $\int_a^b f(x) dx$ ; sinon

- on pose  $N := N + 1$  et on subdivise l'intervalle où l'erreur est maximale (disons  $I_k$ ) en deux sous-intervalles équidistants qu'on denote par  $I_k$  et  $I_{N+1}$ . Ensuite, on calcule  $\text{res}$ ,  $\text{resabs}$  et  $\text{err}$  pour ces deux intervalles. Si (6.5) est vérifié, on arrête le calcul et on accepte

$$\sum_{j=1}^N \text{res}(I_j) \approx \int_a^b f(x) dx \quad (6.6)$$

comme approximation de l'intégrale; sinon on répète la partie (iii) de cet algorithme.

## Estimation de l'erreur (6.4)

Malheureusement, les formules pour l'erreur, obtenues dans le paragraphe I.2, ne sont pas très utiles pour un programme général, car on ne connaît que très rarement la  $p^{\text{ème}}$  dérivée de la fonction  $f(x)$  (dans notre situation  $p = 30$ ).

L'idée est d'appliquer une deuxième formule de quadrature  $(\hat{b}_i, \hat{c}_i)_{i=1}^{\hat{s}}$  et d'utiliser la différence de deux approximations numériques comme estimation de l'erreur du moins bon résultat. Pour que le travail supplémentaire soit négligeable, on suppose  $\hat{s} \leq s$  et on reprend les mêmes évaluations de  $f$ , c.-à-d., on suppose  $\hat{c}_i = c_i$  pour tous  $i$ . Une telle formule de quadrature s'appelle *formule emboîtée*, si pour au moins un indice  $i$  on a  $\hat{b}_i \neq b_i$ .

*Remarque.* Si  $(b_i, c_i)_{i=1}^s$  est une formule de quadrature d'ordre  $p \geq s$ , l'ordre d'une formule emboîtée est  $\hat{p} \leq s - 1$ . Ce résultat découle du fait que, pour une formule de quadrature d'ordre  $\geq s$ , les poids  $b_i$  sont uniquement déterminés par ses nœuds  $c_i$ .

Pour la formule de Gauss ( $s = 15$ ,  $p = 30$ ), on obtient une formule emboîtée  $(\hat{b}_i, c_i)_{i=1}^s$  d'ordre 14 en enlevant le point milieu  $c_8 = 1/2$  (voir fig. I.10), c.-à-d., on pose  $\hat{b}_8 = 0$ . L'expression calculable

$$\text{ERR1} := h \sum_{i=1}^s b_i f(x_0 + c_i h) - h \sum_{i=1}^s \hat{b}_i f(x_0 + c_i h) \approx C_1 h^{15} \quad (6.7)$$

est une approximation de l'erreur de la formule emboîtée, car

$$\begin{aligned} \text{ERR1} &= \underbrace{\left( h \sum_{i=1}^s b_i f(x_0 + c_i h) - \int_{x_0}^{x_0+h} f(x) dx \right)}_{= Ch^{31} + \mathcal{O}(h^{32})} + \underbrace{\left( \int_{x_0}^{x_0+h} f(x) dx - h \sum_{i=1}^s \hat{b}_i f(x_0 + c_i h) \right)}_{= Ch^{15} + \mathcal{O}(h^{16})}. \end{aligned}$$

Pour le programme TEGRAL, on considère encore une deuxième formule emboîtée qui a pour nœuds  $\{c_2, c_4, c_6, c_{10}, c_{12}, c_{14}\}$  et un ordre 6 (voir la fig. I.10). On dénote les poids de cette formule de quadrature par  $\hat{\bar{b}}_i$  et on définit

$$\text{ERR2} := h \sum_{i=1}^s b_i f(x_0 + c_i h) - h \sum_{i=1}^s \hat{\bar{b}}_i f(x_0 + c_i h) \approx C_2 h^7. \quad (6.8)$$

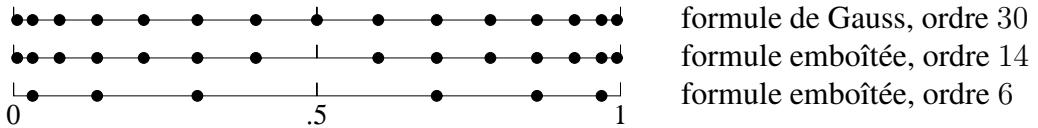


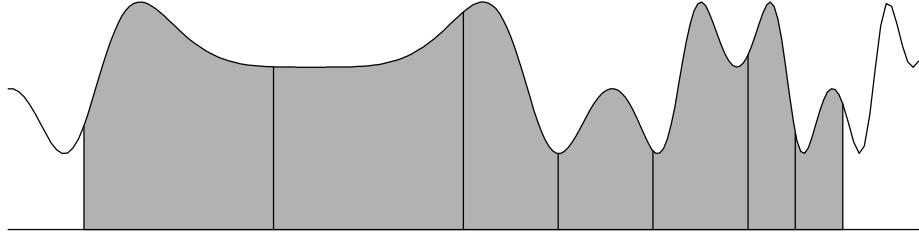
FIG. I.10: Formule de Gauss et ses formules emboîtées

Il y a plusieurs possibilités pour définir  $\text{err}(x_0, x_0 + h)$ :

- $\text{err}(x_0, x_0 + h) := \text{ERR1}$ ; cette estimation est trop pessimiste. En général, la formule de Gauss donne un résultat largement meilleur que la formule emboîtée d'ordre 14.
- dans le programme TEGRAL, on a choisi l'approximation

$$\text{err}(x_0, x_0 + h) := \text{ERR1} \cdot \left( \frac{\text{ERR1}}{\text{ERR2}} \right)^2, \quad \left( \approx h^{15} \cdot \left( \frac{h^{15}}{h^7} \right)^2 \approx h^{31} \right)$$

ce qui donne de bons résultats.

FIG. I.11: Division choisie par TEGRAL pour  $f(x) = 2 + \sin(3 \cos(0.002(x-40)^2))$  sur  $(10, 110)$ 

## Exemples

- 1) Si l'on applique le programme TEGRAL avec  $\text{TOL} = 10^{-10}$  à la fonction de la fig. I.1, on obtient le résultat avec une erreur de  $2.0 \cdot 10^{-14}$ . La division choisie est donnée dans la fig. I.11.
- 2) Appliquons le même programme à la fonction

$$f(x) = \sqrt{x} \cdot \log x \quad \text{sur } (0, 1), \quad (6.9)$$

qui a une singularité au point 0 dans la dérivée. Les divisions successives de l'intervalle par l'algorithme sont présentées dans la fig. I.12. On voit que l'intervalle où l'erreur est maximale est toujours celui qui est tout à gauche. Les erreurs sont données dans le tableau I.2. La convergence est très lente et on se demande s'il n'y a pas de possibilité d'accélérer la convergence de la suite  $\{S_N\}$ .

TAB. I.2: Résultat de TEGRAL pour (6.9)

$N$	$S_N$	$err_N = S_N - \int_0^1 f(x) dx$	$err_N/err_{N-1}$
1	-0.4446200164956040	$-0.176 \cdot 10^{-3}$	—
2	-0.4445133092592463	$-0.689 \cdot 10^{-4}$	0.392
3	-0.4444711927155809	$-0.267 \cdot 10^{-4}$	0.388
4	-0.4444547502264998	$-0.103 \cdot 10^{-4}$	0.385
5	-0.4444483881989292	$-0.394 \cdot 10^{-5}$	0.383
6	-0.4444459448772270	$-0.150 \cdot 10^{-5}$	0.380
...	...	...	...
21	-0.44444444444449657	$-0.521 \cdot 10^{-12}$	0.366
22	-0.44444444444446350	$-0.191 \cdot 10^{-12}$	0.366

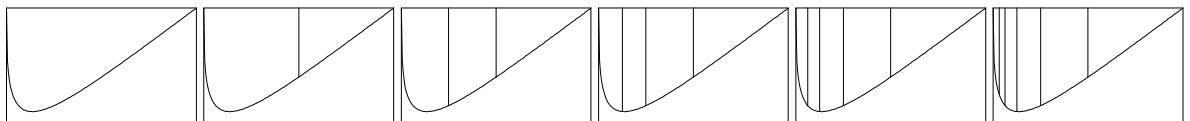


FIG. I.12: Division choisie par TEGRAL pour (6.9)

## I.7 L'epsilon-algorithme

Etant donnée une suite  $\{S_0, S_1, S_2, \dots\}$  qui converge lentement vers la valeur  $S$ . Le but est de trouver une autre suite avec la même limite, mais qui converge plus rapidement.

Souvent, on peut observer que la suite satisfait

$$S_{n+1} - S \approx \rho \cdot (S_n - S), \quad (7.1)$$

ou de façon équivalente

$$S_n \approx S + C \cdot \rho^n. \quad (7.2)$$

Par exemple, la suite  $S_n$  du tableau I.2 satisfait approximativement (7.1) avec  $\rho = 0.366$ . Un autre exemple fréquent est donné par la méthode des approximations successives  $S_{n+1} = g(S_n)$  pour calculer un point fixe de  $g(x)$ , c.-à-d. un  $S$  satisfaisant  $S = g(S)$ . Si  $g$  est différentiable,

$$S_{n+1} - S = g(S_n) - g(S) \approx g'(S) \cdot (S_n - S).$$

Ceci est (7.1) avec  $\rho = g'(S)$ .

### Le procédé $\Delta^2$ d'Aitken (1926)

L'idée est de remplacer “ $\approx$ ” par “ $=$ ” dans (7.2) et de calculer  $\rho$ ,  $C$  et  $S$  de trois formules consécutives. Avec la notation

$$\Delta S_n := S_{n+1} - S_n \quad (\text{différence finie}), \quad (7.3)$$

on obtient alors

$$\Delta S_n = C\rho^n(\rho - 1), \quad \Delta S_{n+1} = C\rho^{n+1}(\rho - 1), \quad \Delta^2 S_n = C\rho^n(\rho - 1)^2,$$

où  $\Delta^2 S_n = \Delta(\Delta S_n) = \Delta S_{n+1} - \Delta S_n = S_{n+2} - 2S_{n+1} + S_n$  est la deuxième différence finie. On en déduit que

$$S = S_{n+1} - C\rho^{n+1} = S_{n+1} - \frac{\Delta S_n \cdot \Delta S_{n+1}}{\Delta^2 S_n}. \quad (7.4)$$

Si (7.2) n'est pas satisfait exactement, la valeur de  $S$  dans (7.4) va dépendre de  $n$ . On obtient ainsi une autre suite  $\{S'_n\}$  définie par (procédé  $\Delta^2$  d'Aitken)

$$S'_n = S_{n+1} - \frac{\Delta S_n \cdot \Delta S_{n+1}}{\Delta^2 S_n}, \quad (7.5)$$

qui, en général, converge plus rapidement vers  $S$  que la suite originale  $\{S_n\}$ .

**Exemple.** Pour la suite  $\{S_n\}$  du tableau I.2, le résultat est donné dans le tableau I.3.

TAB. I.3: Accélération de la convergence pour  $\{S_n\}$  du tableau I.2

$n$	$S_n$	$S'_n$	$S''_n$
1	-0.4446200164956040	-0.4444437305042874	-0.4444444444444444
2	-0.4445133092592463	-0.4444442199284397	-0.4444444444444444
3	-0.4444711927155809	-0.444443729666139	-0.4444444444444444
4	-0.4444547502264998	-0.444444214607878	-0.4444444444444444
5	-0.4444483881989292	-0.444444369930052	-0.4444444444444444
6	-0.4444459448772270	-0.444444420118825	-0.4444444444444444

## L'epsilon-algorithme

Pour généraliser l'idée d'Aitken, on considère une suite  $\{S_n\}$  pour laquelle on suppose, au lieu de (7.2),

$$S_n \approx S + C_1 \cdot \rho_1^n + C_2 \cdot \rho_2^n. \quad (7.6)$$

Cette fois, on a 5 paramètres à déterminer. Alors, on prend 5 formules consécutives de (7.6), on suppose égalité, et on calcule  $S, C_1, \rho_1, C_2, \rho_2$ . La valeur de  $S$  ainsi obtenue est dénotée par  $S''_n$ . Shanks (1955) a fait ce calcul et il a trouvé la formule (nous ajoutons une formule semblable pour  $S'_n$ )

$$S'_n = \frac{\det \begin{pmatrix} S_n & S_{n+1} \\ S_{n+1} & S_{n+2} \end{pmatrix}}{\Delta^2 S_n}, \quad S''_n = \frac{\det \begin{pmatrix} S_n & S_{n+1} & S_{n+2} \\ S_{n+1} & S_{n+2} & S_{n+3} \\ S_{n+2} & S_{n+3} & S_{n+4} \end{pmatrix}}{\det \begin{pmatrix} \Delta^2 S_n & \Delta^2 S_{n+1} \\ \Delta^2 S_{n+1} & \Delta^2 S_{n+2} \end{pmatrix}}$$

(sans démonstration). Mais, pour un calcul numérique, ces formules ne sont pas très pratiques. Wynn (1956) a trouvé une formule beaucoup plus simple:

**Théorème 7.1 ( $\epsilon$ -algorithme)** *Etant donnée la suite  $\{S_0, S_1, S_2, \dots\}$ . Si l'on définit  $\epsilon_k^{(n)}$  par*

$$\begin{aligned} \epsilon_{-1}^{(n)} &= 0, & \epsilon_0^{(n)} &= S_n, \\ \epsilon_{k+1}^{(n)} &= \epsilon_{k-1}^{(n+1)} + \frac{1}{\epsilon_{k+1}^{(n+1)} - \epsilon_k^{(n)}}, & k \geq 0, n \geq 0, \end{aligned} \quad (7.7)$$

alors,  $\epsilon_2^{(n)} = S'_n$ ,  $\epsilon_4^{(n)} = S''_n$ ,  $\epsilon_6^{(n)} = S'''_n, \dots$

La démonstration de  $\epsilon_2^{(n)} = S'_n$  se fait par un simple calcul de  $\epsilon_1^{(n)}$  et de  $\epsilon_2^{(n)}$ :

$$\begin{aligned} \epsilon_1^{(n)} &= 0 + \frac{1}{S_{n+1} - S_n} = \frac{1}{\Delta S_n}, \\ \epsilon_2^{(n)} &= S_{n+1} + \frac{1}{\frac{1}{\Delta S_{n+1}} - \frac{1}{\Delta S_n}} = S_{n+1} + \frac{\Delta S_n \cdot \Delta S_{n+1}}{\Delta S_n - \Delta S_{n+1}} = S'_n. \end{aligned}$$

Le cas général est moins évident et est donné sans démonstration. Tous les détails (démonstration et autres propriétés) sont donnés dans un livre de Brezinski<sup>2</sup>.

*Exemple.* Pour la suite

$$S_n = \sum_{i=1}^n \frac{(-1)^{i+1}}{i}, \quad (7.8)$$

qui converge vers  $\log(2)$ , les erreurs de  $\epsilon_k^{(n)}$ ,  $k = 0, 2, 4, 6, \dots$  sont données dans la fig. I.13. L'amélioration de la convergence par l' $\epsilon$ -algorithme se voit clairement.

---

<sup>2</sup>C. Brezinski (1977): *Accélération de la Convergence en Analyse Numérique*. Lecture Notes in Mathematics, Nr. 584, Springer-Verlag. [MA 00.04/3 584]

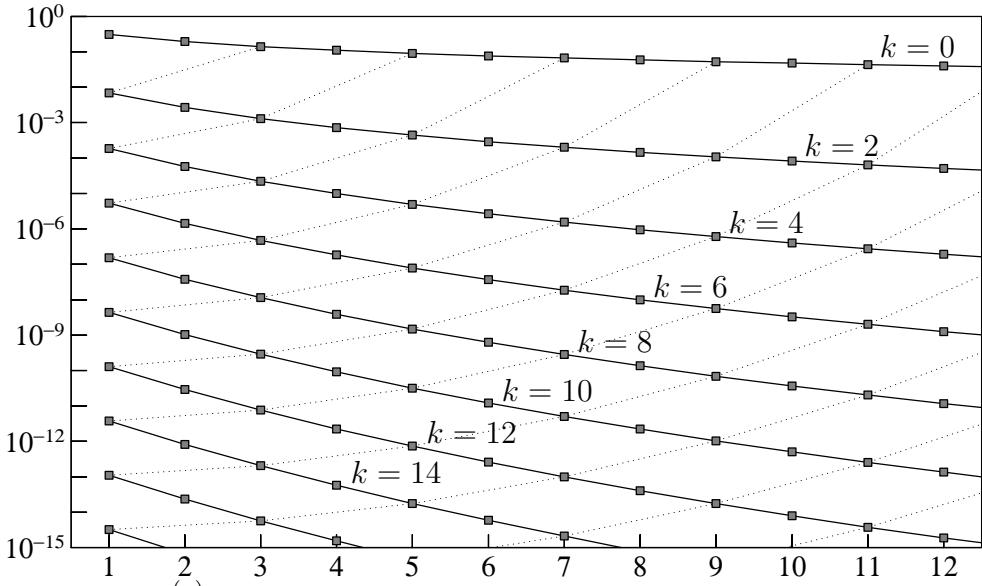


FIG. I.13: Erreur de  $\epsilon_k^{(n)}$  en fonction de  $n$  pour la suite (7.8); les lignes pointillées indiquent que le même nombre des  $S_j$  est utilisé.

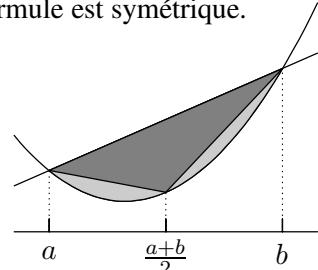
## I.8 Exercices

- Soit  $(b_i, c_i)_{i=1}^s$  une formule de quadrature d'ordre  $\geq s$ . Montrer que

$$b_i = \int_0^1 \ell_i(x) dx \quad \text{où} \quad \ell_i(x) = \prod_{\substack{j=1 \\ j \neq i}}^s \frac{(x - c_j)}{(c_i - c_j)}.$$

- Si les nœuds d'une formule de quadrature sont  $c_i = 1 - c_{s+1-i}$  (pour tous  $i$ ) et si la formule a un ordre  $p \geq s$ , alors on a nécessairement  $b_i = b_{s+1-i}$ , c'est-à-dire la formule est symétrique.
- Soient une parabole et une droite se coupant comme sur le petit dessin. Utiliser la règle de Simpson pour montrer que (Archimète, 283–212 av. J.-C.)

$$\text{aire(parabole)} = \frac{4}{3} \text{aire(triangle)}.$$



- Calculer les formules de Newton-Cotes pour

$$(c_i) = (0, 1/3, 2/3, 1), \quad (c_i) = (0, 1/4, 2/4, 3/4, 1)$$

et déterminer l'ordre de ces formules de quadrature.

*Indication.* Les calculs se simplifient en utilisant l'exercice 2.

- Calculer la constante d'erreur pour la formule de Simpson et de Newton. Expliquer pourquoi, malgré le fait que la méthode de Newton possède une constante d'erreur plus petite, la méthode de Simpson est meilleure si on compare l'erreur avec le travail *fe* (voir la fig. I.4).
- Calculer les noyaux de Peano  $N_k(\tau)$  ( $k = 1, 2$ ) pour la règle du trapèze et les dessiner. Remarquer une relation avec les polynômes de Bernoulli et la formule d'Euler-Maclaurin.<sup>3</sup>
- Montrer que, pour une formule de quadrature symétrique, les noyaux de Peano satisfont

$$N_k(1 - \tau) = (-1)^k N_k(\tau).$$

---

<sup>3</sup>Hairer & Wanner, *Analysis by Its History*, Undergraduate Texts in Mathematics, Springer, 2nd printing 1997.

8. Calculer les noyaux de Peano  $N_k(\tau)$  ( $k = 1, 2, 3, 4$ ) pour la formule de Simpson. Les dessiner. Est-ce que  $N_4(\tau)$  change de signe sur l'intervalle  $[0, 1]$ ?
9. Soit  $p$  l'ordre d'une formule de quadrature et supposons que le noyau de Peano  $N_p(\tau)$  ne change pas de signe sur  $[0, 1]$ . Montrer qu'avec un  $\xi \in (x_0, x_0 + h)$

$$\int_{x_0}^{x_0+h} f(x) dx - h \sum_{i=1}^s b_i f(x_0 + c_i h) = \frac{h^{p+1}}{p!} \left( \frac{1}{p+1} - \sum_{i=1}^s b_i c_i^p \right) f^{(p)}(\xi).$$

10. (Formule de Radau). Déterminer  $c_2, b_1, b_2$  dans la formule de quadrature

$$\int_0^1 g(t) dt \approx b_1 g(0) + b_2 g(c_2)$$

afin que son ordre soit maximal.

Résultat.  $c_2 = 2/3, b_1 = 1/4, b_2 = 3/4$  et  $p = 3$ .

11. Pour les *polynômes de Legendre* démontrer la formule

$$(1 - \tau^2)P'_k(\tau) = -k\tau P_k(\tau) + kP_{k-1}(\tau). \quad (8.1)$$

Indication. Écrire le polynôme  $(1 - \tau^2)P'_k(\tau) + k\tau P_k(\tau)$  sous forme d'une combinaison linéaire de  $P_{k+1}(\tau), P_{k-1}(\tau), \dots$  comme dans la démonstration du théorème 4.3.

12. Calculer les racines du polynôme de Legendre  $P_{15}(\tau)$  en utilisant la *méthode de bisection*.

- (a) Localiser les racines en calculant  $P_{15}(i/n)$  pour  $i = 0, 1, \dots, n$  et avec un  $n$  grand;
- (b) Si  $[a, b]$  est un intervalle avec  $P_n(a) \cdot P_n(b) < 0$  alors

```

10      CENTR=(A+B)/2.D0
        PC=P(N,CENTR)
        IF (CENTR.EQ.A.OR.CENTR.EQ.B) GOTO 40
        IF (PA*PC.LT.0.D0) THEN
          B=CENTR
          PB=PC
        ELSE
          A=CENTR
          PA=PC
        END IF
        GOTO 10
40      CONTINUE

```

Pour écrire la FUNCTION  $P(N, X)$  qui calcule la valeur de la fonction  $P_k(\tau)$ , utiliser la formule de récurrence (4.5) et  $P_0(\tau) = 1, P_1(\tau) = \tau$ .

13. Calculer la constante d'erreur  $C_s$  pour la formule de Gauss avec  $s$  noeuds d'ordre  $2s$ .

Indication.  $C_s$  est l'erreur de la formule quand elle est appliquée à un polynôme de degré  $2s$  de la forme  $t^{2s}/(2s)! + \dots$ . Essayer  $K \cdot (P_s(2t-1))^2$ , et utiliser la formule de récurrence de l'exercice 11 pour evaluer  $\int_{-1}^1 P_s(\tau)^2 d\tau$ . Le résultat est

$$C_s = \frac{(s!)^4}{(2s+1)(2s!)^3}.$$

14. Montrer que pour les formules de quadrature de Gauss (ordre  $p = 2s$ ) le noyau de Peano  $N_{2s}(\tau)$  ne change pas de signe.

Indication. Faire une démonstration par l'absurde et utiliser le théorème de Rolle.

15. Soient  $c_1^{(s)}, c_2^{(s)}, \dots, c_s^{(s)}$  les nœuds de la formule de quadrature de Gauss d'ordre  $2s$  et  $c_1^{(s+1)}, c_2^{(s+1)}, \dots, c_{s+1}^{(s+1)}$  ceux de la formule d'ordre  $2s + 2$ . Montrer qu'on a alors

$$0 < c_1^{(s+1)} < c_1^{(s)} < c_2^{(s+1)} < c_2^{(s)} < \dots < c_s^{(s)} < c_{s+1}^{(s+1)} < 1.$$

*Indication.* Procéder par récurrence en utilisant le fait que si  $P_s(\tau) = 0$ , alors  $P_{s-1}(\tau)$  et  $P_{s+1}(\tau)$  sont de signe opposé (voir formule (4.6)).

16. (Formules de Radau). Montrer qu'il existe une unique formule de quadrature d'ordre  $2s - 1$  qui satisfait  $c_s = 1$ .

*Indication.* Utiliser  $M(t) = \text{Const} \cdot (P_s(x) - P_{s-1}(x))$ .

17. Considérons une formule de quadrature d'ordre  $p \geq 1$  satisfaisant  $0 \leq c_i \leq 1$ . Montrer que, pour toute fonction  $f : [a, b] \rightarrow \mathbb{R}$  intégrable au sens de Riemann, on a

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

lorsque  $h = \max_j h_j$  tend vers zéro.

*Indication.* Pour un  $i$  fixé, l'expression  $\sum_{j=0}^N h_j f(x_j + c_i h_j)$  est une somme de Riemann.

18. Soit  $f : \mathbb{R} \rightarrow \mathbb{R}$ , donne par

$$f(x) = a_0 + \sum_{k=1}^m (a_k \cos(kx) + b_k \sin(kx)), \quad a_k, b_k \in \mathbb{R}.$$

- (a) Quelle est la valeur exacte de  $\int_0^{2\pi} f(x) dx$  ?
  - (b) Appliquer la règle du trapèze à  $\int_0^{2\pi} f(x) dx$  avec  $h = 2\pi/N$ . A partir de quelle valeur de  $N$ , le résultat est exacte?
  - (c) Appliquer une formule de quadrature d'ordre plus élevé (par exemple celle de Gauss d'ordre 6, voir exemple 3.3) et répondez à la même question que sous (b).
  - (d) Quelle formule de quadrature proposez-vous pour l'intégration numérique d'une fonction périodique?
19. Considérons la suite  $\{x_n\}$  donnée par  $x_{n+1} = x_n + 1 - x_n^2/2$ ,  $x_0 = 0$ .
- (a) Quelle est sa limite?
  - (b) Appliquer l'algorithme  $\Delta^2$  d'Aitken pour accélérer la convergence.
  - (c) En utilisant  $x_0, x_1, \dots, x_8$  comparer l'erreur des suites obtenues avec ou sans  $\Delta^2$  d'Aitken.
20. Considérons une suite  $\{S_n\}$  qui satisfait

$$(S_{n+1} - S) = \rho_n(S_n - S) \quad \text{avec} \quad \rho_n \longrightarrow \rho \quad \text{et} \quad \rho \neq 1.$$

- a) Montrer que la suite  $\{S'_n\}$ , donnée par le procédé  $\Delta^2$  d'Aitken converge plus vite vers  $S$  que la suite originale, c.-à-d.,

$$\frac{S'_n - S}{S_n - S} \longrightarrow 0 \quad \text{pour} \quad n \longrightarrow \infty.$$

- b) Donner une suite divergente  $\{S_n\}$  pour laquelle la suite  $\{S'_n\}$  converge.

*Indication.* Nous savons que  $\Delta S_n = (\rho_n - 1)(S_n - S)$ , trouver une formule similaire pour  $\Delta^2 S_n$ .

# Chapitre II

## Interpolation et Approximation

Le problème de l'*interpolation* consiste à chercher des fonctions “simples” (polynômes, polynômes par morceaux, polynômes trigonométriques) passant par des points donnés

$$(x_0, y_0), (x_1, y_1), \dots, (x_n, y_n), \quad (0.1)$$

c.-à-d., on cherche  $p(x)$  avec  $p(x_i) = y_i$  pour  $i = 0, 1, \dots, n$ . Si les valeurs de  $y_i$  satisfont  $y_i = f(x_i)$  où  $f(x)$  est une fonction donnée, il est intéressant d’étudier l’erreur de l’*approximation*

$$f(x) - p(x) = ? \quad (0.2)$$

### Bibliographie sur ce chapitre

- J.H. Ahlberg, E.N. Nilson & J.L. Walsh (1967): *The Theory of Splines and Their Applications*. Academic Press, New York. [MA 65/4]
- C. de Boor (1978): *A Practical Guide to Splines*. Springer-Verlag. [MA 65/141]
- G.D. Knott (2000): *Interpolating Cubic Splines*. Birkhäuser. [MA 65/431]
- H.J. Nussbaumer (1981): *Fast Fourier Transform and Convolution Algorithms*. Springer-Verlag.
- H. Späth (1995): *One Dimensional Spline Interpolation*. AK Peters. [MA 65/362]

### II.1 Différences divisées et formule de Newton

Étant donnés les  $n + 1$  points (0.1) où les  $x_i$  sont distincts mais pas nécessairement ordonnés, on cherche un polynôme  $p(x)$  de degré  $n$  qui satisfasse

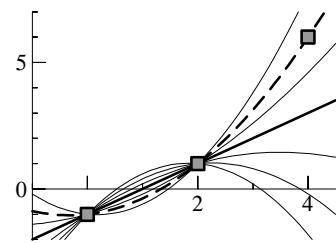
$$p(x_i) = y_i \quad \text{pour} \quad i = 0, 1, \dots, n. \quad (1.1)$$

*Cas  $n = 1$ .* Le polynôme de degré 1 (une droite) qui passe par  $(x_0, y_0), (x_1, y_1)$  est donné par

$$p(x) = y_0 + (x - x_0) \frac{y_1 - y_0}{x_1 - x_0}. \quad (1.2)$$

*Cas  $n = 2$ .* Pour obtenir un polynôme de degré 2 (une parabole) qui passe par les trois points  $(x_0, y_0), (x_1, y_1), (x_2, y_2)$ , on ajoute un terme de correction (de degré 2) à la formule (1.2). Comme ce terme doit être zéro aux points  $x_0$  et  $x_1$ , on a nécessairement

$$p(x) = y_0 + (x - x_0) \frac{y_1 - y_0}{x_1 - x_0} + a \cdot (x - x_0)(x - x_1). \quad (1.3)$$



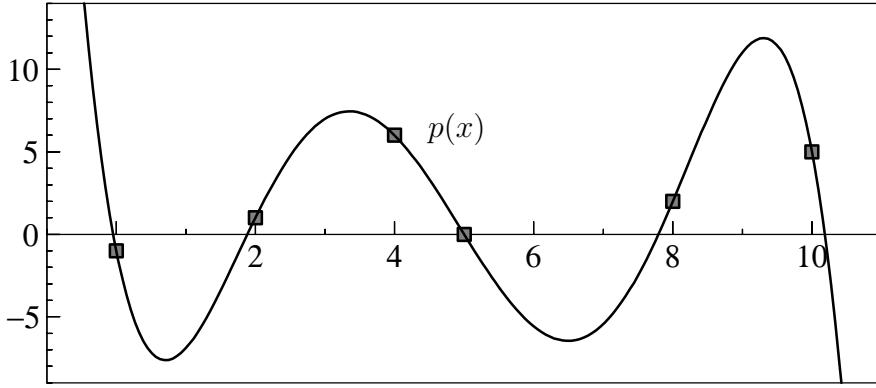


FIG. II.1: Polynôme d'interpolation de degré 5

Le coefficient  $a$  est déterminé par  $p(x_2) = y_2$ . Un calcul simple (soustraire  $p(x_1)$  de  $p(x_2)$  et diviser par  $(x_2 - x_0)(x_2 - x_1)$ ) nous donne

$$a = \frac{1}{x_2 - x_0} \left( \frac{y_2 - y_1}{x_2 - x_1} - \frac{y_1 - y_0}{x_1 - x_0} \right). \quad (1.4)$$

Un exemple pour le cas  $n = 5$  avec des  $x_i$  non équidistants est donné dans la fig. II.1. Pour les cas  $n > 2$  les formules deviennent plus complexes et il est recommandé d'introduire une notation convenable pour simplifier les expressions comme dans (1.4).

**Définition 1.1 (différences divisées)** Pour  $(x_i, y_i)$  donnés ( $x_i$  distincts) on définit

$$\begin{aligned} y[x_i] &:= y_i \\ \delta y[x_i, x_j] &:= \frac{y[x_j] - y[x_i]}{x_j - x_i} \\ \delta^2 y[x_i, x_j, x_k] &:= \frac{\delta y[x_j, x_k] - \delta y[x_i, x_j]}{x_k - x_i} \\ \delta^n y[x_{i_0}, x_{i_1}, \dots, x_{i_n}] &:= \frac{\delta^{n-1} y[x_{i_1}, \dots, x_{i_n}] - \delta^{n-1} y[x_{i_0}, \dots, x_{i_{n-1}}]}{x_{i_n} - x_{i_0}}. \end{aligned}$$

**Théorème 1.2 (formule de Newton)** *Le polynôme d'interpolation de degré  $n$  qui passe par les  $n + 1$  points  $(x_0, y_0), (x_1, y_1), \dots, (x_n, y_n)$ , où les  $x_i$  sont distincts, est unique et donné par*

$$\begin{aligned} p(x) &= y[x_0] + (x - x_0) \delta y[x_0, x_1] + (x - x_0)(x - x_1) \delta^2 y[x_0, x_1, x_2] \\ &\quad + \dots + (x - x_0)(x - x_1) \dots (x - x_{n-1}) \delta^n y[x_0, x_1, \dots, x_n]. \end{aligned} \quad (1.5)$$

*Démonstration.* Pour  $n = 1$  et  $n = 2$  la formule (1.5) est équivalente à (1.2) et à (1.3)–(1.4). Pour démontrer le cas général, nous procédons par récurrence. Supposons que

$$p_1(x) = y[x_0] + (x - x_0) \delta y[x_0, x_1] + \dots + (x - x_0) \dots (x - x_{n-2}) \delta^{n-1} y[x_0, x_1, \dots, x_{n-1}]$$

soit le polynôme unique de degré  $n - 1$  qui passe par  $(x_i, y_i)$  pour  $i = 0, 1, \dots, n - 1$ . Alors, comme dans (1.3), le polynôme  $p(x)$  est nécessairement de la forme

$$p(x) = p_1(x) + a \cdot (x - x_0)(x - x_1) \dots (x - x_{n-1}), \quad (1.6)$$

où  $a$  est déterminé par  $p(x_n) = y_n$ . Il en résulte l'unicité du polynôme d'interpolation.

Pour montrer que  $a = \delta^n y[x_0, x_1, \dots, x_n]$ , ce qui achève la démonstration, nous considérons également le polynôme de degré  $n - 1$

$$p_2(x) = y[x_1] + (x - x_1) \delta y[x_1, x_2] + \dots + (x - x_1) \cdot \dots \cdot (x - x_{n-1}) \delta^{n-1} y[x_1, x_2, \dots, x_n]$$

qui passe par  $(x_i, y_i)$  pour  $i = 1, \dots, n$ . Ensuite, on pose (idée d'Aitken–Neville, 1929, 1932)

$$q(x) := \frac{1}{x_n - x_0} ((x_n - x)p_1(x) + (x - x_0)p_2(x)). \quad (1.7)$$

Il s'agit d'un polynôme de degré  $n$  qui satisfait la condition (1.1) pour le point  $x_0$  (ici, le facteur  $(x - x_0)$  est nul), pour le point  $x_n$  (ici, le facteur  $(x_n - x)$  est nul), et pour les points  $x_i$ ,  $i = 1, \dots, n - 1$  (ici, les deux polynômes  $p_1(x)$  et  $p_2(x)$  sont égaux à  $y_i$ ). Par l'unicité du polynôme d'interpolation on a alors  $q(x) = p(x)$ . En comparant le coefficient de  $x^n$  dans (1.7) avec celui de (1.6), nous obtenons

$$a = \frac{1}{x_n - x_0} (\delta^{n-1} y[x_1, \dots, x_n] - \delta^{n-1} y[x_0, \dots, x_{n-1}]) = \delta^n y[x_0, \dots, x_n],$$

ce qui démontre la formule (1.5).  $\square$

TAB. II.1: Différences divisées pour les données de la fig. II.1

$x_i$	$y_i$	$\delta y$	$\delta^2 y$	$\delta^3 y$	$\delta^4 y$	$\delta^5 y$
0	-1					
2	1	1				
4	6	5/2	3/8			
5	0	-6	-17/6	-77/120		
8	2	2/3	5/3	3/4	167/960	
10	5	3/2	1/6	-1/4	-1/8	-287/9600

**Exemple 1.3** Pour les données de la fig. II.1, les différences divisées sont présentées dans le tableau II.1. Le polynôme d'interpolation est alors donné par

$$\begin{aligned} p(x) = & -1 + x + x(x - 2)\frac{3}{8} - x(x - 2)(x - 4)\frac{77}{120} + x(x - 2)(x - 4)(x - 5)\frac{167}{960} \\ & - x(x - 2)(x - 4)(x - 5)(x - 8)\frac{287}{9600}. \end{aligned}$$

ou mieux encore pour la programmation (ou le calcul à la main)

$$p(x) = -1 + x \left( 1 + (x - 2) \left( \frac{3}{8} + (x - 4) \left( -\frac{77}{120} + (x - 5) \left( \frac{167}{960} - (x - 8) \frac{287}{9600} \right) \right) \right) \right).$$

*Remarque.* L'ordre des  $\{x_i\}$  n'a aucune importance pour la formule de Newton (1.5). Si l'on permute les données  $(x_i, y_i)$ , on obtient évidemment le même polynôme. Pour l'exemple ci-dessus et pour les  $\{x_i\}$  choisis dans l'ordre  $\{4, 5, 2, 8, 0, 10\}$ , on obtient ainsi

$$p(x) = 6 + (x - 4) \left( -6 + (x - 5) \left( -\frac{17}{6} + (x - 2) \left( \frac{3}{4} + (x - 8) \left( \frac{167}{960} - x \frac{287}{9600} \right) \right) \right) \right).$$

En observant que  $\delta^n y[x_{i_0}, \dots, x_{i_n}]$  est une fonction symétrique de ses arguments (par exemple,  $\delta^2 y[x_2, x_3, x_1] = \delta^2 y[x_1, x_2, x_3]$ , voir l'exercice 2), on peut utiliser les valeurs calculées dans le tableau II.1.

Pour diminuer l'influence des erreurs d'arrondi, il est recommandé d'ordonner les  $\{x_i\}$  de manière à ce que les valeurs situées au milieu soient prises d'abord et les valeurs aux extrémités à la fin. Pour ce choix, les expressions  $(x - x_0)$ ,  $(x - x_0)(x - x_1)$ ,  $(x - x_0)(x - x_1)(x - x_2)$ , etc., sont en général plus petites que pour un autre choix et l'amplification des erreurs dans les différences divisées est moins importante.

## II.2 Erreur de l'interpolation

Supposons que les points  $(x_i, y_i)$  soient sur le graphe d'une fonction  $f : [a, b] \rightarrow \mathbb{R}$ , c.-à-d.,

$$y_i = f(x_i), \quad i = 0, 1, \dots, n, \quad (2.1)$$

étudions alors l'erreur  $f(x) - p(x)$  du polynôme d'interpolation  $p(x)$ . Deux exemples sont donnés dans la fig. II.2. A gauche, on voit un polynôme d'interpolation pour la fonction  $f(x) = \sin x$ , et à droite pour la fonction  $1/(1 + x^2)$ . Pour mieux voir l'erreur, on a dessiné la fonction  $f(x)$  en une courbe pointillée.

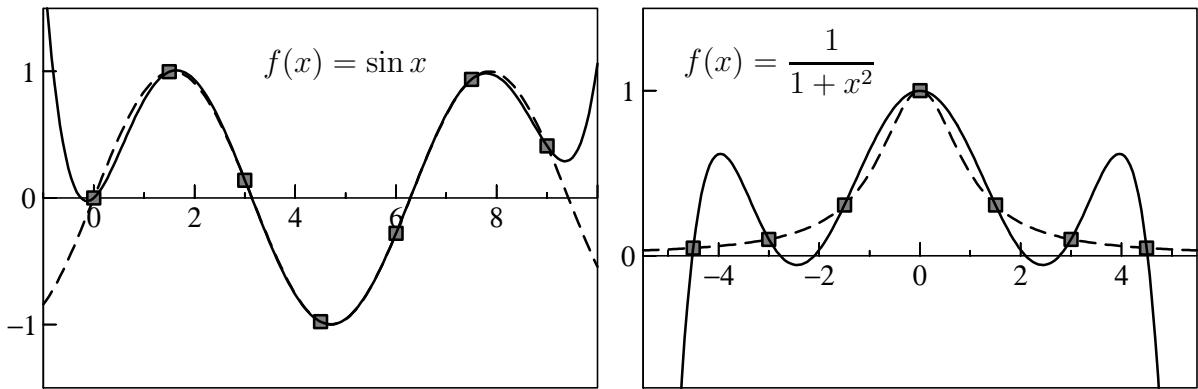


FIG. II.2: Polynôme d'interpolation pour  $\sin x$  (gauche) et pour  $1/(1 + x^2)$  (droite)

Les résultats suivants sont dus à Cauchy (1840, *Sur les fonctions interpolaires*, C.R. XI, p. 775-789, *Oeuvres* ser. 1, vol. V, p. 409-424). Commençons par une relation intéressante entre les différences divisées pour (2.1) et les dérivées de la fonction  $f(x)$ .

**Lemme 2.1** Soit  $f(x)$   $n$ -fois différentiable et  $y_i = f(x_i)$  pour  $i = 0, 1, \dots, n$  ( $x_i$  distincts). Alors, il existe un  $\xi \in (\min x_i, \max x_i)$  tel que

$$\delta^n y[x_0, x_1, \dots, x_n] = \frac{f^{(n)}(\xi)}{n!}. \quad (2.2)$$

*Démonstration.* Soit  $p(x)$  le polynôme d'interpolation de degré  $n$  passant par  $(x_i, y_i)$  et notons  $d(x) = f(x) - p(x)$ . Par définition de  $p(x)$ , la différence  $d(x)$  s'annule en  $n + 1$  points distincts :

$$d(x_i) = 0 \quad \text{pour} \quad i = 0, 1, \dots, n.$$

Comme  $d(x)$  est différentiable, on peut appliquer  $n$  fois le théorème de Rolle (voir le cours d'Analyse I) et on en déduit que

$$d'(x) \quad \text{a } n \text{ zéros distincts dans } (\min x_i, \max x_i).$$

Le même argument appliqué à  $d'(x)$  donne

$$d''(x) \quad \text{a } n-1 \text{ zéros distincts dans } (\min x_i, \max x_i),$$

et finalement encore

$$d^{(n)}(x) \quad \text{a 1 zéro dans } (\min x_i, \max x_i).$$

Notons ce zéro de  $d^{(n)}(x)$  par  $\xi$ . Alors, on a

$$f^{(n)}(\xi) = p^{(n)}(\xi) = n! \cdot \delta^n y[x_0, x_1, \dots, x_n]. \quad (2.3)$$

La deuxième identité dans (2.3) résulte du fait que  $\delta^n y[x_0, x_1, \dots, x_n]$  est le coefficient de  $x^n$  dans  $p(x)$ .  $\square$

**Théorème 2.2** Soit  $f : [a, b] \rightarrow \mathbb{R}$  ( $n+1$ )-fois différentiable et soit  $p(x)$  le polynôme d'interpolation de degré  $n$  qui passe par  $(x_i, f(x_i))$  pour  $i = 0, 1, \dots, n$ . Alors, pour  $x \in [a, b]$ , il existe un  $\xi \in (\min(x_i, x), \max(x_i, x))$  tel que

$$f(x) - p(x) = (x - x_0) \cdot \dots \cdot (x - x_n) \cdot \frac{f^{(n+1)}(\xi)}{(n+1)!}. \quad (2.4)$$

*Démonstration.* Si  $x = x_i$  pour un indice  $i \in \{0, 1, \dots, n\}$ , la formule (2.4) est vérifiée car  $p(x_i) = f(x_i)$ . Fixons alors un  $\bar{x}$  dans  $[a, b]$  qui soit différent de  $x_i$  et montrons la formule (2.4) pour  $x = \bar{x}$ .

L'idée est de considérer le polynôme  $\bar{p}(x)$  de degré  $n+1$  qui passe par  $(x_i, f(x_i))$  pour  $i = 0, 1, \dots, n$  et par  $(\bar{x}, f(\bar{x}))$ . La formule de Newton donne

$$\bar{p}(x) = p(x) + (x - x_0) \cdot \dots \cdot (x - x_n) \cdot \delta^{n+1} y[x_0, \dots, x_n, \bar{x}]. \quad (2.5)$$

Si l'on remplace la différence divisée dans (2.5) par  $f^{(n+1)}(\xi)/(n+1)!$  (voir le lemme précédent) et si l'on pose  $x = \bar{x}$ , on obtient le résultat (2.4) pour  $x = \bar{x}$  car  $\bar{p}(\bar{x}) = f(\bar{x})$ . Comme  $\bar{x}$  est arbitraire, la formule (2.4) est vérifiée pour tout  $x$ .  $\square$

**Exemple 2.3** Dans la situation de la fig. II.2, on a  $n+1 = 7$ . Comme la 7ème dérivée de  $\sin x$  est majorée par 1, on a que

$$|p(x) - \sin x| \leq |x(x - 1.5)(x - 3)(x - 4.5)(x - 6)(x - 7.5)(x - 9)| \cdot \frac{1}{7!},$$

par exemple

$$|p(4) - \sin 4| \leq 0.035 \quad \text{ou} \quad |p(1) - \sin 1| \leq 0.181.$$

Pour le deuxième exemple,  $f(x) = 1/(1+x^2)$ , la 7ème dérivée est donnée par

$$f^{(7)}(x) = -8! \cdot \frac{(x+1)(x-1)x(x^2-2x-1)(x^2+2x-1)}{(1+x^2)^8},$$

qui est maximale pour  $x \approx \pm 0.17632698$ . On obtient ainsi

$$\left| p(x) - \frac{1}{1+x^2} \right| \leq |(x^2-20.25)(x^2-9)(x^2-2.25)x| \cdot \frac{4392}{7!}.$$

Alors, l'erreur peut être 4392 fois plus grande que pour l'interpolation de  $\sin x$ .

## II.3 Polynômes de Chebyshev

La formule (2.4) montre que l'erreur de l'interpolation est un produit de la  $(n + 1)$ <sup>ème</sup> dérivée de  $f(x)$ , évaluée à un point inconnu, avec l'expression  $(x - x_0) \cdot \dots \cdot (x - x_n)$  qui ne dépend que de la division  $\{x_0, \dots, x_n\}$ .

**Problème intéressante :** chercher, pour un  $n$  donné, la division de  $[a, b]$  pour laquelle

$$L = \max_{x \in [a, b]} |(x - x_0) \cdot \dots \cdot (x - x_n)| \quad \text{est minimal.} \quad (3.1)$$

**Exemple 3.1** Considérons l'intervalle  $[-1, 1]$ , le cas  $n = 2$ , et une distribution symétrique des  $x_i$ . On cherche alors  $p(x) = (x + \alpha)x(x - \alpha) = x^3 - ax$  satisfaisant (3.1). Nous voyons en figure II.3 que, pour  $a = 0$ , nous avons  $L = 1$ ; puis cette valeur diminue quand  $a$  croît, jusqu'au moment où la courbe touche les bornes  $+L$  et  $-L$  à l'intérieur de  $[-1, 1]$  (pour  $a = 3/4$ ); après,  $L$  recommence de grandir. La solution optimale est donc  $p(x) = x^3 - 3x/4$ .

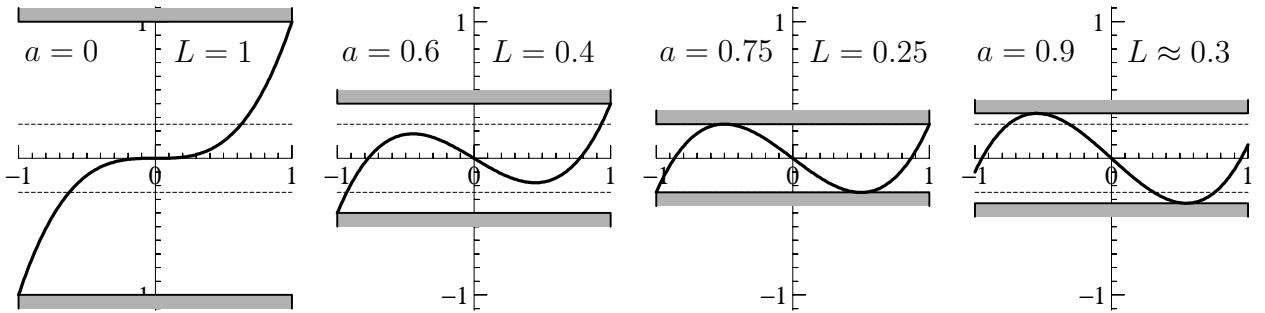


FIG. II.3: Valeurs maximales de  $\tau(x) = x^3 - ax$

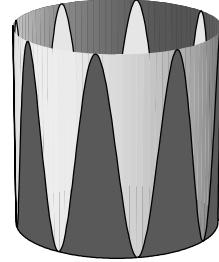
Pour  $n$  arbitraire, la réponse au problème posé se trouve dans un travail de P.L. Chebyshev (transcription française : “Tchebychef”, 1854, *Oeuvres I*, p. 109). Elle peut être donnée à l'aide de polynômes de Chebyshev; voir la figure II.4.<sup>1</sup>

**Définition 3.2 (Polynômes de Chebyshev)**

Pour  $n = 0, 1, 2, \dots$  et pour  $x \in [-1, 1]$ , on définit

$$T_n(x) = \cos(n \arccos x) \quad (3.2)$$

(projection du cosinus sur un tambour).



**Propriétés des polynômes de Chebyshev.**

a) Les fonctions  $T_n(x)$  satisfont la récurrence

$$T_0(x) = 1, \quad T_1(x) = x, \quad T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x). \quad (3.3)$$

Par conséquence,  $T_n(x)$  est un polynôme de degré  $n$  dont le coefficient de  $x^n$  est  $2^{n-1}$ , c.-à-d.,  $T_n(x) = 2^{n-1}x^n + \dots$

b)  $|T_n(x)| \leq 1$  pour  $x \in [-1, 1]$ .

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

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

<sup>1</sup>Pour une étude des “courbes blanches” dans la fig. II.4 (à droite) voir la page 209 du livre: Th.J. Rivlin, *Chebyshev Polynomials*. 2nd ed., John Wiley & Sons, 1990 [MA 41/36]

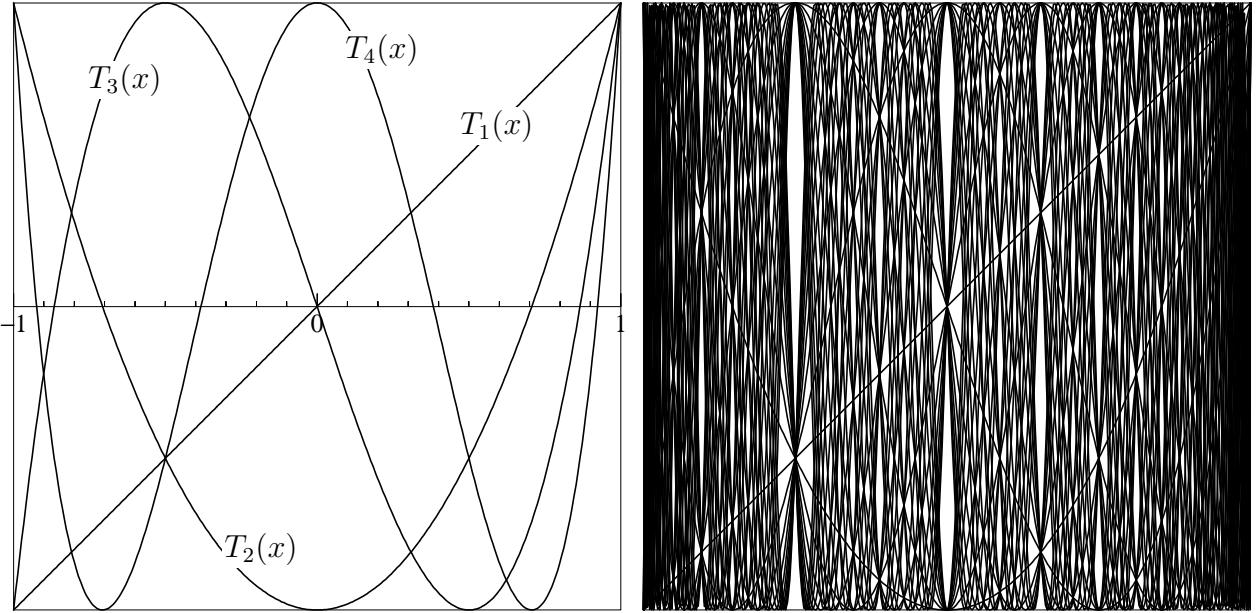


FIG. II.4: Les premiers 4 (à gauche) respectivement 30 (à droite) polynômes de Chebyshev

e) Les polynômes  $T_n(x)$  sont orthogonaux par rapport à la fonction de poids  $1/\sqrt{1-x^2}$

$$\int_{-1}^1 \frac{1}{\sqrt{1-x^2}} T_n(x) T_m(x) dx = \begin{cases} \pi & \text{si } n = m = 0 \\ \pi/2 & \text{si } n = m \neq 0 \\ 0 & \text{si } n \neq m. \end{cases}$$

*Démonstration.* La formule (3.3) est une conséquence de

$$\cos((n+1)\varphi) + \cos((n-1)\varphi) = 2 \cos \varphi \cdot \cos(n\varphi)$$

si l'on pose  $\cos \varphi = x$  et  $\varphi = \arccos x$ . La même transformation donne

$$\int_{-1}^1 \frac{1}{\sqrt{1-x^2}} T_n(x) T_m(x) dx = \int_0^\pi \cos(n\varphi) \cos(m\varphi) d\varphi$$

et la propriété (e) résulte de l'orthogonalité de  $\cos(n\varphi)$ .  $\square$

Revenons maintenant à la question de trouver une division satisfaisant à (3.1).

**Lemme 3.3** Soit  $q(x)$  un polynôme de degré  $n$  dont le coefficient de  $x^n$  est  $2^{n-1}$  (comme pour le polynôme de Chebyshev) et soit  $q(x) \not\equiv T_n(x)$ . Alors,

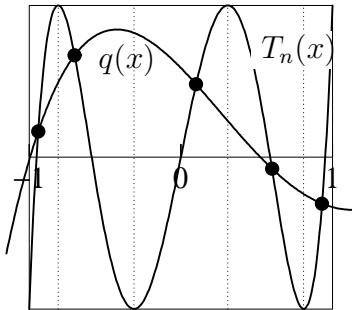
$$\max_{x \in [-1,1]} |q(x)| > \max_{x \in [-1,1]} |T_n(x)| = 1. \quad (3.4)$$

*Démonstration.* Supposons, par l'absurde, que

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

(voir la figure pour  $n = 5$ ) et considérons la différence  $d(x) = q(x) - T_n(x)$ . Cette fonction s'annule au moins une fois dans chacun des intervalles fermés

$$\left[ \cos\left(\frac{(k+1)\pi}{n}\right), \cos\left(\frac{k\pi}{n}\right) \right], \quad k = 0, 1, \dots, n-1.$$



(3.5)

Alors,  $d(x)$  possède  $n$  zéros dans  $[-1, 1]$  (si une racine  $\alpha \in (-1, 1)$  est à l'extrémité de l'intervalle (3.5), elle doit être comptée deux fois car à un tel point  $T'_n(\alpha) = 0$  et  $q'(\alpha) = 0$ ). Comme  $d(x)$  est un polynôme de degré  $n-1$  (le coefficient de  $x^n$  est le même pour  $q(x)$  et  $T_n(x)$ ), ceci est une contradiction à  $d(x) \not\equiv 0$ .  $\square$

Le lemme précédent montre que

$$\max_{x \in [-1,1]} |(x - x_0) \cdot \dots \cdot (x - x_n)| \quad \text{est minimal}$$

si et seulement si  $(x - x_0) \cdot \dots \cdot (x - x_n) = 2^{-n}T_{n+1}(x)$ , c.-à-d., si

$$x_k = \cos\left(\frac{(2k+1)\pi}{2n+2}\right), \quad k = 0, 1, \dots, n \quad (3.6)$$

(points de Chebyshev). Pour répondre à la question (3.1), il faut encore utiliser la translation  $x \mapsto \frac{a+b}{2} + \frac{b-a}{2}x$ , qui envoie l'intervalle  $[-1, 1]$  sur  $[a, b]$ . On obtient alors

**Théorème 3.4** *L'expression (3.1) est minimale parmi toutes les divisions  $\{x_0, x_1, \dots, x_n\}$  si et seulement si*

$$x_k = \frac{a+b}{2} + \frac{b-a}{2} \cdot \cos\left(\frac{(2k+1)\pi}{2n+2}\right), \quad k = 0, 1, \dots, n. \quad (3.7)$$

**Exemple 3.5** Considérons la fonction  $f(x) = 1/(1+25x^2)$  sur l'intervalle  $[-1, 1]$  (cf. la fig. II.2). Dans la fig. II.5, on compare le polynôme d'interpolation basé sur des points équidistants ( $n = 8$ ) avec celui basé sur les points de Chebyshev (aussi  $n = 8$ ). On observe une nette amélioration.

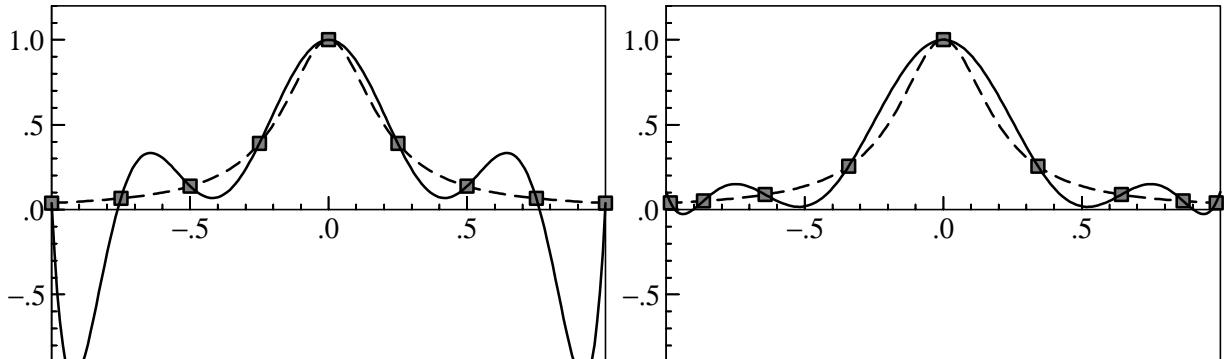


FIG. II.5: Interpolation avec des points équidistants (à gauche) et les points de Chebyshev (à droite)

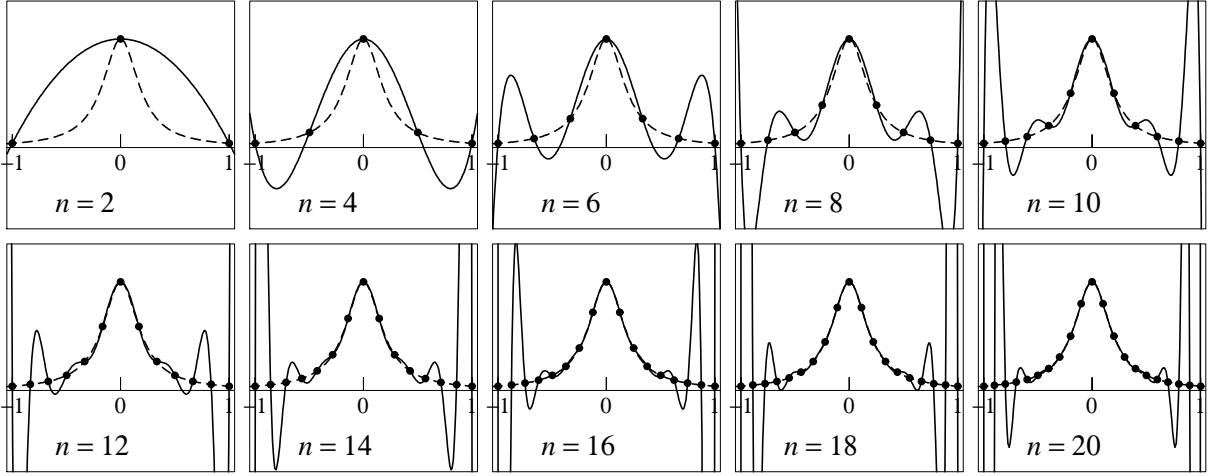
## II.4 Convergence de l'interpolation

La précision du polynôme d'interpolation (figure II.5 à gauche) est certainement insuffisante, quand  $x$  se rapproche des bords. On pourrait naïvement croire que le degré  $n$  n'est pas encore assez élevé. Mais le contraire se produit : quand on augmente  $n$ , la catastrophe s'intensifie davantage. C'est le *phénomène de Runge* (voir le dessin de la figure II.6).

**Le phénomène de Runge (1901).**<sup>2</sup> Nous considérons des fonctions rationnelles (ou méromorphes)  $f(x)$ , définies sur l'intervalle  $[-1, 1]$ , et la suite des divisions *équidistantes*

$$x_k^{(n)} = -1 + \frac{2k}{n}, \quad k = 0, 1, \dots, n. \quad (4.1)$$

<sup>2</sup>Carl David Tolmé Runge (1856-1927) est le père des mathématiques appliquées en Allemagne. L'article sur ce phénomène a été publié en 1901 dans le journal *Zeitschr. Math. u. Physik* vol. 46.

FIG. II.6: Le phénomène de Runge pour la fonction  $f(x) = 1/(1 + 25x^2)$ 

Nous notons par  $p_n(x)$  le polynôme d'interpolation de degré  $n$  satisfaisant à  $p(x_k^{(n)}) = f(x_k^{(n)})$  pour  $k = 0, 1, \dots, n$ . Le but est d'étudier la convergence de  $p_n(x)$  vers  $f(x)$ . On peut observer dans la fig. II.6 que, dans un certain intervalle autour de l'origine,  $p_n(x)$  converge vers  $f(x)$ , mais au bord de l'intervalle  $[-1, 1]$ , on a divergence.

**Excursion en analyse complexe.** Le meilleur moyen de comprendre ce phénomène est la formule de Cauchy (1831), voir le cours “Analyse II”,

$$f(x) = \frac{1}{2\pi i} \int_{\gamma} \frac{f(z)}{z-x} dz. \quad (4.2)$$

Dans cette formule,  $\gamma$  est une courbe fermée autour de  $x$ , telle que  $f(z)$  n'a pas de singularité à l'intérieur de la courbe. Si la courbe est donnée par la paramétrisation  $\gamma : [a, b] \rightarrow \mathbb{C}$  avec  $\gamma(a) = \gamma(b)$ , l'intégrale dans (4.2) est définie par

$$\frac{1}{2\pi i} \int_{\gamma} \frac{f(z)}{z-x} dz = \frac{1}{2\pi i} \int_a^b \frac{f(\gamma(t))}{\gamma(t)-x} \cdot \gamma'(t) dt.$$

**Une formule pour le polynôme d'interpolation.** En utilisant la notation

$$\pi_n(x) := (x - x_0^{(n)}) \cdot (x - x_1^{(n)}) \cdot \dots \cdot (x - x_n^{(n)}) \quad (4.3)$$

pour la division équidistante (4.1), nous obtenons pour le polynôme d'interpolation

$$p_n(x) = \frac{1}{2\pi i} \int_{\gamma} \frac{f(z)}{z-x} \cdot \frac{\pi_n(z) - \pi_n(x)}{\pi_n(z)} dz. \quad (4.4)$$

Ici,  $\gamma$  est une courbe fermée autour du segment  $[-1, 1]$  telle que  $f(z)$  n'ait pas de singularité (pôle) à l'intérieur de la courbe (voir la petite figure dans (4.2)).

En effet, la partie droite de (4.4) est un polynôme de degré  $n$  en  $x$  car  $(\pi_n(z) - \pi_n(x))/(z-x)$  en est un. En posant  $x = x_k^{(n)}$  pour un  $k \in \{0, \dots, n\}$ , on a

$$\frac{1}{2\pi i} \int_{\gamma} \frac{f(z)}{z-x_k^{(n)}} \cdot \frac{\pi_n(z) - \pi_n(x_k^{(n)})}{\pi_n(z)} dz = \frac{1}{2\pi i} \int_{\gamma} \frac{f(z)}{z-x_k^{(n)}} dz = f(x_k^{(n)}),$$

ce qui montre la formule (4.4). La différence de (4.4) et de (4.2) nous donne la formule suivante pour l'erreur de l'interpolation :

$$f(x) - p_n(x) = \frac{1}{2\pi i} \int_{\gamma} \frac{f(z)}{z-x} \cdot \frac{\pi_n(x)}{\pi_n(z)} dz. \quad (4.5)$$

Il est important de se rappeler que  $\gamma$  est une courbe fermée autour du segment  $[-1, 1]$  telle que  $f(z)$  n'ait pas de singularité à l'intérieur de  $\gamma$ .

**Étude de la convergence.** La formule (4.5) nous permet de procéder de la manière suivante : si, pour un  $x \in [-1, 1]$ , on peut choisir la courbe  $\gamma$  telle que

$$|\pi_n(x)| \leq \alpha^n |\pi_n(z)| \quad \text{pour tout } z \in \gamma \quad (4.6)$$

avec un  $\alpha < 1$ , alors,

$$|f(x) - p_n(x)| \leq \frac{\alpha^n}{2\pi} \int_{\gamma} \frac{|f(z)|}{|z-x|} |dz| \quad (4.7)$$

et on a convergence pour  $n \rightarrow \infty$ . Il faut alors étudier la fonction  $\sqrt[n]{|\pi_n(z)|}$  pour  $n \rightarrow \infty$ .

**Lemme 4.1** Pour la division équidistante (4.1) et pour le polynôme  $\pi_n(z)$  de (4.3) on a

$$\lim_{n \rightarrow \infty} \sqrt[n]{|\pi_n(z)|} = G(z)$$

où

$$G(z) = \exp\left(\frac{1}{2}\Re((z+1)\log(z+1) - (z-1)\log(z-1)) - 1\right). \quad (4.8)$$

*Démonstration.* En prenant le logarithme de  $\sqrt[n]{|\pi_n(z)|}$ , on obtient

$$\log \sqrt[n]{|\pi_n(z)|} = \frac{1}{n} \log |\pi_n(z)| = \frac{1}{n} (\log |z - x_0^{(n)}| + \dots + \log |z - x_n^{(n)}|),$$

ce qui représente une somme de Riemann pour la fonction  $t \mapsto \frac{1}{2} \log |z-t|$ ,  $x_{k+1}^{(n)} - x_k^{(n)} = 2/n$ . On obtient ainsi (observer que  $\log w = \log |w| + i \arg w$ )

$$\lim_{n \rightarrow \infty} \log \sqrt[n]{|\pi_n(z)|} = \frac{1}{2} \int_{-1}^1 \log |z-t| dt = \frac{1}{2} \Re \int_{-1}^1 \log(z-t) dt.$$

En considérant  $z$  comme paramètre, une primitive de la fonction  $g(t) = \log(z-t)$  est donnée par  $(z-t) - (z-t) \log(z-t)$ . Ceci implique

$$\lim_{n \rightarrow \infty} \log \sqrt[n]{|\pi_n(z)|} = \frac{1}{2} \Re ((z+1)\log(z+1) - (z-1)\log(z-1) - (z+1) + (z-1)),$$

et démontre la formule du lemme. □

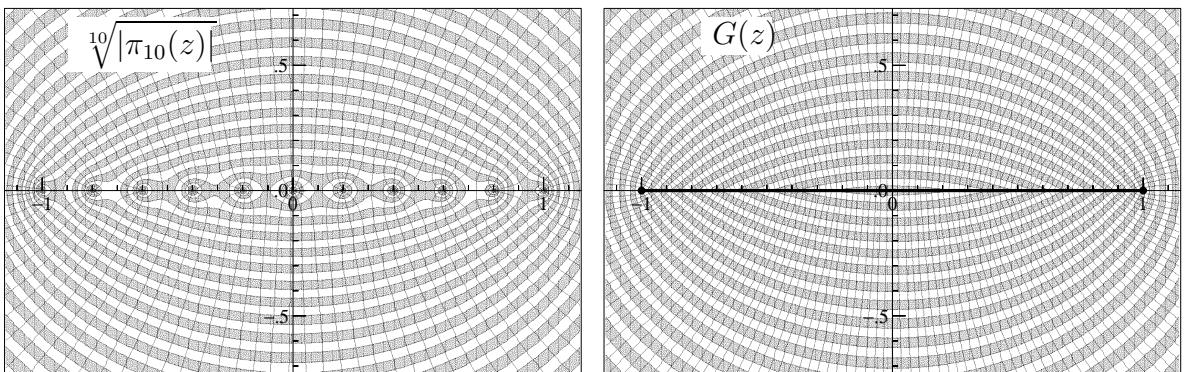
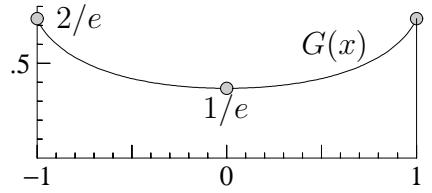


FIG. II.7: Lignes de niveau pour les fonctions  $\sqrt[10]{|\pi_{10}(z)|}$  et  $G(z)$ ; division équidistante.

Pour un  $x$  réel et compris dans  $(-1, 1)$ , on a

$$G(x) = e^{-1} \sqrt{(1+x)^{1+x}(1-x)^{1-x}},$$

en particulier  $G(0) = 1/e$ ,  $G(\pm 1) = 2/e$ . Les lignes de niveau  $\{z \mid G(z) = \text{Const}\}$  sont dessinées dans la fig. II.7 (dessin à droite).



**Théorème 4.2 (Runge 1901)** Si  $f(z)$  n'a pas de singularité dans  $\{z \mid G(z) \leq G(\beta)\}$ , alors

$$\lim_{n \rightarrow \infty} p_n(x) = f(x) \quad \text{pour } x \in (-\beta, \beta).$$

*Démonstration.* Nous considérons une courbe  $\gamma$  telle que l'intervalle  $[-1, 1]$  ainsi que l'ensemble  $\{z \mid G(z) \leq G(\beta)\}$  sont à l'intérieur, mais tous les pôles de  $f(z)$  à l'extérieur. Pour un  $x \in (-\beta, \beta)$  fixé, ceci entraîne que

$$G(x) < G(\beta) \leq \min_{z \in \gamma} G(z).$$

Comme  $\sqrt[n]{|\pi_n(x)|} \rightarrow G(x)$  et  $\min_{z \in \gamma} \sqrt[n]{|\pi_n(z)|} \rightarrow \min_{z \in \gamma} G(z)$  (convergence uniforme sur  $\gamma$ ; pour le voir il faut élaborer la démonstration du lemme 4.1), on peut trouver un  $\alpha$  ( $0 < \alpha < 1$ ) tel que, pour  $n$  suffisamment grand, on a

$$\sqrt[n]{|\pi_n(x)|} \leq \alpha \cdot \sqrt[n]{|\pi_n(z)|} \quad \text{pour } z \in \gamma.$$

Ceci vérifie (4.6) pour  $x \in (-\beta, \beta)$  et la convergence est une conséquence de (4.7).  $\square$

*Exemple.* La fonction  $f(x) = 1/(1+25x^2)$  possède une singularité en  $\pm i/5$ . La ligne de niveau de  $G(z)$  qui passe par  $\pm i/5$  coupe l'axe réel au point  $\beta \approx 0.726$ . Alors, la convergence du polynôme d'interpolation (avec des points équidistants) est établie pour  $|x| < 0.726$  (voir fig. II.6).

**Convergence en utilisant des points de Chebyshev.** Faisons la même étude en remplaçant la division équidistante (4.1) par les points de Chebyshev

$$x_k^{(n)} = \cos\left(\frac{(2k+1)\pi}{2n+2}\right), \quad k = 0, 1, \dots, n. \quad (4.9)$$

L'expérience de la fig. II.6 avec la fonction  $f(x) = 1/(1+25x^2)$  est répétée dans la fig. II.8. On observe une convergence uniforme sur toute l'intervalle  $[-1, 1]$ .

Pour comprendre l'absence d'un "phénomène de Runge", considérons le polynôme  $\pi_n(x)$  de (4.3) mais pour la division (4.9). Dans ce cas (voir (3.2) et (3.6))

$$\pi_n(z) = 2^{-n} T_{n+1}(z) = 2^{-n} \cos((n+1)\varphi) = \frac{u^{n+1} + u^{-(n+1)}}{2^{n+1}} \quad (4.10)$$

où  $\varphi = \arccos z$  et  $u = e^{i\varphi}$ .

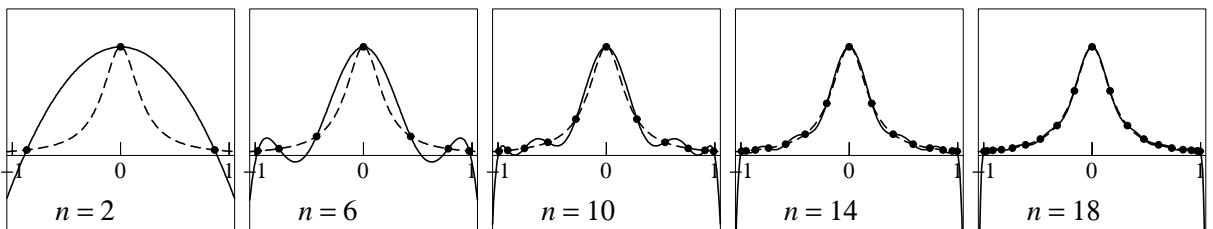


FIG. II.8: Interpolation de la fonction  $f(x) = 1/(1+25x^2)$  en utilisant des points de Chebyshev.

La relation  $z = \cos \varphi = \frac{1}{2}(e^{i\varphi} + e^{-i\varphi}) = \frac{1}{2}(u + u^{-1})$  donne l'équation  $u^2 - 2uz + 1 = 0$  pour  $u$ , dont les solutions sont  $u = z + \sqrt{z^2 - 1}$ ,  $u^{-1} = z - \sqrt{z^2 - 1}$ . Élevées à la  $(n+1)$ -ème puissance, la plus grande de ces grandeurs va finir par dominer l'autre. Ainsi,

$$G(z) = \lim_{n \rightarrow \infty} \sqrt[n]{|\pi_n(z)|} = \frac{1}{2} \max\{|z + \sqrt{z^2 - 1}|, |z - \sqrt{z^2 - 1}|\}. \quad (4.11)$$

Les lignes de niveau de la fonction limite deviennent parallèles à l'intervalle  $[-1, 1]$  (voir fig. II.9). Ainsi, un pôle se trouvant en dehors de l'intervalle  $[-1, 1]$  ne peut empêcher le polynôme  $p_n(x)$  de converger sur tout l'intervalle (voir fig. II.8).

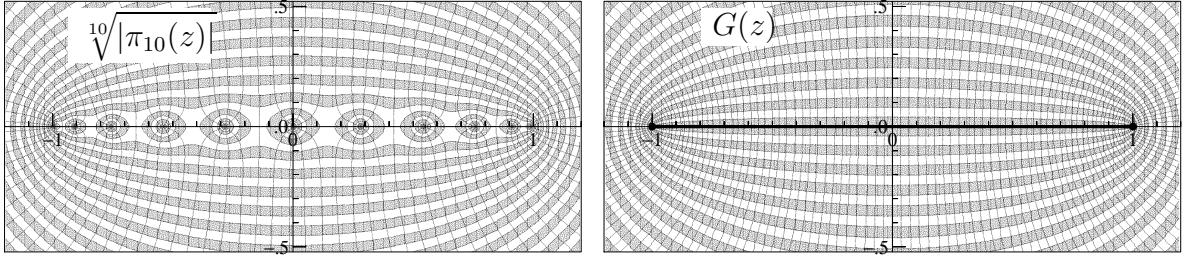


FIG. II.9: Lignes de niveau pour les fonctions  $\sqrt[10]{|\pi_{10}(z)|}$  et  $G(z)$ ; points de Chebyshev.

*Remarque.* La convergence (uniforme) de la suite de polynômes d'interpolation avec les points de Chebyshev peut être démontré pour toute fonction  $f(x)$  qui est continûment différentiable sur  $[-1, 1]$ . La *démonstration* utilise le théorème de Stone-Weierstraß (voir par exemple le chapitre II du livre de Werner & Schaback<sup>3</sup> pour plus de détails).

## II.5 Influence des erreurs d'arrondi sur l'interpolation

**Représentation en virgule flottante.** Chaque nombre réel  $x \neq 0$  peut s'écrire sous la forme

$$x = \pm a \cdot 10^b \quad (5.1)$$

où  $a$ , la *mantisso*, satisfait  $0.1 \leq a < 1$  et l'*exposant*  $b$  est un nombre entier. Cette représentation de  $x$  est unique. Supposons maintenant qu'on ait seulement un nombre fini de chiffres (disons  $\ell$ ) à disposition pour la mantisse, et pas de restriction pour l'exposant. Si  $\bar{a}$  dénote l'arrondi de  $a$ , on va calculer en fait avec le nombre

$$\text{arr}(x) = \pm \bar{a} \cdot 10^b$$

au lieu de  $x$ . Par exemple, le nombre  $\pi = 3.141592653\dots$  est représenté par

$$\text{arr}(\pi) = 0.31415927 \cdot 10^1 \quad (5.2)$$

si l'on calcule avec  $\ell = 8$  chiffres en base 10.

**Précision de l'ordinateur.** On dénote par  $\text{eps}$  le plus petit nombre positif tel que

$$\text{arr}(1 + \text{eps}) > 1.$$

Pour un calcul en base 10 avec  $\ell$  chiffres dans la mantisse on a

$$\begin{aligned} \text{arr}(0.\underbrace{10\dots 0}_{\ell}49\dots \cdot 10^1) &= 1 \\ \text{arr}(0.\underbrace{10\dots 0}_{\ell}50\dots \cdot 10^1) &= 0.\underbrace{10\dots 1}_{\ell} \cdot 10^1 > 1. \end{aligned}$$

<sup>3</sup>H. Werner & R. Schaback (1979), *Praktische Mathematik II*. Springer-Verlag, 2. Auflage. [MA 65/23]

Dans cette situation, on a alors

$$\text{eps} = 5 \cdot 10^{-\ell}. \quad (5.3)$$

Si l'on fait le même calcul en base 2 (comme tous les ordinateurs le font) on obtient

$$\text{eps} = 2^{-\ell}. \quad (5.4)$$

Par exemple, sur une SUN workstation on a

$$\begin{aligned} \text{REAL * 4}, \quad \text{eps} &= 2^{-24} \approx 5.96 \cdot 10^{-8} \\ \text{REAL * 8}, \quad \text{eps} &= 2^{-53} \approx 1.11 \cdot 10^{-16} \\ \text{REAL * 16}, \quad \text{eps} &= 2^{-113} \approx 9.63 \cdot 10^{-35}. \end{aligned}$$

**Théorème 5.1** *Pour un  $x \neq 0$  on a*

$$\frac{|\text{arr}(x) - x|}{|x|} \leq \text{eps}, \quad (5.5)$$

c.-à-d., l'erreur relative due à l'arrondissement est bornée par  $\text{eps}$ .

*Démonstration.* Soit  $x = a \cdot 10^b$  et  $\text{arr}(x) = \bar{a} \cdot 10^b$ . Si l'on arrondit à  $\ell$  chiffres significatifs, on a  $|\bar{a} - a| \leq 5 \cdot 10^{-\ell-1}$ . Il en résulte

$$\frac{|\text{arr}(x) - x|}{|x|} = \frac{|\bar{a} - a| \cdot 10^b}{|a| \cdot 10^b} \leq \frac{5 \cdot 10^{-\ell-1}}{10^{-1}} = 5 \cdot 10^{-\ell} = \text{eps}$$

car  $|a| \geq 1/10$ . □

L'estimation (5.5) peut aussi être écrite sous la forme

$$\text{arr}(x) = x(1 + \epsilon) \quad \text{où} \quad |\epsilon| \leq \text{eps}. \quad (5.6)$$

Cette formule est la base pour toute étude d'erreurs d'arrondi.

**Influence des erreurs dans  $y_i$  sur le polynôme d'interpolation.** Supposons que les données  $y_i$  soient erronées et qu'on calcule en fait avec

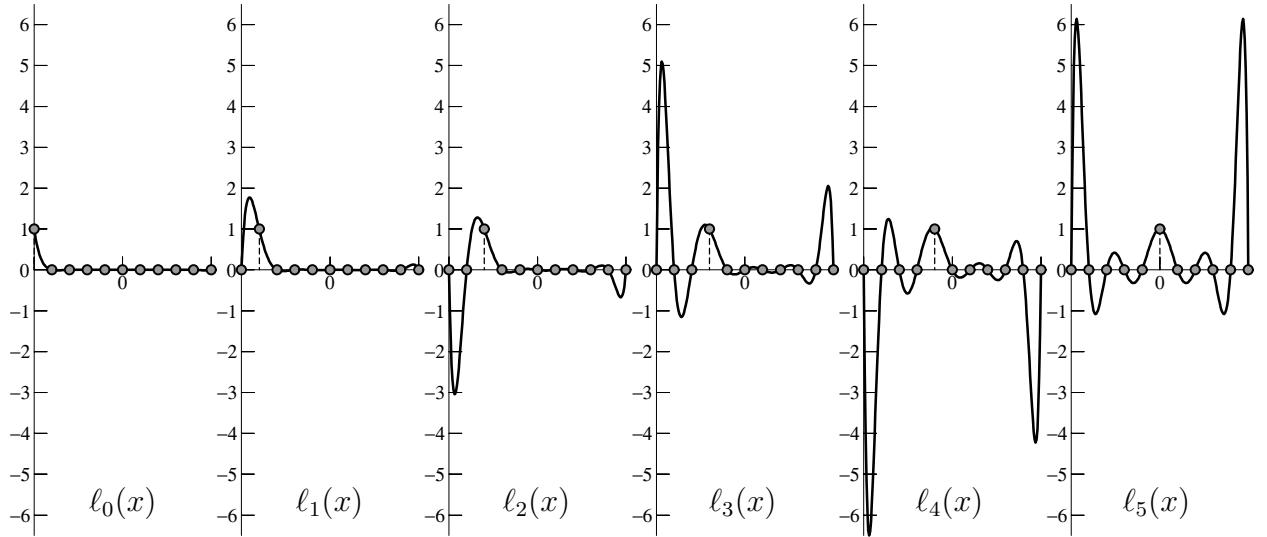
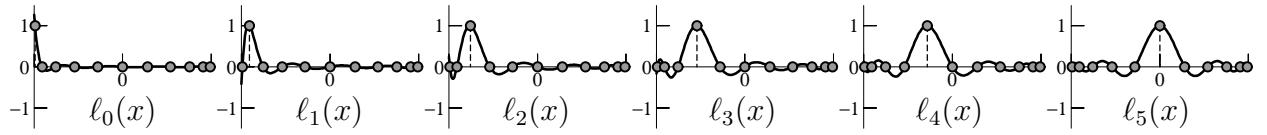
$$\hat{y}_i = y_i(1 + \epsilon_i) \quad \text{où} \quad |\epsilon_i| \leq \text{eps}. \quad (5.7)$$

Pour étudier la différence entre le polynôme qui passe par  $(x_i, y_i)$  et celui qui passe par  $(x_i, \hat{y}_i)$ , on utilise la formule du lemme suivant.

**Lemme 5.2 (formule de Lagrange)** *Le polynôme d'interpolation  $p(x)$  qui passe par  $(x_i, y_i)$  pour  $i = 0, 1, \dots, n$  est donné par*

$$p(x) = \sum_{i=0}^n y_i \ell_i(x) \quad \text{où} \quad \ell_i(x) = \prod_{\substack{j=0 \\ j \neq i}}^n \frac{x - x_j}{x_i - x_j}. \quad (5.8)$$

*Démonstration.* Le polynôme  $\ell_i(x)$  satisfait  $\ell_i(x_i) = 1$  et  $\ell_i(x_k) = 0$  si  $k \neq i$ . Ainsi, les deux côtés de la formule (5.8) valent  $y_k$  pour  $x = x_k$  ( $k = 0, 1, \dots, n$ ). Comme  $p(x)$  et  $\sum_{i=0}^n y_i \ell_i(x)$  sont des polynômes de degré  $n$ , cette formule est une conséquence de l'unicité du polynôme d'interpolation (voir le théorème 1.2). □

FIG. II.10: Polynômes de Lagrange à points équidistants pour  $n = 10$ FIG. II.11: Polynômes de Lagrange à points de Chebyshev pour  $n = 10$ 

Les polynômes passant par  $(x_i, y_i)$  et  $(x_i, \hat{y}_i)$  sont respectivement

$$p(x) = \sum_{i=0}^n y_i \ell_i(x) \quad \text{et} \quad \hat{p}(x) = \sum_{i=0}^n \hat{y}_i \ell_i(x).$$

Si  $\hat{y}_i$  est donné par (5.7), la différence satisfait  $\hat{p}(x) - p(x) = \sum_{i=0}^n \epsilon_i y_i \ell_i(x)$  et on obtient

$$|\hat{p}(x) - p(x)| \leq \text{eps} \cdot \max_{k=0, \dots, n} |y_k| \cdot \sum_{i=0}^n |\ell_i(x)|. \quad (5.9)$$

La fonction  $\sum_{i=0}^n |\ell_i(x)|$  décrit l'amplification de l'erreur dans les données. Sa valeur maximale

$$\Lambda_n := \max_{x \in [a, b]} \sum_{i=0}^n |\ell_i(x)| \quad (5.10)$$

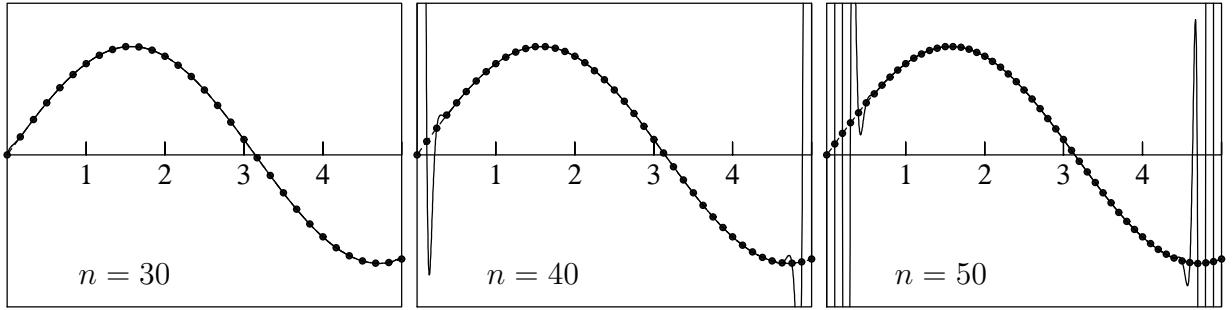
s'appelle la *constante de Lebesgue* associée aux points  $x_0, x_1, \dots, x_n$  et à l'intervalle  $[a, b]$ . Cette constante peut être calculée numériquement (voir les exercices 9 and 10).

**Points équidistants.** Pour la division  $x_k = a + \frac{k}{n}(b - a)$  de l'intervalle  $[a, b]$ , on a

$$\Lambda_{20} \approx 3 \cdot 10^4, \quad \Lambda_{40} \approx 10^{10}, \quad \Lambda_n \approx \frac{2^{n+1}}{e \cdot n \cdot \log n}.$$

**Points de Chebyshev.** Si l'on choisit  $x_k = \frac{a+b}{2} + \frac{b-a}{2} \cos\left(\frac{(2k+1)\pi}{2n+2}\right)$ , on a

$$\begin{aligned} \Lambda_n &\leq 3 & \text{pour } n \leq 20 \\ \Lambda_n &\leq 4 & \text{pour } n \leq 100 \\ \Lambda_n &\approx \frac{2}{\pi} \log n & \text{si } n \text{ est grand.} \end{aligned}$$

FIG. II.12: Influence des erreurs dans les  $y_i$  sur l'interpolation

*Expérience numérique.* Considérons la fonction  $f(x) = \sin x$  sur l'intervalle  $[0, 5]$ . Pour  $n \geq 25$ , l'erreur de l'interpolation est bornée par  $5^{26}/26! \approx 3.69 \cdot 10^{-9}$  quelle que soit la division choisie (voir le théorème 2.2). Prenons les  $y_i = \sin x_i$  en simple précision (erreur relative  $\approx \text{eps} \approx 5.96 \cdot 10^{-8}$ ) et faisons le calcul de  $p(x)$  en double précision. Dans la fig. II.12 nous voyons le polynôme d'interpolation obtenu de cette manière (avec des points équidistants). Son erreur devient bien visible à partir de  $n = 32$ , la valeur où  $\text{eps} \cdot \Lambda_n$  dépasse 1. Evidemment, l'interpolation avec les points de Chebyshev ne montrera pas ce phénomène.

## II.6 Transformée de Fourier discrète (DFT)

Dans le traitement de signaux, où on est confronté à une immense quantité (plusieurs milliers ou millions) de valeurs numériques, la transformée de Fourier est un outil inévitable. De plus, les données ont souvent une certaine périodicité ce qui rend la transformée plus efficace. Elle est, par exemple, très utilisée dans le traitement des sons (voir la fig. II.13) ainsi que pour la compression d'images (voir le paragraphe II.7).

**Transformée de Fourier (continue) ou série de Fourier.** Une série trigonométrique (ou série de Fourier) est une série de la forme

$$f(x) = \frac{a_0}{2} + \sum_{k=1}^{\infty} (a_k \cos kx + b_k \sin kx). \quad (6.1)$$

En cas de convergence, elle représente une fonction  $2\pi$ -périodique, c.-à-d.  $f(x+2\pi) = f(x)$  pour tout  $x \in \mathbb{R}$ . Les formules deviennent plus simples en passant aux complexes. Grâce aux identités  $e^{ix} = \cos x + i \sin x$ ,  $\cos x = (e^{ix} + e^{-ix})/2$  et  $\sin x = (e^{ix} - e^{-ix})/2i$ , la série précédente devient simplement

$$f(x) = \sum_{k=-\infty}^{\infty} c_k e^{ikx}. \quad (6.2)$$

La clé fondamentale permettant le calcul des séries trigonométriques a été découverte par Euler (1777, *Opera vol 16, Pars I*, p. 333) et réside dans la *relation d'orthogonalité*

$$\int_0^{2\pi} e^{-i\ell x} \cdot e^{ikx} dx = \int_0^{2\pi} e^{i(k-\ell)x} dx = \begin{cases} 0 & \text{si } k \neq \ell \\ 2\pi & \text{si } k = \ell. \end{cases} \quad (6.3)$$

Cette propriété nous permet de calculer les coefficients  $c_k$ . Il suffit de multiplier (6.2) par  $e^{-i\ell x}$  et intégrer terme par terme<sup>4</sup> de 0 à  $2\pi$ . Tous les termes, sauf un, disparaissent et on obtient

$$c_k = \frac{1}{2\pi} \int_0^{2\pi} f(x) e^{-ikx} dx. \quad (6.4)$$

<sup>4</sup>Pour une justification voir les cours *Analyse II* ou *Mathématiques pour Informaticiens*.

Pour marquer la dépendance de  $f(x)$ , nous écrivons souvent  $\hat{f}(k) := c_k$  pour les coefficients de Fourier. Rappelons que les coefficients  $\hat{f}(k)$  convergent vers zéro pour  $k \rightarrow \infty$ , et on a  $\hat{f}(k) = \mathcal{O}(|k|^{-p})$  si le prolongement  $2\pi$ -périodique de  $f(x)$  est  $p$ -fois continûment différentiable.

**Transformée de Fourier discrète (DFT).** Supposons maintenant que la fonction  $2\pi$ -périodique  $f(x)$  soit seulement connue pour les  $x$  de la division équidistante

$$x_j = \frac{2\pi j}{N}, \quad j = 0, 1, \dots, N - 1$$

et posons  $y_j = f(x_j)$ . Si nécessaire, on peut prolonger  $\{y_j\}$  à une suite  $N$ -périodique en posant  $y_{j+N} = y_j$  pour tout entier  $j$  ( $N = 8$  dans la petite figure).

Par analogie avec (6.2) on cherche à exprimer cette suite par

$$y_j = \sum_{k=0}^{N-1} z_k e^{ikx_j} = \sum_{k=0}^{N-1} z_k \omega^{kj} \quad \text{avec} \quad \omega = e^{2\pi i/N}. \quad (6.5)$$

Cette fois, la *relation d'orthogonalité discrète* (observer que  $\omega^N = 1$ )

$$\sum_{j=0}^{N-1} \omega^{-\ell j} \cdot \omega^{kj} = \sum_{j=0}^{N-1} \omega^{(k-\ell)j} = \begin{cases} \frac{1-\omega^{(k-\ell)N}}{1-\omega^{k-\ell}} = 0 & \text{si } k \neq \ell \pmod{N} \\ N & \text{si } k = \ell \pmod{N} \end{cases} \quad (6.6)$$

nous aide à trouver les  $z_k$  à partir de (6.5). Par parfaite analogie avec la preuve ci-dessus pour (6.4), on multiplie l'équation (6.5) par  $\omega^{-\ell j}$  et on additionne de  $j = 0$  à  $j = N - 1$ . Tous les termes, sauf un, disparaissent et on obtient la *transformée de Fourier discrète* (DFT)

$$z_k = \frac{1}{N} \sum_{j=0}^{N-1} y_j \omega^{-kj}. \quad (6.7)$$

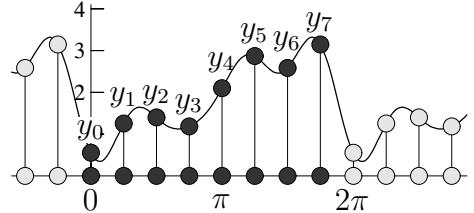
Si  $y_j = f(x_j)$ , nous écrivons aussi  $\hat{f}_N(k) := z_k$  pour les coefficients de la transformée de Fourier discrète. Comme  $y_N = y_0$ , la valeur  $z_k$  de (6.7) peut être interprétée comme le résultat de la règle du trapèze appliquée à l'intégrale dans (6.4). Toutefois,  $z_k = \hat{f}_N(k)$  n'est pas une approximation de  $c_k = \hat{f}(k)$  pour tout  $k$ , car  $\{z_k\}$  est une suite  $N$ -périodique (ceci est une conséquence immédiate de  $\omega^N = 1$ ) alors que  $\hat{f}(k)$  converge vers zéro si  $k \rightarrow \infty$  (voir le paragraphe II.9 pour une étude détaillée de l'erreur  $\hat{f}(k) - \hat{f}_N(k)$ ).

**Compression des données.** Étant donnée une suite  $N$ -périodique  $\{y_j\}$  et sa transformée de Fourier discrète  $\{z_k\}$ . L'idée est de supprimer dans la représentation (6.5) pour  $y_j$  tous les termes dont la valeur absolue de  $z_k$  est en-dessous d'un certain seuil (par exemple, 3% du coefficient maximal).

*Exemple.* Le premier dessin de la fig. II.13 montre la digitalisation d'un son. On a enregistré 22000 impulsions par seconde, dont 1024 sont dessinées (ceci correspond à  $1024/22 \approx 46.5$  millisecondes). On observe bien une certaine périodicité des données.

Pour les  $N = 1024$  nombres  $\{y_j\}$  on a calculé la transformée de Fourier discrète  $\{z_k\}$ . La suite de leurs valeurs absolues  $\{|z_k|\}$  est dessinée dans la deuxième image de la fig. II.13 pour  $k = 1, \dots, 170$  (comme les  $y_i$  sont réels, on a  $z_{N-k} = z_{-k} = \bar{z}_k$ , et il n'est pas nécessaire de dessiner les valeurs pour  $k \geq N/2$ ; pour  $170 < k < N/2$  les  $|z_k|$  sont inférieurs à 0.072).

La théorie de ce paragraphe est basée sur le fait que  $f(x)$  est une fonction périodique. Cependant, la période du signal n'est visiblement pas égale à  $N = 1024$ , mais elle est plutôt proche de



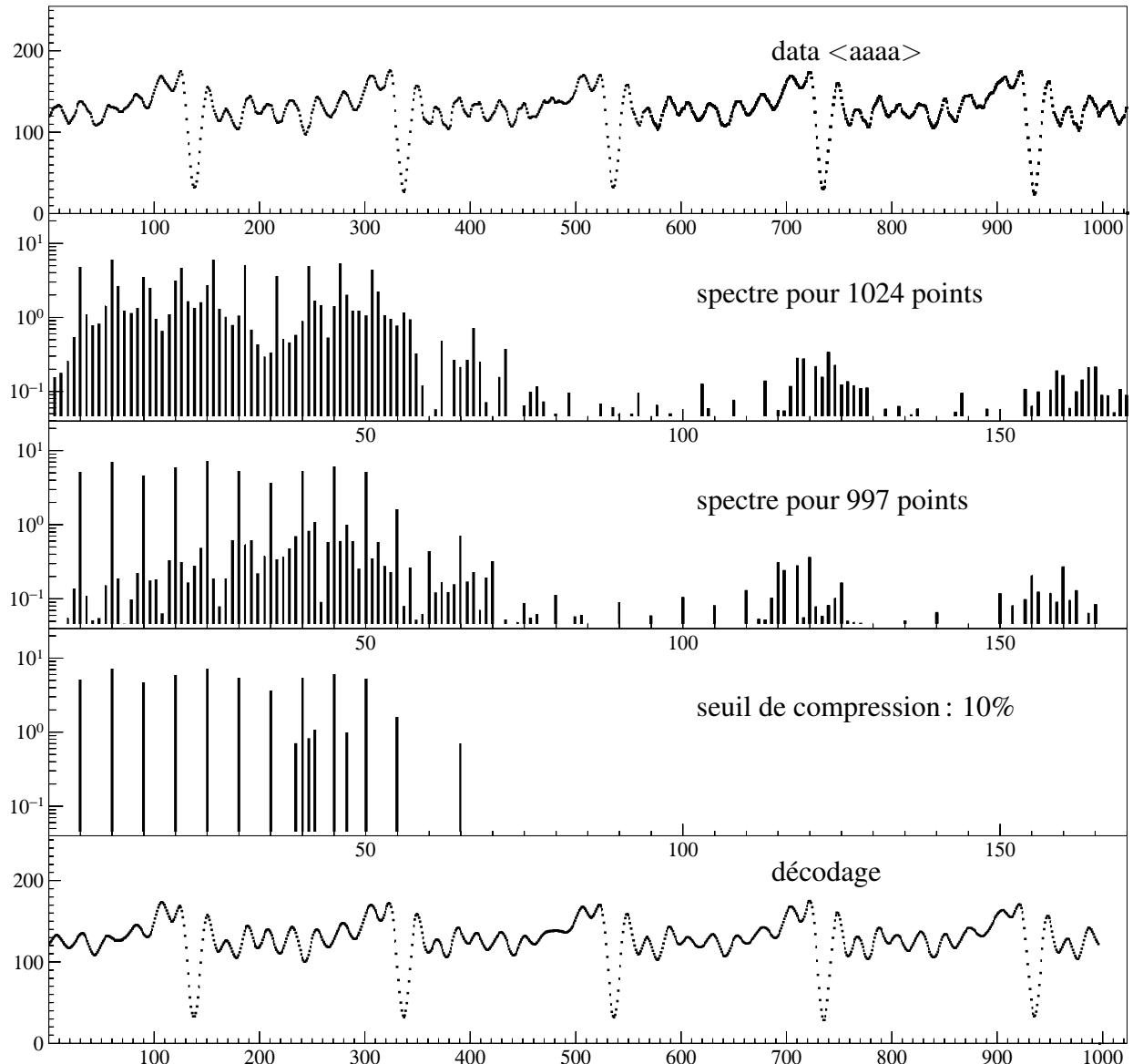


FIG. II.13: Spectrogramme, compression et décodage du son “a” prononcé par Martin

$N = 997$ . Si l'on calcule 1024 valeurs de  $f(x)$  sur une période exacte (par interpolation linéaire) ainsi que leur transformée de Fourier discrète, on obtient la troisième image de la fig. II.13. Cette fois, on peut beaucoup mieux observer la fréquence principale ( $5 * 22000 / 997 \approx 110$  Hz) ainsi que les harmoniques (multiples de la fréquence principale).

Maintenant nous supprimons tous les coefficients  $z_k$  dont la valeur absolue est inférieure à 10% de la valeur maximale. Les 16 coefficients restant sont dessinés dans le quatrième dessin de la fig. II.13. Pour un seuil de 3% ils resteraient 46 coefficients, et pour un seuil de 1% pas plus de 87. Ainsi, la vraie information contenue dans le signal ne contient que 16 (respectivement 46 ou 87) nombres complexes au lieu des 997 valeurs réelles  $y_j$ .

Pour décoder le signal, nous utilisons la formule (6.5) avec les  $z_k$  restant après la compression. Le résultat (décodage) est déssiné en bas de la fig. II.13. On peut constater que le signal original est très bien reproduit.

La conclusion de cette expérience est la suivante : au lieu de stocker les 997 valeurs du signal original, il suffit de stocker quelques coefficients de la transformée de Fourier discrète sans perdre de l'information visible.

## II.7 Transformée cosinus discrète (DCT) et JPEG

L'algorithme du paragraphe précédent marche très bien si les données ont une certaine périodicité. Si elles ne sont pas périodiques (par exemple pour la compression d'images), on utilise souvent une variante de la transformée de Fourier discrète. Cette variante a en plus l'avantage d'éviter le calcul avec des nombres complexes.

**Transformée de Fourier en cosinus.** Soit  $f(x)$  une fonction continue, définie sur l'intervalle  $[0, \pi]$ . On la prolonge en une fonction paire par  $f(-x) = f(x)$  et en une fonction  $2\pi$ -périodique par  $f(x + 2\pi) = f(x)$ . La série de Fourier (6.1) d'une telle fonction contient seulement les termes en cosinus et s'écrit

$$f(x) = \frac{a_0}{2} + \sum_{k=1}^{\infty} a_k \cos kx. \quad (7.1)$$

Sur l'intervalle  $[0, \pi]$ , les fonctions  $\cos kx$  (pour  $k \geq 0$ ) vérifient la *relation d'orthogonalité*

$$\int_0^\pi \cos \ell x \cdot \cos kx dx = \frac{1}{2} \int_0^\pi (\cos(\ell+k)x + \cos(\ell-k)x) dx = \begin{cases} 0 & \text{si } k \neq \ell \\ \pi/2 & \text{si } k = \ell \neq 0 \\ \pi & \text{si } k = \ell = 0. \end{cases} \quad (7.2)$$

La démarche habituelle (multiplier (7.1) par  $\cos \ell x$  et intégrer terme par terme de 0 à  $\pi$ ) nous donne

$$a_k = \frac{2}{\pi} \int_0^\pi f(x) \cos kx dx. \quad (7.3)$$

**Transformée cosinus discrète (DCT).** Comme  $f(0) \neq f(\pi)$  en général, nous considérons les  $N$  points au milieu des sous-intervalles, c.-à-d. les points

$$x_j = \frac{(2j+1)\pi}{2N}, \quad j = 0, 1, \dots, N-1$$

et nous posons  $y_j = f(x_j)$  ( $N = 4$  dans la petite figure). Par analogie avec (7.1), nous exprimons cette suite par (voir la fig. II.14 pour les fonctions de base  $\cos kx_j$ )

$$y_j = \frac{z_0}{2} + \sum_{k=1}^{N-1} z_k \cos kx_j. \quad (7.4)$$

Avec la *relation d'orthogonalité discrète* (pour  $0 \leq k, \ell \leq N-1$ )

$$\sum_{j=0}^{N-1} \cos \ell x_j \cdot \cos kx_j = \frac{1}{2} \sum_{j=0}^{N-1} (\cos(\ell+k)x_j + \cos(\ell-k)x_j) = \begin{cases} 0 & \text{si } k \neq \ell \\ N/2 & \text{si } k = \ell \neq 0 \\ N & \text{si } k = \ell = 0. \end{cases} \quad (7.5)$$

nous trouvons (multiplier l'équation (7.4) par  $\cos \ell x_j$  et additionner de  $j = 0$  à  $j = N-1$ ) la *transformée cosinus discrète (DCT)*

$$z_k = \frac{2}{N} \sum_{j=0}^{N-1} y_j \cos kx_j. \quad (7.6)$$

La valeur  $z_k$  de (7.6) est le résultat de la règle du point milieu appliquée à l'intégrale dans (7.3).

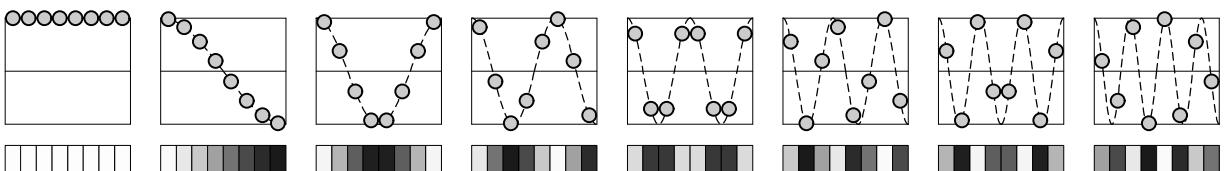


FIG. II.14: Fonctions de base pour la transformée cosinus discrète ( $N = 8$ )

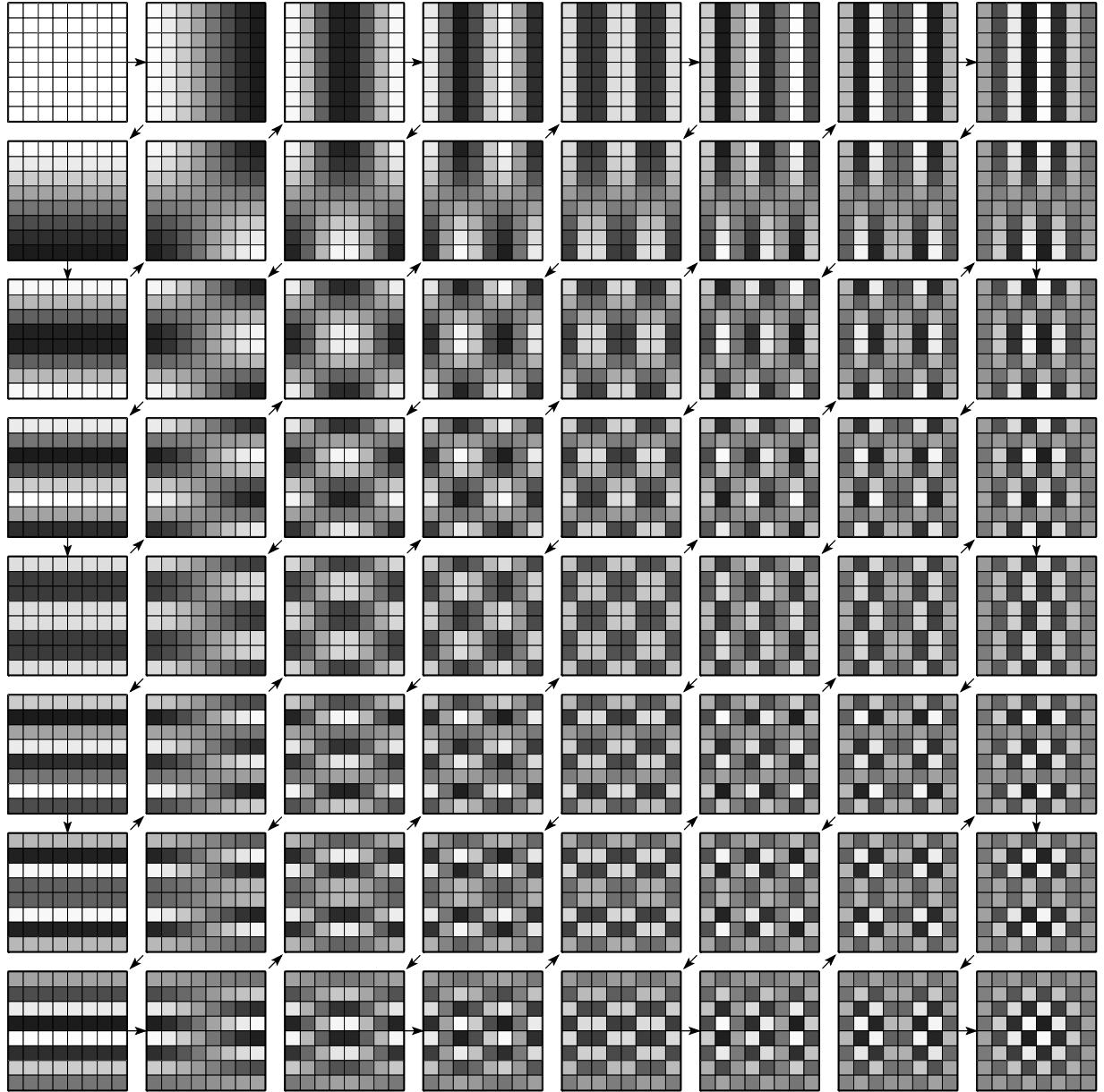


FIG. II.15: Fonctions de base pour la transformée cosinus discrète en 2 dimensions (8 × 8 pixels)

**Transformée cosinus discrète en dimension 2.** Une image digitale est donnée par un tableau  $\{Y_{i,j}\}$ , où  $i$  parcourt les pixels verticalement et  $j$  horizontalement. La valeur de  $Y_{i,j}$  représente le niveau de gris (ou la couleur) du pixel  $(i, j)$ . Motivé par l'algorithme précédent, nous essayons d'exprimer ces données par

$$Y_{i,j} = \sum_{k=0}^{N-1} \sum_{\ell=0}^{N-1} \tilde{Z}_{k,\ell} \cos kx_i \cos \ell x_j \quad (7.7)$$

où  $x_j = (2j+1)\pi/2N$  comme pour la transformée cosinus discrète. Pour compenser le facteur  $1/2$  dans (7.4), nous utilisons la notation  $\tilde{Z}_{0,0} = Z_{0,0}/4$ ,  $\tilde{Z}_{k,0} = Z_{k,0}/2$ ,  $\tilde{Z}_{0,\ell} = Z_{0,\ell}/2$  et  $\tilde{Z}_{k,\ell} = Z_{k,\ell}$ . La relation d'orthogonalité (7.5) nous permet de calculer les  $Z_{k,\ell}$  par la formule

$$Z_{k,\ell} = \frac{4}{N^2} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} Y_{i,j} \cos kx_i \cos \ell x_j. \quad (7.8)$$

Les fonctions de base  $\cos kx_i \cos \ell x_j$  ( $k, \ell = 0, \dots, N - 1$ ) sont illustrées dans la fig. II.15.

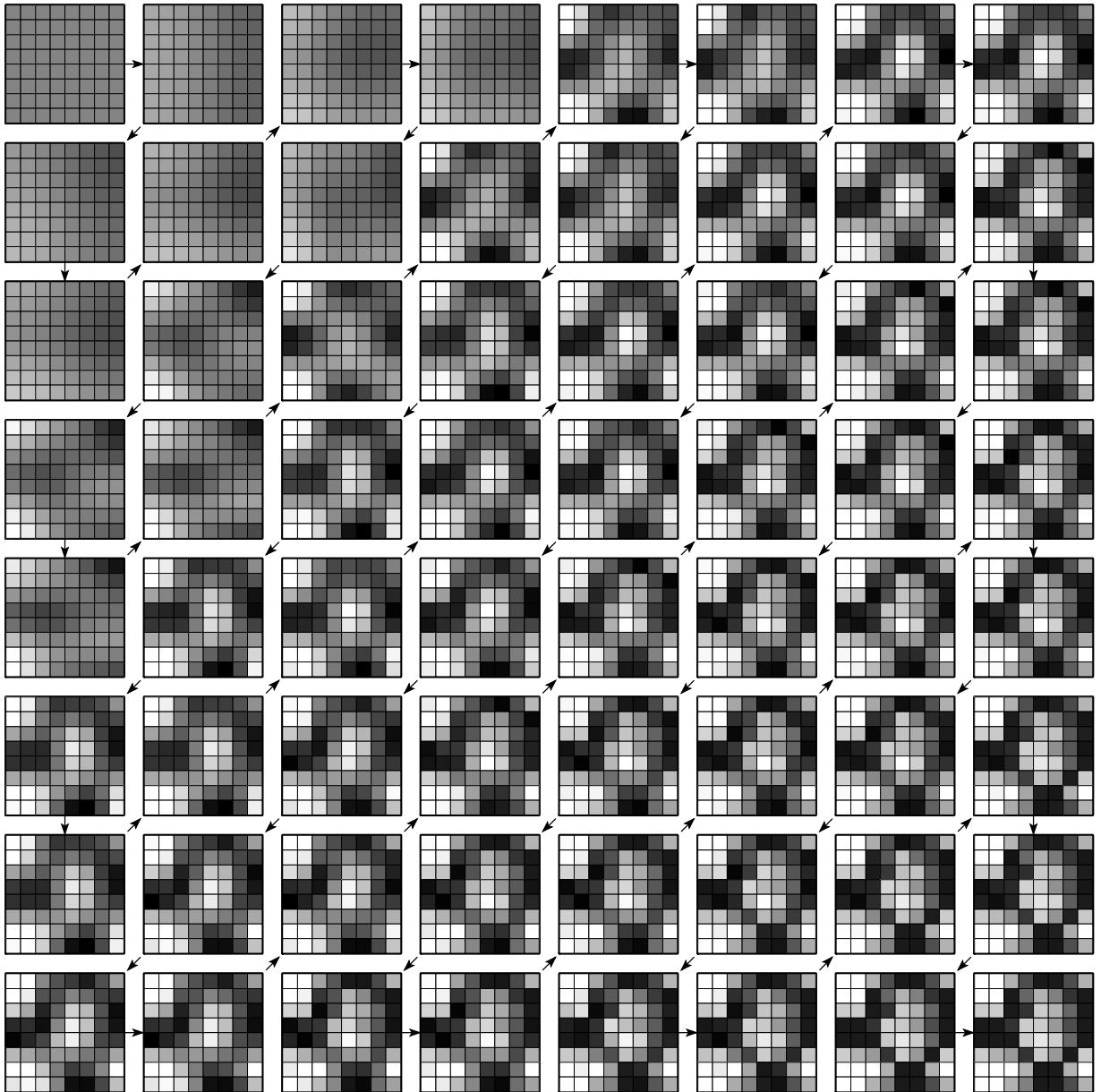
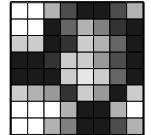


FIG. II.16: Une image toujours mieux reconstituée en zig-zag par JPEG

**JPEG.** L'utilisation de cette base dans la compression d'images est due à Ahmed, Natarajan et Rao (IEEE 1974)<sup>5</sup>. On décompose l'image entière en blocs de  $8 \times 8$  pixels et on calcule pour chaque bloc la transformée cosinus discrète (7.8). Les coefficients  $Z_{k,\ell}$  sont alors quantifiés et ceux qui sont en-dessous d'un seuil sont remplacés par zéro. Pour des images contenant des parties uniformes (par exemple : ciel bleu), seulement deux ou trois coefficients vont rester ce qui donne un important facteur de compression.

La reconstruction de l'image à droite (un conglomérat entre Astérix, Mickey Mouse et Donald Duck, dessiné par Gerhard) est illustrée dans la fig. II.16. Le premier dessin (en haut à gauche) correspond au premier terme de la somme (7.7). En rajoutant un terme après l'autre, et en suivant un chemin astucieux en zig-zag, la reconstruction est démontrée.



<sup>5</sup>Pour des explications plus détaillées et références voir *JPEG still image data compression standard* par W. B. Pennebaker et J. L. Mitchell, Van Nostrand Reinhold, New York, 1992.

## II.8 Transformée de Fourier rapide (FFT)

Pour  $\{y_j\}_{j=0}^{N-1}$  donnés, un calcul direct de la transformée de Fourier discrète  $\{z_k\}_{k=0}^{N-1}$  (voir (6.7)) nécessite  $\approx N^2$  multiplications et additions. Dans ce paragraphe, nous présentons un algorithme (un des “Top 10 Algorithms” du 20ème siècle) qui fait le même travail en  $N \log_2 N$  opérations. Cet algorithme est dû à Cooley & Tukey (1965); il est basé sur des idées de Runge (1925).

Nous supprimons le facteur  $1/N$  dans (6.7) et nous utilisons la notation

$$\mathcal{F}_N y = z \quad \text{ou} \quad z_k = \sum_{j=0}^{N-1} y_j \omega_N^{-kj}, \quad \omega_N = e^{2\pi i/N} \quad (8.1)$$

**Lemme 8.1** Soient  $u = (u_0, u_1, \dots, u_{N-1})$ ,  $v = (v_0, v_1, \dots, v_{N-1})$  et définissons

$$y = (u_0, v_0, u_1, v_1, \dots, u_{N-1}, v_{N-1}). \quad (8.2)$$

Alors, pour  $k = 0, 1, \dots, N - 1$ , on a ( $\omega_{2N} = e^{2\pi i / 2N} = e^{\pi i / N}$ )

$$\begin{aligned} (\mathcal{F}_{2N} y)_k &= (\mathcal{F}_N u)_k + \omega_{2N}^{-k} (\mathcal{F}_N v)_k \\ (\mathcal{F}_{2N} y)_{k+N} &= (\mathcal{F}_N u)_k - \omega_{2N}^{-k} (\mathcal{F}_N v)_k. \end{aligned} \tag{8.3}$$

*Démonstration.* En utilisant  $\omega_{2N}^2 = \omega_N$ , une séparation des indices paires et impaires nous donne pour un  $k$  arbitraire

$$(\mathcal{F}_{2N} y)_k = \sum_{j=0}^{2N-1} y_j \omega_{2N}^{-jk} = \sum_{\ell=0}^{N-1} \underbrace{y_{2\ell}}_{\mathcal{U}_\ell} \underbrace{\omega_{2N}^{-2\ell k}}_{\omega_N^{-\ell k}} + \sum_{\ell=0}^{N-1} \underbrace{y_{2\ell+1}}_{\mathcal{V}_\ell} \underbrace{\omega_{2N}^{-(2\ell+1)k}}_{\omega_{2N}^{-k} \cdot \omega_N^{-\ell k}} = (\mathcal{F}_N u)_k + \omega_{2N}^{-k} (\mathcal{F}_N v)_k.$$

La deuxième formule de (8.3) résulte de  $\omega_N^N = 1$  et de  $\omega_{2N}^N = -1$ .

La formule (8.3) nous permet de calculer (avec  $N$  multiplications et  $2N$  additions) le vecteur  $\mathcal{F}_{2N} y$  à partir de  $\mathcal{F}_N u$  et  $\mathcal{F}_N v$ . La même procédure peut être appliquée récursivement aux suites  $u$  et  $v$  si elles ont une longueur paire. Si l'on suppose que  $N = 2^m$ , on obtient l'algorithme présenté dans le schéma suivant (pour  $N = 8 = 2^3$ )

$$\mathcal{F}_N \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{pmatrix} \quad \left( \begin{array}{c|ccccc} & \mathcal{F}_{N/2} & & \mathcal{F}_{N/4} & & \mathcal{F}_{N/8} \\ \hline & \begin{pmatrix} y_0 \\ y_2 \\ y_4 \\ y_6 \end{pmatrix} & & \begin{pmatrix} y_0 \\ y_4 \end{pmatrix} & & y_0 = y_0 \\ & & & & & y_4 = y_4 \\ & & & \mathcal{F}_{N/4} & \begin{pmatrix} y_2 \\ y_6 \end{pmatrix} & y_2 = y_2 \\ & & & & & y_6 = y_6 \\ & & & & & \end{array} \right) \quad \left( \begin{array}{c|ccccc} & \mathcal{F}_{N/2} & & \mathcal{F}_{N/4} & & \mathcal{F}_{N/8} \\ \hline & \begin{pmatrix} y_1 \\ y_3 \\ y_5 \\ y_7 \end{pmatrix} & & \begin{pmatrix} y_1 \\ y_5 \end{pmatrix} & & y_1 = y_1 \\ & & & & & y_5 = y_5 \\ & & & \mathcal{F}_{N/4} & \begin{pmatrix} y_3 \\ y_7 \end{pmatrix} & y_3 = y_3 \\ & & & & & y_7 = y_7 \\ & & & & & \end{array} \right) \quad (8.4)$$

La programmation de cet algorithme se fait de droit à gauche. D'abord, on met les  $\{y_j\}$  dans l'ordre exigé par l'algorithme (8.4). Après, on effectue les opérations de (8.3) comme indiqué dans le schéma (8.4). Pour une explication détaillée de la programmation voir le livre “Numerical Recipies”.<sup>6</sup>

---

<sup>6</sup>W.H. Press, B.R. Flannery, S.A. Teukolsky & W.T. Vetterling (1989): *Numerical Recipes. The Art of Scientific Computing* (FORTRAN Version). Cambridge University Press.

TAB. II.2: Comparaison de nombres d'opérations

$N$	$N^2$	$N \log_2 N$	quotient
$2^5 = 32$	$\approx 10^3$	160	$\approx 6.4$
$2^{10} \approx 10^3$	$\approx 10^6$	$\approx 10^4$	$\approx 100$
$2^{20} \approx 10^6$	$\approx 10^{12}$	$\approx 2 \cdot 10^7$	$\approx 5 \cdot 10^4$

Pour passer d'une colonne à une autre (dans le schéma (8.4)) on a besoin de  $N/2$  multiplications complexes et de  $N$  additions (ou soustractions). Comme  $m = \log_2 N$  passages sont nécessaires, on a le résultat suivant.

**Théorème 8.2** Pour  $N = 2^m$ , le calcul de  $\mathcal{F}_N y$  peut être effectué en  $\frac{N}{2} \log_2 N$  multiplications complexes et  $N \log_2 N$  additions complexes.

Pour mieux illustrer l'importance de cet algorithme, nous comparons dans le tableau II.2 le nombre d'opérations nécessaires pour le calcul de  $\mathcal{F}_N y$  – avec ou sans FFT.

**L'inverse de la transformée de Fourier discrète.** Pour le décodage il faut calculer les  $y_j$  à partir des  $z_k$  à l'aide de la formule (6.5). Pour en obtenir un algorithme rapide, il suffit de remplacer  $\omega_{2N}^{-k}$  dans (8.3) par  $\omega_{2N}^k$ .

**Transformée cosinus rapide.** Pour les données réelles  $(y_0, y_1, \dots, y_{N-1})$ , nous considérons le vecteur  $\tilde{y} = (y_0, y_1, \dots, y_{N-1}, y_{N-1}, \dots, y_1, y_0)$  illustré dans le petit dessin avant la formule (7.4). La relation  $z_k = \frac{1}{2}\omega_{2N}^{-k/2}(\mathcal{F}_{2N}\tilde{y})_k$  pour la transformée cosinus discrète (7.6) nous permet de trouver un algorithme rapide (même deux fois plus rapide que la FFT).<sup>7</sup>

**Transformée cosinus rapide en dimension 2.** Pour le calcul de (7.8) il faut évaluer la somme sur  $j$  pour chaque  $i \in \{0, 1, \dots, N-1\}$ , et ensuite encore la somme sur  $i$ . Ceci correspond à  $N+1$  transformées cosinus discrètes. Le travail est alors proportionnel à  $(N+1)N \log_2 N$  à la place de  $N^4$  pour un calcul direct. En plus, les  $N$  premières DCT peuvent être calculées en parallèle.

## II.9 Interpolation trigonométrique

Pour la division équidistante

$$x_j = \frac{2\pi j}{N}, \quad j = 0, 1, \dots, N$$

de l'intervalle  $[0, 2\pi]$  et pour  $y_0, y_1, \dots, y_{N-1}$  donnés, on cherche un polynôme trigonométrique (une combinaison linéaire finie de fonctions  $e^{ikx}$ ) passant par  $(x_j, y_j)$  pour  $j = 0, 1, \dots, N-1$ .

**Théorème 9.1 (polynôme d'interpolation trigonométrique)** Pour  $y_0, y_1, \dots, y_{N-1}$  donnés, soit  $\{z_k\}$  sa transformée de Fourier discrète (6.7). Si  $N$  est pair, le polynôme trigonométrique

$$p_N(x) = \sum_{k=-N/2}^{N/2} z_k e^{ikx} := \frac{1}{2} (z_{-N/2} e^{-iNx/2} + z_{N/2} e^{iNx/2}) + \sum_{|k| < N/2} z_k e^{ikx} \quad (9.1)$$

satisfait  $p_N(x_j) = y_j$  pour  $j = 0, 1, \dots, N-1$ .

*Démonstration.* La suite  $z_k e^{ikx_j}$  est  $N$ -périodique en  $k$ . Ainsi,  $p_N(x_j) = \sum_{k=-N/2}^{N/2} z_k e^{ikx_j} = \sum_{k=0}^{N-1} z_k e^{ikx_j}$ . Comme  $e^{ikx_j} = \omega_N^{kj}$ , une comparaison avec (6.5) montre que  $p_N(x_j) = y_j$ .  $\square$

<sup>7</sup>Van Loan, Computational Frameworks for the Fast Fourier Transform. SIAM, Philadelphia, 1992. [MA 65/326]

*Remarque.* Si les  $y_k$  sont réels, les  $\{z_k\}$  satisfont à  $z_{-k} = \bar{z}_k$  et le polynôme  $p_N(x)$  est une fonction réelle, c.-à-d. une combinaison réelle de  $\sin kx$  et  $\cos kx$ .

**Erreur de l'interpolation trigonométrique.** Supposons maintenant que  $y_j = f(x_j)$  où

$$f(x) = \sum_{k=-\infty}^{\infty} c_k e^{ikx}, \quad c_k = \hat{f}(k) \quad (9.2)$$

est une série de Fourier qui converge absolument pour tout  $x$ . Pour étudier l'erreur  $|f(x) - p_N(x)|$ , où  $p_N(x)$  est le polynôme d'interpolation trigonométrique du théorème 9.1, il faut savoir comment les  $z_k = \hat{f}_N(k)$  de la transformée de Fourier discrète approximent les coefficients de Fourier  $c_k = \hat{f}(k)$ .

**Lemme 9.2** *Si la série (9.2) est absolument convergente, alors*

$$\hat{f}_N(k) - \hat{f}(k) = \sum_{\ell \neq 0} \hat{f}(k + \ell N). \quad (9.3)$$

*Démonstration.* La transformée de Fourier discrète avec  $y_j = f(x_j)$  de (9.2) est donnée par

$$\hat{f}_N(k) = \frac{1}{N} \sum_{j=0}^{N-1} \left( \sum_{n=-\infty}^{\infty} \hat{f}(n) e^{inx_j} \right) \omega_N^{-kj} = \sum_{n=-\infty}^{\infty} \hat{f}(n) \left( \frac{1}{N} \sum_{j=0}^{N-1} \omega_N^{(n-k)j} \right) = \sum_{\ell=-\infty}^{\infty} \hat{f}(k + \ell N).$$

La dernière égalité est une conséquence de la relation d'orthogonalité (6.6).  $\square$

On sait que, si la fonction  $f(x)$  est globalement lisse, les coefficients de Fourier convergent rapidement vers zéro. Pour une telle fonction,  $\hat{f}_N(k)$  est une bonne approximation de  $\hat{f}(k)$  pour  $|k| \leq N/2$ , mais elle est mauvaise pour  $|k| > N/2$ .

**Théorème 9.3** *Soit  $f(x)$  donnée par (9.2) avec une série absolument convergente. Alors, le polynôme trigonométrique (9.1) pour  $y_j = f(x_j)$  satisfait pour tout  $x \in \mathbb{R}$*

$$|f(x) - p_N(x)| \leq 2 \sum'_{|k| \geq N/2} |\hat{f}(k)|. \quad (9.4)$$

*Démonstration.* On soustrait (9.1) de (9.2) :

$$f(x) - p_N(x) = \sum'_{k=-N/2}^{N/2} (\hat{f}(k) - \hat{f}_N(k)) e^{ikx} + \sum'_{|k| \geq N/2} \hat{f}(k) e^{ikx}.$$

L'assertion est donc une conséquence de (9.3) et de l'inégalité de triangle.  $\square$

Ce théorème permet une interprétation intéressante. Considérons une fonction  $2\pi$ -périodique de fréquence maximale  $M$  (c.-à-d.,  $\hat{f}(k) = 0$  pour  $|k| > M$ ). Alors, le polynôme trigonométrique  $p_N(x)$  donne le résultat exact ( $p_N(x) = f(x)$  pour tout  $x$ ) si

$$N > 2M. \quad (9.5)$$

Ce résultat – le *théorème d'échantillonnage* – nous donne une formule pour le nombre d'échantillons nécessaires pour une représentation exacte d'une telle fonction.

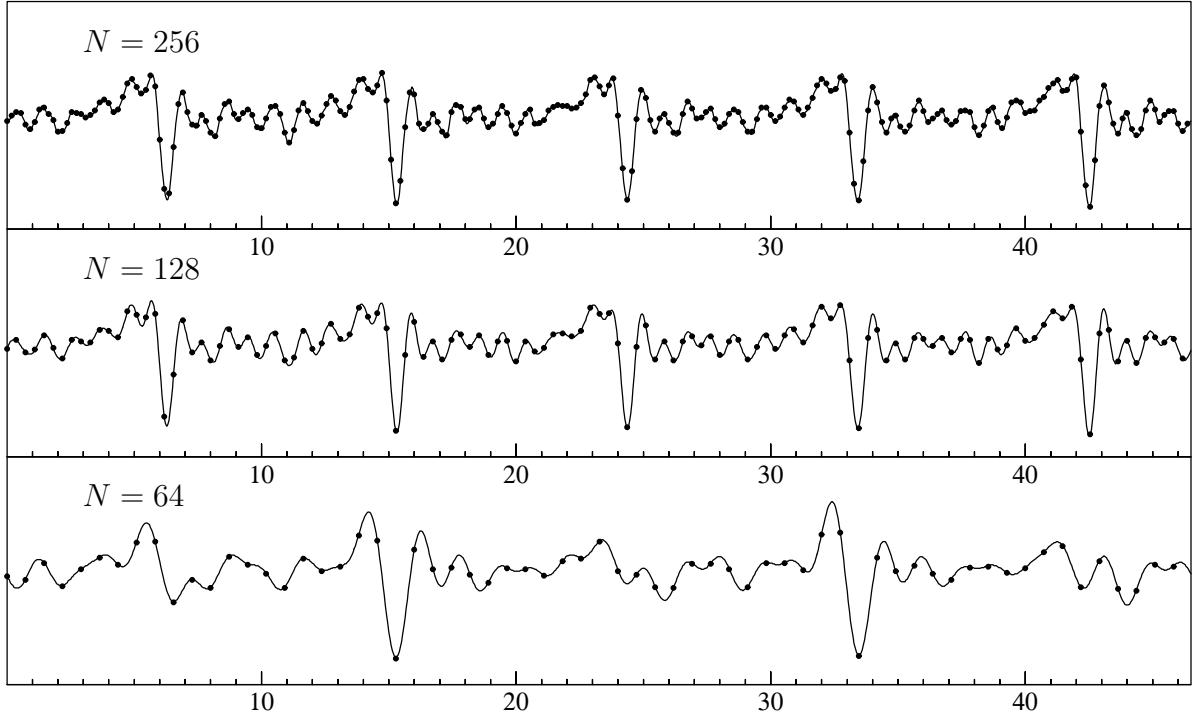


FIG. II.17: Interpolation trigonométrique

Reprendons l'exemple de la fig. II.13. Dans le deuxième dessin de cette figure on voit que les fréquences dominantes du son sont situées dans l'intervalle  $|k| \leq 60$ . Comme la longueur de l'intervalle dans la fig. II.13 est de  $1024/22000$  secondes, la valeur maximale  $M = 60$  correspond à  $60 * 22000/1024 \approx 1390$  Hz. Alors,  $N = 128$  échantillons sont suffisants pour représenter correctement le signal. Dans la fig. II.17 sont dessinés les polynômes trigonométriques  $p_N(x)$  (pour  $N = 64, 128$  et  $256$ ) passant par

$$y_j \quad \text{pour} \quad j = 0 \pmod{1024/N}$$

où  $y_j$  sont les données de la fig. II.13. On voit que la représentation est bonne à partir de  $N = 128$ . Il suffit alors d'utiliser chaque 8<sup>ème</sup> échantillon (une autre possibilité de compression des données).

## II.10 Interpolation par fonctions spline

Le mot “spline” (anglais) signifie “languette élastique”. On s'intéresse à la courbe décrite par une languette forcée de passer par un nombre fini de points donnés (disons par  $(x_i, y_i)$  pour  $i =$

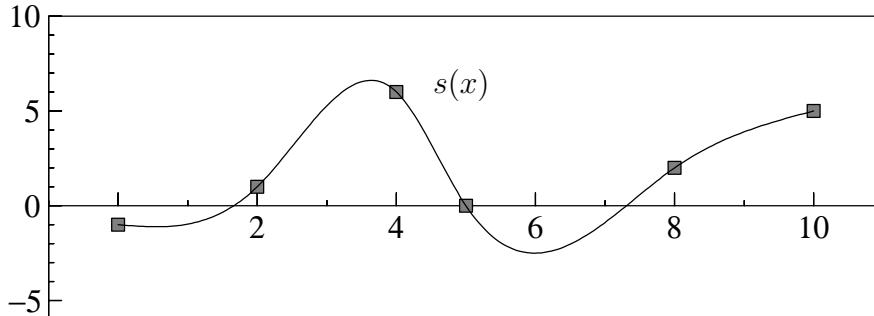


FIG. II.18: Spline cubique (à comparer avec fig. II.1)

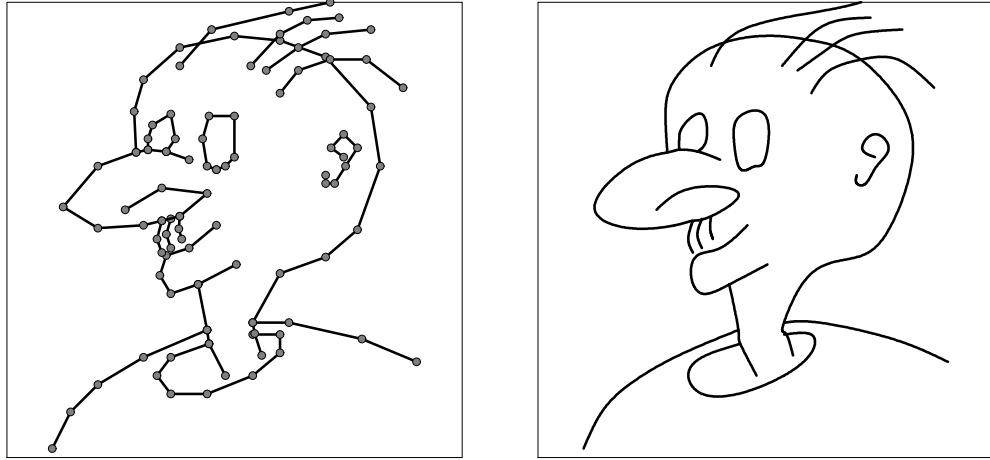


FIG. II.19: Un dessin en zig-zag (à gauche) et en splines (à droite)

$0, 1, \dots, n$ ). La fig. II.18 montre le spline passant par les mêmes données que pour la fig. II.1 (pour pouvoir le comparer avec le polynôme d'interpolation).

La théorie des splines a été développée dans les années 1950 par I.J. Schoenberg pour servir au calcul scientifique (approximations, ...) ; de nos jours elle est constamment appliquée pour la représentation de courbes et surfaces en Computer Graphics (voir la fig. II.19).

**Formulation mathématique du problème :** pour des points  $(x_i, y_i)$  avec des  $x_i$  ordonnés, on cherche une fonction  $s : [a, b] \rightarrow \mathbb{R}$  (où  $a = x_0, b = x_n$ ) satisfaisant à

- (S1)  $s(x_i) = y_i$  pour  $i = 0, 1, \dots, n$ ;
- (S2)  $s(x)$  est 2 fois continûment différentiable (c.-à-d. de classe  $\mathcal{C}^2$ ) ;
- (S3)  $\int_a^b (s''(x))^2 dx \rightarrow \min.$

L'intégrale dans (S3) représente l'énergie de la languette déformée qui, par le principe de Maupertius, est supposée minimale.

**Théorème 10.1** Soit  $a = x_0 < x_1 < \dots < x_n = b$  une division donnée et soit  $s : [a, b] \rightarrow \mathbb{R}$  une fonction vérifiant (S1), (S2) et qui est un polynôme de degré 3 sur chaque sous-intervalle  $[x_{i-1}, x_i]$ .

Pour toute fonction  $f : [a, b] \rightarrow \mathbb{R}$  vérifiant (S1), (S2) et

$$s''(b)(f'(b) - s'(b)) = s''(a)(f'(a) - s'(a)) \quad (10.1)$$

on a alors que

$$\int_a^b (s''(x))^2 dx \leq \int_a^b (f''(x))^2 dx. \quad (10.2)$$

*Démonstration.* Chaque fonction vérifiant (S1) et (S2) peut être écrite sous la forme  $f(x) = s(x) + \epsilon h(x)$  où  $\epsilon \in \mathbb{R}$  et  $h(x)$  est de classe  $\mathcal{C}^2$  satisfaisant à

$$h(x_i) = 0 \quad \text{pour } i = 0, 1, \dots, n. \quad (10.3)$$

La condition (10.2) devient alors

$$\begin{aligned} \int_a^b (s''(x))^2 dx &\leq \int_a^b (s''(x) + \epsilon h''(x))^2 dx \\ &= \int_a^b (s''(x))^2 dx + 2\epsilon \int_a^b s''(x)h''(x) dx + \epsilon^2 \int_a^b (h''(x))^2 dx. \end{aligned}$$

Si l'on fixe  $h(x)$ , cette condition est satisfaite pour tout  $\epsilon$  (positive et négative) si et seulement si

$$\int_a^b s''(x)h''(x) dx = 0 \quad (10.4)$$

ce qui est équivalent à (après une intégration par parties)

$$s''(x)h'(x) \Big|_a^b - \int_a^b s'''(x)h'(x) dx = 0 \quad (10.5)$$

L'hypothèse (10.1) implique que la première expression de (10.5) est nulle. Comme  $s'''(x)$  est constant sur  $(x_{i-1}, x_i)$ , disons égal à  $\alpha_i$ , la deuxième expression de (10.5) devient

$$\int_a^b s'''(x)h'(x) dx = \sum_{i=1}^n \alpha_i \int_{x_{i-1}}^{x_i} h'(x) dx = \sum_{i=0}^n \alpha_i (h(x_i) - h(x_{i-1})) = 0$$

par (10.3). Ainsi, (10.4) et par conséquent (10.2) aussi sont vérifiés.  $\square$

Le théorème précédent montre que les candidats à la solution de (S1)-(S3) sont des fonctions de classe  $C^2$  qui sont des polynômes de degré 3 par morceaux.

**Définition 10.2 (spline cubique)** Soit  $a = x_0 < x_1 < \dots < x_n = b$  une division de  $[a, b]$ . Une fonction  $s : [a, b] \rightarrow \mathbb{R}$  s'appelle spline (cubique) si elle est 2 fois continûment différentiable et si, sur chaque intervalle  $[x_{i-1}, x_i]$ , elle est un polynôme de degré 3.

Pour satisfaire la condition (10.1), on a plusieurs possibilités:

- *spline naturel*: on suppose que

$$s''(a) = 0 \quad \text{et} \quad s''(b) = 0. \quad (10.6)$$

- *spline scellé*: on suppose données les pentes aux extrémités

$$s'(a) = p_0 \quad \text{et} \quad s'(b) = p_n. \quad (10.7)$$

- *spline périodique*: on suppose que

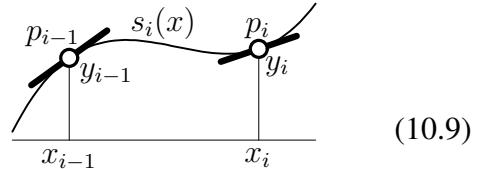
$$s'(a) = s'(b) \quad \text{et} \quad s''(a) = s''(b). \quad (10.8)$$

Evidemment, pour le spline scellé, la condition (10.1) est seulement satisfaite si la fonction  $f(x)$  vérifie aussi la condition (10.7). Alors,  $s(x)$  minimise l'intégrale de (S3) seulement dans la classe de fonctions dont les pentes sont fixées aux extrémités. Pour le spline périodique, la situation est analogue.

Le but suivant est de dériver une construction du spline vérifiant  $s(x_i) = y_i$  pour  $i = 0, 1, \dots, n$  et une des conditions (10.6)-(10.8).

**Interpolation d'Hermite.** Considérons un seul sous-intervalle  $[x_{i-1}, x_i]$  et cherchons un polynôme  $s_i(x)$  de degré 3 vérifiant

$$\begin{aligned} s_i(x_{i-1}) &= y_{i-1}, & s_i(x_i) &= y_i, \\ s'_i(x_{i-1}) &= p_{i-1}, & s'_i(x_i) &= p_i. \end{aligned}$$



La solution peut être obtenue par la formule de Newton en remplaçant les deux dernières conditions de (10.9) par  $s_i(x_{i-1} + \epsilon) = y_{i-1} + \epsilon p_{i-1}$  et  $s_i(x_i - \epsilon) = y_i - \epsilon p_i$ , et en considérant la limite  $\epsilon \rightarrow 0$ . Les différences divisées correspondantes aux données (10.9) sont présentées dans le tableau II.3 ( $h_{i-1} := x_i - x_{i-1}$ ). En utilisant les valeurs encadrées pour la formule de Newton, on obtient

$$s_i(x) = y_{i-1} + (x - x_{i-1}) \delta y[x_i, x_{i-1}] \quad (10.10)$$

$$+ \frac{(x - x_{i-1})(x - x_i)}{h_{i-1}^2} \left( (p_i - \delta y[x_i, x_{i-1}]) (x - x_{i-1}) + (p_{i-1} - \delta y[x_i, x_{i-1}]) (x - x_i) \right).$$

TAB. II.3: Différences divisées pour l'interpolation d'Hermite

$x_{i-1}$	$y_{i-1}$		
$x_{i-1}$	$y_{i-1}$	$p_{i-1}$	$h_{i-1}^{-1}(\delta y[x_i, x_{i-1}] - p_{i-1})$
$x_i$	$y_i$	$\delta y[x_i, x_{i-1}]$	$h_{i-1}^{-1}(p_i - \delta y[x_i, x_{i-1}])$
$x_i$	$y_i$	$p_i$	$h_{i-1}^{-2}(p_i + p_{i-1} - 2\delta y[x_i, x_{i-1}])$
$x_i$	$y_i$		

**Construction du spline interpolant.** Pour chaque choix de pentes  $p_0, p_1, \dots, p_n$ , la fonction  $s : [a, b] \rightarrow \mathbb{R}$ , définie par  $s(x) = s_i(x)$  pour  $x \in [x_{i-1}, x_i]$ , satisfait à

- a)  $s(x_i) = y_i$  pour  $i = 0, 1, \dots, n$ ;
- b)  $s(x)$  est de classe  $C^1$  et  $s'(x_i) = p_i$  pour  $i = 0, 1, \dots, n$ ;
- c) sur chaque intervalle  $[x_{i-1}, x_i]$ ,  $s(x)$  est un polynôme de degré 3.

Pour construire le spline interpolant, il reste à déterminer les pentes  $p_0, p_1, \dots, p_n$  de manière à ce que  $s''(x)$  soit continue, c.-à-d.,

$$s''_i(x_i) = s''_{i+1}(x_i), \quad i = 1, \dots, n-1, \quad (10.11)$$

et qu'une des conditions (10.6)-(10.8) soit satisfaite. En dérivant (10.10) deux fois, on obtient

$$\begin{aligned} s''_i(x_i) &= \frac{2}{h_{i-1}}(2p_i + p_{i-1} - 3\delta y[x_i, x_{i-1}]) \\ s''_i(x_{i-1}) &= -\frac{2}{h_{i-1}}(p_i + 2p_{i-1} - 3\delta y[x_i, x_{i-1}]). \end{aligned}$$

La condition (10.11) devient alors (pour  $i = 1, \dots, n-1$ )

$$\frac{p_{i-1}}{h_{i-1}} + p_i \cdot 2 \cdot \left( \frac{1}{h_{i-1}} + \frac{1}{h_i} \right) + \frac{p_{i+1}}{h_i} = 3 \left( \frac{\delta y[x_i, x_{i-1}]}{h_{i-1}} + \frac{\delta y[x_{i+1}, x_i]}{h_i} \right). \quad (10.12)$$

Ceci donne  $n-1$  équations linéaires pour les  $n+1$  inconnues  $p_0, p_1, \dots, p_n$ . Les deux dernières conditions sont données par le type du spline. Par exemple, pour le *spline scellé*, les valeurs de  $p_0$  et  $p_n$  sont explicitement données et le système linéaire ainsi obtenu s'écrit matriciellement

$$\underbrace{\begin{pmatrix} 2\left(\frac{1}{h_0} + \frac{1}{h_1}\right) & \frac{1}{h_1} & & & & \\ \frac{1}{h_1} & 2\left(\frac{1}{h_1} + \frac{1}{h_2}\right) & \frac{1}{h_2} & & & \\ & \frac{1}{h_2} & 2\left(\frac{1}{h_2} + \frac{1}{h_3}\right) & \ddots & & \\ & & \ddots & \ddots & \frac{1}{h_{n-2}} & \\ & & & & \ddots & \\ & & & & & 2\left(\frac{1}{h_{n-2}} + \frac{1}{h_{n-1}}\right) \end{pmatrix}}_A \begin{pmatrix} p_1 \\ p_2 \\ p_3 \\ \vdots \\ p_{n-1} \end{pmatrix} = \begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ \vdots \\ c_{n-1} \end{pmatrix} \quad (10.13)$$

avec dec  $c_i$  donnés par (10.12). La matrice  $A$  est symétrique et tridiagonale.

**Théorème 10.3** Soit  $Ap = c$  avec des  $c_i$  satisfaisant  $|c_i| \leq \gamma$  pour tout  $i$ . Alors,

$$|p_i| \leq \frac{h}{2} \gamma \quad \text{où} \quad h = \max_{i=0, \dots, n-1} h_i. \quad (10.14)$$

En particulier,  $A$  est inversible (poser  $\gamma = 0$ ).

*Démonstration.* Soit  $p$  une solution de  $Ap = c$  et choisissons l'indice  $\ell$  de manière à ce que  $|p_\ell| \geq |p_j|$  pour tout  $j$ . La ligne  $\ell$  du système  $Ap = c$  donne

$$p_\ell \cdot 2 \cdot \left( \frac{1}{h_{\ell-1}} + \frac{1}{h_\ell} \right) = -\frac{p_{\ell-1}}{h_{\ell-1}} - \frac{p_{\ell+1}}{h_\ell} + c_\ell$$

et, en prenant sa valeur absolue,  $|p_\ell| \cdot 2 \cdot \left( \frac{1}{h_{\ell-1}} + \frac{1}{h_\ell} \right) \leq |p_\ell| \cdot \left( \frac{1}{h_{\ell-1}} + \frac{1}{h_\ell} \right) + \gamma$ , on en déduit

$$|p_\ell| \leq \frac{h_\ell \cdot h_{\ell-1}}{h_\ell + h_{\ell-1}} \gamma \leq \frac{\max(h_\ell, h_{\ell-1})}{2} \gamma \leq \frac{h}{2} \gamma$$

car  $h_\ell + h_{\ell-1} \geq 2 \cdot \min(h_\ell, h_{\ell-1})$ .  $\square$

*Conclusion.* Le spline scellé existe toujours et il est unique. La même démonstration s'applique aussi au spline naturel et au spline périodique.

La résolution du système linéaire (10.13) se fait par élimination. On élimine la variable  $p_1$  dans la ligne 2 à l'aide de la ligne 1, puis la variable  $p_2$  dans la ligne 3 à l'aide de la ligne 2, etc. On obtient alors un système bidiagonal qui est facile à résoudre.

## II.11 L'erreur du spline

Soient une fonction différentiable  $f : [a, b] \rightarrow I\!\!R$  et une division  $a = x_0 < x_1 < \dots < x_n = b$ . Considérons le spline  $s(x)$  satisfaisant (spline scellé)

$$s(x_i) = f(x_i), \quad i = 0, 1, \dots, n \quad (8.1a)$$

$$s'(x_0) = f'(x_0), \quad s'(x_n) = f'(x_n). \quad (8.1b)$$

Ce spline est donné par (10.10) où les coefficients  $p_i$  satisfont (10.12),  $p_0 = f'(x_0)$  et  $p_n = f'(x_n)$ . Le but est d'étudier l'erreur  $f(x) - s(x)$  pour  $x \in [a, b]$ .

L'idée importante est de considérer (sur  $[x_{i-1}, x_i]$ ) également le polynôme d'interpolation d'Hermite  $q_i(x)$ , défini par

$$q_i(x_{i-1}) = f(x_{i-1}), \quad q_i(x_i) = f(x_i), \quad q'_i(x_{i-1}) = f'(x_{i-1}), \quad q'_i(x_i) = f'(x_i). \quad (11.1)$$

Il est aussi donné par (10.10) si l'on remplace  $p_j$  par  $f'(x_j)$ . On va estimer séparément les deux termes dans la formule

$$f(x) - s_i(x) = (f(x) - q_i(x)) + (q_i(x) - s_i(x)). \quad (11.2)$$

**Théorème 11.1** Soit  $f(x)$  de classe  $C^4$  et  $q_i(x)$  le polynôme de degré 3 satisfaisant (11.1) (interpolation d'Hermite). Alors, pour  $x \in [x_{i-1}, x_i]$ , on a

$$|f(x) - q_i(x)| \leq \frac{h_{i-1}^4}{384} \cdot \max_{\xi \in [x_{i-1}, x_i]} |f^{(4)}(\xi)|. \quad (11.3)$$

*Démonstration.* Si l'on calcule le polynôme d'interpolation de degré 3 passant par

$$(x_{i-1}, f(x_{i-1})), \quad (x_{i-1} + \epsilon, f(x_{i-1} + \epsilon)), \quad (x_i - \epsilon, f(x_i - \epsilon)), \quad (x_i, f(x_i))$$

avec la formule de Newton, on obtient pour  $\epsilon \rightarrow 0$  exactement le polynôme  $q_i(x)$  (voir le tableau II.3 des différences divisées). L'erreur de ce polynôme peut être majorée par (voir le para-

graphique II.2)

$$|(x - x_{i-1})(x - x_{i-1} - \epsilon)(x - x_i + \epsilon)(x - x_i)| \cdot \frac{|f^{(4)}(\xi)|}{4!}.$$

Donc, pour  $\epsilon \rightarrow 0$ ,

$$|f(x) - q_i(x)| \leq (x - x_{i-1})^2(x - x_i)^2 \cdot \frac{1}{4!} \cdot \max_{\xi \in [x_{i-1}, x_i]} |f^{(4)}(\xi)|.$$

Comme la fonction  $(x - x_{i-1})^2(x - x_i)^2$  possède son maximum au milieu de l'intervalle  $[x_{i-1}, x_i]$ , on obtient l'estimation (11.3).  $\square$

Pour estimer la deuxième expression de (11.2), nous soustrayons la formule (10.10) pour  $s_i(x)$  de la formule analogue pour  $q_i(x)$  et nous obtenons

$$q_i(x) - s_i(x) = \frac{(x - x_{i-1})(x - x_i)}{h_{i-1}^2} \left( (f'(x_i) - p_i)(x - x_{i-1}) + (f'(x_{i-1}) - p_{i-1})(x - x_i) \right). \quad (11.4)$$

Il nous faut encore une estimation de  $f'(x_i) - p_i$ .

**Lemme 11.2** Soit  $f(x)$  de classe  $C^4$  sur  $[a, b]$  et notons  $h = \max_i h_i$  où  $h_i = x_{i+1} - x_i$ . Si les  $p_i$  satisfont (10.12),  $p_0 = f'(x_0)$  et  $p_n = f'(x_n)$ ,

$$|f'(x_i) - p_i| \leq \frac{h^3}{24} \cdot \max_{x \in [a, b]} |f^{(4)}(x)|. \quad (11.5)$$

Si la division est équidistante et si  $f \in C^5[a, b]$ , alors

$$|f'(x_i) - p_i| \leq \frac{h^4}{60} \cdot \max_{x \in [a, b]} |f^{(5)}(x)|. \quad (11.6)$$

*Démonstration.* Voici les idées pour le cas équidistant. Les  $p_i$  sont définis par (voir (10.12))

$$\frac{1}{h} (p_{i-1} + 4p_i + p_{i+1}) - \frac{3}{h^2} (f(x_{i+1}) - f(x_{i-1})) = 0. \quad (11.7)$$

Pour estimer la différence  $f'(x_i) - p_i$ , nous calculons le défaut qu'on obtient en remplaçant  $p_i$  par  $f'(x_i)$  dans (11.7) :

$$\frac{1}{h} (f'(x_{i-1}) + 4f'(x_i) + f'(x_{i+1})) - \frac{3}{h^2} (f(x_{i+1}) - f(x_{i-1})) =: d_i. \quad (11.8)$$

En utilisant la formule de Taylor (voir le cours d'Analyse I) pour  $f(x_i \pm h)$  et  $f'(x_i \pm h)$ , on obtient

$$d_i = h^3 \int_0^1 \left( \frac{(1-t)^3}{3!} - 3 \frac{(1-t)^4}{4!} \right) (f^{(5)}(x_i + th) + f^{(5)}(x_i - th)) dt.$$

Comme la fonction  $(1-t)^3/3! - 3(1-t)^4/4!$  ne change pas de signe sur  $[0, 1]$ , ceci nous donne l'estimation

$$|d_i| \leq h^3 \underbrace{\int_0^1 \left( \frac{(1-t)^3}{3!} - 3 \frac{(1-t)^4}{4!} \right) dt}_{= 1/30} \cdot 2 \cdot \max_{x \in [x_{i-1}, x_{i+1}]} |f^{(5)}(x)|.$$

Notons  $d = (d_1, \dots, d_{n-1})^T$  et  $e = (e_1, \dots, e_{n-1})^T$  où  $e_i = f'(x_i) - p_i$ . En prenant la différence entre (11.8) et (11.7), on obtient  $Ae = d$  où  $A$  est la matrice de (10.13). Le théorème 10.3 implique

$$|e_i| \leq \frac{h}{2} \max_j |d_j| \leq \frac{h^4}{60} \cdot \max_{x \in [a, b]} |f^{(5)}(x)|,$$

ce qu'il fallait démontrer.  $\square$

**Théorème 11.3 (erreur du spline scellé)** Soit  $f : [a, b] \rightarrow \mathbb{R}$  de classe  $\mathcal{C}^4$ ,  $a = x_0 < x_1 < \dots < x_n = b$  une division arbitraire et  $s(x)$  le spline qui passe par  $(x_i, f(x_i))$  pour  $i = 0, 1, \dots, n$  et qui satisfait  $s'(x_0) = f'(x_0)$  et  $s'(x_n) = f'(x_n)$ . Alors, avec  $h_i = x_{i+1} - x_i$  et  $h = \max_i h_i$  on a

$$|f(x) - s(x)| \leq \frac{5}{384} \cdot h^4 \cdot \max_{\xi \in [a, b]} |f^{(4)}(\xi)|. \quad (11.9)$$

Si de plus la division est équidistante et si  $f \in \mathcal{C}^5[a, b]$ ,

$$|f(x) - s(x)| \leq \frac{h^4}{384} \cdot \max_{\xi \in [a, b]} |f^{(4)}(\xi)| + \frac{h^5}{240} \cdot \max_{\xi \in [a, b]} |f^{(5)}(\xi)|. \quad (11.10)$$

*Démonstration.* Considérons de nouveau le cas équidistant. Pour  $x \in [x_{i-1}, x_i]$ , nous estimons séparément les deux termes dans (11.2). L'estimation du premier terme résulte de (11.3). Pour le deuxième terme, nous utilisons (11.4) et (11.6). Ceci donne

$$|q_i(x) - s_i(x)| \leq \frac{(x - x_{i-1})(x_i - x)}{h_{i-1}^2} \cdot h_{i-1} \cdot \frac{h^4}{60} \cdot \max_{\xi \in [a, b]} |f^{(5)}(\xi)| \leq \frac{h^5}{4 \cdot 60} \cdot \max_{\xi \in [a, b]} |f^{(5)}(\xi)|. \quad \square$$

**Exemple.** Considérons encore une fois (voir les fig. II.5 et II.6) la fonction

$$f(x) = \frac{1}{1 + 25x^2} \quad \text{sur} \quad [-1, 1] \quad (11.11)$$

et calculons le spline interpolant (scellé) pour la division équidistante  $x_i = -1 + 2i/n$ ,  $i = 0, 1, \dots, n$ . Dans la fig. II.20, les 4 dessins du haut montrent le spline  $s(x)$  pour  $n = 3$ ,  $n = 9$ ,  $n = 27$  et  $n = 81$ . La fonction  $f(x)$  est dessinée en pointillés. Les 4 dessins du bas montrent les erreurs. Cette fois, la fonction  $h^4 \cdot f^{(4)}(x)/384$  ( $h = 2/n$ ) est inclue en pointillés (on a choisi l'échelle sur l'axe  $y$  de manière à ce que  $h^4 \cdot f^{(4)}(0)/384$  soit toujours au même endroit). Pour des petits  $h$ , quand le deuxième terme de (11.10) est négligeable, on peut très bien observer la validité de l'estimation (11.10).

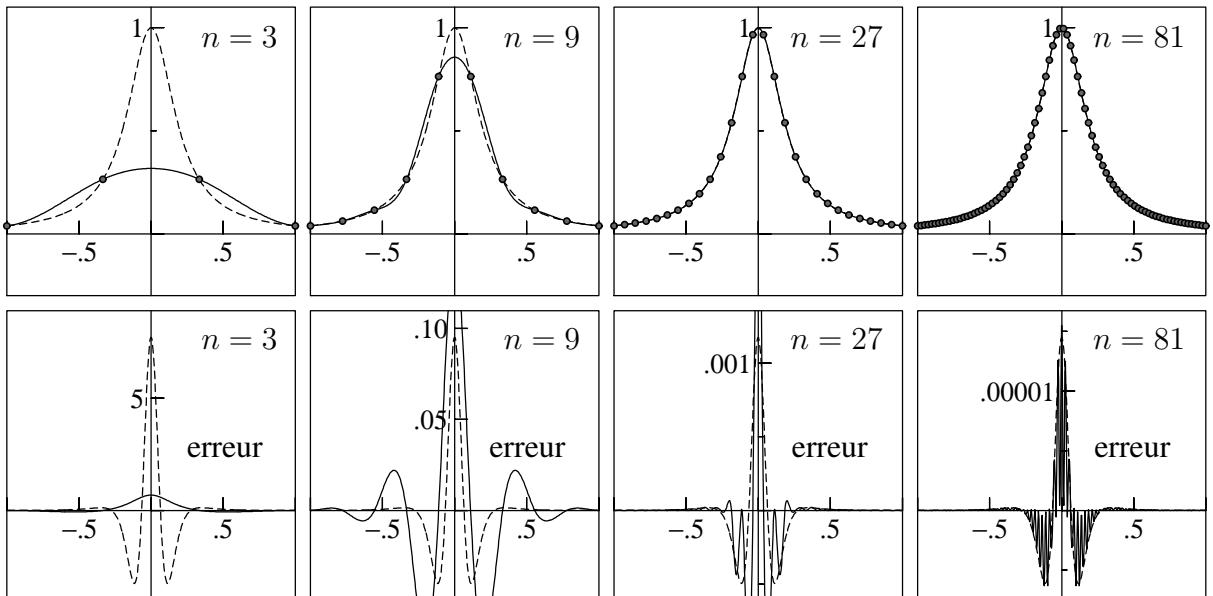


FIG. II.20: Spline interpolant (scellé) pour la fonction (11.11)

## II.12 Exercices

1. Calculer le polynôme d'interpolation passant par les points

$$(0, 0), \quad (1, 3), \quad (3, 1), \quad (5, 2), \quad (8, 2),$$

en utilisant la formule de Newton.

2. Démontrer que la différence divisée  $\delta^n y[x_0, x_1, \dots, x_n]$  est une fonction symétrique; c.-à-d.

$$\delta^n y[x_{\sigma(0)}, x_{\sigma(1)}, \dots, x_{\sigma(n)}] = \delta^n y[x_0, x_1, \dots, x_n]$$

pour toute permutation  $\sigma$  de  $\{0, 1, \dots, n\}$ .

3. Nous savons que l'erreur pour l'interpolation linéaire de  $f$  aux points  $x_0, x_1$  est

$$f(x) - p(x) = (x - x_0)(x - x_1) \frac{f''(\zeta(x))}{2}, \quad x_0 < x < x_1,$$

si  $f \in C^2[x_0, x_1]$ . Déterminer la fonction  $\zeta(x)$  explicitement dans le cas où  $f(x) = \frac{1}{x}$ ,  $x_0 = 1$ ,  $x_1 = 2$ , et trouver  $\max_{1 \leq x \leq 2} \zeta(x)$  et  $\min_{1 \leq x \leq 2} \zeta(x)$ .

4. On veut tabuler la fonction  $y = \sin x$  aux points équidistants  $x_j = jh$ ,  $j \geq 0$ .

(a) Majorer l'erreur d'interpolation dans l'intervalle  $[x_i, x_{i+1}]$  lorsqu'on fait passer un polynôme de degré 3 par  $x_{i-1}, x_i, x_{i+1}, x_{i+2}$ .

(b) Quel  $h$  peut-on prendre pour que cette erreur soit  $\leq 10^{-8}$  pour tout  $i > 0$ ?

5. (a) Montrer que la formule de quadrature

$$\int_{-1}^1 \frac{f(x)}{\sqrt{1-x^2}} dx \approx \frac{\pi}{s} \sum_{k=1}^s f(c_k), \quad c_k = \cos\left(\frac{(2k-1)\pi}{2s}\right) \quad (12.1)$$

est exacte pour tout polynôme  $f(x)$  de degré  $\leq 2s - 1$ .

*Indication.* Vérifier (12.1) pour les polynômes de Chebyshev  $T_0(x), \dots, T_{2s-1}(x)$ .

(b) Effectuez le changement de variables  $x = \cos(t)$  dans (12.1), à quelle formule de quadrature se ramène t'on ?

6. Considérer la fonction  $f(x) = x^3$  sur l'intervalle  $[-1, 1]$ .

(a) Déterminer  $p$  tel que la droite  $d(x) = px$  satisfait

$$\max_{x \in [-1, 1]} |f(x) - d(x)| \rightarrow \min.$$

(b) Pour quels  $x_i$ , la droite trouvée peut être interprétée comme un polynôme d'interpolation pour  $f(x)$  ?

(c) Y-a-t'il une relation avec les points de Chebyshev? Si oui, expliquer laquelle.

7. Pour un polynôme  $p(x) = a_0 + a_1x + \dots + a_nx^n$ , l'algorithme de Horner permet de calculer  $b_0 = p(x_0)$  par:

$$\begin{aligned} b_n &= a_n \\ b_i &= a_i + x_0b_{i+1}, \quad i = n-1, \dots, 1, 0. \end{aligned}$$

Notons  $q(x) = b_1 + b_2x + \dots + b_nx^{n-1}$ .

a) Montrer que  $p(x) = b_0 + (x - x_0)q(x)$  et que  $p'(x_0) = q(x_0)$ .

b) Généraliser l'algorithme de Horner pour calculer  $p(x_0)$  et  $p'(x_0)$  en même temps.

8. Sur l'intervalle  $[-1, 1]$ , on considère la fonction

$$f(x) = \frac{3x - 4}{(x^2 - 9)(5x^2 - 8x + 4)}.$$

Faire une représentation graphique de  $f(x)$  et calculer ses pôles. Utiliser la figure II.7 pour déterminer approximativement l'intervalle maximal  $[\alpha, \beta] \subset [-1, 1]$  où la suite des polynômes d'interpolations (pour les divisions équidistantes  $x_i = -1 + 2i/n$ ) converge (quand  $n \rightarrow \infty$ ).

9. Pour une division  $x_0 < x_1 < \dots < x_n$  ( $n \geq 2$ ), étudier la fonction

$$\Lambda(x) = \sum_{i=0}^n |\ell_i(x)|, \quad \ell_i(x) = \prod_{j \neq i} \frac{(x - x_j)}{(x_i - x_j)}. \quad (12.2)$$

a) Dessiner  $\Lambda(x)$  pour  $n = 2$  en considérant la division  $\{-1, 0, 1\}$ .

b) Montrer que, sur l'intervalle  $[x_{j-1}, x_j]$ , on a l'identité

$$\Lambda|_{[x_{j-1}, x_j]}(x) = \sum_{i=0}^n \epsilon_i \ell_i(x), \quad \text{avec} \quad \epsilon_i = \begin{cases} (-1)^{j-i+1} & \text{si } i \leq j-1 \\ (-1)^{i-j} & \text{si } i \geq j. \end{cases} \quad (12.3)$$

c) Montrer que la fonction  $\Lambda(x)$  de (12.2) ne possède qu'un seul maximum local sur chaque intervalle  $[x_{j-1}, x_j]$ .

*Indication.* Etudier les extréma du polynôme (12.3).

10. Calculer les constantes de Lebesgue

$$\Lambda_n = \max_{x \in [-1, 1]} \sum_{i=0}^n |\ell_i(x)|, \quad n = 1, 2, 3, \dots$$

a) pour la division équidistante  $x_k = -1 + 2k/n$ ,

b) pour les points de Chebyshev  $x_k = -\cos((2k+1)\pi/(2n+2))$ .

Pour calculer le maximum de la fonction  $f(x) = \sum_{i=0}^n |\ell_i(x)|$  sur  $[x_{j-1}, x_j]$  utiliser la recherche de Fibonacci.

11. Calculer à la main la transformée de Fourier discrète de la suite  $\{0, 1, 2, 3, 0, -3, -2, -1\}$ .

12. Pour une fonction continue  $f : [a, b] \rightarrow \mathbb{R}$  et une formule de quadrature  $(b_i, c_i)_{i=1}^s$  à poids  $b_i$  positifs, estimer la différence

$$I_\Delta - \widehat{I}_\Delta,$$

où  $I_\Delta$  est le résultat pour la division  $\Delta = \{a = x_0 < x_1 < \dots < x_n = b\}$  (sans tenir compte des erreurs d'arrondi), et  $\widehat{I}_\Delta$  est le résultat obtenu en tenant compte des erreurs d'arrondi dans l'évaluation de  $f(x)$ , c.-à-d. en remplaçant  $f(x_j + c_i h_j)$  par

$$\widehat{y}_{ij} = f(x_j + c_i h_j)(1 + \varepsilon_{ij}), \quad |\varepsilon_{ij}| \leq \varepsilon.$$

Justifier la condition (6.1) du paragraphe I.6.

13. (a) Calculer les coefficients  $\widehat{f}(k)$  de la série de Fourier pour la fonction  $2\pi$ -périodique

$$f(x) = \begin{cases} 4x/\pi & \text{pour } |x| < \pi, \\ 0 & \text{pour } x = \pi. \end{cases}$$

- (b) Calculer la transformée de Fourier discrète pour  $\{y_k\}_{k=0}^N$  où  $y_l = f(2\pi l/N)$ ,  $l = 0, \dots, N$  avec  $f$ , la fonction de l'exercice précédent.

- (c) Vérifier le résultat obtenu dans l'exercice 11 pour  $N = 8$ .

- (d) Estimer la différence  $|z_k - \widehat{f}_N(k)|$ .

14. Pour deux suites  $N$ -périodiques  $y$  et  $z$ , on définit la *convolution*  $y * z$  par

$$(y * z)_k = \sum_{j=0}^{N-1} y_{k-j} z_j.$$

Montrer que  $y * z$  est également  $N$ -périodique et que

$$\mathcal{F}_N(y * z) = \mathcal{F}_N y \cdot \mathcal{F}_N z \quad (12.4)$$

où la multiplication dans (12.4) est effectuée élément par élément.

15. Démontrer l'égalité de Parseval (1806) pour la transformée de Fourier discrète.

$$N \sum_{k=0}^{N-1} |z_k|^2 = \sum_{k=0}^{N-1} |y_k|^2.$$

16. Soit  $N$  un nombre pair et  $p_N(x)$  le polynôme trigonométrique qui passe par  $(x_\ell, y_\ell)$  pour  $\ell = 0, 1, \dots, N-1$  (voir la formule (9.1)). Montrer que

$$p_N(x) = \sum_{\ell=0}^{N-1} y_\ell S_N(x - x_\ell) \quad (12.5)$$

où

$$S_N(x) = \frac{\sin(xN/2)}{N} \cdot \frac{\cos(x/2)}{\sin(x/2)}. \quad (12.6)$$

17. La fonction  $S_N(x)$  de (12.6) permet d'étudier l'influence des erreurs dans les  $y_\ell$  sur le polynôme trigonométrique interpolant. Montrer que  $S_N(x)$  est  $2\pi$ -périodique et que pour tout  $x \in [-\pi, \pi]$

$$\left| S_N(x) - \frac{\sin(xN/2)}{xN/2} \right| \leq \frac{2}{\pi \cdot N}.$$

18. Calculer la main le spline naturel ( $s''(x_0) = s''(x_n) = 0$ ) qui passe par les points

$$(-3, 0), \quad (-2, 0), \quad (-1, 0), \quad (0, 1), \quad (1, 0), \quad (2, 0), \quad (3, 0).$$

Dessiner les graphes (si possible avant les calculs!) de  $s(x)$ ,  $s'(x)$ ,  $s''(x)$ ,  $s'''(x)$ .

*Indication.* Pour des données symétriques par rapport  $x = 0$ , le spline naturel est une fonction paire. Avec cette observation, on peut réduire la dimension du système des  $p_i$ .

19. Pour la fonction

$$f(x) = \begin{cases} x^3 & \text{si } x \geq 0 \\ 0 & \text{si } x \leq 0, \end{cases}$$

calculer les différences

$$\Delta f(x), \quad \Delta^2 f(x), \quad \Delta^3 f(x), \quad \text{et} \quad B(x) = \Delta^4 f(x)$$

où  $\Delta g(x) := g(x+1) - g(x)$  et  $\Delta^2 g := \Delta(\Delta g)$ , etc.

Montrer que  $B(x)$  est une fonction “spline” (appelée “B-spline”) à support compact, c.-à-d. qu'elle est nulle en dehors d'un intervalle borné.

Calculer les valeurs de  $B(x)$  pour  $x$  entier et dessiner  $B(x)$ .

20. (*Spline périodique*). Soient donnés  $(x_i, y_i)$  pour  $i = 0, 1, \dots, n$  avec  $y_n = y_0$ . Montrer l'existence et l'unicité d'un spline qui passe par tous les  $(x_i, y_i)$  et qui satisfait

$$s'(x_n) = s'(x_0), \quad s''(x_n) = s''(x_0).$$

21. (Formule de *Newton-Gregory*). Soit  $f : [a, b] \rightarrow \mathbb{R}$  une fonction différentiable et  $x_i = a + ih$ , (avec  $h = (b - a)/n$ ) une division équidistante de  $[a, b]$ . Montrer que l'intégration du spline, défini par  $s(x_i) = f(x_i)$  pour  $i = 0, 1, \dots, n$ ,  $s'(a) = f'(a)$  et  $s'(b) = f'(b)$ , donne la formule de quadrature suivante:

$$\int_a^b f(x)dx \approx h \left( \frac{1}{2}f(x_0) + f(x_1) + \dots + f(x_{n-1}) + \frac{1}{2}f(x_n) \right) - \frac{h^2}{12} (f'(x_n) - f'(x_0)).$$

Pourquoi cette formule est-elle exacte pour tout polynôme de degré 3?

# Chapitre III

## Equations Différentielles Ordinaires

Ce chapitre est consacré à la résolution numérique d'un système d'équations différentielles ordinaires

$$\begin{aligned} y'_1 &= f_1(t, y_1, \dots, y_n), & y_1(t_0) &= y_{10}, \\ &\vdots \\ y'_n &= f_n(t, y_1, \dots, y_n), & y_n(t_0) &= y_{n0}. \end{aligned} \tag{0.1}$$

En notation vectorielle, ce système s'écrit

$$y' = f(t, y), \quad y(t_0) = y_0 \tag{0.2}$$

où  $y = (y_1, \dots, y_n)^T$  et  $f : I\!\!R \times I\!\!R^n \rightarrow I\!\!R^n$ . Voici quelques livres qui traitent de ce sujet.

### Bibliographie sur ce chapitre

- K. Burrage (1995): *Parallel and sequential methods for ordinary differential equations*. The Clarendon Press, Oxford University Press. [MA 65/369]
- J.C. Butcher (1987): *The Numerical Analysis of Ordinary Differential Equations*. John Wiley & Sons. [MA 65/276]
- J.C. Butcher (2003): *Numerical Methods for Ordinary Differential Equations*. John Wiley & Sons. [MA 65/470]
- M. Crouzeix & A.L. Mignot (1984): *Analyse Numérique des Equations Différentielles*. Masson. [MA 65/217]
- P. Deuflhard & F. Bornemann (1994): *Numerische Mathematik II. Integration gewöhnlicher Differentialgleichungen*. Walter de Gruyter. [MA 65/309]
- E. Hairer, S.P. Nørsett & G. Wanner (1993): *Solving Ordinary Differential Equations I. Nonstiff Problems*. Springer Series in Comput. Math., vol. 8, 2nd edition. [MA 65/245]
- E. Hairer & G. Wanner (1996): *Solving Ordinary Differential Equations II. Stiff and Differential-Algebraic Problems*. Springer Series in Comput. Math., vol. 14, 2nd edition. [MA 65/245]
- E. Hairer, C. Lubich & G. Wanner (2002): *Geometric Numerical Integration. Structure-Preserving Algorithms for Ordinary Differential Equations*. Springer Series in Comput. Math., vol. 31, 2nd edition in preparation. [MA 65/448]
- P. Henrici (1962): *Discrete Variable Methods in Ordinary Differential Equations*. John Wiley & Sons. [MA 65/50]
- A. Iserles (1996): *A First Course in the Numerical Analysis of Differential Equations*. Cambridge Texts in Applied Mathematics, Cambridge University Press.
- J.D. Lambert (1991): *Numerical Methods for Ordinary Differential Equations*. John Wiley & Sons. [MA 65/367]
- A.M. Stuart & A.R. Humphries (1996): *Dynamical Systems and Numerical Analysis*. Cambridge Univ. Press. [MA 65/377]

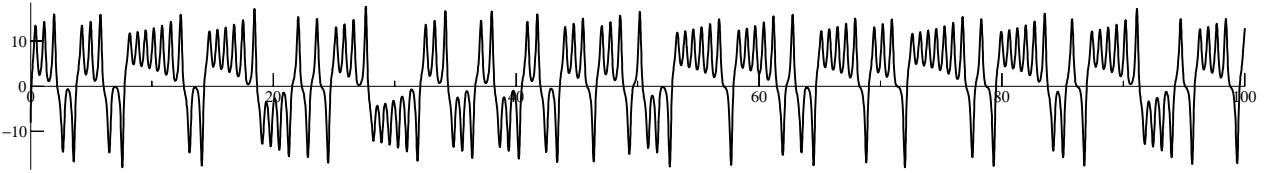
### III.1 Quelques exemples typiques

Pour des équations différentielles d'un intérêt pratique on trouve rarement la solution  $y(t)$  exprimée avec une formule exacte. On est alors obligé d'utiliser des méthodes numériques.

**Exemple 1.1 (modèle de Lorenz)** Une équation très célèbre est celle de Lorenz (1979)

$$\begin{aligned} y'_1 &= -\sigma y_1 + \sigma y_2 & y_1(0) &= -8 \\ y'_2 &= -y_1 y_3 + r y_1 - y_2 & y_2(0) &= 8 \\ y'_3 &= y_1 y_2 - b y_3 & y_3(0) &= r - 1 \end{aligned} \quad (1.1)$$

avec  $\sigma = 10$ ,  $r = 28$  et  $b = 8/3$ . 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 le paragraphe III.2) ou les *méthodes multipas* (paragraphe III.5) nous permettent de trouver sans difficultés des bonnes approximations.

La solution  $y(t) = (y_1(t), y_2(t), y_3(t))^T$  peut aussi être interprétée comme une courbe paramétrique dans l'espace  $\mathbb{R}^3$  (avec paramètre  $t$ ). Leurs projections sur le plan des composantes  $(y_1, y_2)$  et  $(y_1, y_3)$  sont dessinées dans la figure III.1. L'intervalle d'intégration est  $[0, 25]$ . La valeur initiale est marquée par un point noir épais.

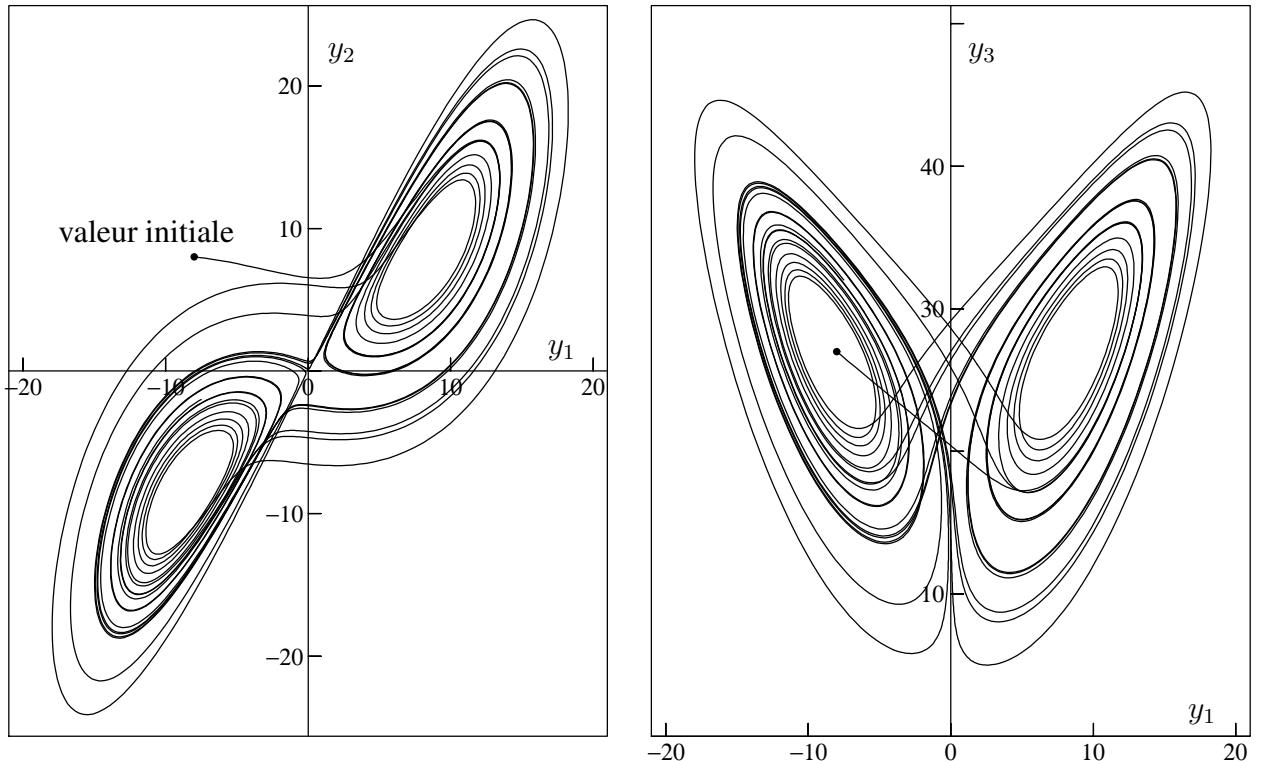
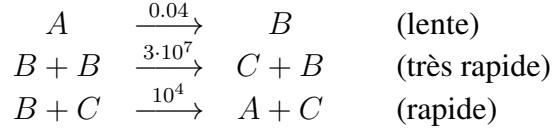


FIG. III.1: Deux vues de la solution  $y(t) = (y_1(t), y_2(t), y_3(t))^T$  du problème de Lorenz (1.1)

When the equations represent the behaviour of a system containing a number of fast and slow reactions, a forward integration of these equations becomes difficult. (H.H. Robertson 1966)

**Exemple 1.2 (réactions chimiques)** L'exemple suivant de Robertson (1966) est devenu célèbre comme équation test pour des études numériques (Willoughby 1974): la réaction chimique



conduit au système d'*équations différentielles raides*

$$\begin{aligned} A : \quad y'_1 &= -0.04y_1 + 10^4y_2y_3 & y_1(0) &= 1 \\ B : \quad y'_2 &= 0.04y_1 - 10^4y_2y_3 - 3 \cdot 10^7y_2^2 & y_2(0) &= 0 \\ C : \quad y'_3 &= 3 \cdot 10^7y_2^2 & y_3(0) &= 0. \end{aligned} \quad (1.2)$$

Si on utilise une méthode classique (par exemple, la méthode de Runge–Kutta DOPRI5) on est obligé de prendre des pas d'intégration très petits pour obtenir une approximation correcte (voir la figure III.2). Dans le paragraphe III.9 nous étudierons des méthodes numériques adaptées à ce type de problèmes (dont un représentant est RADAU5). Elles donnent une grande précision avec des grands pas d'intégration.

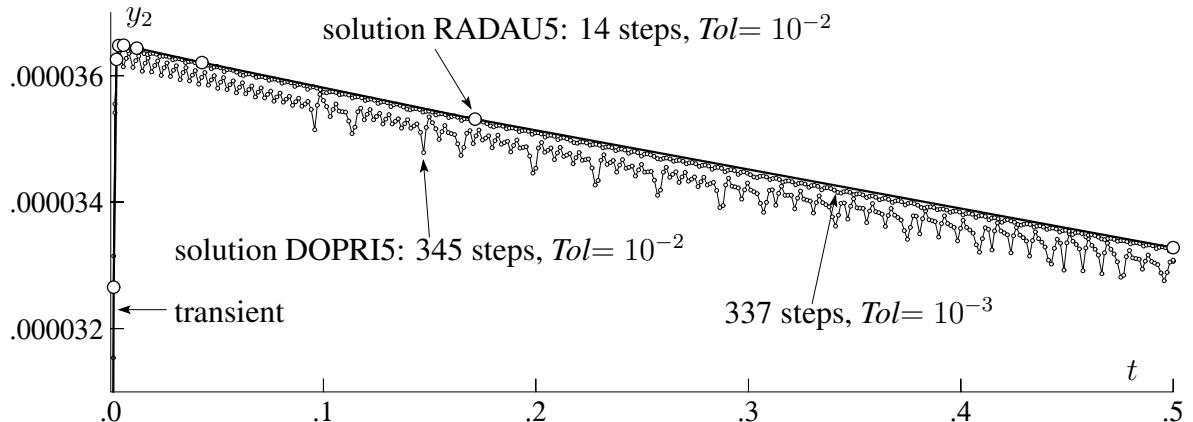


FIG. III.2: Solution numérique pour (1.2) obtenue par une méthode classique (DOPRI5) et par une méthode pour des équations différentielles raides (RADAU5)

**Exemple 1.3 (intégration à long terme du système planétaire)** Comme dernier exemple, considérons le problème à  $N$  corps qui est important aussi bien dans l'astronomie (mouvement des planètes) que dans la biologie moléculaire (mouvement des atomes). Les équations sont

$$y''_i = G \sum_{j \neq i} m_j \frac{y_j - y_i}{\|y_j - y_i\|^3} \quad (1.3)$$

où  $y_i \in I\!\!R^3$  est la position du  $i$ ème corps,  $m_i$  sa masse, et  $G$  la constante gravitationnelle. En introduisant la vitesse  $v_i = y'_i$  comme nouvelle variable, on obtient un système d'équations différentielles pour les  $y_i$  et  $v_i$  de dimension  $6N$  où  $N$  est le nombre des corps considérés.

Comme exemple concret, considérons le mouvement de cinq planètes extérieures autour du soleil ( $N = 6$  corps).<sup>1</sup> La figure III.3 montre la différence entre une méthode classique (méthode d'Euler explicite) et une méthode adaptée à ce problème; voir le paragraphe III.10 pour une explication.

<sup>1</sup>La constante  $G$ , les masses  $m_i$  et les valeurs initiales pour les positions et vitesses sont données dans le paragraphe I.2.3 du livre *Geometric Numerical Integration* de Hairer, Lubich & Wanner (2002).

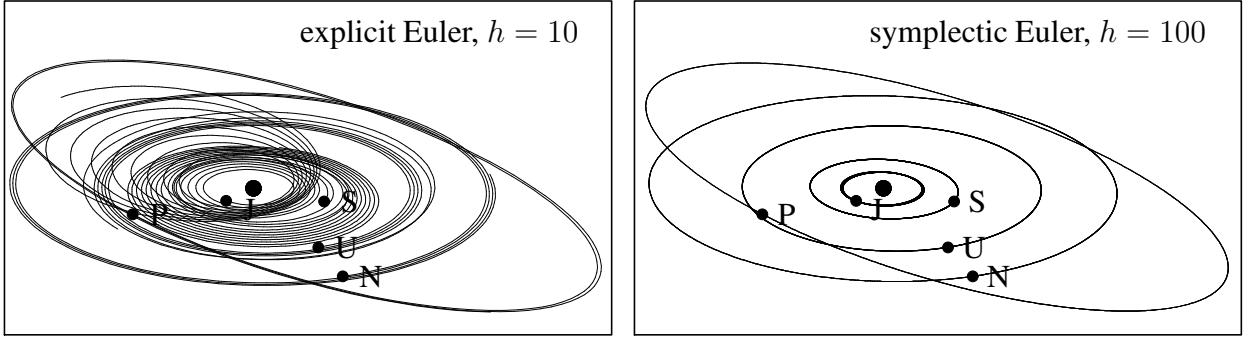


FIG. III.3: Solution numérique pour (1.3) obtenue par une méthode classique (méthode d'Euler explicite) et par une méthode étudiée au paragraphe III.10

Avant de discuter la résolution numérique des équations différentielles, nous rappelons un théorème sur l'existence et l'unicité de la solution (pour une démonstration, voir le cours d'Analyse II).

**Théorème 1.4** *Soit  $f(t, y)$  une fonction continûment différentiable dans un voisinage de  $(t_0, y_0)$ . Alors, il existe  $\alpha > 0$  tel que le problème  $y' = f(t, y), y(t_0) = y_0$  possède exactement une solution sur l'intervalle  $(t_0 - \alpha, t_0 + \alpha)$ .*  $\square$

## III.2 Méthodes de Runge-Kutta

Pour calculer une approximation de la solution de

$$y' = f(t, y), \quad y(t_0) = y_0 \quad (2.1)$$

sur l'intervalle  $[t_0, T]$ , on procède comme suit: on subdivise  $[t_0, T]$  en sous-intervalles d'extrémités  $t_0 < t_1 < \dots < t_N = T$ , on dénote  $h_n = t_{n+1} - t_n$  et on calcule l'approximation  $y_n \approx y(t_n)$  par une formule de type

$$y_{n+1} = y_n + h_n \Phi(t_n, y_n, h_n). \quad (2.2)$$

Une telle formule s'appelle “méthode à un pas”, car le calcul de  $y_{n+1}$  utilise uniquement les valeurs  $h_n, t_n, y_n$  et non  $h_{n-1}, t_{n-1}, y_{n-1}, \dots$

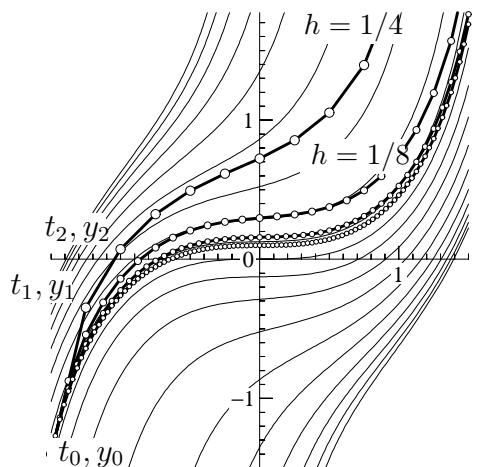
**Méthode d'Euler (1768).** La méthode la plus simple est

$$y_1 = y_0 + h f(t_0, y_0)$$

(pour simplifier la notation nous considérons uniquement le premier pas ( $n = 0$  dans (2.2)) et nous notons  $h_0 = h$ ). Elle est obtenue en remplaçant la solution  $y(t)$  par sa tangente au point  $(t_0, y_0)$ . Le dessin à droite montre la solution numérique pour le problème

$$y' = t^2 + y^2, \quad y(-1.5) = -1.4$$

et pour  $h = 1/4$ ,  $h = 1/8$ , etc. On peut observer la convergence vers une fonction qui, comme on verra dans le paragraphe III.3, est la solution du problème.



Pour la dérivation d'autres méthodes numériques, intégrons (2.1) de  $t_0$  à  $t_0 + h$

$$y(t_0 + h) = y_0 + \int_{t_0}^{t_0+h} f(\tau, y(\tau)) d\tau. \quad (2.3)$$

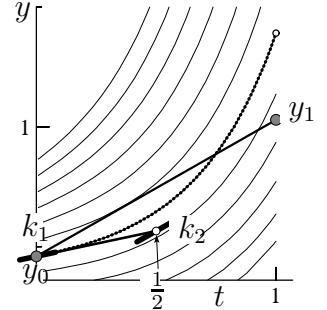
Si l'on remplace l'intégrale de (2.3) par  $hf(t_0, y_0)$ , on obtient la méthode d'Euler. L'idée évidente est d'approcher cette intégrale par une formule de quadrature ayant un ordre plus élevé.

**Méthode de Runge (1895).** On prend la formule du point milieu

$$y(t_0 + h) \approx y_0 + hf\left(t_0 + \frac{h}{2}, y(t_0 + \frac{h}{2})\right)$$

et on remplace la valeur inconnue  $y(t_0 + h/2)$  par la méthode d'Euler. Ceci nous donne

$$y_1 = y_0 + hf\left(t_0 + \frac{h}{2}, y_0 + \frac{h}{2}f(t_0, y_0)\right).$$



**Méthode de Heun (1900).** On prend une formule de quadrature d'ordre 3

$$y(t_0 + h) \approx y_0 + \frac{h}{4} \left( f(t_0, y_0) + 3f\left(t_0 + \frac{2h}{3}, y(t_0 + \frac{2h}{3})\right) \right) \quad (2.4)$$

et on remplace la valeur inconnue  $y(t_0 + 2h/3)$  par l'approximation de la méthode de Runge. Ceci nous donne

$$y_1 = y_0 + \frac{h}{4} \left( f(t_0, y_0) + 3f\left(t_0 + \frac{2h}{3}, y_0 + \frac{2h}{3}f\left(t_0 + \frac{h}{3}, y_0 + \frac{h}{3}f(t_0, y_0)\right)\right) \right). \quad (2.5)$$

En généralisant cette idée à une formule de quadrature plus générale et en introduisant la notation  $k_i = f(\dots)$  pour les expressions de  $f(t, y)$  qui apparaissent, on est conduit à la définition suivante (Kutta 1901).

**Définition 2.1** Une méthode de Runge-Kutta à  $s$  étages est donnée par

$$\begin{aligned} k_1 &= f(t_0, y_0) \\ k_2 &= f(t_0 + c_2 h, y_0 + h a_{21} k_1) \\ k_3 &= f(t_0 + c_3 h, y_0 + h(a_{31} k_1 + a_{32} k_2)) \\ &\dots \\ k_s &= f(t_0 + c_s h, y_0 + h(a_{s1} k_1 + \dots + a_{s,s-1} k_{s-1})) \\ y_1 &= y_0 + h(b_1 k_1 + \dots + b_s k_s) \end{aligned} \quad (2.6)$$

où  $c_i, a_{ij}, b_j$  sont des coefficients. On la représente à l'aide du schéma

$c_i$	$a_{ij}$
	$b_i$

**Exemples.** Les méthodes d'Euler, de Runge et de Heun sont données par les tableaux suivants:

$0$	$1/2$	$1/2$	$0$	$1/3$	$1/3$
$1$	0	1	0	2/3	2/3

Par la suite nous supposerons toujours que les  $c_i$  satisfont

$$c_1 = 0, \quad c_i = \sum_{j=1}^{i-1} a_{ij}, \quad i = 2, \dots, s. \quad (2.7)$$

Ceci signifie que  $k_i = f(t_0 + c_i h, y(t_0 + c_i h)) + \mathcal{O}(h^2)$ . Nous étendons maintenant la notion de l'ordre (voir la Définition I.1.2 pour les formules de quadrature) aux méthodes de Runge-Kutta.

**Définition 2.2** On dit que la méthode (2.6) a l'ordre  $p$  si, pour chaque problème  $y' = f(t, y)$ ,  $y(t_0) = y_0$  (avec  $f(t, y)$  suffisamment différentiable), l'erreur après un pas satisfait

$$y_1 - y(t_0 + h) = \mathcal{O}(h^{p+1}) \quad \text{pour} \quad h \rightarrow 0. \quad (2.8)$$

La différence (2.8) s'appelle erreur locale de la méthode.

La méthode d'Euler est une méthode d'ordre  $p = 1$ , car

$$y(t_0 + h) = y_0 + hy'(t_0) + \mathcal{O}(h^2) = y_0 + hf(t_0, y_0) + \mathcal{O}(h^2) = y_1 + \mathcal{O}(h^2).$$

La méthode de Runge est basée sur la formule du point milieu qui est une formule de quadrature d'ordre 2:

$$y(t_0 + h) = y_0 + hf\left(t_0 + \frac{h}{2}, y(t_0 + \frac{h}{2})\right) + \mathcal{O}(h^3).$$

En remplaçant  $y(t_0 + h/2)$  par la valeur  $y_0 + (h/2)f(t_0, y_0)$  de la méthode d'Euler, on ajoute un terme d'erreur qui est de  $\mathcal{O}(h^3)$  grâce au facteur  $h$  devant  $f$ . Ainsi, cette méthode a l'ordre  $p = 2$ . De la même manière on voit que la méthode de Heun a l'ordre  $p = 3$ .

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 série de Taylor autour de  $h = 0$ . Une comparaison des coefficients de  $h^i$  pour  $i = 1, \dots, p$  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 triviale (8 conditions pour l'ordre  $p = 4$ , 200 pour  $p = 8$  et 1205 pour  $p = 10$ ). Mais c'est de cette manière que Kutta (1901) a trouvé des méthodes d'ordre 4. Les plus célèbres sont données dans le tableau III.1. Celle de gauche est basée sur la formule de Simpson, l'autre sur la formule de quadrature de Newton.

TAB. III.1: Méthodes de Kutta (1901)

$\begin{array}{c ccccc} 0 & & & & & \\ 1/2 & 1/2 & & & & \\ 1/2 & 0 & 1/2 & & & \\ 1 & 0 & 0 & 1 & & \\ \hline & 1/6 & 2/6 & 2/6 & 1/6 \end{array}$	$\begin{array}{c ccccc} 0 & & & & & \\ 1/3 & 1/3 & & & & \\ 2/3 & -1/3 & 1 & & & \\ 1 & 1 & -1 & 1 & & \\ \hline & 1/8 & 3/8 & 3/8 & 1/8 \end{array}$
“La” méthode de Runge-Kutta	règle 3/8

Les méthodes de Runge-Kutta, qui sont actuellement les plus utilisées, ont été construites autour de 1980 par Dormand et Prince (Angleterre). Pour des méthodes d'ordre  $p = 5$  avec  $s = 6$  et d'ordre  $p = 8$  avec  $s = 12$ , des programmes informatiques DOPRI5 et DOP853 sont disponibles sur la page internet <<http://www.unige.ch/~hairer>>.

**Expérience numérique.** Considérons les cinq méthodes vues jusqu'à maintenant (Euler, Runge, Heun et les deux méthodes du tableau III.1 de Kutta) et comparons leurs performances pour le problème (équation Van der Pol)

$$\begin{aligned} y'_1 &= y_2 & y_1(0) &= 2.00861986087484313650940188 \\ y'_2 &= (1 - y_1^2)y_2 - y_1 & y_2(0) &= 0. \end{aligned} \quad (2.9)$$

La valeur initiale est choisie pour que la solution soit périodique de période

$$T = 6.6632868593231301896996820305.$$

Nous subdivisons l'intervalle  $[0, T]$  en  $n$  parties équidistantes et appliquons  $n$  fois la méthode. L'erreur à la fin de l'intervalle est alors dessinée en fonction du travail (nombre total d'évaluations de  $f$ ) dans la figure III.4.

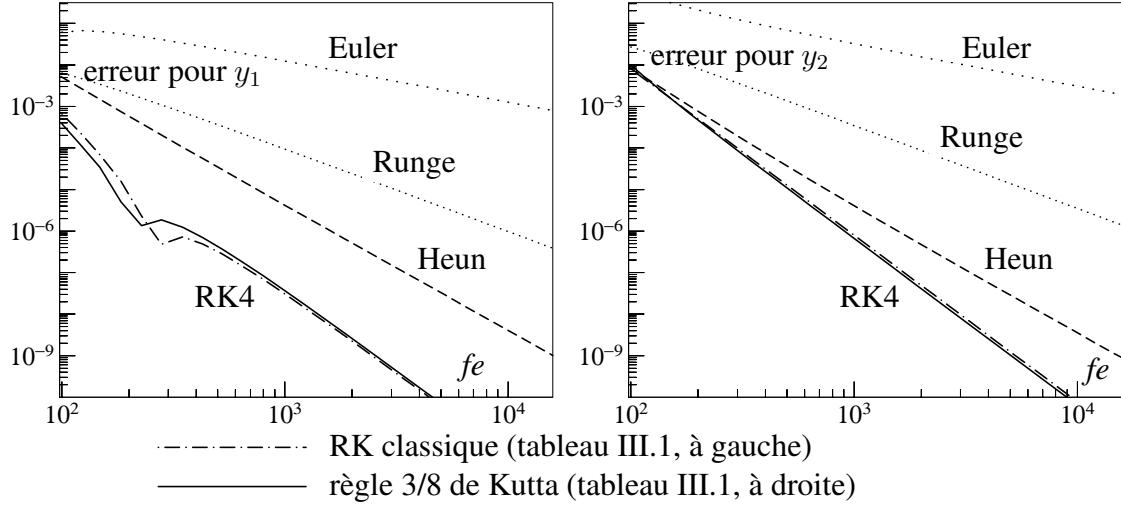


FIG. III.4: Erreur globale en fonction du travail numérique

Comme dans la fig. I.4 (intégration numérique), on peut constater que  $\log_{10}(err)$  dépend linéairement de  $\log_{10}(fe)$  et que cette droite est de pente  $-p$ , où  $p$  est l'ordre de la méthode. Il est donc important d'utiliser des méthodes d'ordre élevé.

### III.3 Convergence des méthodes de Runge-Kutta

Dans la figure III.4, on a constaté que pour un calcul avec des pas constants, l'erreur globale se comporte comme  $\log_{10}(err) \approx C_0 - p \cdot \log_{10}(fe)$ , ce qui est équivalent à  $err \approx C_1(fe)^{-p} \approx C_2 h^p$ . Ceci montre que la solution numérique converge vers la solution exacte si  $h \rightarrow 0$ . Dans ce paragraphe, nous allons démontrer ce résultat.

Nous appliquons une méthode à un pas

$$y_{n+1} = y_n + h_n \Phi(t_n, y_n, h_n) \quad (3.1)$$

à une équation différentielle  $y' = f(t, y)$ ,  $y(t_0) = y_0$ , et nous cherchons à estimer l'*erreur globale*  $y(t_n) - y_n$ .

**Théorème 3.1** Soit  $y(t)$  la solution de  $y' = f(t, y)$ ,  $y(t_0) = y_0$  sur  $[t_0, T]$ . Supposons que

a) l'*erreur locale* satisfasse pour  $t \in [t_0, T]$  et  $h \leq h_{\max}$

$$\|y(t+h) - y(t) - h\Phi(t, y(t), h)\| \leq C \cdot h^{p+1} \quad (3.2)$$

b) la fonction  $\Phi(t, y, h)$  satisfasse une condition de Lipschitz

$$\|\Phi(t, y, h) - \Phi(t, z, h)\| \leq \Lambda \cdot \|y - z\| \quad (3.3)$$

pour  $h \leq h_{\max}$  et  $(t, y), (t, z)$  dans un voisinage de la solution.

Alors, l'*erreur globale* admet pour  $t_n \leq T$  l'estimation

$$\|y(t_n) - y_n\| \leq h^p \cdot \frac{C}{\Lambda} \cdot (e^{\Lambda(t_n - t_0)} - 1) \quad (3.4)$$

où  $h = \max_i h_i$ , sous la condition que  $h$  soit suffisamment petit.

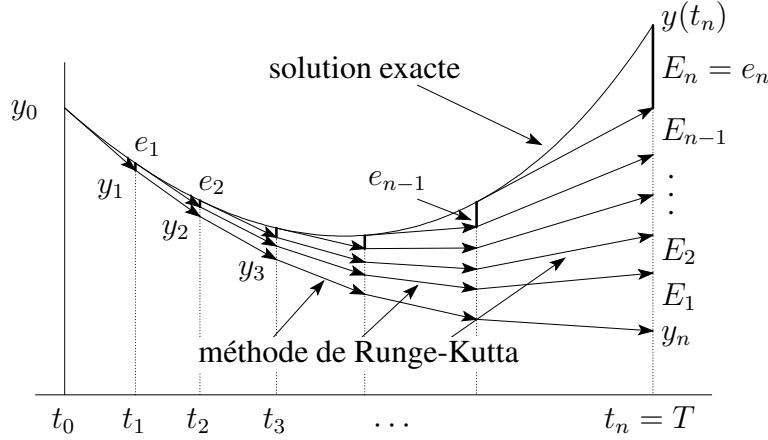


FIG. III.5: Estimation de l'erreur globale

*Démonstration.* L'idée est d'étudier l'influence de l'erreur locale, commise au  $i^{\text{ème}}$  pas, sur l'approximation  $y_n$ . Ensuite, on va additionner les erreurs accumulées.

*Propagation de l'erreur.* Soient  $\{y_n\}$  et  $\{z_n\}$  deux solutions numériques obtenues par (3.1) avec pour valeurs initiales  $y_0$  et  $z_0$ , respectivement. En utilisant la condition de Lipschitz (3.3), leur différence peut être estimée comme

$$\|y_{n+1} - z_{n+1}\| \leq \|y_n - z_n\| + h_n \Lambda \|y_n - z_n\| = (1 + h_n \Lambda) \|y_n - z_n\| \leq e^{h_n \Lambda} \|y_n - z_n\|. \quad (3.5)$$

Récursevement, on obtient alors

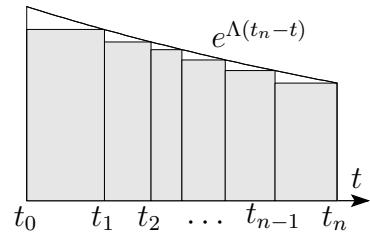
$$\|y_n - z_n\| \leq e^{h_{n-1}\Lambda} \cdot e^{h_{n-2}\Lambda} \cdot \dots \cdot e^{h_i\Lambda} \|y_i - z_i\| = e^{\Lambda(t_n - t_i)} \|y_i - z_i\|.$$

et l'erreur propagée  $E_i$  (voir la figure III.5) satisfait

$$\|E_i\| \leq e^{\Lambda(t_n - t_i)} \|e_i\| \leq Ch_{i-1}^{p+1} e^{\Lambda(t_n - t_i)}. \quad (3.6)$$

*Accumulation des erreurs propagées.* L'inégalité du triangle et l'estimation (3.6) nous donnent

$$\begin{aligned}
\|y(t_n) - y_n\| &\leq \sum_{i=1}^n \|E_i\| \leq C \sum_{i=1}^n h_{i-1}^{p+1} e^{\Lambda(t_n - t_i)} \\
&\leq Ch^p \left( h_0 e^{\Lambda(t_n - t_1)} + \dots + h_{n-2} e^{\Lambda(t_n - x_{t-1})} + h_{n-1} \right) \\
&\leq Ch^p \int_{t_0}^{t_n} e^{\Lambda(t_n - \tau)} d\tau = Ch^p \frac{1}{-\Lambda} e^{\Lambda(t_n - \tau)} \Big|_{t_0}^{t_n} \\
&= \frac{Ch^p}{\Lambda} \left( e^{\Lambda(t_n - t_0)} - 1 \right).
\end{aligned}$$



Cette estimation montre que, pour  $h$  suffisamment petit, la solution numérique ne sort pas du voisinage où  $\Phi$  satisfait à une condition de Lipschitz. Ceci justifie les estimations dans (3.5).  $\square$

Supposons maintenant que (3.1) représente une méthode de Runge-Kutta et vérifions les hypothèses du théorème précédent. La condition (3.2) est satisfaite pour une méthode d'ordre  $p$  (par définition de l'ordre). Il reste à vérifier la condition de Lipschitz (3.3) pour la fonction

$$\Phi(t, y, h) = \sum_{i=1}^s b_i k_i(y) \quad (3.7)$$

où  $k_i(y) = f(t + c_i h, y + h \sum_{j=1}^{i-1} a_{ij} k_j(y))$ .

**Lemme 3.2** Si  $f(t, y)$  satisfait une condition de Lipschitz  $\|f(t, y) - f(t, z)\| \leq L\|y - z\|$  dans un voisinage de la solution de  $y' = f(t, y)$ , l'expression  $\Phi(t, y, h)$  de (3.7) vérifie (3.3) avec

$$\Lambda = L \left( \sum_i |b_i| + (h_{\max} L) \sum_{i,j} |b_i a_{ij}| + (h_{\max} L)^2 \sum_{i,j,k} |b_i a_{ij} a_{jk}| + \dots \right). \quad (3.8)$$

*Démonstration.* La condition de Lipschitz pour  $f(t, y)$  appliquée à  $k_i(y) - k_i(z)$  nous donne

$$\begin{aligned} \|k_1(y) - k_1(z)\| &= \|f(t, y) - f(t, z)\| \leq L\|y - z\| \\ \|k_2(y) - k_2(z)\| &\leq L\|y - z + ha_{21}(k_1(y) - k_1(z))\| \leq L(1 + h_{\max} L |a_{21}|) \|y - z\| \end{aligned}$$

etc. Ces estimations insérées dans

$$\|\Phi(t, y, h) - \Phi(t, z, h)\| \leq \sum_{i=1}^s |b_i| \cdot \|k_i(y) - k_i(z)\|$$

impliquent (3.3) avec  $\Lambda$  donné par (3.8).  $\square$

### III.4 Un programme à pas variables

Pour résoudre un problème réaliste, un calcul à pas constants est en général inefficace. Mais comment choisir la division? L'idée est de choisir les pas afin que l'erreur locale soit partout environ égale à  $Tol$  (fourni par l'utilisateur). A cette fin, il faut connaître une estimation de l'erreur locale. Inspiré par le programme TEGRAL pour l'intégration numérique (voir le paragraphe I.6), nous construisons une deuxième méthode de Runge-Kutta avec  $\hat{y}_1$  comme approximation numérique, et nous utilisons la différence  $\hat{y}_1 - y_1$  comme estimation de l'erreur locale du moins bon résultat.

**Méthode emboîtée.** Soit donnée une méthode d'ordre  $p$  à  $s$  étages (coefficients  $c_i, a_{ij}, b_j$ ). On cherche une approximation  $\hat{y}_1$  d'ordre  $\hat{p} < p$  qui utilise les mêmes évaluations de  $f$ , c.-à-d.,

$$\hat{y}_1 = y_0 + h(\hat{b}_1 k_1 + \dots + \hat{b}_s k_s)$$

où les  $k_i$  sont donnés par la méthode (2.6). Pour avoir plus de liberté, on ajoute souvent un terme avec  $f(x_1, y_1)$ , qu'il faut en tous cas calculer pour le pas suivant, et on cherche  $\hat{y}_1$  de la forme

$$\hat{y}_1 = y_0 + h(\hat{b}_1 k_1 + \dots + \hat{b}_s k_s + \hat{b}_{s+1} f(x_1, y_1)). \quad (4.1)$$

**Exemple.** Pour la méthode de Runge, basée sur la règle du point milieu, on peut prendre la méthode d'Euler comme méthode emboîtée. L'expression  $err = h(k_2 - k_1)$  est donc une approximation de l'erreur locale (pour la méthode d'Euler).

Pour une méthode générale, il faut développer les  $k_i$  et  $f(x_1, y_1)$  en série de Taylor et comparer avec la solution exacte. Comme les  $c_i$  et les  $a_{ij}$  sont déjà connus, on obtient un système linéaire pour le  $\hat{b}_i$ .

En faisant ce calcul pour la méthode “règle 3/8” (ordre  $p = 4$ , tableau III.1), les coefficients d'une méthode emboîtée avec  $\hat{p} = 3$  sont

$$\hat{b}_1 = b_1 - \frac{c}{24} \quad \hat{b}_2 = b_2 + \frac{c}{8} \quad \hat{b}_3 = b_3 - \frac{c}{8} \quad \hat{b}_4 = b_4 - \frac{c}{8} \quad \hat{b}_5 = \frac{c}{6}, \quad (4.2)$$

et avec  $c = 1$  on obtient donc

$$err = \frac{h}{24} (-k_1 + 3k_2 - 3k_3 - 3k_4 + 4f(x_1, y_1)) \quad (4.3)$$

comme estimation de l'erreur locale.

**Calcul du  $h$  “optimal”.** Si l’on applique la méthode avec une certaine valeur  $h$ , l’estimation de l’erreur satisfait ( $\hat{p} < p$ )

$$y_1 - \hat{y}_1 = (y_1 - y(t_0 + h)) + (y(t_0 + h) - \hat{y}_1) = \mathcal{O}(h^{p+1}) + \mathcal{O}(h^{\hat{p}+1}) \approx C \cdot h^{\hat{p}+1}. \quad (4.4)$$

Le  $h$  “optimal”, noté par  $h_{\text{opt}}$ , est celui où cette estimation est proche de  $Tol$ :

$$Tol \approx C \cdot h_{\text{opt}}^{\hat{p}+1}. \quad (4.5)$$

En éliminant  $C$  de (4.4) et de (4.5), on obtient

$$h_{\text{opt}} = 0.9 \cdot h \cdot \sqrt[{\hat{p}+1}]{\frac{Tol}{\|y_1 - \hat{y}_1\|}}. \quad (4.6)$$

Le facteur 0.9 est ajouté pour rendre le programme plus sûr. En général on suppose en plus une restriction du type  $0.2h \leq h_{\text{opt}} \leq 5h$  pour éviter des grandes variations dans  $h$ .

Pour la norme dans (4.6) on utilise en général

$$\|y_1 - \hat{y}_1\| = \sqrt{\frac{1}{n} \sum_{i=1}^n \left( \frac{y_{i1} - \hat{y}_{i1}}{sc_i} \right)^2} \quad \text{où} \quad sc_i = 1 + \max(|y_{i0}|, |y_{i1}|) \quad (4.7)$$

( $y_{i0}, y_{i1}, \hat{y}_{i1}$  est la  $i$ ème composante de  $y_0, y_1, \hat{y}_1$ , respectivement). Ceci représente un mélange entre erreur relative et erreur absolue.

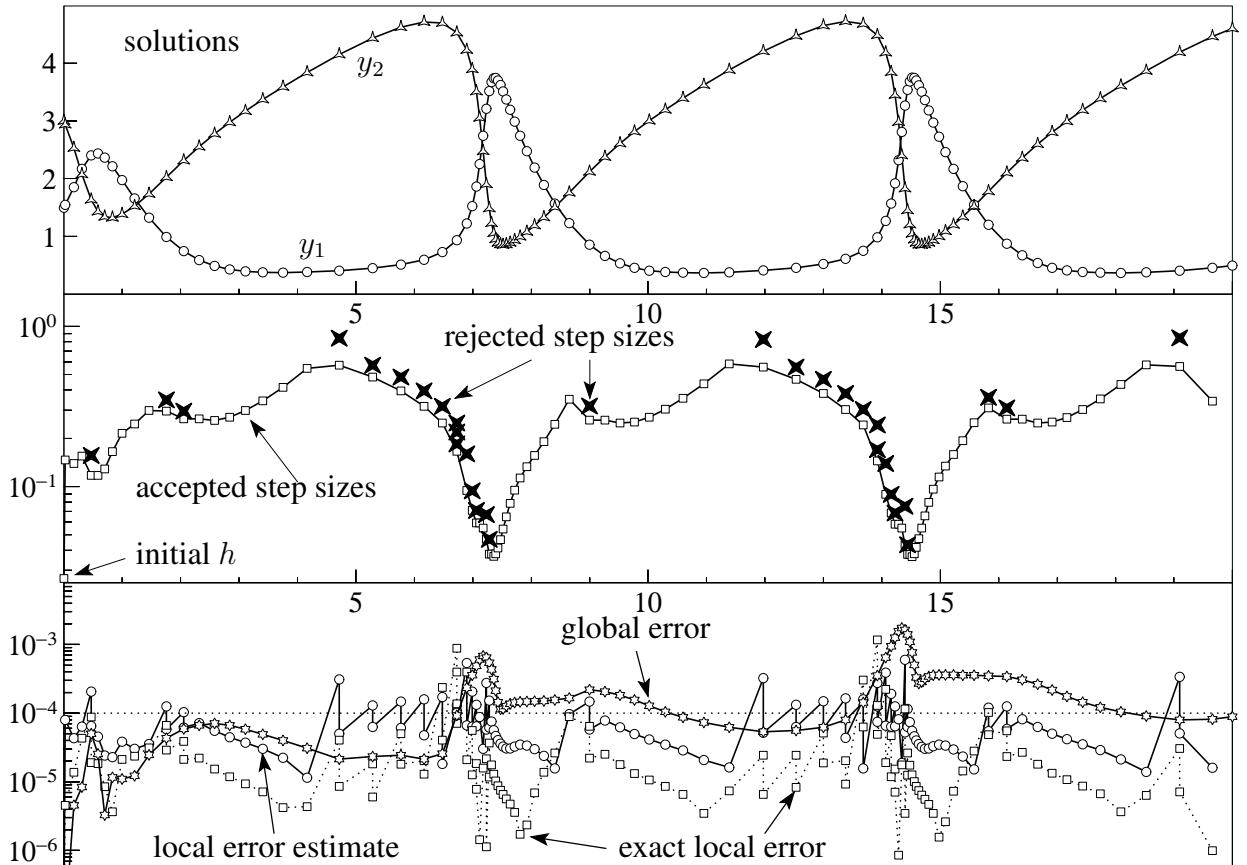


FIG. III.6: Sélection du pas,  $Tol = 10^{-4}$ , 96 pas acceptés + 32 pas rejetés

**Algorithme pour la sélection automatique du pas.** Au début, l'utilisateur fournit un sous-programme qui calcule la valeur de  $f(t, y)$ , les valeurs initiales  $t_0, y_0$  et un premier choix de  $h$ .

A) Avec  $h$ , calculer  $y_1$ ,  $err = \|y_1 - \hat{y}_1\|$  et  $h_{opt}$  de (4.6).  
B) **If**  $err \leq Tol$  (le pas est accepté) **then**  
     $t_0 := t_0 + h$ ,  $y_0 := y_1$ ,  $h := \min(h_{opt}, t_{end} - t_0)$   
**else** (le pas est rejeté)  
     $h := h_{opt}$   
**end if**

C) Si  $t_0 = t_{end}$  on a terminé, sinon on recommence en (A) et on calcule le pas suivant.

**Exemple numérique.** Cet algorithme a été programmé (en utilisant la “règle 3/8” et l'estimation de l'erreur (4.3)) et il a été appliqué au problème (une réaction chimique, le “Brusselator”)

$$\begin{aligned} y'_1 &= 1 + y_1^2 y_2 - 4y_1 & y_1(0) &= 1.5 \\ y'_2 &= 3y_1 - y_1^2 y_2 & y_2(0) &= 3 \end{aligned} \quad (4.8)$$

sur l'intervalle  $[0, 20]$ . Les résultats obtenus avec  $Tol = 10^{-4}$  sont présentés dans la figure III.6 :

- i) en haut, les deux composantes de la solution avec tous les pas acceptés;
- ii) au milieu les pas; les pas acceptés étant reliés, les pas rejetés étant indiqués par  $\times$ ;
- iii) les dessins du bas montre l'estimation de l'erreur locale  $err$ , ainsi que les valeurs exactes de l'erreur locale et de l'erreur globale.

## III.5 Méthodes multipas (multistep methods)

Déjà longtemps avant la parution des premières méthodes de Runge-Kutta, J.C. Adams a résolu numériquement des équations différentielles (1855, publié dans un livre de Bashforth 1883). Son idée était d'utiliser l'information de plusieurs pas précédents (en particulier  $y_n, y_{n-1}, \dots, y_{n-k+1}$ ) pour obtenir une approximation précise de  $y(t_{n+1})$ . C'est la raison pour laquelle ces méthodes s'appellent aujourd'hui méthodes multipas (contrairement aux méthodes à un pas).

**Méthodes d'Adams explicites.** Soit donnée une division  $t_0 < \dots < t_n < t_{n+1} < \dots$  de l'intervalle sur lequel on cherche à résoudre l'équation différentielle  $y' = f(t, y)$  et supposons qu'on connaisse des approximations  $y_n, y_{n-1}, \dots, y_{n-k+1}$  de la solution pour  $k$  pas consécutifs ( $y_j \approx y(t_j)$ ). Comme pour la dérivation des méthodes de Runge-Kutta, on intègre l'équation différentielle pour obtenir

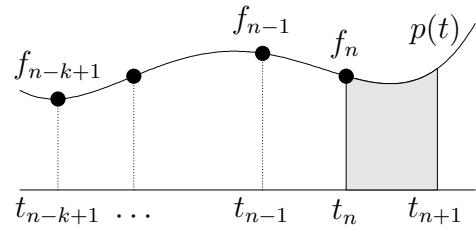
$$y(t_{n+1}) = y(t_n) + \int_{t_n}^{t_{n+1}} f(\tau, y(\tau)) d\tau. \quad (5.1)$$

Mais, au lieu d'appliquer une formule de quadrature standard à l'intégrale de (5.1), on remplace  $f(t, y(t))$  par le polynôme  $p(t)$  de degré  $k - 1$  qui satisfait (on utilise l'abréviation  $f_j = f(t_j, y_j)$ )

$$\begin{aligned} p(t_j) &= f_j \\ \text{pour} \\ j &= n, n-1, \dots, n-k+1. \end{aligned}$$

L'approximation de  $y(t_{n+1})$  est alors définie par

$$y_{n+1} = y_n + \int_{t_n}^{t_{n+1}} p(\tau) d\tau. \quad (5.2)$$



Si l'on représente le polynôme  $p(t)$  par la formule de Newton (voir le paragraphe II.1)

$$p(t) = \sum_{j=0}^{k-1} \left( \prod_{i=0}^{j-1} (t - t_{n-i}) \right) \cdot \delta^j f[t_n, t_{n-1}, \dots, t_{n-j}] \quad (5.3)$$

la méthode (5.2) devient

$$y_{n+1} = y_n + \sum_{j=0}^{k-1} \left( \int_{t_n}^{t_{n+1}} \prod_{i=0}^{j-1} (t - t_{n-i}) dt \right) \cdot \delta^j f[t_n, t_{n-1}, \dots, t_{n-j}]. \quad (5.4)$$

*Cas équidistant.* Dans la situation  $t_j = t_0 + jh$  les différences divisées peuvent être écrites sous la forme

$$\delta^j f[t_n, t_{n-1}, \dots, t_{n-j}] = \frac{\nabla^j f_n}{j! h^j} \quad (5.5)$$

où  $\nabla^0 f_n = f_n$ ,  $\nabla f_n = f_n - f_{n-1}$ ,  $\nabla^2 f_n = \nabla(\nabla f_n) = f_n - 2f_{n-1} + f_{n-2}, \dots$  sont les *différences finies régressives* (à distinguer des différences finies progressives  $\Delta f_n = f_{n+1} - f_n$ ). La formule (5.4) devient alors (poser  $t = t_n + sh$ )

$$y_{n+1} = y_n + h \sum_{j=0}^{k-1} \gamma_j \nabla^j f_n \quad (5.6)$$

où

$$\gamma_j = \frac{1}{j!} \int_0^1 \prod_{i=0}^{j-1} (s + i) ds = \int_0^1 \binom{s+j-1}{j} ds. \quad (5.7)$$

Les premiers coefficients  $\gamma_j$  sont donnés dans le tableau III.2.

TAB. III.2: Coefficients pour les méthodes d'Adams explicites

$j$	0	1	2	3	4	5	6	7	8
$\gamma_j$	1	$\frac{1}{2}$	$\frac{5}{12}$	$\frac{3}{8}$	$\frac{251}{720}$	$\frac{95}{288}$	$\frac{19087}{60480}$	$\frac{5257}{17280}$	$\frac{1070017}{3628800}$

Des cas particuliers sont:

$$\begin{aligned} k = 1 : \quad & y_{n+1} = y_n + h f_n && \text{(méthode d'Euler)} \\ k = 2 : \quad & y_{n+1} = y_n + h \left( \frac{3}{2} f_n - \frac{1}{2} f_{n-1} \right) \\ k = 3 : \quad & y_{n+1} = y_n + h \left( \frac{23}{12} f_n - \frac{16}{12} f_{n-1} + \frac{5}{12} f_{n-2} \right) \\ k = 4 : \quad & y_{n+1} = y_n + h \left( \frac{55}{24} f_n - \frac{59}{24} f_{n-1} + \frac{37}{24} f_{n-2} - \frac{9}{24} f_{n-3} \right). \end{aligned}$$

Si l'on veut appliquer cette méthode (par exemple avec  $k = 3$ ) à la résolution de  $y' = f(t, y)$ ,  $y(t_0) = y_0$ , il faut connaître trois approximations initiales  $y_0, y_1$  et  $y_2$ . Ensuite, on peut utiliser la formule récursivement pour calculer  $y_3, y_4$ , etc. Adams a calculé la série de Taylor de la solution exacte (autour de la valeur initiale) pour déterminer les approximations initiales qui manquent. Evidemment, on peut aussi les obtenir par une méthode à un pas.

*Remarque.* Dans la construction de la méthode (5.4), on a utilisé le polynôme d'interpolation  $p(t)$  en-dehors de l'intervalle  $[t_{n-k+1}, t_n]$ . On sait bien (voir le chapitre II) que ceci peut provoquer de grandes erreurs. La modification suivante est aussi due à J.C. Adams (1855).

**Méthodes d'Adams implicites.** L'idée est de considérer le polynôme  $p^*(t)$  de degré  $k$  qui satisfasse

$$p^*(t_j) = f_j \quad \text{pour } j = n+1, n, n-1, \dots, n-k+1 \quad (5.8)$$

(attention:  $f_{n+1} = f(t_{n+1}, y_{n+1})$  est encore inconnu) et de définir l'approximation numérique par

$$y_{n+1} = y_n + \int_{t_n}^{t_{n+1}} p^*(\tau) d\tau. \quad (5.9)$$

Comme précédemment, la formule de Newton donne

$$p^*(t) = \sum_{j=0}^k \left( \prod_{i=0}^{j-1} (t - t_{n-i+1}) \right) \cdot \delta^j f[t_{n+1}, t_n, \dots, t_{n-j+1}] \quad (5.10)$$

et la méthode devient

$$y_{n+1} = y_n + \sum_{j=0}^k \left( \int_{t_n}^{t_{n+1}} \prod_{i=0}^{j-1} (t - t_{n-i+1}) dt \right) \cdot \delta^j f[t_{n+1}, t_n, \dots, t_{n-j+1}]. \quad (5.11)$$

Pour le cas équidistant, on a

$$y_{n+1} = y_n + h \sum_{j=0}^k \gamma_j^* \nabla^j f_{n+1} \quad (5.12)$$

où les coefficients  $\gamma_j^*$  sont donnés par (voir tableau III.3)

$$\gamma_j^* = \frac{1}{j!} \int_0^1 \prod_{i=0}^{j-1} (i-1+s) ds = \int_0^1 \binom{s+j-2}{j} ds. \quad (5.13)$$

TAB. III.3: Coefficients pour les méthodes d'Adams implicites

$j$	0	1	2	3	4	5	6	7	8
$\gamma_j^*$	1	$-\frac{1}{2}$	$-\frac{1}{12}$	$-\frac{1}{24}$	$-\frac{19}{720}$	$-\frac{3}{160}$	$-\frac{863}{60480}$	$-\frac{275}{24192}$	$-\frac{33953}{3628800}$

Des cas particuliers sont:

$$\begin{aligned} k = 0 : \quad & y_{n+1} = y_n + h f_{n+1} = y_n + h f(t_{n+1}, y_{n+1}) \\ k = 1 : \quad & y_{n+1} = y_n + h \left( \frac{1}{2} f_{n+1} + \frac{1}{2} f_n \right) \\ k = 2 : \quad & y_{n+1} = y_n + h \left( \frac{5}{12} f_{n+1} + \frac{8}{12} f_n - \frac{1}{12} f_{n-1} \right) \\ k = 3 : \quad & y_{n+1} = y_n + h \left( \frac{9}{24} f_{n+1} + \frac{19}{24} f_n - \frac{5}{24} f_{n-1} + \frac{1}{24} f_{n-2} \right). \end{aligned}$$

Chacune de ces formules représente une équation non linéaire pour  $y_{n+1}$ , de la forme

$$y_{n+1} = \eta_n + h \beta f(t_{n+1}, y_{n+1}) \quad (5.14)$$

(par exemple, pour  $k = 2$  on a  $\beta = 5/12$  et  $\eta_n = y_n + h(8f_n - f_{n-1})/12$ ). On peut résoudre ce système par les méthodes du chapitre VI (méthode de Newton) ou simplement par la méthode des approximations successives.

**Méthodes prédicteur-correcteur.** La solution de (5.14) est elle-même seulement une approximation de  $y(t_{n+1})$ . Ainsi, il n'est pas important de résoudre (5.14) à une très grande précision. L'idée est de calculer une première approximation par une méthode explicite et de corriger cette valeur (une ou plusieurs fois) par la formule (5.14). Avec cet algorithme, un pas de la méthode prend la forme suivante:

**P:** on calcule le prédicteur  $\hat{y}_{n+1} = y_n + h \sum_{j=0}^{k-1} \gamma_j \nabla^j f_n$  par la méthode d'Adams explicite;  $\hat{y}_{n+1}$  est déjà une approximation de  $y(t_{n+1})$ ;

**E:** évaluation de la fonction: on calcule  $\hat{f}_{n+1} = f(t_{n+1}, \hat{y}_{n+1})$ ;

**C:** l'approximation corrigée est alors donnée par  $y_{n+1} = \eta_n + h\beta \hat{f}_{n+1}$ ;

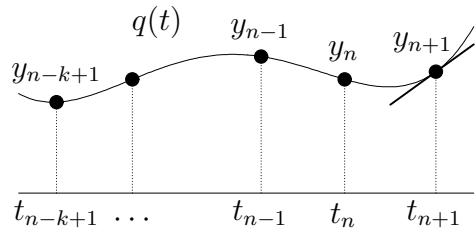
**E:** calculer  $f_{n+1} = f(t_{n+1}, y_{n+1})$ .

Cette procédure, qu'on dénote PECE, est la plus utilisée. D'autres possibilités sont: de faire plusieurs itérations, par exemple PECECE, ou d'omettre la dernière évaluation de  $f$  (c.-à-d. PEC) et de prendre  $\hat{f}_{n+1}$  à la place de  $f_{n+1}$  pour le pas suivant.

**Méthodes BDF (backward differentiation formulas).** Au lieu de travailler avec un polynôme qui passe par les  $f_j$ , on considère le polynôme  $q(t)$  de degré  $k$ , défini par

$$\begin{aligned} q(t_j) &= y_j \\ \text{pour} \\ j &= n+1, n, \dots, n-k+1 \end{aligned}$$

et on détermine  $y_{n+1}$  de façon telle que



$$q'(t_{n+1}) = f(t_{n+1}, q(t_{n+1})). \quad (5.15)$$

Comme dans (5.10), la formule de Newton donne

$$q(t) = \sum_{j=0}^k \left( \prod_{i=0}^{j-1} (t - t_{n-i+1}) \right) \cdot \delta^j y[t_{n+1}, t_n, \dots, t_{n-j+1}]. \quad (5.16)$$

Chaque terme de cette somme contient le facteur  $(t - t_{n+1})$ . Alors, on calcule facilement  $q'(t_{n+1})$  et on obtient

$$\sum_{j=1}^k \left( \prod_{i=1}^{j-1} (t_{n+1} - t_{n-i+1}) \right) \cdot \delta^j y[t_{n+1}, t_n, \dots, t_{n-j+1}] = f(t_{n+1}, y_{n+1}). \quad (5.17)$$

Pour le cas équidistant, cette formule devient (utiliser (5.5))

$$\sum_{j=1}^k \frac{1}{j} \nabla^j y_{n+1} = h f_{n+1}. \quad (5.18)$$

Des cas particuliers sont:

$$\begin{aligned} k = 1 : \quad & y_{n+1} - y_n = h f_{n+1} \\ k = 2 : \quad & \frac{3}{2} y_{n+1} - 2 y_n + \frac{1}{2} y_{n-1} = h f_{n+1} \\ k = 3 : \quad & \frac{11}{6} y_{n+1} - 3 y_n + \frac{3}{2} y_{n-1} - \frac{1}{3} y_{n-2} = h f_{n+1} \\ k = 4 : \quad & \frac{25}{12} y_{n+1} - 4 y_n + 3 y_{n-1} - \frac{4}{3} y_{n-2} + \frac{1}{4} y_{n-3} = h f_{n+1} \end{aligned}$$

De nouveau, chaque formule définit implicitement l'approximation numérique  $y_{n+1}$  (les méthodes BDF sont très importantes pour la résolution de problèmes dits "raides", voir le paragraphe III.9).

**Expérience numérique.** Pour plusieurs valeurs de  $k$ , nous avons appliqué la méthode d'Adams explicite (en pointillés dans la figure III.7,  $k = 1, 2, 3, 4$ ) ainsi que la méthode d'Adams implicite (sous la forme PECE, trait continu,  $k = 0, 1, 2, 3, 4$ ) au problème (2.9). Comme dans la figure III.4, le travail numérique est dessiné en fonction de l'erreur globale à la fin de l'intervalle.

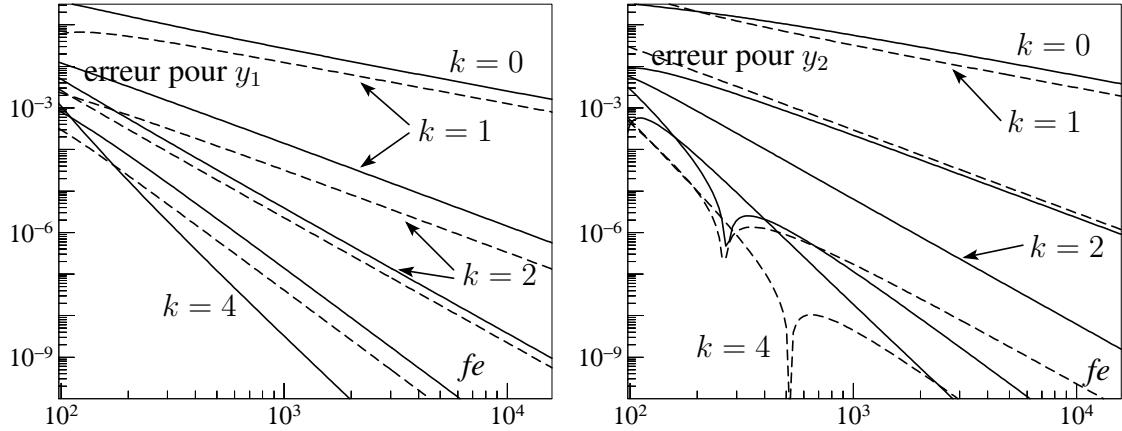


FIG. III.7: Erreur globale par rapport au travail numérique

On constate que cette erreur se comporte comme  $h^k$  pour les méthodes explicites et comme  $h^{k+1}$  pour les méthodes implicites. Pour pouvoir expliquer ce comportement, nous allons étudier l'erreur locale (ordre), la stabilité et la convergence des méthodes multipas.

### III.6 Étude de l'erreur locale

Toutes les méthodes du paragraphe précédent sont de la forme

$$\sum_{i=0}^k \alpha_i y_{n+i} = h \sum_{i=0}^k \beta_i f_{n+i} \quad (6.1)$$

où  $\alpha_k \neq 0$  et  $|\alpha_0| + |\beta_0| > 0$ . Une méthode est explicite si  $\beta_k = 0$ , sinon elle est implicite.

**Définition 6.1** Soit  $y(t)$  une solution de  $y' = f(t, y)$  et soit  $y_{n+k}$  la valeur obtenue par la méthode (6.1) en utilisant  $y_i = y(t_i)$  pour  $i = n, \dots, n+k-1$  (valeurs sur la solution exacte, voir la figure). Alors,

$$\text{erreur locale} := y(t_{n+k}) - y_{n+k}.$$

On dit que la méthode (6.1) a l'ordre  $p$  si l'erreur locale est  $\mathcal{O}(h^{p+1})$ .

**Théorème 6.2** Une méthode multipas a l'ordre  $p$ , si et seulement si ses coefficients satisfont

$$\sum_{i=0}^k \alpha_i = 0 \quad \text{et} \quad \sum_{i=0}^k \alpha_i i^q = q \sum_{i=0}^k \beta_i i^{q-1} \quad \text{pour } q = 1, \dots, p. \quad (6.2)$$

*Démonstration.* Le développement en série de Taylor du défaut de (6.1) donne

$$\sum_{i=0}^k \alpha_i y(t + ih) - h \sum_{i=0}^k \beta_i y'(t + ih) = \sum_{q \geq 0} d_q y^{(q)}(t) \frac{h^q}{q!} \quad (6.3)$$

où  $d_0 = \sum_i \alpha_i$  et  $d_q = \sum_i \alpha_i i^q - q \sum_i \beta_i i^{q-1}$ . Comme  $y_i = y(t_i)$  pour  $i = n, \dots, n+k-1$  dans la définition 6.1, on a  $f_i = f(t_i, y(t_i)) = y'(t_i)$  pour  $i = n, \dots, n+k-1$ , et en soustrayant la formule (6.1) de (6.3) on obtient

$$\alpha_k (y(t_{n+k}) - y_{n+k}) - h \beta_k (f(t_{n+k}, y(t_{n+k})) - f(t_{n+k}, y_{n+k})) = \sum_{q \geq 0} d_q y^{(q)}(t_n) \frac{h^q}{q!}.$$

et la condition que l'erreur locale soit  $\mathcal{O}(h^{p+1})$  devient  $d_0 = d_1 = \dots = d_p = 0$ .  $\square$

**Exemple** (méthode d'Adams explicite à  $k$  pas). Pour  $q \leq k$ , considérons l'équation différentielle  $y' = qt^{q-1}$  avec comme solution  $y(t) = t^q$ . Dans cette situation, le polynôme  $p(t)$  de (5.2) est égal à  $f(t, y(t))$  et la méthode d'Adams explicite donne le résultat exact, c.-à-d. le défaut (6.3) est zéro. Ceci implique  $d_q = 0$ . Ainsi, l'ordre de cette méthode est  $\geq k$  (on peut en fait montrer qu'il est égal à  $k$ ).

De la même manière, on montre que la méthode d'Adams implicite a l'ordre  $p = k + 1$  et la méthode BDF l'ordre  $p = k$ .

### III.7 Stabilité

La structure simple des conditions d'ordre pour les méthodes multipas (voir (6.2)) permet de construire des méthodes avec un ordre maximal. Mais, ces méthodes sont-elles utiles?

**Exemple** (Dahlquist 1956). Posons  $k = 2$  et construisons une méthode explicite ( $\beta_2 = 0$ ; normalisation  $\alpha_2 = 1$ ) avec un ordre maximal. Les conditions (6.2) avec  $p = 3$  nous donnent la méthode d'ordre 3 suivante

$$y_{n+2} + 4y_{n+1} - 5y_n = h(4f_{n+1} + 2f_n). \quad (7.1)$$

Une application à l'équation différentielle  $y' = y$ ,  $y(0) = 1$  donne la formule de récurrence

$$y_{n+2} + 4(1-h)y_{n+1} - (5+2h)y_n = 0. \quad (7.2)$$

Pour résoudre (7.2), nous insérons  $y_n = \zeta^n$  et obtenons l'équation caractéristique

$$\zeta^2 + 4(1-h)\zeta - (5+2h) = 0 \quad (7.3)$$

avec comme solution  $\zeta_1 = 1 + h + \mathcal{O}(h^2)$ ,  $\zeta_2 = -5 + \mathcal{O}(h)$ . La solution de (7.2) est alors

$$y_n = C_1\zeta_1^n + C_2\zeta_2^n \quad (7.4)$$

où les constantes  $C_1$  et  $C_2$  sont déterminées par  $y_0 = 1$  et  $y_1 = e^h$  (on a choisi la valeur  $y_1$  sur la solution exacte). Pour  $n$  grand, le terme  $C_2\zeta_2^n \approx C_2(-5)^n$  est dominant et on n'a aucun espoir que la solution numérique converge vers la solution exacte  $e^x$  (figure III.8).

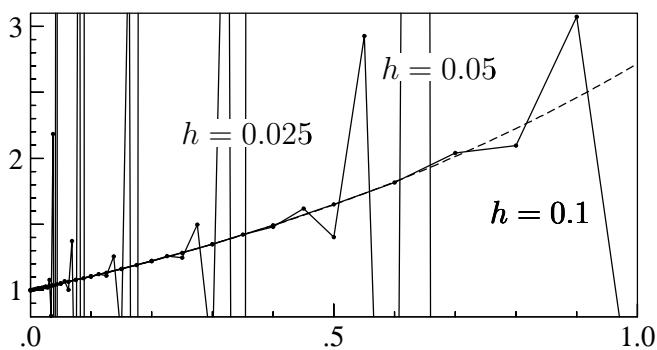


FIG. III.8: Instabilité de la méthode (7.1)

La raison de la divergence de la solution numérique dans l'exemple précédent est que le polynôme

$$\rho(\zeta) := \sum_{i=0}^k \alpha_i \zeta^i \quad (7.5)$$

possède une racine qui est plus grande que 1 en valeur absolue.

Pour trouver une condition nécessaire pour la convergence, considérons le problème  $y' = 0$  avec des valeurs initiales  $y_0, y_1, \dots, y_{k-1}$  perturbées. La solution numérique  $y_n$  satisfait

$$\alpha_k y_{n+k} + \dots + \alpha_0 y_n = 0 \quad (7.6)$$

et est donnée par une combinaison linéaire de

$$\begin{aligned} \zeta^n &\dots & \text{si } \zeta \text{ est une racine simple de } \rho(\zeta) = 0, \\ \zeta^n, n\zeta^n &\dots & \text{si } \zeta \text{ est une racine double de } \rho(\zeta) = 0, \\ \zeta^n, n\zeta^n, \dots, n^{\ell-1}\zeta^n &\dots & \text{si } \zeta \text{ est une racine de multiplicité } \ell. \end{aligned}$$

Pour que la solution numérique reste bornée, il faut que les conditions de la définition suivante soient remplies.

**Définition 7.1** Une méthode multipas est stable, si les racines du polynôme  $\rho(\zeta)$  satisfont

- i) si  $\rho(\hat{\zeta}) = 0$  alors  $|\hat{\zeta}| \leq 1$ ,
- ii) si  $\rho(\hat{\zeta}) = 0$  et  $|\hat{\zeta}| = 1$  alors  $\hat{\zeta}$  est une racine simple de  $\rho(\zeta)$ .

Pour les méthodes d'Adams, on a

$$\rho(\zeta) = \zeta^{k-1}(\zeta - 1). \quad (7.7)$$

Elles sont donc stables. Les méthodes BDF sont stables seulement pour  $k \leq 6$  (voir les exercices 17 et 18).

*Remarque.* Donnons encore sans démonstration un résultat intéressant qui s'appelle “la première barrière de Dahlquist”. Pour une méthode stable, l'ordre  $p$  satisfait  $p \leq k+2$  (si  $k$  est pair),  $p \leq k+1$  (si  $k$  est impair) et  $p \leq k$  (si la méthode est explicite).

## III.8 Convergence des méthodes multipas

Pour l'étude de la convergence des méthodes multipas, nous nous contentons du cas équidistant, le cas général étant trop technique.

**Théorème 8.1** Supposons que les  $k$  valeurs de départ satisfassent  $\|y(t_i) - y_i\| \leq C_0 h^p$  pour  $i = 0, 1, \dots, k-1$ . Si la méthode multipas (6.1) est d'ordre  $p$  et stable, alors elle est convergente d'ordre  $p$ , c.-à-d. que l'erreur globale satisfait

$$\|y(t_n) - y_n\| \leq Ch^p \quad \text{pour} \quad x_n - x_0 = nh \leq \text{Const.} \quad (8.1)$$

*Démonstration.* Le point essentiel de la démonstration est le suivant: on écrit formellement la méthode multipas (6.1) sous la forme d'une méthode à un pas et on applique les idées de la démonstration du paragraphe III.3.

*Formulation comme une méthode à un pas.* Avec  $\alpha_k = 1$ , la méthode multipas (6.1) devient

$$y_{n+k} = - \sum_{i=0}^{k-1} \alpha_i y_{n+i} + h\Psi(t_n, y_n, \dots, y_{n+k-1}, h). \quad (8.2)$$

Pour une méthode explicite ( $\beta_k = 0$ ), l'expression  $\Psi$  est donné par

$$\Psi(t_n, y_n, \dots, y_{n+k-1}, h) = \sum_{i=0}^{k-1} \beta_i f(t_{n+i}, y_{n+i}),$$

sinon elle est définie implicitement. Considérons maintenant les super-vecteurs

$$Y_n := (y_{n+k-1}, \dots, y_{n+1}, y_n)^T$$

et écrivons la méthode (8.2) sous la forme

$$Y_{n+1} = AY_n + h\Phi(t_n, Y_n, h) \quad \text{où} \quad (8.3)$$

$$A = \begin{pmatrix} -\alpha_{k-1} & -\alpha_{k-2} & \dots & -\alpha_1 & -\alpha_0 \\ 1 & 0 & \dots & 0 & 0 \\ & 1 & \ddots & & 0 \\ & & \ddots & \ddots & \vdots \\ & & & 1 & 0 \end{pmatrix} \quad \text{et} \quad \Phi(t, Y, h) = \begin{pmatrix} \Psi(t, Y, h) \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \quad (8.4)$$

(si les  $y_j$  étaient déjà des vecteurs, il faudrait remplacer  $A$  dans (8.3) par  $A \otimes I$ , etc; cependant nous n'utiliserons pas cette notation jugée trop lourde).

*Erreur locale.* Considérons des valeurs  $y_n, \dots, y_{n+k-1}$  sur la solution exacte, notons

$$Y(t_n) := (y(t_{n+k-1}), \dots, y(t_{n+1}), y(t_n))^T \quad (8.5)$$

et appliquons une fois la méthode multipas. Ceci donne

$$\hat{Y}_{n+1} = AY(t_n) + h\Phi(t_n, Y(t_n), h).$$

La première composante de  $\hat{Y}_{n+1} - Y(t_{n+1})$  est exactement l'erreur locale (définition 6.1), tandis que les autres composantes sont égales à zéro. Comme la méthode a l'ordre  $p$  par hypothèse, nous avons

$$\|\hat{Y}_{n+1} - Y(t_{n+1})\| \leq C_1 h^{p+1} \quad \text{pour} \quad t_{n+1} - t_0 = (n+1)h \leq \text{Const.} \quad (8.6)$$

*Propagation de l'erreur (stabilité).* Considérons une deuxième solution numérique, définie par

$$Z_{n+1} = AZ_n + h\Phi(t_n, Z_n, h)$$

et estimons la différence  $Y_{n+1} - Z_{n+1}$ . Pour les méthodes d'Adams, on a

$$\begin{pmatrix} y_{n+k} - z_{n+k} \\ y_{n+k-1} - z_{n+k-1} \\ \vdots \\ y_{n+1} - z_{n+1} \end{pmatrix} = \begin{pmatrix} y_{n+k-1} - z_{n+k-1} \\ y_{n+k-1} - z_{n+k-1} \\ \vdots \\ y_{n+1} - z_{n+1} \end{pmatrix} + h \begin{pmatrix} \Psi(t_n, Y_n, h) - \Psi(t_n, Z_n, h) \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

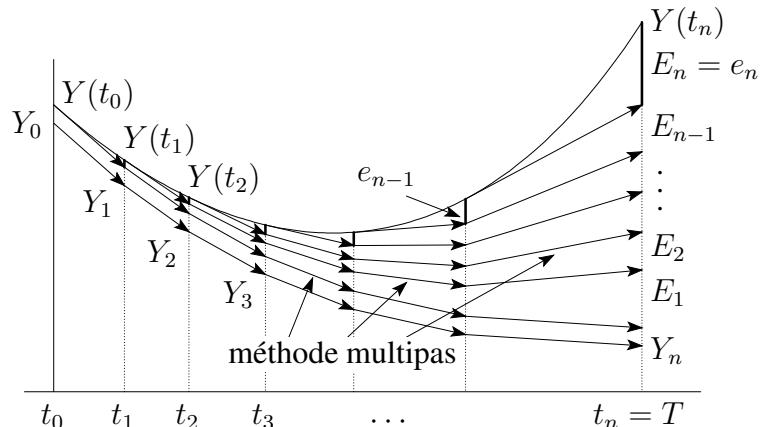


FIG. III.9: Estimation de l'erreur globale pour des méthodes multipas

En utilisant la norme infinie et une condition de Lipschitz pour  $\Psi$  (qui est une conséquence de celle de  $f(t, y)$ ), on obtient

$$\|Y_{n+1} - Z_{n+1}\| \leq (1 + h\Lambda)\|Y_n - Z_n\|. \quad (8.7)$$

Pour une méthode générale, on est obligé de choisir une autre norme pour arriver à (8.7). La stabilité de la méthode implique que ceci est possible (voir Hairer, Nørsett & Wanner (1993), paragraphe III.4).

*Accumulation des erreurs propagées.* Cette partie de la démonstration est exactement la même que pour les méthodes à un pas (voir le paragraphe III.3 et la figure III.9). Au lieu de (3.2) et (3.5), on utilise (8.6) et (8.7).  $\square$

### III.9 Equations différentielles raides (stiff)

The most pragmatical opinion is also historically the first one (Curtiss & Hirschfelder 1952): *stiff equations are equations where certain implicit methods, in particular BDF, perform better, usually tremendously better, than explicit ones.* (Hairer & Wanner 1991)

L'exemple 1.2 d'une équation différentielle, modélisant une réaction chimique, a montré qu'une méthode de Runge–Kutta explicite est obligée de prendre des pas d'intégration très petits pour obtenir une approximation acceptable. Une caractéristique de cette équation différentielle est que la solution cherchée est très lisse et les autres solutions s'approchent rapidement de celle-ci.

Pour mieux comprendre le phénomène de l'exemple 1.2, considérons le problème plus simple

$$\varepsilon y' = -y + \cos t, \quad 0 < \varepsilon \ll 1 \quad (9.1)$$

qui possède les mêmes caractéristiques (voir la figure III.10).

**Solution exacte.** L'équation différentielle (9.1) est linéaire inhomogène. Cherchons une solution particulière de la forme  $y(t) = A \cos t + B \sin t$ . En introduisant cette fonction dans (9.1)

$$-\varepsilon A \sin t + \varepsilon B \cos t = -A \cos t - B \sin t + \cos t,$$

une comparaison des coefficients donne  $A = 1/(1 + \varepsilon^2)$  et  $B = \varepsilon/(1 + \varepsilon^2)$ . Comme la solution générale de (9.1) est la somme de la solution générale de l'équation homogène et d'une solution particulière, nous obtenons

$$y(t) = e^{-t/\varepsilon} C + \cos t + \varepsilon \sin t + \mathcal{O}(\varepsilon^2). \quad (9.2)$$

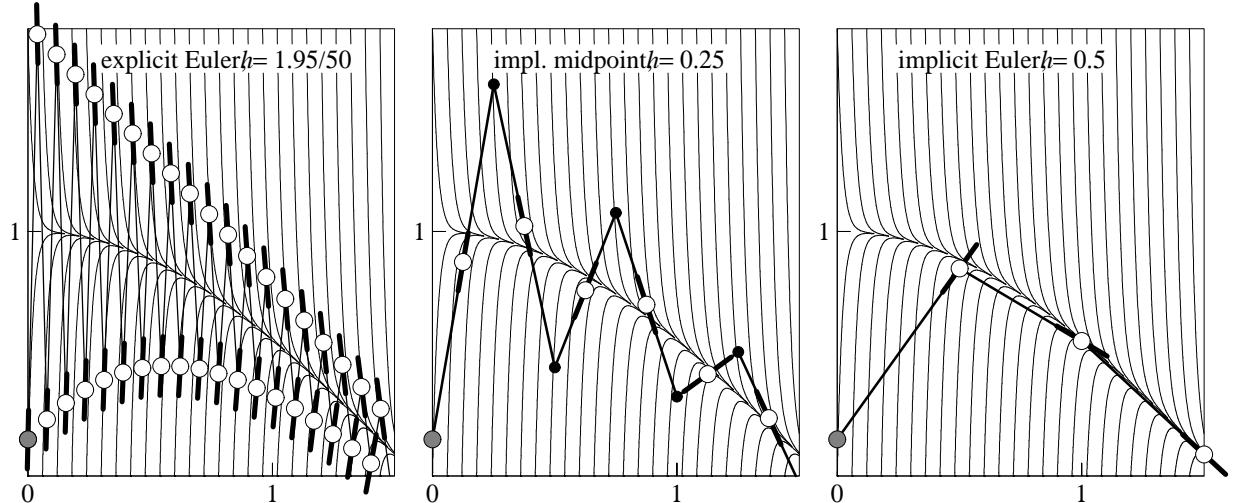


FIG. III.10: Solutions exactes et numériques pour le problème (9.1) de Curtiss & Hirschfelder,  $\varepsilon = 1/50$ .

**Solution numérique (Euler explicite).** La méthode d'Euler explicite  $y_{n+1} = y_n + hf(t_n, y_n)$ , appliquée au problème (9.1) avec des pas constants, donne avec  $t_n = nh$

$$y_{n+1} = \left(1 - \frac{h}{\varepsilon}\right)y_n + \frac{h}{\varepsilon} \cos t_n. \quad (9.3)$$

Ceci est une équation aux différences finies qui est linéaire et inhomogène. La solution est obtenue comme pour une équation différentielle. On cherche d'abord une solution particulière de la forme  $y_n = A \cos t_n + B \sin t_n$ . On l'introduit dans (9.3) et, en utilisant  $t_{n+1} = t_n + h$  et les théorèmes d'addition pour  $\sin(t_n + h)$  et  $\cos(t_n + h)$ , on obtient ainsi

$$\begin{aligned} A(\cos t_n \cdot \cos h - \sin t_n \cdot \sin h) + B(\sin t_n \cdot \cos h + \cos t_n \cdot \sin h) \\ = \left(1 - \frac{h}{\varepsilon}\right)(A \cos t_n + B \sin t_n) + \frac{h}{\varepsilon} \cos t_n. \end{aligned}$$

En comparant les coefficients de  $\cos t_n$  et  $\sin t_n$ , on obtient deux équations linéaires pour  $A$  et  $B$  dont la solution est  $A = 1 + \mathcal{O}(h\varepsilon)$  et  $B = \varepsilon + \mathcal{O}(h^2\varepsilon)$ . En ajoutant la solution générale de l'équation homogène à la solution particulière, on obtient (dessin à gauche de la figure III.10)

$$y_n = \left(1 - \frac{h}{\varepsilon}\right)^n C + \cos t_n + \varepsilon \sin t_n + \mathcal{O}(h\varepsilon). \quad (9.4)$$

On voit que la solution numérique  $y_n$  est proche de la solution exacte seulement si  $|1 - h/\varepsilon| < 1$ , c.-à-d. si  $h < 2\varepsilon$ . Si  $\varepsilon$  est très petit (par exemple  $\varepsilon = 10^{-6}$ ) une telle restriction est inacceptable.

**Solution numérique (Euler implicite).** Pour la méthode d'Euler implicite, le même calcul donne

$$\left(1 + \frac{h}{\varepsilon}\right)y_{n+1} = y_n + \frac{h}{\varepsilon} \cos t_{n+1}, \quad (9.5)$$

dont la solution peut être écrite sous la forme

$$y_n = \left(1 + \frac{h}{\varepsilon}\right)^{-n} C + \cos t_n + \varepsilon \sin t_n + \mathcal{O}(h\varepsilon). \quad (9.6)$$

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 figure III.10 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 pose 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

$$y' = \lambda y \quad (9.7)$$

comme *équation de test* (Dahlquist 1963). Sa solution exacte est  $y(t) = e^{\lambda t} C$  et elle reste bornée pour  $t \geq 0$  si  $\Re \lambda \leq 0$ . La solution numérique d'une méthode de Runge–Kutta ou d'une méthode multipas, appliquée avec des pas constants au problème (9.7), ne dépend que du produit  $h\lambda$ . Il est alors intéressant d'étudier pour quelle valeur de  $h\lambda$  la solution numérique reste bornée.

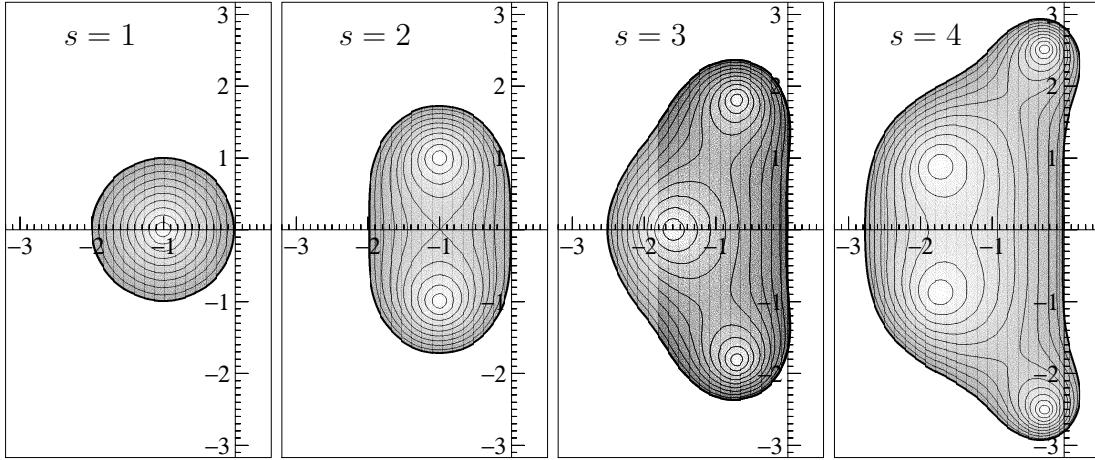
**Définition 9.1 (A-stabilité)** Considérons une méthode dont la solution numérique  $\{y_n\}_{n \geq 0}$  pour l'équation de test (9.7) est une fonction de  $z = h\lambda$ . Alors, l'ensemble

$$S := \{z \in \mathbb{C} ; \{y_n\}_{n \geq 0} \text{ est bornée}\} \quad (9.8)$$

s'appelle domaine de stabilité de la méthode. On dit que la méthode est A-stable si

$$S \supset \mathbb{C}^- \quad \text{où} \quad \mathbb{C}^- = \{z \in \mathbb{C} ; \Re z \leq 0\}.$$

Pour la méthode d'Euler explicite le domaine de stabilité est  $S = \{z ; |1 + z| \leq 1\}$ , le disque de rayon 1 et de centre  $-1$ . Pour la méthode d'Euler implicite il est  $S = \{z ; |1 - z| \geq 1\}$ , l'extérieur du disque de rayon 1 et de centre  $+1$ . Seulement la méthode implicite est A-stable.

FIG. III.11: Domaines de stabilité pour les méthodes de Runge–Kutta avec  $s$  étages et d'ordre  $p = s$ 

**Domaine de stabilité des méthodes à un pas.** Pour une méthode de Runge–Kutta, la solution numérique est de la forme  $y_{n+1} = R(h\lambda)y_n$ , où la fonction  $R(z)$  s'appelle *fonction de stabilité* (voir l'exercice 2). Le domaine de stabilité est alors

$$S = \{z ; |R(z)| \leq 1\}.$$

Si la méthode explicite à  $s$  étages (définition 2.1) a l'ordre  $p = s$ ,  $R(z)$  est le polynôme de degré  $s$  obtenu par troncature de la série  $e^z = 1 + z + \frac{1}{2!}z^2 + \dots$ . Les domaines de stabilité pour ces méthodes d'ordre 1 jusqu'à 4 sont dessinés dans la figure III.11. Comme  $R(z)$  est un polynôme,  $S$  est borné et la condition de stabilité  $h\lambda \in S$  impose une restriction sévère à  $h$ . Ces méthodes ne sont donc pas recommandées pour la résolution des équations différentielles raides.

**Domaine de stabilité des méthodes multipas.** Si l'on applique une méthode multipas (6.1) au problème (9.7), on obtient l'équation aux différences finies

$$\sum_{j=0}^k (\alpha_j - h\lambda\beta_j) y_{n+j} = 0. \quad (9.9)$$

La solution numérique est une combinaison linéaire de  $\zeta_j(h\lambda)^n$ ,  $j = 1, \dots, k$ , où  $\zeta_j(z)$  est un zéro du polynôme caractéristique (en supposant que les zéros soient distincts)

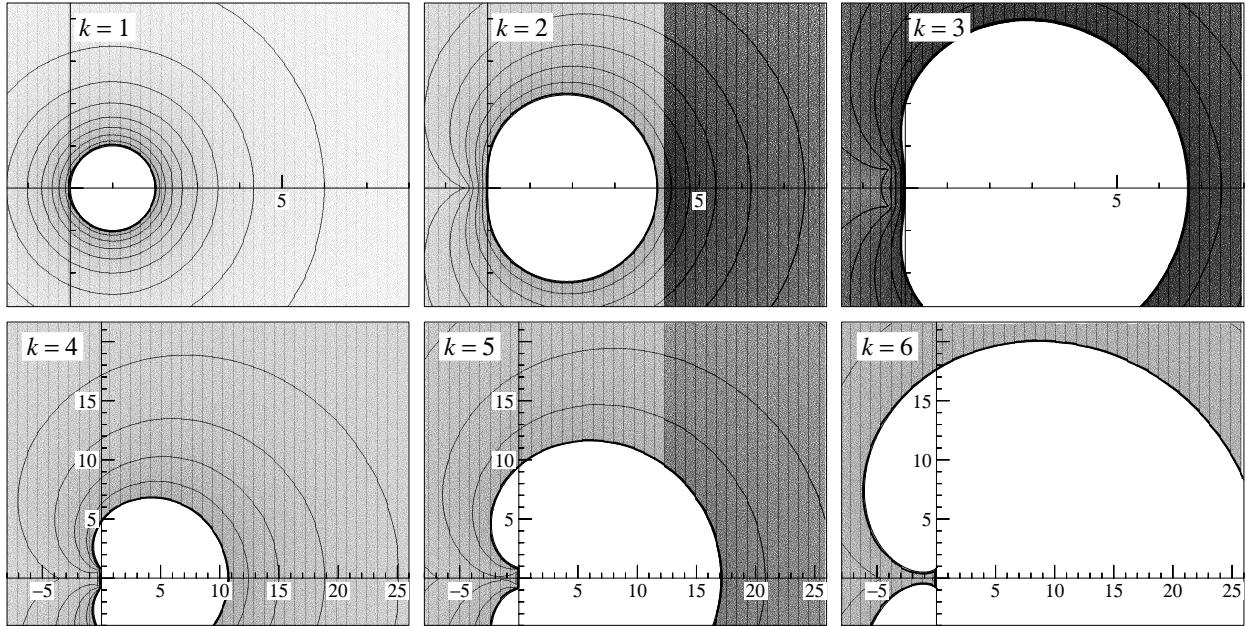
$$\rho(\zeta) - z\sigma(\zeta) = 0, \quad \rho(\zeta) = \sum_{j=0}^k \alpha_j \zeta^j, \quad \sigma(\zeta) = \sum_{j=0}^k \beta_j \zeta^j. \quad (9.10)$$

Le domaine de stabilité  $S$  d'une méthode multipas est alors l'ensemble des  $z \in \mathbb{C}$ , tel que toutes les racines de (9.10) sont majorées par 1. Pour étudier et dessiner  $S$ , il est plus simple de considérer le bord  $\partial S$ , car (par continuité de  $\zeta_j(z)$ )

$$z \in \partial S \quad \Rightarrow \quad \rho(e^{i\theta}) - z\sigma(e^{i\theta}) = 0 \quad \text{pour un } \theta \in [0, 2\pi].$$

Il suffit alors de dessiner la courbe  $\theta \mapsto \rho(e^{i\theta})/\sigma(e^{i\theta})$  (la “root locus curve”) qui sépare l'ensemble  $S$  du reste. Pour savoir, quelle composante connexe appartient à  $S$ , il suffit de le vérifier pour un seul point.

Les méthodes d'Adams explicites et implicites (à l'exception de la méthode d'Euler implicite et de la règle du trapèze) ont toutes un domaine de stabilité borné et petit. Ces méthodes ne sont donc pas utilisables pour des problèmes raides. Pour les méthodes BDF, par contre, le domaine de

FIG. III.12: Domaines de stabilité pour les méthodes BDF à  $k$  pas.

stabilité contient une grande partie du demi-plan gauche (voir la figure III.12). La méthode BDF avec  $k = 2$  (ordre  $p = 2$ ) est même A-stable. Ceci est une conséquence du fait que la “root locus curve”

$$z = \rho(e^{i\theta})/\sigma(e^{i\theta}) = \frac{3}{2} - 2e^{-i\theta} + \frac{1}{2}e^{-2i\theta}$$

satisfait

$$\Re z = \frac{3}{2} - 2 \cos \theta + \frac{1}{2} \cos 2\theta = (1 - \cos \theta)^2 \geq 0.$$

Les méthodes BDF sont beaucoup utilisées pour résoudre des équations différentielles raides même si elles sont A-stables seulement pour  $k \leq 2$  (voir la figure III.12). La célèbre *barrière de Dahlquist* (1963) dit que l’ordre d’une méthode multipas A-stable ne peut pas être plus grand que 2. Ce résultat négatif a motivé la recherche d’autres méthodes d’intégration qui permettent de combiner A-stabilité avec un ordre élevé.

### Méthodes de Runge–Kutta implicites (Radau IIA)

Comme pour la dérivation des méthodes de Runge–Kutta explicites, nous partons de la formule intégrée (2.3) de l’équation différentielle. Nous appliquons une formule de quadrature avec  $c_s = 1$  ayant l’ordre maximal  $2s - 1$  (formules de Radau, à comparer avec les exercices I.10 et I.16). Par exemple, pour  $s = 2$  on a

$$y(t_0 + h) = y(t_0) + \frac{h}{4} \left( 3f\left(t_0 + \frac{h}{3}, y(t_0 + \frac{h}{3})\right) + f\left(t_0 + h, y(t_0 + h)\right) \right) + \mathcal{O}(h^4). \quad (9.11)$$

Pour approximer la valeur  $y(t_0 + h/3)$ , nous intégrons l’équation différentielle de  $t_0$  à  $t_0 + h/3$  et nous appliquons une formule de quadrature qui utilise les mêmes évaluations de  $f$  que dans (9.11):

$$y(t_0 + \frac{h}{3}) = y(t_0) + \frac{h}{12} \left( 5f\left(t_0 + \frac{h}{3}, y(t_0 + \frac{h}{3})\right) - f\left(t_0 + h, y(t_0 + h)\right) \right) + \mathcal{O}(h^3). \quad (9.12)$$

En supprimant les termes du reste et en notant  $k_1$  et  $k_2$  les deux évaluations de  $f$ , nous arrivons à

$$\begin{aligned} k_1 &= f\left(t_0 + \frac{h}{3}, y_0 + \frac{h}{12}(5k_1 - k_2)\right) \\ k_2 &= f\left(t_0 + h, y_0 + \frac{h}{4}(3k_1 + k_2)\right) \\ y_1 &= y_0 + \frac{h}{4}(3k_1 + k_2) \end{aligned} \quad (9.13)$$

TAB. III.4: Coefficients  $c_i$ ,  $a_{ij}$  et  $b_i$  pour les méthodes Radau IIA avec  $s = 2$  et  $s = 3$ , ordre  $p = 2s - 1$ 

			$\frac{4 - \sqrt{6}}{10}$	$\frac{88 - 7\sqrt{6}}{360}$	$\frac{296 - 169\sqrt{6}}{1800}$	$\frac{-2 + 3\sqrt{6}}{225}$
$\frac{1}{3}$	$\frac{5}{12}$	$-\frac{1}{12}$	$\frac{4 + \sqrt{6}}{10}$	$\frac{296 + 169\sqrt{6}}{1800}$	$\frac{88 + 7\sqrt{6}}{360}$	$\frac{-2 - 3\sqrt{6}}{225}$
1	$\frac{3}{4}$	$\frac{1}{4}$	1	$\frac{16 - \sqrt{6}}{36}$	$\frac{16 + \sqrt{6}}{36}$	$\frac{1}{9}$
	$\frac{3}{4}$	$\frac{1}{4}$		$\frac{16 - \sqrt{6}}{36}$	$\frac{16 + \sqrt{6}}{36}$	$\frac{1}{9}$

qui est une méthode de Runge–Kutta comme dans la définition 2.1, mais où la matrice  $(a_{ij})$  n'est plus triangulaire inférieure (voir aussi le tableau III.4). Les deux premières équations de (9.13) constituent un système non linéaire pour  $k_1$  et  $k_2$ , qu'il faut résoudre avec les techniques du chapitre VI (méthode de Newton simplifiée).

**Lemme 9.2** *La méthode (9.13) a l'ordre 3 et elle est A-stable.*

*Démonstration.* L'ordre 3 est une conséquence des formules (9.11) et (9.12). Il suffit que la formule (9.12) soit d'ordre 2 car le terme correspondant dans (9.11) est multiplié par  $h$ .

*A-stabilité.* Si l'on applique la méthode à  $y' = \lambda y$ , on obtient avec  $z = h\lambda$

$$hk_1 = zy_0 + \frac{z}{12}(5hk_1 - hk_1), \quad hk_2 = zy_0 + \frac{z}{4}(3hk_1 + hk_2).$$

On résoud ce système linéaire pour  $hk_1$  et  $hk_2$ , et on introduit la solution dans la troisième formule de (9.13). Ceci donne  $y_1 = R(z)y_0$  avec comme fonction de stabilité

$$R(z) = \frac{P(z)}{Q(z)} = \frac{1 + z/3}{1 - 2z/3 + z^2/6}. \quad (9.14)$$

Sur l'axe imaginaire, on a

$$|Q(iy)|^2 - |P(iy)|^2 = \left(1 - \frac{y^2}{6}\right)^2 + \frac{4}{9}y^2 - \left(1 + \frac{y^2}{9}\right) = \frac{1}{36}y^4 \geq 0.$$

Ceci implique  $|Q(iy)| \geq |P(iy)|$  et aussi  $|R(iy)| \leq 1$ . Les singularités de  $R(z)$  (les zéros de  $Q(z)$ ) sont  $z_{12} = 2 \pm i\sqrt{2}$  dans le demi-plan droit. Donc  $R(z)$  est analytique dans le demi-plan gauche et, par le principe du maximum,  $R(z)$  est majoré par 1 pour  $\Re z \leq 0$ .  $\square$

Cette construction peut être généralisée pour obtenir des méthodes de Runge–Kutta implicites qui sont A-stables et d'ordre  $2s - 1$ . Elles s'appellent méthodes Radau IIA et sont, comme les méthodes BDF, souvent employées pour résoudre des équations différentielles raides. La méthode RADAU5, utilisée pour le calcul de la figure III.2, est celle avec  $s = 3$  et ordre  $p = 5$ . Les coefficients de la méthode (9.13) et de celle avec  $s = 3$  sont données dans le tableau III.4.

**Remarque.** Après la publication remarquable de Dahlquist en 1963, la recherche sur la résolution des équations différentielles raides a rapidement pris une place importante en analyse numérique. Des nouvelles méthodes d'intégration, des nouvelles théories (par exemple, étoiles d'ordre) et des programmes informatiques performants ont été développés. Plus de détails peuvent être trouvés dans le livre “*Solving Ordinary Differential Equations II. Stiff and Differential-Algebraic Problems*” par Hairer & Wanner (1996).

Beaucoup plus récente est l'étude sur les propriétés géométriques des intégrateurs numériques. Ca sera le contenu du paragraphe suivant. Le livre “*Geometric Numerical Integration*” par Hairer, Lubich & Wanner (2002) est consacré à ce sujet.

### III.10 Intégration géométrique

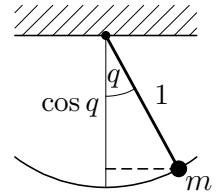
On cherche des méthodes d'intégration pour des problèmes similaires à l'exemple 1.3 qui donnent des bonnes approximations pour un calcul à long terme. Avec  $q_i$  à la place de  $y_i$  et avec  $p_i = m_i q'_i$  l'équation différentielle (1.3) peut être écrite sous la forme (système Hamiltonien)

$$p' = -\nabla U(q), \quad q' = \nabla T(p), \quad (10.1)$$

où  $T(p) = \frac{1}{2} \sum_i m_i^{-1} p_i^T p_i$  est l'énergie cinétique et  $U(q) = -G \sum_{i>j} m_i m_j / \|q_i - q_j\|$  est l'énergie potentielle du système. Pour simplifier la présentation, nous supposons par la suite que  $p$  et  $q$  sont des fonctions scalaires (par conséquent,  $\nabla U(q) = U'(q)$  et  $\nabla T(p) = T'(p)$ ).

Un exemple typique et intéressant est l'équation du pendule mathématique où l'énergie cinétique est  $T(p) = p^2/2$ , l'énergie potentielle  $U(q) = -\cos q$  et l'énergie totale

$$H(p, q) = \frac{1}{2} p^2 - \cos q.$$



Le système (10.1) possède deux propriétés très intéressantes:

**Conservation de l'énergie.** Un calcul direct montre que l'énergie totale  $H(p, q) = T(p) + U(q)$  reste constante le long des solutions de (10.1); figure III.13 (gauche). Ceci découle du fait que

$$\frac{d}{dt} H(p(t), q(t)) = T'(p(t))p'(t) + U'(q(t))q'(t) = 0.$$

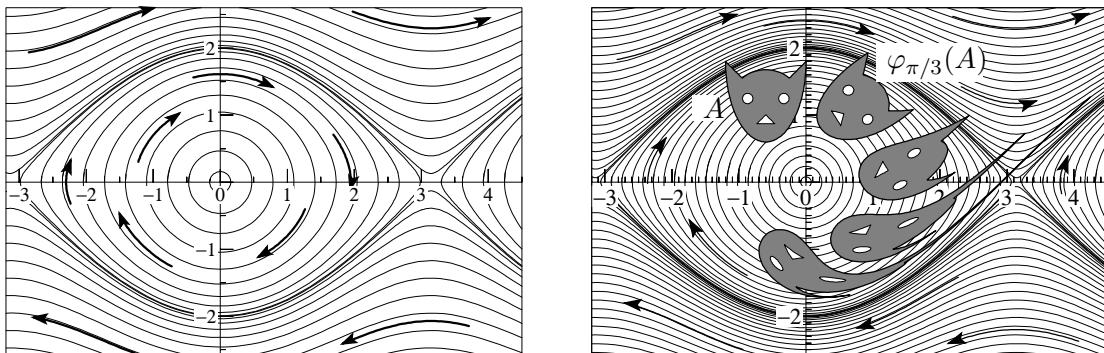


FIG. III.13: Conservation de l'énergie (gauche) et symplecticité (droite) du flot pour l'équation du pendule.

**Symplecticité (préservation de l'aire).** Nous utilisons la notation  $\varphi_t(p_0, q_0) := (p(t), q(t))^T$  pour le flot de l'équation différentielle (10.1) et nous allons démontrer que, pour un sous-ensemble  $A$  de l'espace  $(p, q)$ , l'aire de  $A$  reste invariante par rapport à l'application  $\varphi_t$ , c.-à-d.

$$\text{aire}(\varphi_t(A)) = \text{aire}(A) \quad \text{pour tout } t \quad (10.2)$$

(voir la figure III.13, droite). Comme  $\text{aire}(A) = \iint_A d(p_0, q_0)$  et<sup>2</sup>

$$\text{aire}(\varphi_t(A)) = \iint_{\varphi_t(A)} d(p, q) = \iint_A \left| \det \left( \frac{\partial \varphi_t(p_0, q_0)}{\partial (p_0, q_0)} \right) \right| d(p_0, q_0),$$

<sup>2</sup>La formule de changement de variables pour les intégrales doubles peut être trouvée dans le livre “Analyse au fil de l'histoire” de Hairer & Wanner à la page 338.

il suffit de démontrer que le déterminant de la matrice Jacobienne de  $\varphi_t$  est égal à 1 en valeur absolue. En dérivant l'équation différentielle (10.1) par rapport à la valeur initiale  $p_0$ , on obtient

$$\frac{\partial p'(t)}{\partial p_0} = -U''(q(t)) \frac{\partial q(t)}{\partial p_0}, \quad \frac{\partial q'(t)}{\partial p_0} = T''(p(t)) \frac{\partial p(t)}{\partial p_0}.$$

Une différentiation par rapport à  $q_0$  donne des formules analogues. En changeant l'ordre de la différentiation, on en déduit que

$$\frac{d}{dt} \det\left(\frac{\partial \varphi_t(p_0, q_0)}{\partial(p_0, q_0)}\right) = \frac{d}{dt} \left( \frac{\partial p(t)}{\partial p_0} \frac{\partial q(t)}{\partial q_0} - \frac{\partial p(t)}{\partial q_0} \frac{\partial q(t)}{\partial p_0} \right) = \dots = 0.$$

Il suit de  $\varphi_0(p_0, q_0) = (p_0, q_0)^T$  que le déterminant de la Jacobienne de  $\varphi_0$  vaut 1. Par conséquent, il vaut 1 pour tout  $t$ , ce qui démontre (10.2).

Dans une simulation numérique d'un système (10.1) on aimerait que les propriétés géométriques du flot exact soient préservées aussi bien que possible. Regardons ce qui se passe avec des méthodes classiques. Les figures III.14 et III.15 montrent le résultat pour la méthode d'Euler explicite et pour la méthode d'Euler implicite. Pour la première, la solution numérique tourne vers l'extérieur, l'énergie augmente et l'aire d'un ensemble croît. Pour la méthode implicite, c'est exactement l'inverse. Aucune de ces deux méthodes donne une approximation de la solution qui est qualitativement acceptable.

**Méthode d'Euler symplectique.** Nous traitons une équation de (10.1) par la méthode d'Euler explicite et l'autre par la méthode implicite. Ceci donne

$$\begin{aligned} p_{n+1} &= p_n - h \nabla U(q_n) & p_{n+1} &= p_n - h \nabla U(q_{n+1}) \\ q_{n+1} &= q_n + h \nabla T(p_{n+1}) & \text{ou} & q_{n+1} = q_n + h \nabla T(p_n). \end{aligned} \tag{10.3}$$

La variante (A) est la méthode de gauche, la variante (B) celle de droite.

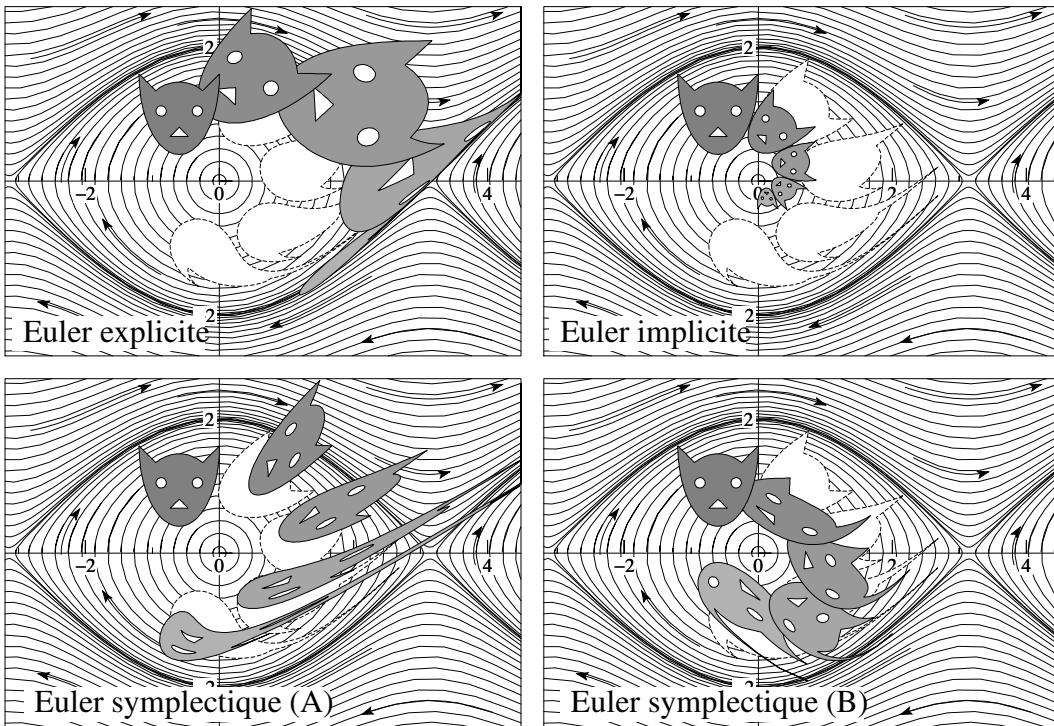


FIG. III.14: Illustration du flot numérique pour l'équation du pendule,  $h = \pi/4$ .

Les deux méthodes sont parfaitement explicites. Pour la première on calcule d'abord  $p_{n+1}$  et ensuite  $q_{n+1}$ ; pour la deuxième dans l'ordre inverse. La figure III.14 montre une nette amélioration pour ce qui concerne la conservation de l'aire.

**Lemme 10.1** Si  $\Phi_h : (p_n, q_n) \mapsto (p_{n+1}, q_{n+1})$  représente une des méthodes de (10.3), alors  $\Phi_h$  préserve l'aire, c.-à-d.  $\text{aire}(\Phi_h(A)) = \text{aire}(A)$ . On dit que cette méthode est symplectique.

*Démonstration.* Nous partageons le membre droit du système (10.1) comme

$$\begin{aligned} p' &= 0 & p' &= -\nabla U(q) \\ q' &= \nabla T(p) & q' &= 0. \end{aligned} \quad (10.4)$$

Les deux systèmes peuvent être résolus de manière exacte. Leurs solutions sont

$$\begin{aligned} p(t) &= p_0 & p(t) &= p_0 - t\nabla U(q_0) \\ q(t) &= q_0 + t\nabla T(p_0) & q(t) &= q_0. \end{aligned}$$

Les flots  $\varphi_t^{(T)}$  et  $\varphi_t^{(U)}$  des deux systèmes (10.4) préservent l'aire car ils sont les deux sous la forme (10.1). L'affirmation du lemme est alors une conséquence du fait que les deux méthodes (10.3) sont respectivement  $\varphi_h^{(U)} \circ \varphi_h^{(T)}$  et  $\varphi_h^{(T)} \circ \varphi_h^{(U)}$ .  $\square$

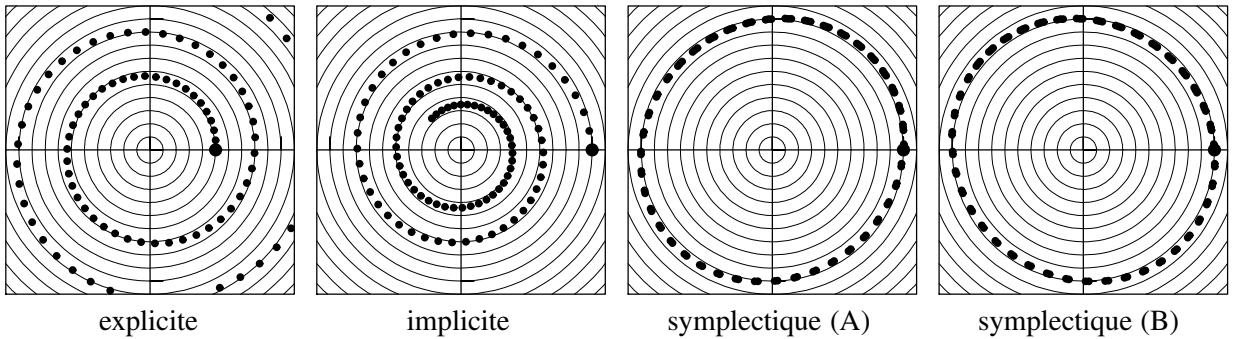


FIG. III.15: Solution numérique de différentes méthodes d'Euler appliquées à l'oscillateur harmonique (10.5) avec  $h = 0.15$ . La valeur initiale est le grand point noir.

On voit par des exemples très simples qu'on n'a pas de *conservation de l'énergie totale*, ni pour les méthodes classiques (Euler explicite ou implicite) ni pour la méthode d'Euler symplectique. L'expérience de la figure III.15 montre la solution numérique pour l'*oscillateur harmonique* qui est un système linéaire  $p' = -q$ ,  $q' = p$  possèdant

$$H(p, q) = \frac{1}{2}(p^2 + q^2) \quad (10.5)$$

comme énergie totale. Seulement les deux variantes de la méthode d'Euler symplectique donnent une solution numérique qui reste proche du cercle  $H(p, q) = \text{Const}$  (solution exacte).

**Lemme 10.2** Pour l'oscillateur harmonique (10.5), la solution numérique  $(p_n, q_n)$  de la méthode d'Euler symplectique reste sur une courbe fermée (l'ellipse  $p^2 + q^2 \pm hpq = \text{Const}$ ) pour tout  $n$ .

*Démonstration.* Considérons, par exemple, la première méthode dans (10.3) et appliquons-la à l'oscillateur harmonique (10.5). On obtient ainsi

$$\begin{pmatrix} 1 & 0 \\ -h & 1 \end{pmatrix} \begin{pmatrix} p_{n+1} \\ q_{n+1} \end{pmatrix} = \begin{pmatrix} 1 & -h \\ 0 & 1 \end{pmatrix} \begin{pmatrix} p_n \\ q_n \end{pmatrix}. \quad (10.6)$$

Une multiplication par  $(p_{n+1}, q_{n+1})$  donne la première égalité de

$$p_{n+1}^2 + q_{n+1}^2 - hp_{n+1}q_{n+1} = p_{n+1}p_n + q_{n+1}q_n - hp_{n+1}q_n = p_n^2 + q_n^2 - hp_nq_n.$$

et une multiplication par  $(p_n, q_n)$  la deuxième. On en déduit que  $p_n^2 + q_n^2 - hp_nq_n = Const$ , ce qui signifie que la solution numérique  $(p_n, q_n)$  reste sur une ellipse pour tout  $n$ .  $\square$

La méthode d'Euler symplectique est malheureusement seulement une méthode d'ordre  $p = 1$ . Peut-on trouver des méthodes avec un ordre plus élevé qui possèdent le même comportement concernant la conservation de l'aire et de l'énergie totale? Voici une méthode d'ordre deux qui est la plus importante dans le contexte de l'intégration géométrique. Elle est beaucoup utilisée dans des simulations en dynamique moléculaire et elle est la base pour plusieurs généralisations.

**La méthode “Störmer–Verlet”.** Pour introduire plus de symétrie dans la discréétisation, nous considérons la composition d'un demi-pas d'une des variantes de la méthode d'Euler symplectique avec un demi-pas de l'autre. Ceci conduit à

$$\begin{aligned} p_{n+1/2} &= p_n - \frac{h}{2} \nabla U(q_n) & q_{n+1/2} &= q_n + \frac{h}{2} \nabla T(p_n) \\ q_{n+1} &= q_n + h \nabla T(p_{n+1/2}) & \text{ou} & \\ p_{n+1} &= p_{n+1/2} - \frac{h}{2} \nabla U(q_{n+1}) & p_{n+1} &= p_n - h \nabla U(q_{n+1/2}) \\ & & q_{n+1} &= q_{n+1/2} + \frac{h}{2} \nabla T(p_{n+1}). \end{aligned} \quad (10.7)$$

De nouveau on appelle variante (A) la méthode à gauche et variante (B) celle de droite.

**Lemme 10.3** *Les deux variantes (10.7) de la méthode “Störmer–Verlet” sont symplectiques, c.-à-d., considérées comme application  $(p_n, q_n) \mapsto (p_{n+1}, q_{n+1})$  elles préservent l'aire.*

*Pour l'oscillateur harmonique (10.5), leur solution numérique  $(p_n, q_n)$  reste sur une courbe fermée (l'ellipse  $p^2 + (1 - h^2/4)q^2 = Const$ ) pour tout  $n$ .*

*Démonstration.* Avec la notation de la démonstration du lemme 10.1, nous observons que la variante (A) correspond à l'application  $\varphi_{h/2}^{(U)} \circ \varphi_h^{(T)} \circ \varphi_{h/2}^{(U)}$  et la variante (B) à  $\varphi_{h/2}^{(T)} \circ \varphi_h^{(U)} \circ \varphi_{h/2}^{(T)}$ . Comme les flots des deux systèmes (10.4) préservent l'aire, c'est vrai aussi pour la composition.

Pour le problème  $p' = -q$ ,  $q' = p$ , la variante (A) de (10.7) devient

$$\begin{pmatrix} p_{n+1} \\ q_{n+1} \end{pmatrix} = \begin{pmatrix} 1 - h^2/2 & -h + h^3/4 \\ h & 1 - h^2/2 \end{pmatrix} \begin{pmatrix} p_n \\ q_n \end{pmatrix}. \quad (10.8)$$

Un calcul direct montre que  $p_{n+1}^2 + (1 - h^2/4)q_{n+1}^2 = p_n^2 + (1 - h^2/4)q_n^2$ .  $\square$

**Méthodes de “splitting”.** Si l'on a besoin d'une très grande précision, l'ordre deux de la méthode “Störmer–Verlet” ne suffit pas. Une idée élégante pour obtenir des méthodes avec un ordre élevé est de considérer  $(p_{n+1}, q_{n+1}) = \Phi_h(p_n, q_n)$ , où  $\Phi_h$  est une composition de flots (comme dans la démonstration du lemme 10.1)

$$\Phi_h = \varphi_{b_m}^{(T)} \circ \varphi_{a_m}^{(U)} \circ \varphi_{b_{m-1}}^{(T)} \circ \dots \circ \varphi_{a_2}^{(U)} \circ \varphi_{b_1}^{(T)} \circ \varphi_{a_1}^{(U)}. \quad (10.9)$$

Pour  $m = 1$  et  $a_1 = b_1 = 1$  on obtient la variante (B) de la méthode d'Euler symplectique; pour  $m = 2$ ,  $a_1 = 1/2$ ,  $b_1 = 1$ ,  $a_2 = 1/2$ ,  $b_2 = 0$  on obtient une des méthodes Störmer–Verlet. De cette manière on peut construire des méthodes avec un ordre arbitrairement grand et pour lesquelles les affirmations du lemme 10.3 restent vraies.

### III.11 Exercices

1. Trouver une méthode numérique pour le problème  $y' = f(t, y)$ ,  $y(t_0) = y_0$  dans l'esprit de celle de Runge (voir le paragraphe III.2), mais basée sur la règle du trapèze.
2. Appliquer la méthode d'Euler, de Runge et de Heun au problème

$$y' = Ay, \quad y(0) = y_0. \quad (11.1)$$

Montrer que la solution numérique est donnée par

$$y_n = R(hA)^n y_0,$$

et calculer  $R(hA)$  pour les trois méthodes.

3. Ecrire l'équation différentielle

$$z'' + z = 0, \quad z(0) = 1, \quad z'(0) = 1,$$

sous la forme (11.1). Calculer la solution exacte et la solution numérique avec la méthode de Runge sur  $[0, 1]$  avec  $h = 1/2$ .

4. Montrer que l'ordre d'une méthode de Runge-Kutta explicite ne peut pas être plus grand que le nombre d'étages, c.-à-d.  $p \leq s$ .

*Indication.* Appliquer la méthode à  $y' = y$ ,  $y(0) = 1$  et observer que  $y_1$  est un polynôme en  $h$  de degré  $s$ .

5. Donner la famille à un paramètre des méthodes de RK explicites, d'ordre  $p = 2$  à  $s = 2$  étages (avec comme paramètre libre  $c_2$ ). Etudier le comportement de la solution numérique pour cette famille quand  $c_2 \rightarrow 0$ .

6. Pour le problème de Van der Pol

$$\begin{aligned} y'_1 &= y_2, & y_1(0) &= 2, \\ y'_2 &= (1 - y_1^2)y_2 - y_1, & y_2(0) &= 1/2, \end{aligned}$$

calculer le terme dominant de l'erreur locale (*i.e.* le coefficient du terme  $h^{p+1}$ ) pour la méthode de Runge d'ordre 2.

7. Calculer l'erreur locale d'une méthode de Runge-Kutta pour l'équation différentielle

$$\begin{aligned} y'_1 &= x^{r-1} & y_1(0) &= 0 \\ y'_2 &= x^{q-1}y_1 & y_2(0) &= 0 \end{aligned}$$

avec  $r$  et  $q$  des entiers positifs. En déduire la condition

$$\sum_{i=1}^s b_i c_i^{q-1} \sum_{j=1}^{i-1} a_{ij} c_j^{r-1} = \frac{1}{r(q+r)}, \quad r + q \leq p$$

pour une méthode d'ordre  $p$ .

8. (Runge 1905). Considérons une méthode de Runge-Kutta à  $s$  étages avec l'ordre  $p = s \leq 4$  et supposons que tous les coefficients  $a_{ij}$  et  $b_j$  soient non-négatifs. Montrer que la constante de Lipschitz  $\Lambda$  de  $\Phi(x, y, h)$  (voir le lemme du paragraphe III.3) satisfait

$$(1 + h\Lambda) \leq e^{hL}$$

où  $L$  est la constante de Lipschitz pour la fonction  $f(x, y)$ . En déduire que l'estimation (3.4) reste vraie si l'on remplace  $\Lambda$  par  $L$ .

9. Soit  $err_i$  une estimation de l'erreur au  $i$ -ème pas et définissons  $\varphi_i$  par

$$err_i = \varphi_i \cdot h_i^r.$$

Si on a fait le calcul jusqu'au  $n$ -ème pas, c.-à-d., si on connaît les valeurs de  $h_i$  et  $err_i$  pour  $i \leq n$ , il faut trouver une valeur raisonnable pour  $h_{n+1}$ .

- (a) L'hypothèse  $\varphi_{n+1} = \varphi_n$  nous conduit à la formule courante du cours.
- (b) (Gustafsson 1992). Montrer que l'hypothèse  $\Delta \ln \varphi_n = \Delta \ln \varphi_{n-1}$  (c.-à-d.,  $\varphi_{n+1}/\varphi_n = \varphi_n/\varphi_{n-1}$ ) nous conduit à la formule

$$h_{n+1} = 0.9 \cdot \frac{h_n^2}{h_{n-1}} \left( \frac{Tol}{err_n} \cdot \frac{err_{n-1}}{err_n} \right)^{1/r}.$$

10. (“Dense output”). Soit  $\{y_n\}$  la solution numérique obtenue par une méthode de Runge-Kutta d'ordre 4 (avec des pas constants). Pour un  $x \in (x_n, x_{n+1})$ , nous considérons le polynôme  $u(x)$  de degré 3 qui satisfait

$$u(x_n) = y_n, \quad u'(x_n) = f(x_n, y_n), \quad u(x_{n+1}) = y_{n+1}, \quad u'(x_{n+1}) = f(x_{n+1}, y_{n+1})$$

(interpolation d'Hermite, voir II.7). Montrer que, pour tout  $x \in (x_n, x_{n+1})$ , on a

$$u(x) - y(x) = \mathcal{O}(h^4).$$

11. Montrer que la formule de *Milne*

$$y_{n+1} = y_{n-1} + \frac{h}{3} (f_{n+1} + 4f_n + f_{n-1})$$

est une méthode multipas d'ordre 4 qui est stable. Expliquer pourquoi ses coefficients sont les mêmes que pour la formule de quadrature de Simpson.

12. Calculer la solution générale de

$$y_{n+3} - 5y_{n+2} + 8y_{n+1} - 4y_n = 0,$$

puis donner une formule pour la solution particulière qui satisfait  $y_0 = -1$ ,  $y_1 = 0$ ,  $y_2 = 4$ .

13. (a) Appliquer la méthode d'Adams explicite

$$y_{n+1} = y_n + h \left( \frac{3}{2} f_n - \frac{1}{2} f_{n-1} \right)$$

avec  $h = 1/8$  au problème  $y' = y^2$ ,  $y(0) = 1$ ,  $y(1/2) = ?$

Pour  $y_1$  utiliser la valeur obtenue par la méthode d'Euler explicite.

- (b) Appliquer la méthode d'Euler explicite au même problème, également avec  $h = 1/8$ .

- (c) Comparer les deux résultats numériques avec la solution exacte  $y(1/2) = 2$ .

14. En utilisant le théorème binomial généralisé (Analyse I), montrer que pour les coefficients  $\gamma_j$  des méthodes d'Adams explicites, la fonction génératrice  $G(t) := \sum_{j=0}^{\infty} \gamma_j t^j$  devient  $G(t) = -t / ((1-t) \log(1-t))$ .

En déduire la formule

$$\gamma_m + \frac{1}{2} \gamma_{m-1} + \frac{1}{3} \gamma_{m-2} + \dots + \frac{1}{m+1} \gamma_0 = 1. \quad (1)$$

Calculer à l'aide de (1) les  $\gamma_j$  pour  $j = 1, 2, 3, 4$ .

*Indication.* Utiliser l'égalité  $\binom{s+j-1}{j} = (-1)^j \binom{-s}{j}$ .

15. Vérifier que la méthode BDF à  $k$  pas est d'ordre  $p = k$ .
16. Quel est le polynôme  $\varrho(\zeta)$  de la méthode d'Adams explicite à  $k$  pas?
17. Montrer que le polynôme  $\rho(\zeta)$  de la méthode BDF à  $k$  pas est donné par

$$\rho(\zeta) = \sum_{j=1}^k \frac{1}{j} \zeta^{k-j} (\zeta - 1)^j.$$

En utilisant la transformation  $\zeta = 1/(1-z)$ , montrer que la méthode est stable si toutes les racines de

$$p(z) = \sum_{j=1}^k \frac{1}{j} z^j \tag{11.2}$$

sont en dehors du disque  $|z - 1| \leq 1$ ; elle est instable si au moins une racine de (11.2) se trouve dans ce disque.

*Remarque.* Le polynôme (11.2) est une somme partielle de la série pour  $-\log(1-z)$ .

18. En calculant numériquement les racines du polynôme (11.2), montrer que la méthode BDF est stable pour  $1 \leq k \leq 6$ , mais instable pour  $k = 7$ .
19. Une méthode multipas est dite symétrique si

$$\alpha_{k-i} = -\alpha_i, \quad \beta_{k-i} = \beta_i, \quad \text{pour } i = 0, 1, \dots, k.$$

Démontrer que l'ordre (maximal) d'une méthode symétrique est toujours pair.

20. Soit  $\varrho(\zeta)$  un polynôme quelconque de degré  $k$  satisfaisant  $\varrho(1) = 0$  et  $\varrho(1)' \neq 0$ .
  - (a) Montrer qu'il existe une unique méthode multipas implicite d'ordre  $p = k+1$  dont le polynôme caractéristique est  $\varrho(\zeta)$ .
  - (b) Montrer qu'il existe une unique méthode multipas explicite d'ordre  $p = k$  dont le polynome caractéristique est  $\varrho(\zeta)$ .
21. Le polynome caractéristique

$$\varrho(\zeta) = \zeta^{k-2}(\zeta^2 - 1),$$

définit les méthodes de Nyström ou de Milne-Simpson. Calculer pour  $k = 2$  la méthode explicite et implicite à l'aide de l'exercice précédent.

# Chapitre IV

## Systèmes d'Equations Linéaires

Considérons un système d'équations linéaires ( $a_{ij}$ ,  $b_j$  donnés)

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n &= b_2 \\ \vdots &\quad \vdots \quad \vdots \quad \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \dots + a_{nn}x_n &= b_n \end{aligned} \tag{0.1}$$

et cherchons sa solution  $x_1, \dots, x_n$ . Très souvent, il est commode d'utiliser la notation matricielle

$$Ax = b. \tag{0.2}$$

Rappelons que le système (0.2) possède une solution unique si et seulement si  $\det A \neq 0$ .

### Bibliographie sur ce chapitre

- Å. Björck (1996): *Numerical Methods for Least Squares Problems*. SIAM. [MA 65/387]
- P.G. Ciarlet (1982): *Introduction à l'analyse numérique matricielle et à l'optimisation*, Masson.
- J.J. Dongarra, C.B. Moler, J.R. Bunch & G.W. Stewart (1979): *LINPACK Users' Guide*. SIAM.
- D.K. Faddeev & V.N. Faddeeva (1963): *Computational Methods of Linear Algebra*. Freeman & Co. [MA 65/271]
- G.H. Golub & C.F. Van Loan (1989): *Matrix Computations*. Second edition. John Hopkins Univ. Press. [MA 65/214]
- N.J. Higham (1996): *Accuracy and Stability of Numerical Algorithms*. SIAM. [MA 65/379]
- A.S. Householder (1964): *The Theory of Matrices in Numerical Analysis*. Blaisdell Publ. Comp. [MA 65/262]
- G.W. Stewart (1973): *Introduction to Matrix Computations*. Academic Press.
- L.N. Trefethen & D. Bau (1997): *Numerical Linear Algebra*. SIAM. [MA 65/388]
- J.H. Wilkinson (1969): *Rundungsfehler*. Springer-Verlag.
- J.H. Wilkinson & C. Reinsch (1971): *Handbook for Automatic Computation, Volume II, Linear Algebra*. Springer-Verlag.

## IV.1 Elimination de Gauss

L'élimination dite "de Gauss" a été pratiquée pendant des siècles sans grand tam-tam, notamment par Newton et par Lagrange (en 1781 dans ses calculs astronomiques). Toutefois, Gauss ayant le souci de *prouver* l'existence des solutions pour son *principium nostrum* des moindres carrés (voir notices historiques du cours d'*Algèbre Linéaire*, p. 17) décrit l'algorithme explicitement :

Soit donné le système (0.1) et supposons que  $\det A \neq 0$ . Si  $a_{11} \neq 0$ , on peut éliminer la variable  $x_1$  dans les équations 2 à  $n$  à l'aide de l'équation 1, c.-à-d., on calcule

$$\ell_{i1} = \frac{a_{i1}}{a_{11}} \quad \text{pour } i = 2, \dots, n \quad (1.1)$$

et on remplace la ligne  $i$  par

$$\text{ligne } i - \ell_{i1} * \text{ligne } 1.$$

De cette manière, on obtient le système équivalent

$$\begin{aligned} a_{11}^{(1)}x_1 + a_{12}^{(1)}x_2 + \dots + a_{1n}^{(1)}x_n &= b_1^{(1)} \\ a_{22}^{(1)}x_2 + \dots + a_{2n}^{(1)}x_n &= b_2^{(1)} \\ \vdots &\quad \vdots \quad \vdots \\ a_{n2}^{(1)}x_2 + \dots + a_{nn}^{(1)}x_n &= b_n^{(1)} \end{aligned} \quad (1.2)$$

où

$$\begin{aligned} a_{1j}^{(1)} &= a_{1j}, & a_{ij}^{(1)} &= a_{ij} - \ell_{i1}a_{1j} & \text{pour } i = 2, \dots, n \\ b_1^{(1)} &= b_1, & b_i^{(1)} &= b_i - \ell_{i1}b_1 \end{aligned} \quad (1.3)$$

(si  $a_{11} = 0$ , on échange la première ligne de (0.1) avec une autre ligne pour arriver à  $a_{11} \neq 0$ ; ceci est toujours possible car  $\det A \neq 0$ ).

Le système (1.2) contient un sous-système de dimension  $n - 1$  sur lequel on peut répéter la procédure pour éliminer  $x_2$  dans les équations 3 à  $n$ . On multiplie la ligne 2 de (1.2) par  $\ell_{i2} = a_{i2}^{(1)}/a_{22}^{(1)}$  et on la soustrait de la ligne  $i$ . Après  $n - 1$  étapes

$$(A, b) \rightarrow (A^{(1)}, b^{(1)}) \rightarrow (A^{(2)}, b^{(2)}) \rightarrow \dots \rightarrow (A^{(n-1)}, b^{(n-1)}) =: (R, c)$$

on obtient un système triangulaire

$$\begin{aligned} r_{11}x_1 + r_{12}x_2 + \dots + r_{1n}x_n &= c_1 \\ r_{22}x_2 + \dots + r_{2n}x_n &= c_2 \\ \ddots &\quad \vdots \quad \vdots \\ r_{nn}x_n &= c_n \end{aligned} \quad (1.4)$$

qui se résoud facilement par "back substitution"

$$x_n = c_n/r_{nn}, \quad x_i = (c_i - \sum_{j=i+1}^n r_{ij}x_j)/r_{ii} \quad \text{pour } i = n-1, \dots, 1. \quad (1.5)$$

**Théorème 1.1** Soit  $\det A \neq 0$ . L'élimination de Gauss donne

$$PA = LR \quad (1.6)$$

où  $P$  est une matrice de permutation et

$$L = \begin{pmatrix} 1 & & & & \\ \ell_{21} & 1 & & & \\ \vdots & \ddots & \ddots & & \\ \ell_{n1} & \dots & \ell_{n,n-1} & 1 \end{pmatrix}, \quad R = \begin{pmatrix} r_{11} & r_{12} & \dots & r_{1n} \\ r_{22} & \dots & r_{2n} \\ \ddots & & \vdots \\ r_{nn} \end{pmatrix}. \quad (1.7)$$

La formule (1.6) s'appelle décomposition LR (left - right) de la matrice  $A$ .

*Remarque.* Les colonnes et les lignes d'une matrice de permutation  $P$  sont des vecteurs unité. On a  $\det P = \pm 1$ . Un exemple est

$$P = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad P \begin{pmatrix} \text{ligne 1} \\ \text{ligne 2} \\ \text{ligne 3} \end{pmatrix} = \begin{pmatrix} \text{ligne 2} \\ \text{ligne 1} \\ \text{ligne 3} \end{pmatrix}.$$

*Démonstration.* Supposons que toutes les permutations nécessaires soient déjà faites avant que l'on commence l'élimination des variables (par abus de notation, nous écrivons  $A$  au lieu de  $PA$  dans cette démonstration). En utilisant les matrices

$$L_1 = \begin{pmatrix} 1 & & & & \\ -\ell_{21} & 1 & & & \\ -\ell_{31} & 0 & 1 & & \\ \vdots & \vdots & \ddots & \ddots & \\ -\ell_{n1} & 0 & \dots & 0 & 1 \end{pmatrix}, \quad L_2 = \begin{pmatrix} 1 & & & & \\ 0 & 1 & & & \\ 0 & -\ell_{32} & 1 & & \\ \vdots & \vdots & \ddots & \ddots & \\ 0 & -\ell_{n2} & \dots & 0 & 1 \end{pmatrix}, \quad \dots \quad (1.8)$$

le premier pas de l'élimination de Gauss correspond à une multiplication de  $A$  avec  $L_1$ , le deuxième avec  $L_2$ , etc.,

$$L_1 A = A^{(1)}, \quad L_2 A^{(1)} = A^{(2)}, \quad \dots, \quad L_{n-1} A^{(n-2)} = A^{(n-1)} = R.$$

Par conséquent,

$$R = (L_{n-1} L_{n-2} \cdots L_1) \cdot A \quad \text{et} \quad A = (L_{n-1} L_{n-2} \cdots L_1)^{-1} \cdot R.$$

Il reste à montrer que la matrice  $L$  de (1.7) est égale à  $(L_{n-1} L_{n-2} \cdots L_1)^{-1}$ . Pour ceci, nous appliquons la même procédure à la matrice  $L$ . La multiplication de  $L$  avec  $L_1$  élimine les éléments de la première colonne en-dessous de la diagonale, puis la multiplication avec  $L_2$  élimine ceux de la deuxième colonne, etc. Finalement, on obtient  $(L_{n-1} L_{n-2} \cdots L_1) \cdot L = I = \text{identité}$ , ce qu'il fallait démontrer.  $\square$

**Calcul du déterminant d'une matrice.** La formule (1.6) implique que  $\det P \cdot \det A = \det L \cdot \det R$ . Comme  $\det P = (-1)^\sigma$ , où  $\sigma$  est le nombre de permutations dans l'élimination de Gauss, on obtient

$$\det A = (-1)^\sigma \cdot r_{11} \cdot \dots \cdot r_{nn}. \quad (1.9)$$

**Résolution de systèmes linéaires.** En pratique, on rencontre souvent la situation où il faut résoudre une suite de systèmes linéaires  $Ax = b$ ,  $Ax' = b'$ ,  $Ax'' = b''$ , etc., possédant tous la même matrice. Très souvent, on connaît  $b'$  seulement après la résolution du premier système.

C'est la raison pour laquelle on écrit, en général, le programme pour l'élimination de Gauss en deux sous-programmes :

DEC – calculer la décomposition LR (voir (1.6)) de la matrice;

SOL – résoudre le système  $Ax = b$ . D'abord on calcule le vecteur  $c$  (voir (1.4)), défini par  $Lc = Pb$ , puis on résoud le système triangulaire  $Rx = c$ .

Pour le problème ci-dessus, on appelle *une fois* le sous-programme DEC et puis, pour chaque système linéaire, le sous-programme SOL.

**Coût de l'élimination de Gauss.** Pour le passage de  $A$  à  $A^{(1)}$ , on a besoin de

- $n - 1$  divisions (voir (1.1)) et de
- $(n - 1)^2$  multiplications et additions (voir (1.3)).

Le calcul de  $A^{(2)}$  nécessite  $n - 2$  divisions et  $(n - 2)^2$  multiplications et additions, etc. Comme le travail dû aux divisions est ici négligeable, le coût total de la décomposition LR s'élève à environ

$$(n - 1)^2 + (n - 2)^2 + \dots + 2^2 + 1^2 \approx \int_0^n x^2 dx = \frac{n^3}{3} \quad \text{opérations}$$

(opération = multiplication + addition).

Le calcul de  $b^{(1)}$  nécessite  $n - 1$  opérations (voir (1.3)). Par conséquent, on obtient  $c$  avec  $\approx (n - 1) + \dots + 2 + 1 \approx n^2/2$  opérations. Similairement, la résolution du système (1.4) se fait en  $n^2/2$  opérations.

En résumé, l'appel au sous-programme DEC nécessite  $\approx n^3/3$  opérations, tandis que SOL a seulement besoin de  $\approx n^2$  opérations (sur des ordinateurs sériels). A titre de comparaison, la formule habituelle pour le déterminant d'une matrice  $n \times n$  contient  $n! \approx n^n/e^n$  termes.

## IV.2 Le choix du pivot

Dans l'élimination de Gauss, il faut au début choisir une équation (avec  $a_{i1} \neq 0$ ; cet élément s'appelle ‘le pivot’) à l'aide de laquelle on élimine  $x_1$  dans les autres équations. Le choix de cette équation (choix du pivot) peut-il influencer la précision du résultat numérique, si l'on fait le calcul sur ordinateur en virgule flottante?

**Exemple 2.1 (Forsythe)** Considérons le système

$$\begin{aligned} 1.00 \cdot 10^{-4} \cdot x_1 + 1.00 \cdot x_2 &= 1.00 \\ 1.00 \cdot x_1 + 1.00 \cdot x_2 &= 2.00 \end{aligned} \tag{2.1}$$

avec pour solution exacte

$$x_1 = \frac{1}{0.9999} = 1.00010001\dots, \quad x_2 = \frac{0.9998}{0.9999} = 0.99989998\dots \tag{2.2}$$

Appliquons l'élimination de Gauss et simulons un calcul en virgule flottante avec 3 chiffres significatifs (en base 10).

- a) Si l'on prend  $a_{11} = 1.00 \cdot 10^{-4}$  comme pivot, on obtient  $\ell_{21} = a_{21}/a_{11} = 1.00 \cdot 10^4$ ,  $a_{22}^{(1)} = 1.00 - 1.00 \cdot 10^4 = -1.00 \cdot 10^4$  et  $b_2^{(1)} = 2.00 - 1.00 \cdot 10^4 = -1.00 \cdot 10^4$ . Par conséquent,  $x_2 = b_2^{(1)}/a_{22}^{(1)} = 1.00$  (exacte!), mais pour  $x_1$  nous obtenons

$$x_1 = (b_1 - a_{12}x_2)/a_{11} = (1.00 - 1.00 * 1.00)/(1.00 \cdot 10^{-4}) = 0.$$

Le résultat numérique, obtenu pour  $x_1$ , est faux.

- b) Si l'on échange les deux équations de (2.1), le pivot est 1.00 et l'élimination de Gauss donne:  $\ell_{21} = 1.00 \cdot 10^{-4}$ ,  $a_{22}^{(1)} = 1.00 - 1.00 \cdot 10^{-4} = 1.00$  et  $b_2^{(1)} = 1.00 - 2.00 * 1.00 \cdot 10^{-4} = 1.00$ . De nouveau, on obtient  $x_2 = b_2^{(1)}/a_{22}^{(1)} = 1.00$ . Mais cette fois le résultat pour  $x_1$  est

$$x_1 = (b_1 - a_{12}x_2)/a_{11} = (2.00 - 1.00 * 1.00)/1.00 = 1.00.$$

Les deux valeurs numériques (pour  $x_2$  et aussi pour  $x_1$ ) sont correctes.

Pour mieux comprendre dans quelle partie de l'élimination de Gauss on a perdu une information essentielle, considérons les sous-problèmes (addition, soustraction, multiplication, division) séparément et étudions leur "condition".

**Condition d'un problème.** Considérons une application  $\mathcal{P} : \mathbb{R}^n \rightarrow \mathbb{R}$ , le problème consistant à calculer  $\mathcal{P}(x)$  pour les données  $x = (x_1, \dots, x_n)$ . Il est intéressant d'étudier l'influence de perturbations dans  $x$  sur le résultat  $\mathcal{P}(x)$ .

**Définition 2.2** La condition  $\kappa$  d'un problème  $\mathcal{P}$  est le plus petit nombre tel que

$$\frac{|\hat{x}_i - x_i|}{|x_i|} \leq \text{eps} \quad \Rightarrow \quad \frac{|\mathcal{P}(\hat{x}) - \mathcal{P}(x)|}{|\mathcal{P}(x)|} \leq \kappa \cdot \text{eps}. \quad (2.3)$$

On dit que le problème  $\mathcal{P}$  est bien conditionné, si  $\kappa$  n'est pas trop grand. Sinon, il est mal conditionné.

Dans cette définition,  $\text{eps}$  représente un petit nombre. Si  $\text{eps}$  est la précision de l'ordinateur (voir le paragraphe II.5) alors,  $\hat{x}_i$  peut être interprété comme l'arrondi de  $x_i$ . Remarquons encore que la condition  $\kappa$  dépend des données  $x_i$  et du problème  $\mathcal{P}$ , mais qu'elle ne dépend pas de l'algorithme avec lequel on calcule  $\mathcal{P}(x)$ .

**Exemple 2.3 (multiplication de deux nombres réels)** Soient donnés les nombres  $x_1$  et  $x_2$ , considérons le problème de calculer  $\mathcal{P}(x_1, x_2) = x_1 \cdot x_2$ . Pour les deux valeurs perturbées

$$\hat{x}_1 = x_1(1 + \epsilon_1), \quad \hat{x}_2 = x_2(1 + \epsilon_2), \quad |\epsilon_i| \leq \text{eps} \quad (2.4)$$

on a

$$\frac{\hat{x}_1 \cdot \hat{x}_2 - x_1 \cdot x_2}{x_1 \cdot x_2} = (1 + \epsilon_1)(1 + \epsilon_2) - 1 = \epsilon_1 + \epsilon_2 + \epsilon_1 \cdot \epsilon_2.$$

Comme  $\text{eps}$  est un petit nombre, le produit  $\epsilon_1 \cdot \epsilon_2$  est négligeable par rapport à  $|\epsilon_1| + |\epsilon_2|$  et on obtient

$$\left| \frac{\hat{x}_1 \cdot \hat{x}_2 - x_1 \cdot x_2}{x_1 \cdot x_2} \right| \leq 2 \cdot \text{eps}. \quad (2.5)$$

On a donc  $\kappa = 2$  et ce problème est bien conditionné.

**Exemple 2.4 (soustraction)** Pour le problème  $\mathcal{P}(x_1, x_2) = x_1 - x_2$ , un calcul analogue donne

$$\left| \frac{(\hat{x}_1 - \hat{x}_2) - (x_1 - x_2)}{x_1 - x_2} \right| = \left| \frac{x_1 \epsilon_1 - x_2 \epsilon_2}{x_1 - x_2} \right| \leq \underbrace{\frac{|x_1| + |x_2|}{|x_1 - x_2|}}_{\kappa} \cdot \text{eps}. \quad (2.6)$$

Si  $\text{sign}x_1 = -\text{sign}x_2$  (ceci correspond à une addition et non pas à une soustraction) on a  $\kappa = 1$ ; le problème est bien conditionné.

Par contre, si  $x_1 \approx x_2$  la condition  $\kappa = (|x_1| + |x_2|)/|x_1 - x_2|$  devient très grande et on est confronté à un problème qui est extrêmement mal conditionné. Pour mieux illustrer l'effet de cette grande condition, considérons l'exemple numérique

$$x_1 = \frac{1}{51}, \quad x_2 = \frac{1}{52} \quad \text{pour lequel} \quad \kappa \approx \frac{2/50}{(1/50)^2} = 100.$$

En faisant le calcul avec 3 chiffres significatifs (en base 10), on obtient  $\hat{x}_1 = 0.196 \cdot 10^{-1}$ ,  $\hat{x}_2 = 0.192 \cdot 10^{-1}$  et  $\hat{x}_1 - \hat{x}_2 = 0.400 \cdot 10^{-3}$ . Comme les deux premiers chiffres sont les mêmes pour  $\hat{x}_1$  et  $\hat{x}_2$ , la soustraction les fait disparaître et on n'a plus qu'un chiffre qui est significatif (le résultat exact est  $1/(51 \cdot 52) = 0.377 \cdot 10^{-3}$ ). On parle d'*extinction de chiffres*.

**Explication du choix du pivot.** Si  $\ell_{21}$  est très grand (ce qui est le cas dans la situation (a) de l'exemple de Forsythe) alors,

$$\left. \begin{array}{l} a_{22}^{(1)} = a_{22} - \ell_{21}a_{12} \approx -\ell_{21}a_{12} \\ b_2^{(1)} = b_2 - \ell_{21}b_1 \approx -\ell_{21}b_1 \end{array} \right\} \quad x_2 = \frac{b_2^{(1)}}{a_{22}^{(1)}} \approx \frac{b_1}{a_{12}}. \quad (2.7)$$

Cette valeur, obtenue pour  $x_2$ , est en général 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  $a_{12}x_2 \approx b_1$  et, à cause de l'extinction de chiffres, on perd de la précision.

La conclusion de cette étude est qu'il faut éviter des  $\ell_{ij}$  trop grands.

**Recherche partielle de pivot.** L'idée est de ne pas se contenter d'un pivot qui soit différent de zéro ( $a_{11} \neq 0$ ), mais d'échanger les équations de (0.1) afin que  $a_{11}$  soit le plus grand élément (en valeur absolue) de la première colonne de  $A$ . De cette manière, on a toujours  $|\ell_{i1}| \leq 1$ . La même stratégie est répétée pour les sous-systèmes apparaissant dans l'élimination de Gauss.

**Expérience numérique.** Pour chaque  $n = 5, 6, \dots, 55$  nous choisissons 2000 matrices aléatoires avec coefficients  $a_{ij}$  uniformément distribués dans  $[-1, 1]$  et des solutions  $x_i$  uniformément distribuées dans  $[-1, 1]$ . Alors on calcule en *double précision* les  $b_j$  pour cette solution exacte. Ensuite

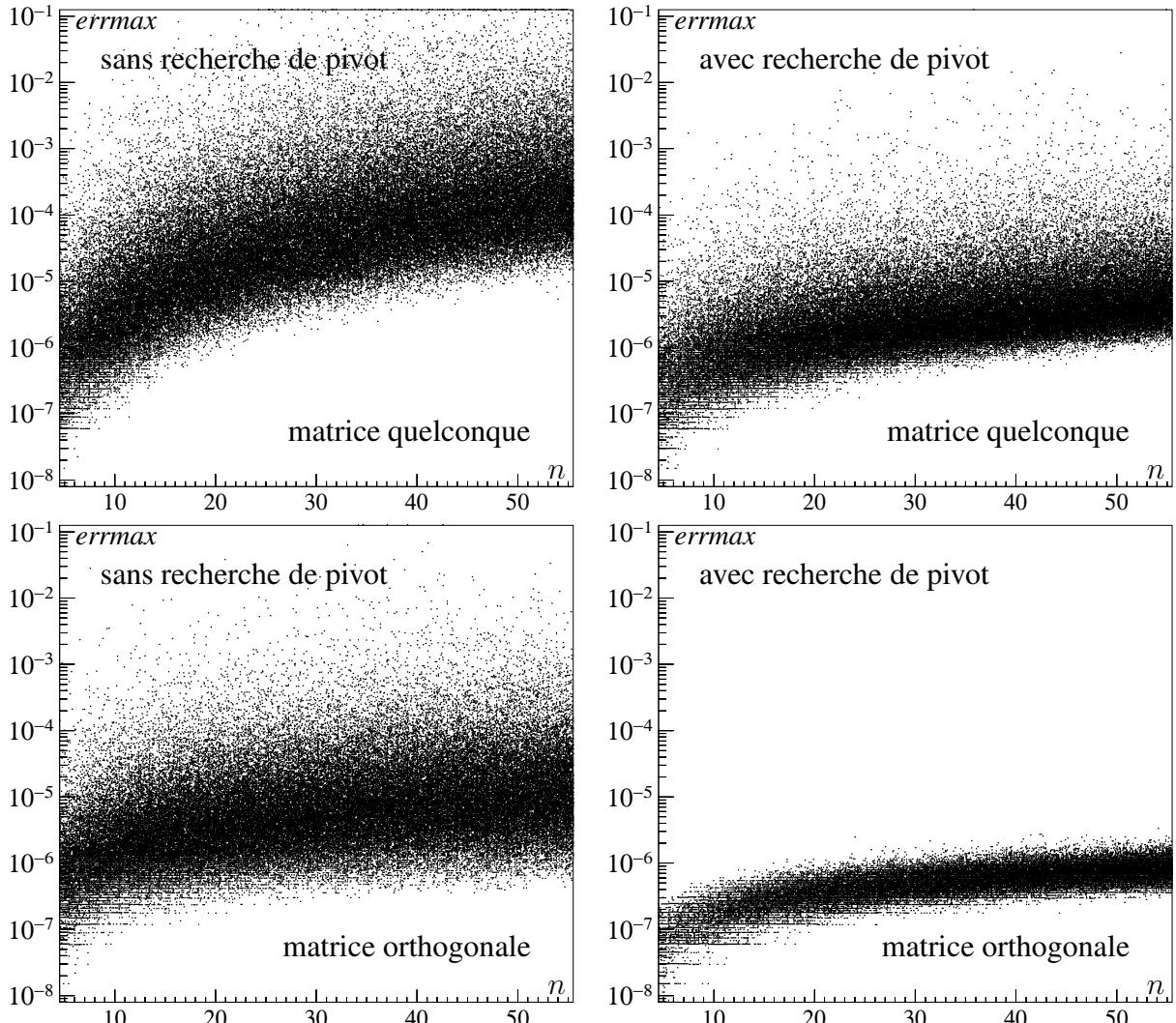


FIG. IV.1: Erreurs pour 1 million de systèmes linéaires de dimensions  $5 \times 5$  à  $55 \times 55$

on applique l'algorithme de Gauss, une fois sans recherche de pivot, et une fois avec recherche de pivot, en *simple précision*. L'erreur  $\max_i |x_i^{\text{num}} - x_i^{\text{ex}}|$  de chaque résultat est représentée par un petit point dans les dessins supérieurs de la figure IV.1. Bien que nous ne soyons pas surpris par les nombreuses erreurs sans recherche de pivot, *quelques* cas demeurent inacceptables à droite ; bon nombre de résultats restent cependant bons !

Faisons une *deuxième* expérience : une matrice avec  $a_{ij}$  uniformément distribués dans  $[-1, 1]$  pour  $j > i$  est complétée par  $a_{ji} = -a_{ij}$ , pour assurer que  $Q = (I - A)^{-1}(I + A)$  soit orthogonale. Cette matrice est calculée en double précision, le reste de l'expérience continue comme auparavant (voir les résultats au bas de la figure IV.1). Cette fois-ci il n'y a pas d'exception dans la bonne performance de l'algorithme de Gauss avec recherche de pivot. Nous allons démontrer ces observations dans les paragraphes suivantes.

### IV.3 La condition d'une matrice

En principe, un problème avec  $m$  données et  $n$  solutions possède  $m \times n$  coefficients décrivant la sensibilité de la  $n$ -ème solution par rapport à la  $m$ -ème donnée. Devant cette myriade de valeurs, il est parfois préférable d'exprimer la condition *par un seul nombre*. On réussira cela à l'aide de normes de vecteurs et de matrices (recherche initiée par A. Turing 1948).

**Rappel sur la norme d'une matrice.** Pour une matrice à  $m$  lignes et  $n$  colonnes, on définit

$$\|A\| = \max_{\|x\|=1} \|Ax\| = \max_{x \neq 0} \frac{\|Ax\|}{\|x\|}, \quad (3.1)$$

c.-à-d., la norme de  $A$  est le plus petit nombre  $\|A\|$  qui possède la propriété

$$\|Ax\| \leq \|A\| \cdot \|x\| \quad \text{pour tout } x \in \mathbb{R}^n. \quad (3.2)$$

Evidemment,  $\|A\|$  dépend des normes choisies dans  $\mathbb{R}^n$  et  $\mathbb{R}^m$ . Il y a des situations où l'on connaît des formules explicites pour  $\|A\|$ . Par exemple, si l'on prend la même norme dans les deux espaces alors,

pour  $\|x\|_1 = \sum_{i=1}^n |x_i|$ , on a

$$\|A\|_1 = \max_{j=1,\dots,n} \left( \sum_{i=1}^m |a_{ij}| \right); \quad (3.3)$$

pour  $\|x\|_2 = (\sum_{i=1}^n |x_i|^2)^{1/2}$ , on a

$$\|A\|_2 = \sqrt{\text{plus grande valeur propre de } A^T A}; \quad (3.4)$$

pour  $\|x\|_\infty = \max_{i=1,\dots,n} |x_i|$ , on a

$$\|A\|_\infty = \max_{i=1,\dots,m} \left( \sum_{j=1}^n |a_{ij}| \right). \quad (3.5)$$

La norme  $\|A\|$  d'une matrice satisfait toutes les propriétés d'une norme. En plus, elle vérifie  $\|I\| = 1$  pour la matrice d'identité et  $\|A \cdot B\| \leq \|A\| \cdot \|B\|$ .

Après ce rappel sur la norme d'une matrice, essayons d'estimer la condition du problème  $Ax = b$ . Pour ceci, considérons un deuxième système linéaire  $\hat{A}\hat{x} = \hat{b}$  avec des données perturbées

$$\begin{aligned} \hat{a}_{ij} &= a_{ij}(1 + \epsilon_{ij}), & |\epsilon_{ij}| &\leq \epsilon_A, \\ \hat{b}_i &= b_i(1 + \epsilon_i), & |\epsilon_i| &\leq \epsilon_b, \end{aligned} \quad (3.6)$$

où  $\epsilon_A$  et  $\epsilon_b$  spécifient la précision des données (par exemple  $\epsilon_A \leq \text{eps}$ ,  $\epsilon_b \leq \text{eps}$  où  $\text{eps}$  est la précision de l'ordinateur). Les hypothèses (3.6) impliquent (au moins pour les normes  $\|\cdot\|_1$  et  $\|\cdot\|_\infty$ ) que

$$\|\hat{A} - A\| \leq \epsilon_A \cdot \|A\|, \quad \|\hat{b} - b\| \leq \epsilon_b \cdot \|b\|. \quad (3.7)$$

Notre premier résultat donne une estimation de  $\|\hat{x} - x\|$ , en supposant que (3.7) soit vrai.

**Théorème 3.1** *Considérons les deux systèmes linéaires  $Ax = b$  et  $\hat{A}\hat{x} = \hat{b}$  où  $A$  est une matrice inversible. Si (3.7) est vérifié et si  $\epsilon_A \cdot \kappa(A) < 1$ , alors on a*

$$\frac{\|\hat{x} - x\|}{\|x\|} \leq \frac{\kappa(A)}{1 - \epsilon_A \cdot \kappa(A)} \cdot (\epsilon_A + \epsilon_b) \quad (3.8)$$

où  $\kappa(A) := \|A\| \cdot \|A^{-1}\|$ . Le nombre  $\kappa(A)$  s'appelle condition de la matrice  $A$ .

*Démonstration.* De  $\hat{b} - b = \hat{A}\hat{x} - Ax = (\hat{A} - A)\hat{x} + A(\hat{x} - x)$ , nous déduisons que

$$\hat{x} - x = A^{-1}(-(\hat{A} - A)\hat{x} + (\hat{b} - b)). \quad (3.9)$$

Maintenant, prenons la norme de (3.9), utilisons l'inégalité du triangle, les estimations (3.7),  $\|\hat{x}\| \leq \|x\| + \|\hat{x} - x\|$  et  $\|b\| = \|Ax\| \leq \|A\| \cdot \|x\|$ . Nous obtenons ainsi

$$\|\hat{x} - x\| \leq \|A^{-1}\| \left( \epsilon_A \cdot \|A\| \cdot (\|x\| + \|\hat{x} - x\|) + \epsilon_b \cdot \|A\| \cdot \|x\| \right).$$

Ceci donne l'estimation (3.8). □

La formule (3.8) montre que pour  $\epsilon_A \cdot \kappa(A) \ll 1$ , l'amplification maximale de l'erreur des données sur le résultat est de  $\kappa(A)$ .

**Propriétés de  $\kappa(A)$ .** Soit  $A$  une matrice inversible. Alors,

- a)  $\kappa(A) \geq 1$  pour toute  $A$ ,
- b)  $\kappa(\alpha A) = \kappa(A)$  pour  $\alpha \neq 0$ ,
- c)  $\kappa(A) = \max_{\|y\|=1} \|Ay\| / \min_{\|z\|=1} \|Az\|$ .

La propriété (c) permet d'étendre la définition de  $\kappa(A)$  aux matrices de dimension  $m \times n$  avec  $m \neq n$ .

*Démonstration.* La propriété (a) est une conséquence de  $1 = \|I\| = \|AA^{-1}\| \leq \|A\| \cdot \|A^{-1}\|$ . La propriété (b) est évidente. Pour montrer (c), nous utilisons

$$\|A^{-1}\| = \max_{x \neq 0} \frac{\|A^{-1}x\|}{\|x\|} = \max_{z \neq 0} \frac{\|z\|}{\|Az\|} = \left( \min_{z \neq 0} \frac{\|Az\|}{\|z\|} \right)^{-1}. \quad \square$$

**Exemples de matrices ayant une grande condition.** Considérons les matrices  $H_n$  (matrice de Hilbert) et  $V_n$  (matrice de Vandermonde) définies par ( $c_j = j/n$ )

$$H_n = \left( \frac{1}{i+j-1} \right)_{i,j=1}^n, \quad V_n = \left( c_j^{i-1} \right)_{i,j=1}^n.$$

Leur condition pour la norme  $\|\cdot\|_\infty$  est donnée dans le tableau IV.1.

**Exemples de matrices ayant une petite condition.** Une matrice  $U$  est orthogonale si  $U^T U = I$ . Pour la norme euclidienne, sa condition vaut 1 car  $\|U\|_2 = 1$  et  $\|U^{-1}\|_2 = 1$  (l'inverse  $U^{-1} = U^T$  est aussi orthogonale).

TAB. IV.1: Condition de matrices de Hilbert et Vandermonde

$n$	2	4	6	8	10	12
$\kappa(H_n)$	27	$2.8 \cdot 10^4$	$2.9 \cdot 10^7$	$3.4 \cdot 10^{10}$	$3.5 \cdot 10^{13}$	$3.8 \cdot 10^{16}$
$\kappa(V_n)$	8	$5.6 \cdot 10^2$	$3.7 \cdot 10^4$	$2.4 \cdot 10^6$	$1.6 \cdot 10^8$	$1.0 \cdot 10^{10}$

Concernant l'interpolation avec des fonctions splines, nous avons rencontré la matrice (voir le paragraphe II.10, cas équidistant)

$$A = \frac{1}{h} \begin{pmatrix} 4 & 1 & & & \\ 1 & 4 & 1 & & \\ & 1 & \ddots & \ddots & \\ & & \ddots & \ddots & \end{pmatrix} \quad \left. \right\} n \quad (3.10)$$

Le facteur  $1/h$  n'influence pas  $\kappa(A)$ . Posons alors  $h = 1$ . Avec la formule (3.5), on vérifie facilement que  $\|A\|_\infty = 6$ . Pour estimer  $\|A^{-1}\|_\infty$ , écrivons  $A$  sous la forme  $A = 4(I + N)$  où  $I$  est l'identité et  $N$  contient le reste. On voit que  $\|N\|_\infty = 1/2$ . En exprimant  $A^{-1}$  par une série géométrique, on obtient

$$\|A^{-1}\|_\infty \leq \frac{1}{4} (1 + \|N\|_\infty + \|N\|_\infty^2 + \|N\|_\infty^3 + \dots) \leq \frac{1}{2}.$$

Par conséquent,  $\kappa_\infty(A) \leq 3$  indépendamment de la dimension du système.

## IV.4 La stabilité d'un algorithme

Le but de ce paragraphe est d'étudier l'influence des erreurs d'arrondi sur le résultat pour l'élimination de Gauss. Commençons par la définition de la stabilité d'un algorithme et avec quelques exemples simples.

Un *algorithme* pour résoudre le problème  $\mathcal{P}(x)$  est une suite d'opérations élémentaires  $f_1, \dots, f_n$  (addition, soustraction, multiplication, division, évaluation d'une racine, d'une fonction élémentaire, ...) telle que

$$\mathcal{P}(x) = f_n(f_{n-1}(\dots f_2(f_1(x)) \dots)). \quad (4.1)$$

En général, il existe beaucoup d'algorithmes différents pour résoudre le même problème  $\mathcal{P}(x)$ .

L'amplification de l'erreur, en faisant l'opération  $f_i$ , est décrite par la condition  $\kappa(f_i)$  (voir la définition dans le paragraphe IV.2). L'estimation

$$\kappa(\mathcal{P}) \leq \kappa(f_1) \cdot \kappa(f_2) \cdot \dots \cdot \kappa(f_n). \quad (4.2)$$

est une conséquence simple de la définition de la condition d'un problème.

**Définition 4.1** *Un algorithme est numériquement stable (au sens de "forward analysis") si*

$$\kappa(f_1) \cdot \kappa(f_2) \cdot \dots \cdot \kappa(f_n) \leq \text{Const} \cdot \kappa(\mathcal{P}) \quad (4.3)$$

où *Const* n'est pas trop grand (par exemple,  $\text{Const} = \mathcal{O}(n)$ ).

La formule (4.3) exprime le fait que l'influence des erreurs d'arrondi durant le calcul de  $\mathcal{P}(x)$  n'est pas beaucoup plus grande que l'influence d'erreurs dans les données (qui sont inévitables).

**Exemple 4.2** Soit  $x = 10^4$  et considérons le problème de calculer  $1/(x(1+x))$ . Examinons les deux algorithmes suivants :

$$\text{a)} \quad x \begin{array}{c} \nearrow x \\ \searrow x+1 \end{array} \rightarrow x(x+1) \rightarrow \frac{1}{x(x+1)}.$$

Toutes ces opérations sont très bien conditionnées (voir le paragraphe IV.3). Ainsi, cet algorithme est numériquement stable.

$$\text{b)} \quad x \begin{array}{c} \nearrow 1/x \\ \searrow x+1 \end{array} \rightarrow 1/(x+1) \rightarrow \frac{1}{x} - \frac{1}{x+1} = \frac{1}{x(x+1)}.$$

Dans cet algorithme, seules les trois premières opérations sont bien conditionnées. La soustraction, à la fin, est très mal conditionnée car  $1/x \approx 1/(x+1)$ . Ainsi, cet algorithme est numériquement *instable*.

La vérification, si un algorithme (non-trivial) est stable (au sens de “forward analysis”), est souvent très complexe et difficile. Pour cette raison, Wilkinson (1961, J. Ass. Comp. Mach. 8) a introduit une autre définition de la stabilité d’un algorithme.

**Définition 4.3** Un algorithme pour résoudre le problème  $\mathcal{P}(x)$  est numériquement stable (au sens de “backward analysis”) si le résultat numérique  $\hat{y}$  peut être interprété comme un résultat exact pour des données perturbées  $\hat{x}$  (c.-à-d.,  $\hat{y} = \mathcal{P}(\hat{x})$ ) et si

$$\frac{|\hat{x}_i - x_i|}{|x_i|} \leq \text{Const} \cdot \text{eps} \quad (4.4)$$

où  $\text{Const}$  n'est pas trop grand et  $\text{eps}$  est la précision de l'ordinateur.

*Remarque.* Pour l'étude de cette stabilité, il ne faut pas connaître la condition du problème.

**Exemple 4.4** Considérons le problème de calculer le produit scalaire  $x_1 \cdot x_2 + x_3 \cdot x_4$ . On utilise l'algorithme

$$(x_1, x_2, x_3, x_4) \begin{array}{c} \nearrow x_1 \cdot x_2 \\ \searrow x_3 \cdot x_4 \end{array} \rightarrow x_1 \cdot x_2 + x_3 \cdot x_4. \quad (4.5)$$

Le résultat numérique (sous l'influence des erreurs d'arrondi) est

$$(x_1(1 + \epsilon_1) \cdot x_2(1 + \epsilon_2)(1 + \eta_1) + x_3(1 + \epsilon_3) \cdot x_4(1 + \epsilon_4)(1 + \eta_2))(1 + \eta_3)$$

où  $|\epsilon_i|, |\eta_j| \leq \text{eps}$ . Ce résultat est égal à  $\hat{x}_1 \cdot \hat{x}_2 + \hat{x}_3 \cdot \hat{x}_4$  si l'on pose

$$\begin{aligned} \hat{x}_1 &= x_1(1 + \epsilon_1)(1 + \eta_1), & \hat{x}_3 &= x_3(1 + \epsilon_3)(1 + \eta_2), \\ \hat{x}_2 &= x_2(1 + \epsilon_2)(1 + \eta_3), & \hat{x}_4 &= x_4(1 + \epsilon_4)(1 + \eta_3). \end{aligned}$$

Ainsi, (4.4) est vérifié pour  $\text{Const} = 2$  (on néglige les produits  $\epsilon_i \cdot \eta_j$ ). En conséquence, l'algorithme (4.5) est toujours numériquement stable (au sens de “backward analysis”).

Cet exemple montre bien qu'un algorithme peut être stable, même si le problème est mal conditionné. Ainsi, il faut bien distinguer les notions “stabilité numérique” et “condition d'un problème”.

**La stabilité de l'élimination de Gauss.** Soit donnée une matrice  $A$  ( $\det A \neq 0$ ) ayant la décomposition  $A = LR$  (on suppose que les permutations nécessaires sont déjà effectuées). En appliquant l'élimination de Gauss, nous obtenons deux matrices  $\hat{L}$  et  $\hat{R}$ , qui représentent la décomposition exacte de la matrice  $\hat{A} := \hat{L} \cdot \hat{R}$ . Pour montrer la stabilité numérique (au sens de “backward analysis”) de l'élimination de Gauss, on a besoin de trouver une estimation de la forme  $|\hat{a}_{ij} - a_{ij}| \leq |a_{ij}| \cdot \text{Const} \cdot \text{eps}$ . En tous cas, il faut estimer la différence  $\hat{a}_{ij} - a_{ij}$ .

**Théorème 4.5 (Wilkinson)** Soit  $A$  une matrice inversible et  $\widehat{L}$ ,  $\widehat{R}$  le résultat numérique de l'élimination de Gauss (avec recherche de pivot, c.-à-d.  $|\widehat{\ell}_{ij}| \leq 1$  pour tout  $i, j$ ). Alors,

$$|\widehat{a}_{ij} - a_{ij}| \leq 2 \cdot a \cdot \min(i-1, j) \cdot \text{eps} \quad (4.6)$$

où  $a = \max_{i,j,k} |a_{ij}^{(k)}|$ .

*Démonstration.* Lors de la  $k$ ème étape de l'élimination de Gauss, on calcule  $\widehat{a}_{ij}^{(k)}$  à partir de  $\widehat{a}_{ij}^{(k-1)}$ . Si l'on prend en considération les erreurs d'arrondi, on obtient

$$\begin{aligned} \widehat{a}_{ij}^{(k)} &= (\widehat{a}_{ij}^{(k-1)} - \widehat{\ell}_{ik} \cdot \widehat{a}_{kj}^{(k-1)} \cdot (1 + \epsilon_{ijk})) (1 + \eta_{ijk}) \\ &= \widehat{a}_{ij}^{(k-1)} - \widehat{\ell}_{ik} \cdot \widehat{a}_{kj}^{(k-1)} + \mu_{ijk} \end{aligned} \quad (4.7)$$

où (en négligeant les termes  $\mathcal{O}(\text{eps}^2)$ )

$$|\mu_{ijk}| \leq |\widehat{a}_{ij}^{(k)}| \cdot |\eta_{ijk}| + |\widehat{\ell}_{ik}| \cdot |\widehat{a}_{kj}^{(k-1)}| \cdot |\epsilon_{ijk}| \leq 2 \cdot a \cdot \text{eps}. \quad (4.8)$$

Par définition de  $\widehat{A}$ , on a  $\widehat{a}_{ij} = \sum_{k=1}^{\min(i,j)} \widehat{\ell}_{ik} \cdot \widehat{r}_{kj} = \sum_{k=1}^{\min(i,j)} \widehat{\ell}_{ik} \cdot \widehat{a}_{kj}^{(k-1)}$  et, en utilisant la formule (4.7), on obtient pour  $i > j$

$$\widehat{a}_{ij} = \sum_{k=1}^j (\widehat{a}_{ij}^{(k-1)} - \widehat{a}_{ij}^{(k)} + \mu_{ijk}) = a_{ij} + \sum_{k=1}^j \mu_{ijk} \quad (4.9a)$$

car  $\widehat{a}_{ij}^{(j)} = 0$  dans cette situation. Pour  $i \leq j$ , on a

$$\widehat{a}_{ij} = \sum_{k=1}^{i-1} (\widehat{a}_{ij}^{(k-1)} - \widehat{a}_{ij}^{(k)} + \mu_{ijk}) + \widehat{\ell}_{ii} \cdot \widehat{a}_{ij}^{(i-1)} = a_{ij} + \sum_{k=1}^{i-1} \mu_{ijk} \quad (4.9b)$$

car  $\widehat{\ell}_{ii} = 1$ . Les formules (4.9) ensemble avec l'estimation (4.8) démontrent l'estimation (4.6).  $\square$

*Conséquence.* L'élimination de Gauss est stable (au sens de “backward analysis”) si le quotient

$$\max_{i,j,k} |a_{ij}^{(k)}| / \max_{i,j} |a_{ij}| \quad (4.10)$$

n'est pas trop grand. En fait, il existe des matrices pathologiques pour lesquelles ce quotient atteint  $2^{n-1}$ , où  $n$  est la dimension de la matrice  $A$  (voir exercice 12). Mais heureusement cette constante est en général beaucoup plus petite. Pour illustrer ceci, en suivant une idée de Trefethen & Schreiber (1990, SIAM J. Matrix Anal. Appl. 11) nous avons pris un grand nombre de matrices de dimensions allant de 2 à 24 dont les éléments sont des nombres aléatoires dans  $[-1, 1]$ . Nous avons dessiné dans la figure IV.2 le quotient (4.10) en fonction de la dimension de la matrice (chaque point représente la moyenne de 30 échantillons). En échelle doublement logarithmique, le résultat semble mystérieusement devenir une droite.

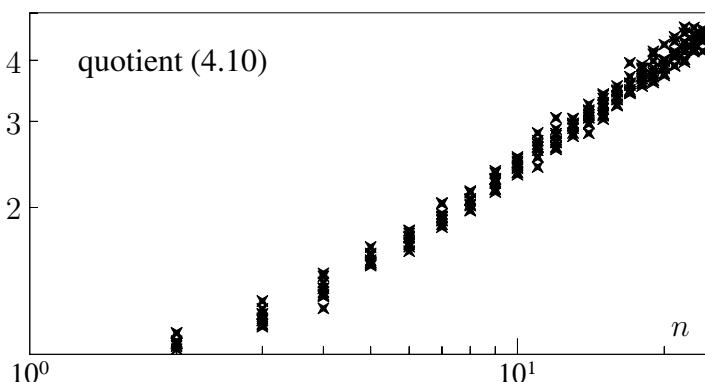


FIG. IV.2: Stabilité numérique de l'élimination de Gauss

## IV.5 L'algorithme de Cholesky

Etudions l'élimination de Gauss pour le cas important où

$A$  est symétrique ( $A^T = A$ ) et

$A$  est définie positive ( $x^T Ax > 0$  pour  $x \neq 0$ ).

Le théorème suivant montre que, dans cette situation particulière, il n'est pas nécessaire d'effectuer une recherche de pivot.

**Théorème 5.1** Soit  $A$  une matrice symétrique et définie positive.

a) L'élimination de Gauss est faisable sans recherche de pivot.

b) La décomposition  $A = LR$  satisfait

$$R = DL^T \quad \text{avec} \quad D = \text{diag}(r_{11}, \dots, r_{nn}). \quad (5.1)$$

*Démonstration.* a) On a  $a_{11} = e_1^T A e_1 > 0$  (avec  $e_1 = (1, 0, \dots, 0)^T$ ) car la matrice  $A$  est définie positive. Alors, on peut choisir  $a_{11}$  comme pivot dans la première étape de l'élimination de Gauss. Ceci donne

$$A = \begin{pmatrix} a_{11} & a^T \\ a & C \end{pmatrix} \quad \longrightarrow \quad A^{(1)} = \begin{pmatrix} a_{11} & a^T \\ 0 & C^{(1)} \end{pmatrix} \quad (5.2)$$

où  $c_{ij}^{(1)} = a_{ij} - \frac{a_{i1}}{a_{11}} \cdot a_{1j}$  pour  $i, j = 2, \dots, n$ , ce qui est équivalent à

$$C^{(1)} = C - \frac{1}{a_{11}} \cdot a \cdot a^T. \quad (5.3)$$

La matrice  $C^{(1)}$  est symétrique (trivial). Montrons qu'elle est aussi définie positive. Pour ceci, nous prenons un  $y \in \mathbb{R}^{n-1}$ ,  $y \neq 0$ . Il faut montrer que  $y^T C^{(1)} y > 0$ . La partition de (5.2) et le fait que  $A$  soit définie positive impliquent

$$(x_1, y^T) \begin{pmatrix} a_{11} & a^T \\ a & C \end{pmatrix} \begin{pmatrix} x_1 \\ y \end{pmatrix} = a_{11}x_1^2 + 2x_1 \cdot y^T a + y^T C y > 0. \quad (5.4)$$

En posant  $x_1 = -y^T a / a_{11}$  dans (5.4), on obtient de (5.3) que

$$y^T C^{(1)} y = y^T C y - \frac{1}{a_{11}} (y^T a)^2 > 0.$$

Par récurrence, on voit que la deuxième et aussi les autres étapes de l'élimination de Gauss sont faisables sans recherche de pivot.

b) La formule (5.1) est une conséquence de l'unicité de l'élimination de Gauss pour des matrices inversibles. En effet, on peut écrire  $R = D\hat{L}^T$  et on obtient  $A = A^T = R^T L^T = \hat{L}(DL^T)$ , d'où  $\hat{L} = L$ .

Pour montrer l'unicité de l'élimination de Gauss, supposons  $A = LR = \hat{L}\hat{R}$  et considérons l'identité  $\hat{L}^{-1}L = \hat{R}R^{-1}$ . Le produit de deux matrices triangulaires inférieures reste une matrice triangulaire inférieure; de même pour les matrices triangulaires supérieures. Comme les éléments de la diagonale de  $\hat{L}^{-1}L$  sont tous égaux à 1, on a

$$\hat{L}^{-1}L = \hat{R}R^{-1} = I, \quad (5.5)$$

ce qui implique l'unicité de l'élimination de Gauss. □

La décomposition

$$A = LDL^T \quad (5.6)$$

s'appelle *décomposition rationnelle de Cholesky*<sup>1</sup>. Comme  $r_{ii} > 0$  ( $A$  est définie positive), on peut considérer la racine  $D^{1/2} = \text{diag}(\sqrt{r_{11}}, \dots, \sqrt{r_{nn}})$ , et la décomposition (5.6) devient  $A = (LD^{1/2})(D^{1/2}L^T) = (LD^{1/2})(LD^{1/2})^T$ . Par abus de notation, en écrivant  $L$  pour  $LD^{1/2}$ , nous obtenons la *décomposition de Cholesky*

$$A = LL^T \quad \text{où} \quad L = \begin{pmatrix} \ell_{11} & & 0 \\ \vdots & \ddots & \\ \ell_{n1} & \dots & \ell_{nn} \end{pmatrix}. \quad (5.7)$$

Une comparaison des coefficients dans l'identité  $A = LL^T$  donne pour

$$\begin{aligned} i = k : \quad a_{kk} &= \ell_{k1}^2 + \ell_{k2}^2 + \dots + \ell_{kk}^2 \\ i > k : \quad a_{ik} &= \ell_{i1}\ell_{k1} + \dots + \ell_{i,k-1}\ell_{k,k-1} + \ell_{ik}\ell_{kk} \end{aligned}$$

et on en déduit l'algorithme suivant:

### Algorithme de Cholesky.

```

for  $k := 1$  to  $n$  do
     $\ell_{kk} := (a_{kk} - \sum_{j=1}^{k-1} \ell_{kj}^2)^{1/2};$ 
    for  $i := k + 1$  to  $n$  do
         $\ell_{ik} := (a_{ik} - \sum_{j=1}^{k-1} \ell_{ij}\ell_{kj})/\ell_{kk}.$ 

```

*Le coût de cet algorithme.* En négligeant les  $n$  racines, le nombre d'opérations nécessaires est

$$\sum_{k=1}^n (n-k) \cdot k \approx \int_0^n (n-x)x \, dx = \frac{n^3}{6}.$$

Ceci correspond à la moitié du coût de la décomposition LR.

Pour résoudre le système  $Ax = b$ , on calcule d'abord la décomposition de Cholesky (5.7). Puis, on résoud successivement les deux systèmes  $Lc = b$  et  $L^Tx = c$ , dont les matrices sont triangulaires.

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

**Théorème 5.2** Soit  $A$  une matrice symétrique et définie positive. Notons  $\widehat{A} = \widehat{L} \cdot \widehat{L}^T$ , où  $\widehat{L}$  est la matrice triangulaire obtenue par l'algorithme de Cholesky. Alors,

$$|\widehat{a}_{ij} - a_{ij}| \leq a_0 \cdot \min(i, j) \cdot \text{eps} \quad (5.8)$$

où  $a_0 = \max_{i,j} |a_{ij}| = \max_i |a_{ii}|$ . □

Ce résultat démontre que l'algorithme de Cholesky est toujours numériquement stable. Il n'est donc pas nécessaire de faire une recherche de pivot, si  $A$  est symétrique et définie positive.

---

<sup>1</sup>Le “Commandant Cholesky” (1875–1918) entra à l’École Polytechnique à l’âge de vingt ans et en sortit dans l’arme de l’Artillerie. Affecté à la Section de Géodésie du Service géographique, en juin 1905, il s’y fit remarquer de suite par une intelligence hors ligne, une grande facilité pour les travaux mathématiques, un esprit chercheur, des idées originales, parfois même paradoxales, mais toujours empreintes d’une grande élévation de sentiments et qu’il soutenait avec une extrême chaleur. (...) Cholesky aborda ce problème en apportant dans ses solutions, ... une originalité marquée. Il imagina pour la résolution des équations de condition par la méthode des moindres carrés un procédé de calcul très ingénieux ... (copié du *Bulletin géodésique* No. 1, 1922).

## IV.6 Systèmes surdéterminés – méthode des moindres carrés

Considérons un système d'équations linéaires

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n &= b_2 \\ \vdots &\quad \vdots \quad \vdots \quad \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \dots + a_{mn}x_n &= b_m \end{aligned} \tag{6.1}$$

où  $m \geq n$  (matriciellement:  $Ax = b$  avec  $x \in \mathbb{R}^n$  et  $b \in \mathbb{R}^m$ ;  $A$  est une matrice  $m \times n$ ). Evidemment, le système (6.1) ne possède, en général, pas de solution. L'idée est de chercher un vecteur  $x$  tel que

$$\|Ax - b\|_2 \rightarrow \min \tag{6.2}$$

pour la norme euclidienne. Une justification probabiliste de cette condition sera donnée dans le paragraphe IV.8. Le nom “méthode des moindres carrés” indique le choix de la norme dans (6.2) (la somme des carrés des erreurs doit être minimale).

**Théorème 6.1** Soit  $A$  une matrice  $m \times n$  (avec  $m \geq n$ ) et soit  $b \in \mathbb{R}^m$ . Le vecteur  $x$  est solution de (6.2) si et seulement si

$$A^T Ax = A^T b. \tag{6.3}$$

Les équations du système (6.3) s'appellent “équations normales”.

*Démonstration.* Les minima de la fonction quadratique

$$f(x) := \|Ax - b\|^2 = (Ax - b)^T (Ax - b) = x^T A^T Ax - 2x^T A^T b + b^T b$$

sont donnés par  $0 = f'(x) = 2(x^T A^T A - b^T A)$ . □

*Interprétation géométrique.* L'ensemble  $E = \{Ax \mid x \in \mathbb{R}^n\}$  est un sous-espace linéaire de  $\mathbb{R}^m$ . Pour un  $b \in \mathbb{R}^m$  arbitraire,  $x$  est une solution de (6.2) si et seulement si  $Ax$  est la projection orthogonale de  $b$  sur  $E$ . Ceci signifie que  $Ax - b \perp Az$  pour tout  $z \in \mathbb{R}^n$ . On en déduit que  $A^T(Ax - b) = 0$  et on a ainsi établi une deuxième démonstration de (6.3).

**Exemple 6.2** Pour étudier le phénomène de la thermo-électricité, on fait l'expérience suivante. On soude un fil de cuivre avec un fil de constantan de manière à obtenir une boucle fermée. Un point de soudure est maintenu à température fixe ( $T_0 \approx 24^\circ\text{C}$ ), alors que l'on fait varier la température  $T$  de l'autre. Ceci génère une tension  $U$ , laquelle est mesurée en fonction de  $T$  (voir le tableau IV.2 et la figure IV.3). Les données du tableau IV.2 sont prises du livre de P.R. Bevington<sup>2</sup>.

On suppose que cette dépendance obéit à la loi

$$U = a + bT + cT^2 \tag{6.4}$$

et on cherche à déterminer les paramètres  $a$ ,  $b$  et  $c$ . Les données du tableau IV.2 nous conduisent au système surdéterminé ( $n = 3$ ,  $m = 21$ )

$$U_i = a + bT_i + cT_i^2, \quad i = 1, \dots, 21. \tag{6.5}$$

En résolvant les équations normales (6.3) pour ce problème, on obtient  $a = -0.886$ ,  $b = 0.0352$  et  $c = 0.598 \cdot 10^{-4}$ . Avec ces paramètres, la fonction (6.4) est dessinée dans la figure IV.3. On observe une très bonne concordance avec les données.

*Remarque.* Les équations normales (6.3) possèdent toujours au moins une solution (la projection sur  $E$  existe toujours). La matrice  $A^T A$  est symétrique et non-négative ( $x^T A^T A x = \|Ax\|^2 \geq 0$ ).

<sup>2</sup>P.R. Bevington (1969): *Data reduction and error analysis for the physical sciences*. McGraw-Hill.

TAB. IV.2: Tensions mesurées en fonction de la température  $T$ 

$i$	$T_i^{\circ}\text{C}$	$U_i$	$i$	$T_i^{\circ}\text{C}$	$U_i$	$i$	$T_i^{\circ}\text{C}$	$U_i$
1	0	-0.89	8	35	0.42	15	70	1.88
2	5	-0.69	9	40	0.61	16	75	2.10
3	10	-0.53	10	45	0.82	17	80	2.31
4	15	-0.34	11	50	1.03	18	85	2.54
5	20	-0.15	12	55	1.22	19	90	2.78
6	25	0.02	13	60	1.45	20	95	3.00
7	30	0.20	14	65	1.68	21	100	3.22

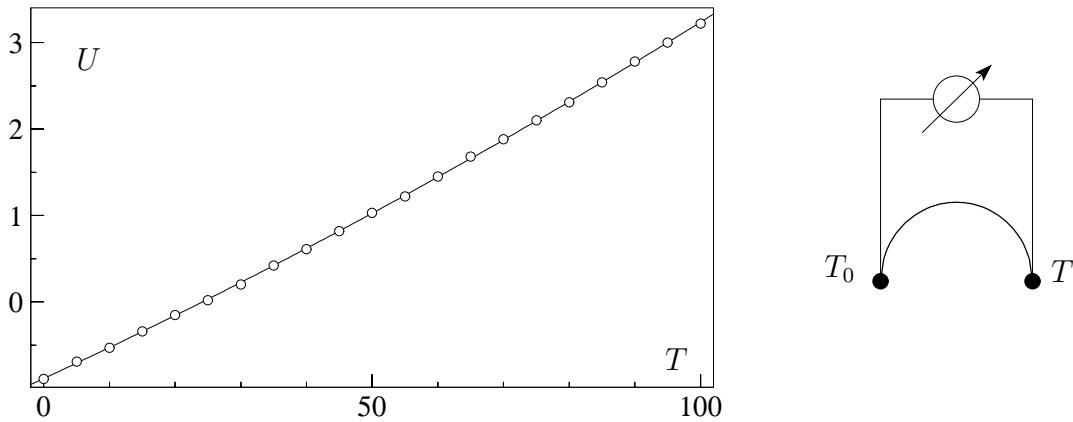


FIG. IV.3: Tension en fonction de la température et schéma de l'expérience

Elle est définie positive si les colonnes de  $A$  sont linéairement indépendantes ( $Ax \neq 0$  pour  $x \neq 0$ ). Dans cette situation, on peut appliquer l'algorithme de Cholesky pour résoudre le système (6.3). Mais, souvent, il est préférable de calculer la solution directement de (6.2) sans passer par les équations normales (6.3).

## IV.7 Décomposition QR d'une matrice

Dans l'élimination de Gauss, on a multiplié l'équation  $Ax = b$  par la matrice triangulaire  $L_{n-1} \cdot \dots \cdot L_2 \cdot L_1$ . De cette manière, on a réduit le problème original à  $Rx = c$  où  $R$  est une matrice triangulaire supérieure. Malheureusement, la multiplication de  $Ax - b$  avec  $L_i$  ne conserve pas la norme du vecteur.

Pour résoudre (6.2), nous cherchons une matrice orthogonale  $Q$  telle que

$$Q^T(Ax - b) = Rx - c = \begin{pmatrix} R' \\ 0 \end{pmatrix} x - \begin{pmatrix} c' \\ c'' \end{pmatrix} \quad (7.1)$$

où  $R'$  (une matrice carrée de dimension  $n$ ) est triangulaire supérieure et  $(c', c'')^T$  est la partition de  $c = Q^T b$  telle que  $c' \in \mathbb{R}^n$  et  $c'' \in \mathbb{R}^{m-n}$ . Comme le produit par une matrice orthogonale ne change pas la norme du vecteur, on a

$$\|Ax - b\|_2^2 = \|Q^T(Ax - b)\|_2^2 = \|Rx - c\|_2^2 = \|R'x - c'\|_2^2 + \|c''\|_2^2. \quad (7.2)$$

On obtient alors la solution de (6.2) en résolvant le système

$$R'x = c'. \quad (7.3)$$

Le problème consiste à calculer une matrice orthogonale  $Q$  (c.-à-d.,  $Q^T Q = I$ ) et une matrice triangulaire supérieure  $R$  telles que  $Q^T A = R$  ou de façon équivalente

$$A = QR. \quad (7.4)$$

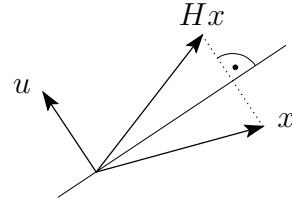
Cette factorisation s'appelle la “décomposition QR” de la matrice  $A$ . Pour arriver à ce but, on peut se servir des rotations de Givens (voir exercice 11 du chapitre V) ou des réflexions de Householder.

**Réflexions de Householder (1958).** Une matrice de la forme

$$H = I - 2uu^T \quad \text{où} \quad u^T u = 1 \quad (7.5)$$

a les propriétés suivantes :

- $H$  est une réflexion à l'hyper-plan  $\{x \mid u^T x = 0\}$  car  $Hx = x - u \cdot (2u^T x)$  et  $Hx + x \perp u$ .
- $H$  est symétrique.
- $H$  est orthogonale, car



$$H^T H = (I - 2uu^T)^T (I - 2uu^T) = I - 4uu^T + 4uu^T uu^T = I.$$

En multipliant  $A$  avec des matrices de Householder, nous allons essayer de transformer  $A$  en une matrice de forme triangulaire.

**L'algorithme de Householder - Businger - Golub.** Dans une *première étape*, on cherche une matrice  $H_1 = I - 2u_1u_1^T$  ( $u_1 \in \mathbb{R}^m$  et  $u_1^T u_1 = 1$ ) telle que

$$H_1 A = \begin{pmatrix} \alpha_1 & \times & \cdots & \times \\ 0 & \times & \cdots & \times \\ \vdots & \vdots & & \vdots \\ 0 & \times & \cdots & \times \end{pmatrix}. \quad (7.6)$$

Si l'on dénote par  $A_1$  la première colonne de  $A$ , il faut que  $H_1 A_1 = \alpha_1 e_1 = (\alpha_1, 0, \dots, 0)^T$  et on obtient  $|\alpha_1| = \|H_1 A_1\|_2 = \|A_1\|_2$ . La forme particulière de  $H_1$  implique que

$$H_1 A_1 = A_1 - 2u_1 \cdot u_1^T A_1 = \alpha_1 e_1.$$

L'expression  $u_1^T A_1$  est un scalaire. Par conséquent,

$$u_1 = C \cdot v_1 \quad \text{où} \quad v_1 = A_1 - \alpha_1 e_1 \quad (7.7)$$

et la constante  $C$  est déterminée par  $\|u_1\|_2 = 1$ . Comme on a encore la liberté de choisir le signe de  $\alpha_1$ , posons

$$\alpha_1 = -\text{sign}(a_{11}) \cdot \|A_1\|_2 \quad (7.8)$$

pour éviter une soustraction mal conditionnée dans le calcul de  $v_1 = A_1 - \alpha_1 e_1$ .

*Calcul de  $H_1 A$ .* Notons par  $A_j$  et  $(H_1 A)_j$  les  $j$ èmes colonnes de  $A$  et  $H_1 A$  respectivement. Alors, on a

$$(H_1 A)_j = A_j - 2u_1 u_1^T A_j = A_j - \beta \cdot v_1^T A_j \cdot v_1 \quad \text{où} \quad \beta = \frac{2}{v_1^T v_1}. \quad (7.9)$$

Le facteur  $\beta$  peut être calculé à l'aide de

$$\beta^{-1} = \frac{v_1^T v_1}{2} = \frac{1}{2} (A_1^T A_1 - 2\alpha_1 a_{11} + \alpha_1^2) = -\alpha_1 (a_{11} - \alpha_1). \quad (7.10)$$

Dans une *deuxième étape*, on applique la procédure précédente à la sous-matrice de dimension  $(m-1) \times (n-1)$  de (7.6). Ceci donne un vecteur  $\bar{u}_2 \in \mathbb{R}^{m-1}$  et une matrice de Householder  $\bar{H}_2 = I - 2\bar{u}_2\bar{u}_2^T$ . En posant  $u_2 = (0, \bar{u}_2)^T$ , une multiplication de (7.6) par la matrice  $H_2 = I - 2u_2u_2^T$  donne

$$H_2 H_1 A = H_2 \begin{pmatrix} \alpha_1 & \times & \cdots & \times \\ 0 & & & \\ \vdots & & C & \\ 0 & & & \end{pmatrix} = \begin{pmatrix} \alpha_1 & \times & \cdots & \times \\ 0 & & & \\ \vdots & & \bar{H}_2 C & \\ 0 & & & \end{pmatrix} = \begin{pmatrix} \alpha_1 & \times & \times & \cdots & \times \\ 0 & \alpha_2 & \times & \cdots & \times \\ 0 & 0 & \times & \cdots & \times \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \times & \cdots & \times \end{pmatrix}.$$

En continuant cette procédure, on obtient après  $n$  étapes (après  $n-1$  étapes si  $m=n$ ) une matrice triangulaire

$$\underbrace{H_n \cdot \dots \cdot H_2 H_1}_Q^T A = R = \begin{pmatrix} R' \\ 0 \end{pmatrix}.$$

Ceci donne la décomposition (7.4) avec  $Q^T = H_n \cdot \dots \cdot H_2 H_1$ .

**Coût de la décomposition QR.** La première étape exige le calcul de  $\alpha_1$  par la formule (7.8) ( $\approx m$  opérations), le calcul de  $2/v_1^T v_1$  par la formule (7.10) (travail négligeable) et le calcul de  $(H_1 A)_j$  pour  $j = 2, \dots, n$  par la formule (7.9) ( $\approx (n-1) \cdot 2 \cdot m$  opérations). En tout, cette étape nécessite environ  $2mn$  opérations. Pour la décomposition QR, on a alors besoin de

$$2(n^2 + (n-1)^2 + \dots + 1) \approx 2n^3/3 \text{ opérations si } m=n \text{ (matrice carrée);}$$

$$2m(n + (n-1) + \dots + 1) \approx mn^2 \text{ opérations si } m \gg n.$$

En comparant encore ce travail avec celui de la résolution des équations normales ( $\approx mn^2/2$  opérations pour le calcul de  $A^T A$  et  $\approx n^3/6$  opérations pour la décomposition de Cholesky de  $A^T A$ ), on voit que la décomposition QR coûte au pire le double.

*Remarque.* Si les colonnes de la matrice  $A$  sont linéairement indépendantes, tous les  $\alpha_i$  sont non nuls et l'algorithme de Householder–Businger–Golub est applicable. Une petite modification (échange des colonnes de  $A$ ) permet de traiter aussi le cas général.

Concernant la programmation, il est important de ne calculer ni les matrices  $H_i$ , ni la matrice  $Q$ . On retient simplement les valeurs  $\alpha_i$  et les vecteurs  $v_i$  (pour  $i = 1, \dots, n$ ) qui contiennent déjà toutes les informations nécessaires pour la décomposition. Comme pour l'élimination de Gauss, on écrit deux sous-programmes. DECQR fournit la décomposition QR de la matrice  $A$  (c.-à-d. les  $\alpha_i$ ,  $v_i$  et la matrice  $R$ ). Le sous-programme SOLQR calcule  $Q^T b$  et la solution du système triangulaire  $R'x = c'$  (voir (7.3)). Le calcul de  $Q^T b = H_n \cdot \dots \cdot H_2 H_1 b$  se fait avec une formule analogue à (7.9).

**Exemple 7.1** Si les colonnes de  $A$  sont “presque” linéairement dépendantes, la résolution du problème (6.2) à l'aide de la décomposition QR est préférable à celle des équations normales. Considérons, par exemple,

$$A = \begin{pmatrix} 1 & 1 \\ \epsilon & 0 \\ 0 & \epsilon \end{pmatrix}, \quad b = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$$

où  $\epsilon$  est une petite constante, disons  $\epsilon^2 < \text{eps}$ . Avec un calcul exact, on obtient

$$A^T A = \begin{pmatrix} 1 + \epsilon^2 & 1 \\ 1 & 1 + \epsilon^2 \end{pmatrix}, \quad A^T b = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

et la solution est donnée par

$$x_1 = x_2 = \frac{1}{2 + \epsilon^2} = \frac{1}{2} + \mathcal{O}(\epsilon^2).$$

Un calcul en virgule flottante fait disparaître le  $\epsilon^2$  dans  $A^T A$  et cette matrice devient singulière. On n'obtient pas de solution.

Par contre, l'algorithme de Householder–Businger–Golub donne (en négligeant  $\epsilon^2$ )  $\alpha_1 = -1$ ,  $v_1 = (2, \epsilon, 0)^T, \dots$  et à la fin

$$R = \begin{pmatrix} -1 & -1 \\ 0 & \sqrt{2} \cdot \epsilon \\ 0 & 0 \end{pmatrix}, \quad Q^T b = \begin{pmatrix} -1 \\ \epsilon/\sqrt{2} \\ -\epsilon/\sqrt{2} \end{pmatrix}.$$

La résolution de (7.3) donne une bonne approximation de la solution exacte.

## IV.8 Etude de l'erreur de la méthode des moindres carrés

Comme c'est le cas dans l'exemple du paragraphe IV.6, nous cherchons les paramètres  $x_1, \dots, x_n$  d'une loi

$$\sum_{j=1}^n c_j(t) x_j = b \tag{8.1}$$

qui relie les variables  $t$  et  $b$  (les fonctions  $c_j(t)$  sont données, p. ex.  $c_j(t) = t^{j-1}$ ). Supposons que pour plusieurs valeurs de  $t$  (disons  $t_1, \dots, t_m$ ,  $m \gg n$ ) l'on puisse mesurer les quantités  $b_1, \dots, b_m$ . On obtient ainsi le système surdéterminé

$$\sum_{j=1}^n a_{ij} x_j = b_i, \quad i = 1, \dots, m \tag{8.2}$$

où  $a_{ij} = c_j(t_i)$ . En pratique, les  $b_i$  sont des mesures légèrement erronées et il est naturel de les considérer comme des valeurs plus ou moins aléatoires. L'étude de l'erreur de la solution  $x$ , obtenue par la méthode des moindres carrés, se fait alors dans le cadre de la théorie des probabilités.

### Rappel sur la théorie des probabilités

Considérons des *variables aléatoires*  $X$  (dites “continues”) qui sont spécifiées par une fonction de densité  $f : \mathbb{R} \rightarrow \mathbb{R}$ , c.-à-d., la probabilité de l'événement que la valeur de  $X$  se trouve dans l'intervalle  $[a, b)$  est donnée par

$$P(a \leq X < b) = \int_a^b f(x) dx \tag{8.3}$$

avec  $f(x) \geq 0$  pour  $x \in \mathbb{R}$  et  $\int_{-\infty}^{\infty} f(x) dx = 1$ .

On appelle *espérance* (mathématique) de la variable aléatoire  $X$  le nombre réel

$$\mu_X = E(X) = \int_{-\infty}^{\infty} x f(x) dx, \tag{8.4}$$

et *variance* la valeur

$$\sigma_X^2 = Var(X) = \int_{-\infty}^{\infty} (x - \mu_X)^2 f(x) dx = \int_{-\infty}^{\infty} x^2 f(x) dx - \mu_X^2. \tag{8.5}$$

**Exemple 8.1** Si une variable aléatoire satisfait (8.3) avec (voir la figure IV.4)

$$f(x) = \frac{1}{\sqrt{2\pi} \cdot \sigma} \cdot \exp\left(-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right) \quad (8.6)$$

alors on dit que la variable aléatoire satisfait la *loi normale* ou la *loi de Gauss – Laplace* que l'on symbolise par  $N(\mu, \sigma^2)$ . On vérifie facilement que  $\mu$  est l'espérance et  $\sigma^2$  la variance de cette variable aléatoire.

La loi normale est parmi les plus importantes en probabilités. Une raison est due au “théorème de la limite centrale” qui implique que les observations pour la plupart des expériences physiques obéissent à cette loi.

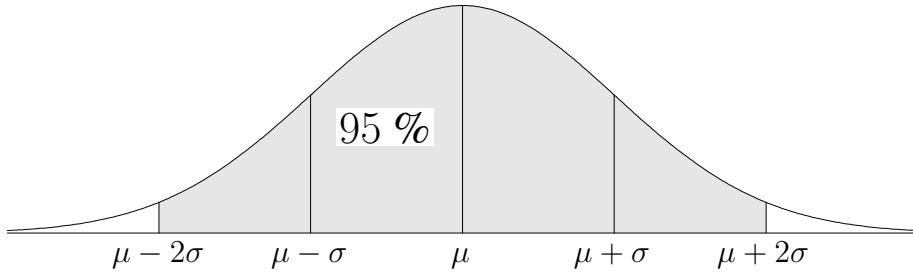


FIG. IV.4: Fonction de densité pour la loi normale

Rappelons aussi que  $n$  variables aléatoires  $X_1, \dots, X_n$  sont indépendantes si, pour tout  $a_i, b_i$ , on a

$$P(a_i \leq X_i < b_i, i = 1, \dots, n) = \prod_{i=1}^n P(a_i \leq X_i < b_i). \quad (8.7)$$

**Lemme 8.2** Soient  $X$  et  $Y$  deux variables aléatoires indépendantes avec comme fonctions de densité  $f(x)$  et  $g(y)$  respectivement et soient  $\alpha, \beta \in \mathbb{R}$  avec  $\alpha \neq 0$ . Alors, les variables aléatoires  $\alpha X + \beta$  et  $X + Y$  possèdent les fonctions de densité

$$\frac{1}{|\alpha|} f\left(\frac{x-\beta}{\alpha}\right) \quad \text{et} \quad (f * g)(z) = \int_{-\infty}^{\infty} f(z-y)g(y) dy. \quad (8.8)$$

Leur espérance mathématique est

$$E(\alpha X + \beta) = \alpha E(X) + \beta, \quad E(X + Y) = E(X) + E(Y) \quad (8.9)$$

et leur variance satisfait

$$\text{Var}(\alpha X + \beta) = \alpha^2 \text{Var}(X), \quad \text{Var}(X + Y) = \text{Var}(X) + \text{Var}(Y). \quad (8.10)$$

*Démonstration.* La fonction de densité pour la variable aléatoire  $\alpha X + \beta$  découle de (pour  $\alpha > 0$ )

$$P(a \leq \alpha X + \beta < b) = P\left(\frac{a-\beta}{\alpha} \leq X < \frac{b-\beta}{\alpha}\right) = \int_{(a-\beta)/\alpha}^{(b-\beta)/\alpha} f(x) dx = \int_a^b \alpha^{-1} f\left(\frac{t-\beta}{\alpha}\right) dt.$$

Les propriétés (8.9) et (8.10) pour  $\alpha X + \beta$  en sont une conséquence directe.

Comme  $X$  et  $Y$  sont supposées indépendantes, on obtient (en posant  $z = x + y$ )

$$P(a \leq X + Y < b) = \iint_{a \leq x+y < b} f(x)g(y) dx dy = \int_a^b \int_{-\infty}^{\infty} f(z-y)g(y) dy dz$$

et on trouve la fonction de densité pour  $X + Y$ . Un calcul direct donne

$$E(X + Y) = \int_{-\infty}^{\infty} z \int_{-\infty}^{\infty} f(z - y)g(y) dy dz = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} (x + y)f(x)g(y) dy dx = E(X) + E(Y)$$

et, de façon similaire, on obtient

$$\begin{aligned} \text{Var}(X + Y) &= \int_{-\infty}^{\infty} z^2 \int_{-\infty}^{\infty} f(z - y)g(y) dy dz - \mu_{X+Y}^2 \\ &= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} (x + y)^2 f(x)g(y) dy dx - (\mu_X + \mu_Y)^2 = \text{Var}(X) + \text{Var}(Y). \quad \square \end{aligned}$$

*Remarque.* Si  $X$  et  $Y$  sont deux variables aléatoires indépendantes qui obéissent à la loi normale, les variables aléatoires  $\alpha X + \beta$  et  $X + Y$  obéissent aussi à cette loi (exercice 16).

Revenons maintenant au problème (8.2). Pour pouvoir estimer l'erreur du résultat numérique  $x$ , faisons les hypothèses suivantes :

**H1:** La valeur  $b_i$  est la réalisation d'une épreuve pour une variable aléatoire  $B_i$ . On suppose que les  $B_i$  soient indépendantes et qu'elles obéissent à la loi de Gauss–Laplace avec  $\beta_i$  comme espérance et  $\sigma_i^2$  comme variance (les  $\beta_i$  sont inconnus, mais les  $\sigma_i^2$  sont supposés connus).

**H2:** Le système surdéterminé (8.2) possède une solution unique si l'on remplace les  $b_i$  par les nombres  $\beta_i$ , c.-à-d. qu'il existe un vecteur  $\xi \in \mathbb{R}^n$  tel que  $A\xi = \beta$  où  $\beta = (\beta_1, \dots, \beta_m)^T$ .

**Motivation de la méthode des moindres carrés.** Par l'hypothèse H1, la probabilité que  $B_i$  soit dans l'intervalle  $[b_i, b_i + db_i]$  avec  $db_i$  (infinitésimamente) petit est

$$P(b_i \leq B_i < b_i + db_i) \approx \frac{1}{\sqrt{2\pi} \cdot \sigma_i} \cdot \exp\left(-\frac{1}{2}\left(\frac{b_i - \beta_i}{\sigma_i}\right)^2\right) \cdot db_i.$$

Comme les  $B_i$  sont indépendants, la formule (8.7) implique que

$$\begin{aligned} P(b_i \leq B_i < b_i + db_i, i = 1, \dots, m) &\approx \prod_{i=1}^m \frac{1}{\sqrt{2\pi} \cdot \sigma_i} \cdot \exp\left(-\frac{1}{2}\left(\frac{b_i - \beta_i}{\sigma_i}\right)^2\right) \cdot db_i \quad (8.11) \\ &= C \cdot \exp\left(-\frac{1}{2} \sum_{i=1}^m \left(\frac{b_i - \beta_i}{\sigma_i}\right)^2\right) = C \cdot \exp\left(-\frac{1}{2} \sum_{i=1}^m \left(\frac{b_i - \sum_{j=1}^n a_{ij}\xi_j}{\sigma_i}\right)^2\right). \end{aligned}$$

Selon une idée de Gauss (1812), la “meilleure” réponse  $x_i$  pour les  $\xi_i$  (inconnus) est celle pour laquelle la probabilité (8.11) est maximale (“maximum likelihood”). Alors, on calcule  $x_1, \dots, x_n$  de façon à ce que

$$\sum_{i=1}^m \left( \frac{b_i}{\sigma_i} - \sum_{j=1}^n \frac{a_{ij}}{\sigma_i} \cdot x_j \right)^2 \rightarrow \min. \quad (8.12)$$

Si l'on remplace  $b_i/\sigma_i$  par  $b_i$  et  $a_{ij}/\sigma_i$  par  $a_{ij}$ , la condition (8.12) est équivalente à (6.2). Par la suite, nous supposerons que cette normalisation soit déjà effectuée (donc,  $\sigma_i = 1$  pour  $i = 1, \dots, n$ ).

## Estimation de l'erreur

La solution de (8.12) est donnée par  $x = (A^T A)^{-1} A^T b$ . La solution théorique satisfait  $\xi = (A^T A)^{-1} A^T \beta$ . Alors,

$$x - \xi = (A^T A)^{-1} A^T (b - \beta) \quad \text{ou} \quad x_i - \xi_i = \sum_{j=1}^m \alpha_{ij} (b_j - \beta_j)$$

où  $\alpha_{ij}$  est l'élément  $(i, j)$  de la matrice  $(A^T A)^{-1} A^T$ . L'idée est de considérer la valeur  $x_i$  comme la réalisation d'une variable aléatoire  $X_i$  définie par

$$X_i = \sum_{j=1}^m \alpha_{ij} B_j \quad \text{ou} \quad X_i - \xi_i = \sum_{j=1}^m \alpha_{ij} (B_j - \beta_j). \quad (8.13)$$

**Théorème 8.3** Soient  $B_1, \dots, B_m$  des variables aléatoires indépendantes avec  $\beta_i$  comme espérance et  $\sigma_i = 1$  comme variance. Alors, la variable aléatoire  $X_i$ , définie par (8.13), satisfait

$$E(X_i) = \xi_i \quad \text{et} \quad \text{Var}(X_i) = \epsilon_{ii} \quad (8.14)$$

où  $\epsilon_{ii}$  est le  $i^{\text{ème}}$  élément de la diagonale de  $(A^T A)^{-1}$ .

*Remarque.* Les autres éléments de  $(A^T A)^{-1}$  sont les covariances de  $X_i$  et  $X_j$ .

*Démonstration.* La formule (8.9) donne  $E(X_i) = \xi_i$ . Pour calculer la variance de  $X_i$ , nous utilisons le fait que  $\text{Var}(B_i) = 1$  et la formule (8.10). Ceci donne avec  $e_i = (0, \dots, 0, 1, 0, \dots, 0)^T$  que

$$\sigma_{X_i}^2 = \sum_{j=1}^m \alpha_{ij}^2 = \|e_i^T (A^T A)^{-1} A^T\|_2^2 = e_i^T (A^T A)^{-1} A^T A (A^T A)^{-1} e_i = e_i^T (A^T A)^{-1} e_i = \epsilon_{ii}. \quad \square$$

**Exemple 8.4** Pour l'expérience sur la thermo-électricité (voir le paragraphe IV.6), on a supposé que les mesures  $b_i$  ont été faites avec une précision correspondant à  $\sigma_i = 0.01$ . Pour le système surdéterminé (on écrit  $x_1, x_2, x_3$  pour  $a, b, c$  et  $b_i$  pour  $U_i$ )

$$\frac{1}{\sigma_i} \cdot x_1 + \frac{T_i}{\sigma_i} \cdot x_2 + \frac{T_i^2}{\sigma_i} \cdot x_3 = \frac{b_i}{\sigma_i}, \quad i = 1, \dots, 21$$

la matrice  $(A^T A)^{-1}$  devient

$$(A^T A)^{-1} = \begin{pmatrix} 0.356 \cdot 10^{-4} & -0.139 \cdot 10^{-5} & 0.113 \cdot 10^{-7} \\ -0.139 \cdot 10^{-5} & 0.765 \cdot 10^{-7} & -0.713 \cdot 10^{-9} \\ 0.113 \cdot 10^{-7} & -0.713 \cdot 10^{-9} & 0.713 \cdot 10^{-11} \end{pmatrix} \quad (8.15)$$

et on obtient

$$\sigma_{X_1} = 0.60 \cdot 10^{-2}, \quad \sigma_{X_2} = 0.28 \cdot 10^{-3}, \quad \sigma_{X_3} = 0.27 \cdot 10^{-5}.$$

Ceci implique qu'avec une probabilité de 95%, la solution exacte (si elle existe) satisfait

$$a = -0.886 \pm 0.012, \quad b = 0.0352 \pm 0.0006, \quad c = 0.598 \cdot 10^{-4} \pm 0.054 \cdot 10^{-4}.$$

## Test de confiance du modèle

Etudions encore si les données  $(t_i, b_i)$  sont compatibles avec la loi (8.1). Ceci revient à justifier l'hypothèse H2.

En utilisant la décomposition QR de la matrice  $A$ , le problème surdéterminé  $Ax = b$  se transforme en (voir (7.1))

$$\begin{pmatrix} R' \\ 0 \end{pmatrix} x = \begin{pmatrix} c' \\ c'' \end{pmatrix} \quad \text{où} \quad \begin{pmatrix} c' \\ c'' \end{pmatrix} = Q^T b. \quad (8.16)$$

La grandeur de  $\|c''\|_2^2$  est une mesure de la qualité du résultat numérique. Théoriquement, si l'on a  $\beta$  à la place de  $b$  et  $\xi$  à la place de  $x$ , cette valeur est nulle.

Notons les éléments de la matrice  $Q$  par  $q_{ij}$ . Alors, les éléments du vecteur  $c = Q^T b$  sont donnés par  $c_i = \sum_{j=1}^m q_{ji} b_j$  et ceux du vecteur  $c''$  satisfont aussi  $c_i = \sum_{j=1}^m q_{ji} (b_j - \beta_j)$ . Il est alors naturel de considérer les variables aléatoires

$$C_i = \sum_{j=1}^m q_{ji} (B_j - \beta_j), \quad i = n+1, \dots, m. \quad (8.17)$$

Le but est d'étudier la fonction de densité de  $\sum_{i=n+1}^m C_i^2$ .

**Lemme 8.5** Soient  $B_1, \dots, B_m$  des variables aléatoires indépendantes satisfaisant la loi normale  $N(\beta_i, 1)$ . Alors, les variables aléatoires  $C_{n+1}, \dots, C_m$ , définies par (8.17), sont indépendantes et satisfont aussi la loi normale avec

$$E(C_i) = 0, \quad \text{Var}(C_i) = 1. \quad (8.18)$$

*Démonstration.* Pour voir que les  $C_i$  sont indépendants, calculons la probabilité  $P(a_i \leq C_i < b_i, i = n+1, \dots, m)$ . Notons par  $S$  l'ensemble  $S = \{y \in \mathbb{R}^m \mid a_i \leq y_i < b_i, i = n+1, \dots, m\}$  et par  $C$  et  $B$  les vecteurs  $(C_1, \dots, C_m)^T$  et  $(B_1, \dots, B_m)^T$ . Alors, on a

$$\begin{aligned} P(a_i \leq C_i < b_i, i = n+1, \dots, m) &= P(C \in S) = P(Q^T(B - \beta) \in S) \\ &= P(B - \beta \in Q(S)) \stackrel{(a)}{=} \iint_{Q(S)} \frac{1}{(\sqrt{2\pi})^m} \exp\left(-\frac{1}{2} \sum_{i=1}^m y_i^2\right) dy_1 \dots dy_m \\ &\stackrel{(b)}{=} \iint_S \frac{1}{(\sqrt{2\pi})^m} \exp\left(-\frac{1}{2} \sum_{i=1}^m z_i^2\right) dz_1 \dots dz_m = \prod_{i=n+1}^m \int_{a_i}^{b_i} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{z_i^2}{2}\right) dz_i. \end{aligned} \quad (8.19)$$

L'identité (a) est une conséquence de l'indépendance des  $B_i$  et (b) découle de la transformation  $y = Qz$ , car  $\det Q = 1$  et  $\sum_i y_i^2 = \sum_i z_i^2$  (la matrice  $Q$  est orthogonale). En utilisant  $S_i = \{y \in \mathbb{R}^m \mid a_i \leq y_i < b_i\}$ , on déduit de la même manière que

$$P(a_i \leq C_i < b_i) = P(C \in S_i) = \dots = \int_{a_i}^{b_i} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{z_i^2}{2}\right) dz_i. \quad (8.20)$$

Une comparaison de (8.19) avec (8.20) démontre l'indépendance de  $C_{n+1}, \dots, C_m$  (voir la définition (8.7)). Le fait que les  $C_i$  satisfont la loi normale  $N(0, 1)$  est une conséquence de (8.20).  $\square$

**Théorème 8.6 (Pearson)** Soient  $Y_1, \dots, Y_n$  des variables aléatoires indépendantes qui obéissent à la loi normale  $N(0, 1)$ . Alors, la fonction de densité de la variable aléatoire

$$Y_1^2 + Y_2^2 + \dots + Y_n^2 \quad (8.21)$$

est donnée par (voir figure IV.5)

$$f_n(x) = \frac{1}{2^{n/2} \cdot \Gamma(n/2)} \cdot x^{n/2-1} \cdot e^{-x/2} \quad (8.22)$$

pour  $x > 0$  et par  $f_n(x) = 0$  pour  $x \leq 0$  ("loi de  $\chi^2$  à  $n$  degrés de liberté"). L'espérance de cette variable aléatoire vaut  $n$  et sa variance  $2n$ .

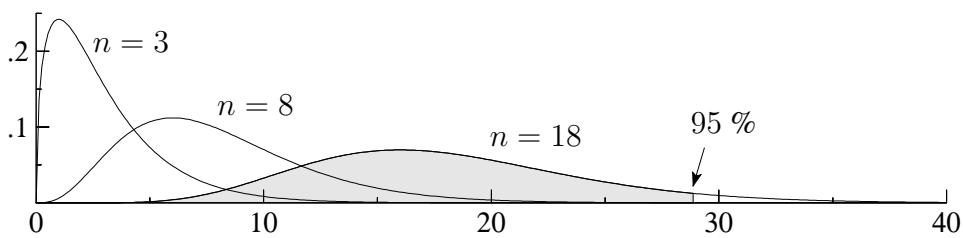


FIG. IV.5: Fonction de densité (8.22)

*Démonstration.* Considérons d'abord le cas  $n = 1$ . Pour  $0 \leq a < b$ , on a

$$\begin{aligned} P(a \leq Y_1^2 < b) &= P(\sqrt{a} \leq Y_1 < \sqrt{b}) + P(-\sqrt{a} \geq Y_1 > -\sqrt{b}) \\ &= 2 \int_{\sqrt{a}}^{\sqrt{b}} \frac{1}{\sqrt{2\pi}} \cdot e^{-x^2/2} dx = \int_a^b \frac{1}{\sqrt{2\pi}} \cdot e^{-t/2} \cdot \frac{dt}{\sqrt{t}}, \end{aligned}$$

ce qui démontre (8.22) pour  $n = 1$  car  $\Gamma(1/2) = \sqrt{\pi}$ .

Pour le cas général, nous procédons par récurrence. Nous utilisons le résultat du Lemme 8.2 qui affirme que la fonction de densité de  $Y_1^2 + \dots + Y_{n+1}^2$  est la convolution de celle de  $Y_1^2 + \dots + Y_n^2$  avec celle de  $Y_{n+1}^2$ . Le calcul

$$\begin{aligned}(f_n * f_1)(x) &= \frac{1}{\sqrt{2} \cdot \Gamma(1/2) \cdot 2^{n/2} \cdot \Gamma(n/2)} \int_0^x (x-t)^{-1/2} e^{-(x-t)/2} t^{n/2-1} e^{-t/2} dt \\ &= \frac{e^{-x/2}}{\sqrt{2} \cdot \Gamma(1/2) \cdot 2^{n/2} \cdot \Gamma(n/2)} \int_0^x (x-t)^{-1/2} t^{n/2-1} dt \\ &= \frac{x^{(n+1)/2-1} e^{-x/2}}{\sqrt{2} \cdot \Gamma(1/2) \cdot 2^{n/2} \cdot \Gamma(n/2)} \int_0^1 (1-s)^{-1/2} s^{n/2-1} ds = f_{n+1}(x)\end{aligned}$$

nous permet de conclure.  $\square$

Pour les variables aléatoires  $C_i$  de (8.17), ce théorème montre que

$$\sum_{i=n+1}^m C_i^2 \tag{8.23}$$

est une variable aléatoire ayant comme fonction de densité  $f_{m-n}(x)$  (on rappelle qu'après normalisation, on a  $\sigma_i = 1$  pour les variables aléatoires  $B_i$ ).

Appliquons ce résultat à l'exemple du paragraphe IV.6 (voir la formulation (8.12)). Dans ce cas, on a  $\|c''\|_2^2 = 25.2$  et  $m - n = 18$  degrés de liberté. La figure IV.5 montre que cette valeur de  $\|c''\|_2^2$  est suffisamment petite pour être probable.

Si l'on avait travaillé avec le modèle plus simple

$$U = a + bT \tag{8.24}$$

(à la place de (6.4)) on aurait trouvé  $\|c''\|_2^2 = 526.3$  et  $m - n = 19$ . Cette valeur est trop grande pour être probable. La conclusion est que, pour les données du tableau IV.2, la loi (8.24) est à rejeter sur la base de ces mesures.

## IV.9 Exercices

1. Pour calculer l'inverse d'une matrice dont la dimension  $n$  est très grande, il existe un algorithme qui exige environ  $n^3$  opérations. Donner cet algorithme.
2. Supposons que la décomposition  $LR$  de la matrice  $A$  est à disposition. Pour  $u, v, b$  des vecteurs donnés, trouver un algorithme efficace pour résoudre le système

$$(A + uv^T)x = b$$

qui utilise uniquement la résolution des systèmes  $A^{-1}b$  et  $A^{-1}u$ . Cet algorithme est connu sous la formule de Sherman - Morrison - Woodbury.

*Indication.* Calculer d'abord une formule pour  $v^T x$ .

3. Considérons le problème de calculer le produit scalaire

$$\langle x, y \rangle = \sum_{i=1}^n x_i y_i .$$

Quelle est sa condition?

4. Soit  $A$  une matrice  $n \times m$  à coefficients réels. Montrer que

$$\|A\|_1 = \|A^T\|_\infty \quad \text{et} \quad \|A\|_2 = \|A^T\|_2.$$

5. Considérons une matrice-bande avec une largeur inférieure  $p_\ell$  et une largeur supérieure  $p_u$  (c'est-à-dire,  $a_{ij} = 0$  si  $i - j > p_\ell$  et si  $j - i > p_u$ ). Montrer que les matrices  $L$  et  $R$  de la décomposition  $LR$  avec et sans la recherche de pivot ont aussi une structure de bande. Pour le cas tridiagonal,  $p_\ell = p_u = 1$ , donner les largeurs des bandes apparaissant dans les décompositions et estimer le coût en opérations des algorithmes.

6. Pour résoudre le système linéaire

$$\sum_{j=1}^n c_j^{i-1} x_j = b_i, \quad i = 1, \dots, n \tag{9.1}$$

(matrice du type Vandermonde), dériver un algorithme qui nécessite seulement  $\mathcal{O}(n^2)$  opérations.

*Indications.*

- (a) Le système (9.1) est équivalent à

$$\sum_{j=1}^n p(c_j) x_j = b(p) \quad \text{pour } \deg p \leq n-1,$$

$$\text{où } b(p) = \sum_{j=1}^n d_j b_j \text{ et } p(i) = \sum_{j=1}^n d_j i^{j-1}.$$

- (b) Choisir pour  $p(t)$  les éléments de la base  $1, t - c_1, (t - c_1)(t - c_2), (t - c_1)(t - c_2)(t - c_3), \dots$

7. (a) Pour la matrice

$$A = \begin{pmatrix} 1 & 0 \\ -1 & 4 \\ 4 & 1 \end{pmatrix}$$

calculer  $\|A\|_1$ ,  $\|A\|_2$  et  $\|A\|_\infty$ .

- (b) Démontrer que pour des matrices symétriques nous avons toujours  $\|A\|_2 \leq \|A\|_1$ .

8. Les valeurs de la suite  $b_k = \exp(k^{2/3} - (k-1)^{2/3})$  peuvent être calculées par les formules:

$$\begin{aligned} b_k &= \exp(k^{2/3} - (k-1)^{2/3}), \\ b_k &= \exp(k^{2/3}) / \exp((k-1)^{2/3})), \\ b_k &= \exp((2k-1)/(k^{4/3} + (k(k-1))^{2/3} + (k-1)^{4/3})). \end{aligned}$$

Calculer à l'aide d'une calculatrice la valeur pour  $k = 100000$  (le résultat est  $b_{100000} = 1.01446656424210809769528199600$ ) et pour  $k$  grand, quelle formule est préférable pour un calcul en virgule flottante?

9. Les racines du polynôme  $x^2 - 2px - q = 0$  peuvent être calculées par

$$\lambda_1 = p + \sqrt{p^2 + q}, \quad \lambda_2 = p - \sqrt{p^2 + q}.$$

Montrer que pour  $p > 0$  (grand) et  $q > 0$  (très petit) cet algorithme est numériquement instable. A l'aide de la relation  $\lambda_1 \lambda_2 = -q$ , trouver un algorithme qui est numériquement stable.

10. Soient donnés  $x_1, x_2, \dots, x_n$ . Une estimation de la variance peut être calculée par chacune des deux formules suivantes:

$$\sigma^2 = \frac{1}{n-1} \left( \sum_{i=1}^n x_i^2 - n\mu^2 \right) \quad \sigma^2 = \frac{1}{n-1} \sum_{i=1}^n (x_i - \mu)^2$$

où  $\mu = \frac{1}{n} \sum_{i=1}^n x_i$  est l'espérance. Quelle formule est la plus stable?

a) Appliquer les deux algorithmes à l'exemple  $n = 2$ ,  $x_1 = 3001$ ,  $x_2 = 3003$  et simuler un calcul en virgule flottante avec 4 chiffres.

b) Etudier l'influence des erreurs d'arrondi pour les deux algorithmes, si  $n = 2$  mais que  $x_1$  et  $x_2$  sont arbitraires.

11. a) Calculer la décomposition de Cholesky  $A = LL^T$  pour la matrice de Hilbert

$$A = \left( \frac{1}{i+j-1} \right)_{i,j=1,\dots,n}, \quad n = 3, 6, 9, 12, 15.$$

b) Comparer le résultat numérique avec les valeurs exactes

$$\ell_{jk} = \frac{\sqrt{2k-1} \cdot (j-1)! \cdot (j-1)!}{(j-k)! \cdot (j+k-1)!}. \quad (9.2)$$

Combien de chiffres sont exacts?

c) Si  $\hat{L}$  dénote le résultat numérique, calculer le résidu  $A - \hat{L}\hat{L}^T$ .

Calculer aussi le résidu  $A - LL^T$  pour la matrice  $L$ , donnée par (9.2).

12. Pour une matrice  $A = (a_{ij})$  notons par  $(a_{ij}^{(k)})$  les matrices des étapes intermédiaires de l'élimination de Gauss. Montrer qu'avec une recherche de pivot partielle, on a

$$\max_{i,j,k} |a_{ij}^{(k)}| \leq 2^{n-1} \cdot \max_{i,j} |a_{ij}|. \quad (9.3)$$

Pour la matrice suivante, on a égalité dans la formule (9.3):

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

13. Soit  $A$  une matrice à  $m$  lignes et  $n$  colonnes ( $m \geq n$ ). On définit pour les matrices non-carrées,

$$\kappa(A) := \max_{\|x\|=1} \|Ax\| / \min_{\|y\|=1} \|Ay\|.$$

Pour la norme Euclidienne, montrer que  $\kappa_2(A^T A) = (\kappa_2(A))^2$ .

*Indication.* Transformer la matrice symétrique  $A^T A$  sous forme diagonale  $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_n \geq 0$  et montrer que

$$\max_{\|x\|_2=1} \|Ax\|_2^2 = \lambda_1, \quad \min_{\|x\|_2=1} \|Ax\|_2^2 = \lambda_n.$$

14. Voici quelques valeurs pour la densité  $\varrho$  de l'eau en fonction de sa température  $T$ .

$T [^\circ C]$	0	5	10	15	20
$\varrho(T)$	0.999868	0.999992	0.999728	0.999126	0.998232

- (a) Approcher ces valeurs par un polynôme de degré 2 (méthode des moindres carrés).
- (b) Pour quelle valeur de  $T$ , la densité est-elle maximale et quelle est cette valeur maximale?

*Indication.* Si vous préférez calculer avec des nombres plus petits, faites la transformation  $x = T/5$ ,  $f(T) = 1 - \varrho(T)$ .

15. Soit  $A$  une matrice inversible de dimension  $n$ . Montrer que la décomposition  $QR$  (où  $Q$  est orthogonale et  $R$  triangulaire supérieure) est unique, si l'on suppose que  $r_{ii} > 0$  pour  $i = 1, \dots, n$ .
16. Soient  $X$  et  $Y$  deux variables aléatoires indépendantes obéissant à la loi normale  $N(\mu_1, \sigma_1)$  et  $N(\mu_2, \sigma_2)$  respectivement. Montrer que  $\alpha X + \beta$  (pour  $\alpha > 0$ ) et  $X + Y$  obéissent aussi à cette loi.
17. Soit  $X$  une variable aléatoire qui obéit à la loi  $\chi^2$  avec  $n$  degrés de liberté (c.-à-d.,  $f_n(x)$  de (8.22) est sa fonction de densité). Montrer que

$$E(X) = n \quad \text{et} \quad \text{Var}(X) = 2n.$$

18. Effectuer une étude complète de l'erreur du modèle trouvé à l'exercice 14. Pour cela, trouver les écarts types des coefficients du polynôme et effectuer un test de confiance du modèle.

*Indication.*  $\|c''\|^2 = \|Ax - b\|^2$ .

19. Les éléments de la diagonale de  $C = (A^T A)^{-1}$  jouent un rôle important pour l'étude de l'erreur de la méthode des moindres carrés. Supposons que nous avons à disposition la décomposition  $QR$  de la matrice  $A$ .
  - (a) Démontrer que  $C = (R^T R)^{-1}$ .
  - (b) Trouver un algorithme pour calculer la diagonale de  $C$  en  $n^3/6$  opérations (  $n$  = nombre de colonnes de  $A$ ; 1 opération = 1 multiplication + 1 addition).

# Chapitre V

## Valeurs et Vecteurs Propres

Les premiers vecteurs et valeurs propres viennent des équations différentielles (Lagrange 1759, *théorie du son*; Lagrange 1781, des matrices  $6 \times 6$  dans le but de calculer les perturbations séculaires des orbites des 6 planètes connues à l'époque, *Oeuvres* V, p. 125-490). Aujourd'hui, le calcul des valeurs et vecteurs propres est indispensable dans toutes les branches de la science, en particulier pour la solution des systèmes des équations différentielles linéaires, en théorie de stabilité, pour les questions de convergence de processus itératifs, et en physique et chimie (mécanique, circuits, cinétique chimique, équation de Schrödinger).

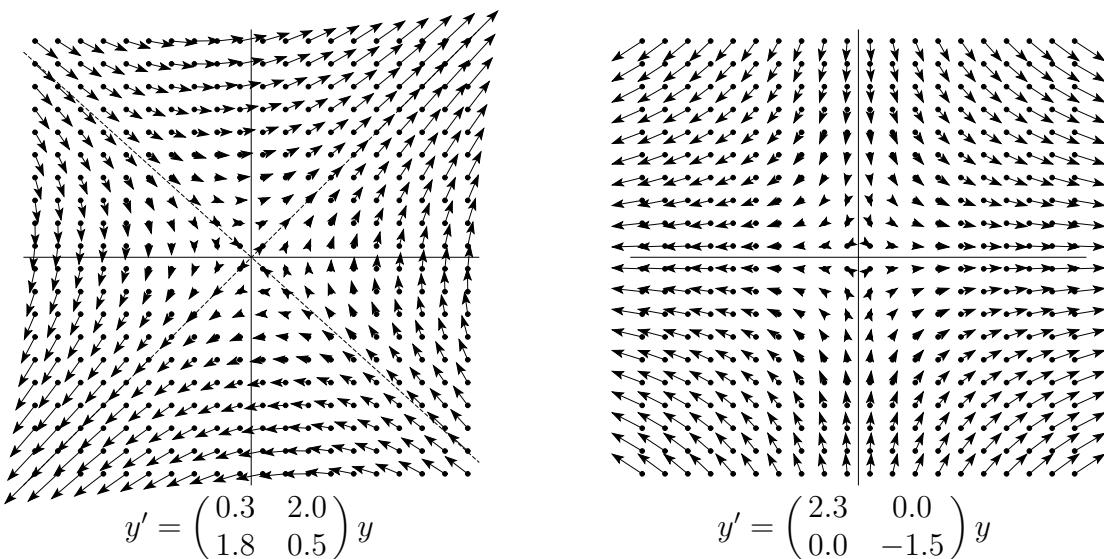


FIG. V.1: Une application linéaire comme champ de vecteurs (à gauche); transformée sur la base des vecteurs propres (à droite).

Observons en figure V.1 (à gauche) le champ de vecteurs d'une équation différentielle  $y' = Ay$ . Deux directions sautent aux yeux : ce sont les directions où le vecteur  $Av$  prend la même direction que le vecteur  $v$ , i.e., où

$$Av = \lambda v \quad \text{ou} \quad (A - \lambda I)v = 0. \quad (0.1)$$

Si cette équation est vérifiée,  $\lambda \in \mathbb{C}$  s'appelle *valeur propre* de la matrice  $A$  et  $v \in \mathbb{C}^n$  ( $v \neq 0$ ) est le *vecteur propre* correspondant. L'équation (0.1) possède une solution  $v$  non nulle si et seulement si

$$\chi_A(\lambda) = \det(A - \lambda I) = 0. \quad (0.2)$$

Le polynôme  $\chi_A(\lambda)$  est le *polynôme caractéristique* de la matrice  $A$ . Les valeurs propres de  $A$  sont alors les zéros du polynôme caractéristique.

### Bibliographie sur ce chapitre

Tous les livres cités dans le chapitre IV, et en plus . . .

B.N. Parlett (1980): *The Symmetric Eigenvalue Problem*. Prentice-Hall, Englewood Cliffs, NJ.

B.T. Smith, J.M. Boyle, Y. Ikebe, V.C. Klema & C.B. Moler (1970): *Matrix Eigensystem Routines: EISPACK Guide*. 2nd ed., Springer-Verlag, New York.

J.H. Wilkinson (1965): *The Algebraic Eigenvalue Problem*. Clarendon Press. [MA 65/72]

J.H. Wilkinson & C. Reinsch (1971): *Handbook for Automatic Computation, Volume II, Linear Algebra*. Springer-Verlag, New York.

## V.1 La condition du calcul des valeurs propres

A cause des erreurs d'arrondi, les éléments d'une matrice  $A$ , pour laquelle on cherche les valeurs propres, ne sont pas exacts. Ils sont plutôt égaux à  $\hat{a}_{ij} = a_{ij}(1 + \epsilon_{ij})$  avec  $|\epsilon_{ij}| \leq \text{eps}$  ( $\text{eps}$ , la précision de l'ordinateur, est supposée être très petite). Il est alors très 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

$$A(\epsilon) = A + \epsilon C \quad \text{où} \quad |\epsilon| \leq \text{eps} \quad \text{et} \quad |c_{ij}| \leq |a_{ij}| \quad (1.1)$$

(souvent, la dernière hypothèse va être remplacée par  $\|C\| \leq \|A\|$ ).

**Théorème 1.1 (Gershgorine)** Soit  $A$  une matrice  $n \times n$  (avec des éléments dans  $\mathbb{R}$  ou dans  $\mathbb{C}$ ).

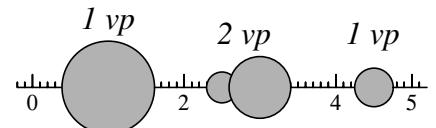
a) Si  $\lambda$  est une valeur propre de  $A$ , alors il existe un indice  $i$  tel que

$$|\lambda - a_{ii}| \leq \sum_{\substack{j=1 \\ j \neq i}}^n |a_{ij}|, \quad (1.2)$$

c.-à-d. que toutes les valeurs propres de  $A$  se trouvent dans

l'union des disques

$$D_i = \{\lambda ; |\lambda - a_{ii}| \leq \sum_{j \neq i} |a_{ij}|\}.$$



b) Si une composante connexe de  $\bigcup_{i=1}^n D_i$  consiste de  $k$  disques, elle contient exactement  $k$  valeurs propres de  $A$ .

*Démonstration.* Soit  $v \neq 0$  un vecteur propre et choisissons l'indice  $i$  tel que  $|v_i| \geq |v_j|$  pour tout  $j$ . La ligne  $i$  de l'équation  $Av = \lambda v$  donne

$$\sum_{j \neq i} a_{ij} v_j = (\lambda - a_{ii}) v_i.$$

En divisant par  $v_i$  et en utilisant l'inégalité de triangle, on obtient

$$|\lambda - a_{ii}| = \left| \sum_{j \neq i} a_{ij} \cdot \frac{v_j}{v_i} \right| \leq \sum_{j \neq i} |a_{ij}|.$$

L'affirmation (b) est vraie si  $A$  est une matrice diagonale. Le cas général est obtenu par un argument de continuité en faisant tendre les éléments en dehors de la diagonale vers zéro (voir le cours “Analyse II” concernant la dépendance continue de racines d'un polynôme en fonction d'un paramètre).  $\square$

**Théorème 1.2** Soit  $A$  une matrice diagonalisable, i.e., il existe  $T$  avec  $T^{-1}AT = \text{diag}(\lambda_1, \dots, \lambda_n)$ , et soit  $A(\epsilon) = A + \epsilon C$ . Alors, pour chaque valeur propre  $\lambda(\epsilon)$  de  $A(\epsilon)$ , il existe un  $\lambda_i$  avec

$$|\lambda(\epsilon) - \lambda_i| \leq |\epsilon| \cdot \kappa_\infty(T) \cdot \|C\|_\infty.$$

*Démonstration.* Nous transformons la matrice  $A(\epsilon) = A + \epsilon C$  par la même matrice, qui transforme  $A$  sous forme diagonale :

$$T^{-1}A(\epsilon)T = \text{diag}(\lambda_1, \dots, \lambda_n) + \epsilon T^{-1}CT.$$

Si l'on dénote par  $e_{ij}$  les éléments de  $T^{-1}CT$ , le théorème de Gershgorine implique l'existence d'un indice  $i$  tel que  $|\lambda(\epsilon) - (\lambda_i + \epsilon e_{ii})| \leq |\epsilon| \sum_{j \neq i} |e_{ij}|$ . L'inégalité de triangle donne alors

$$|\lambda(\epsilon) - \lambda_i| \leq |\epsilon| \cdot \max_i \left( \sum_j |e_{ij}| \right) \leq |\epsilon| \cdot \|T^{-1}CT\|_\infty \leq |\epsilon| \cdot \|T^{-1}\|_\infty \|C\|_\infty \|T\|_\infty,$$

ce qui démontre l'affirmation du théorème, car  $\kappa_\infty(T) = \|T\|_\infty \|T^{-1}\|_\infty$  (condition de  $T$ ).  $\square$

La condition du calcul des valeurs propres depend de la condition de la matrice de transformation  $T$ . Si la matrice  $A$  est symétrique ( $T$  est orthogonale), le problème est bien conditionné. Toutefois, observons qu'on obtient seulement une estimation pour l'*erreur absolue* et non pour l'*erreur relative*.

**Théorème 1.3 (différentiabilité des valeurs propres)** Soit  $\lambda_1$  une racine simple de  $\chi_A(\lambda) = 0$ . Alors, pour  $|\epsilon|$  suffisamment petit, la matrice  $A(\epsilon) = A + \epsilon C$  possède une valeur propre unique  $\lambda_1(\epsilon)$  proche de  $\lambda_1$ . La fonction  $\lambda_1(\epsilon)$  est différentiable (même analytique) et on a

$$\lambda_1(\epsilon) = \lambda_1 + \epsilon \cdot \frac{u_1^* Cv_1}{u_1^* v_1} + \mathcal{O}(\epsilon^2) \quad (1.3)$$

où  $v_1$  est le vecteur propre à droite ( $Av_1 = \lambda_1 v_1$ ) et  $u_1$  est le vecteur propre à gauche ( $u_1^* A = \lambda_1 u_1^*$ ). On peut supposer que  $\|v_1\| = \|u_1\| = 1$ .

*Démonstration.* Soit  $p(\lambda, \epsilon) := \chi_{A+\epsilon C}(\lambda) = \det(A + \epsilon C - \lambda I)$ . Comme

$$p(\lambda_1, 0) = 0 \quad \text{et} \quad \frac{\partial p}{\partial \lambda}(\lambda_1, 0) \neq 0,$$

le théorème des fonctions implicites garantie l'existence d'une fonction différentiable  $\lambda_1(\epsilon)$  (même analytique), tel que  $\lambda_1(0) = \lambda_1$  et  $p(\lambda_1(\epsilon), \epsilon) = 0$ . Il existe donc un vecteur  $v_1(\epsilon) \neq 0$  tel que

$$(A(\epsilon) - \lambda_1(\epsilon)I)v_1(\epsilon) = 0. \quad (1.4)$$

La matrice dans (1.4) étant de rang  $n - 1$ , on peut fixer une composante à 1 et appliquer la règle de Cramer. Ceci montre que les autres composantes sont des fonctions rationnelles des éléments de la matrice  $A + \epsilon C - \lambda_1(\epsilon)I$  et donc différentiables. Après la normalisation à  $v_1(\epsilon)^T v_1(\epsilon) = 1$ , la fonction  $v_1(\epsilon)$  reste différentiable.

Pour calculer  $\lambda'_1(0)$ , nous pouvons dériver l'équation (1.4) par rapport à  $\epsilon$  et poser ensuite  $\epsilon = 0$ . Ceci donne

$$(A - \lambda_1 I)v'_1(0) + (C - \lambda'_1(0)I)v_1 = 0. \quad (1.5)$$

En multipliant cette relation par  $u_1^*$ , on obtient  $u_1^*(C - \lambda'_1(0)I)v_1 = 0$ , ce qui permet de calculer  $\lambda'_1(0)$  et démontre la formule (1.3).  $\square$

**Conséquences.** La formule (1.3) du théorème précédent montre que le plus le vecteur propre de droite est parallèle au vecteur propre de gauche, le mieux la valeur propre correspondante est *bien conditionnée* (par exemple, pour les matrices symétriques les deux vecteurs sont identiques) ; le plus ils se rapprochent de l'orthogonalité, le plus la valeur propre est *mal conditionnée*.

Si la matrice n'est pas symétrique (ou normale), le calcul de  $\lambda_1$  (valeur propre simple) peut être mal conditionné. Considérons par exemple la matrice

$$A = \begin{pmatrix} 1 & \alpha \\ 0 & 2 \end{pmatrix} \quad \text{où} \quad v_1 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad u_1 = \frac{1}{\sqrt{1+\alpha^2}} \begin{pmatrix} 1 \\ -\alpha \end{pmatrix}.$$

Dans cette situation, la formule (1.3) nous donne  $\lambda_1(\epsilon) - \lambda_1 = \epsilon \cdot (c_{11} - \alpha c_{21}) + \mathcal{O}(\epsilon^2)$  et le calcul de  $\lambda_1 = 1$  est mal conditionné si  $\alpha$  est grand.

**Exemple 1.4** Considérons la matrice (boîte de Jordan)

$$A = \left( \begin{array}{cccc} \lambda_1 & 1 & & \\ & \lambda_1 & \ddots & \\ & & \ddots & 1 \\ & & & \lambda_1 \end{array} \right) \Bigg\} n \quad (1.6)$$

Le polynôme caractéristique de  $A + \epsilon C$  satisfait

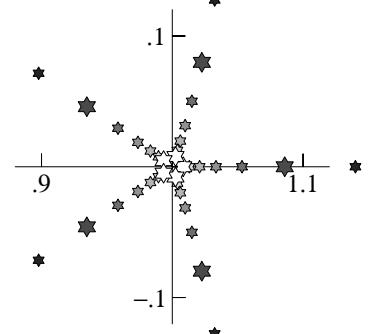
$$\det(A + \epsilon C - \lambda I) = (\lambda_1 - \lambda)^n - (-1)^n \cdot \epsilon \cdot c_{n1} + \mathcal{O}(\epsilon^2) + \mathcal{O}(\epsilon \cdot |\lambda_1 - \lambda|).$$

Si  $c_{n1} \neq 0$ , les termes  $\mathcal{O}(\epsilon^2)$  et  $\mathcal{O}(\epsilon \cdot |\lambda_1 - \lambda|)$  sont négligeables par rapport à  $\epsilon \cdot c_{n1}$ . Les valeurs propres de  $A + \epsilon C$  sont alors approximativement données par les racines de

$$(\lambda_1 - \lambda)^n - (-1)^n \cdot \epsilon \cdot c_{n1} = 0, \quad \text{c.-à-d.} \quad \lambda \approx \lambda_1 + (\epsilon \cdot c_{n1})^{1/n} \quad (1.7)$$

(observer que  $(\epsilon \cdot c_{n1})^{1/n}$  donne  $n$  valeurs complexes distinctes – multiples des racines de l'unité).

**Expérience numérique.** Prenons la matrice (1.6) avec  $\lambda_1 = 1$  et  $n = 5$ . Les éléments de la matrice  $C$  sont des nombres aléatoires dans l'intervalle  $[-1, 1]$ . Le dessin ci-contre montre les 5 valeurs propres de  $A + \epsilon C$  pour  $\epsilon = 10^{-4}, 10^{-5}, \dots, 10^{-10}$ . L'erreur est  $\approx 10^{-1}$  pour  $\epsilon = 10^{-5}$  et  $\approx 10^{-2}$  pour  $\epsilon = 10^{-10}$ , ce qui correspond à la formule (1.7) pour  $n = 5$ .



**Conséquence.** Si la dimension  $n$  d'une boîte de Jordan est plus grande que 1, le calcul de la valeur propre de cette matrice est très mal conditionné.

## Condition du calcul des vecteurs propres

Considérons la situation où toutes les valeurs propres de  $A$  sont distinctes. La démonstration du théorème sur la différentiabilité des valeurs propres montre (voir formule (1.4)) que les vecteurs propres normalisés  $v_i(\epsilon)$  de  $A + \epsilon C$  sont des fonctions différentiables de  $\epsilon$ . Pour étudier la condition du calcul des vecteurs propres, nous exprimons  $v'_1(0)$  dans la base des vecteurs propres (de droite)

$$v'_1(0) = \sum_{i=1}^n \alpha_i v_i. \quad (1.8)$$

La formule (1.5) donne alors

$$\sum_{j=2}^n (\lambda_j - \lambda_1) \alpha_j v_j + (C - \lambda'_1(0)I)v_1 = 0. \quad (1.9)$$

En multipliant (1.9) par le vecteur propre de gauche  $u_i^*$  (observer que  $u_i^* v_j = 0$  pour  $i \neq j$ ), on obtient  $\alpha_i$  (pour  $i \geq 2$ ) de la relation  $(\lambda_i - \lambda_1)\alpha_i u_i^* v_i + u_i^* Cv_1 = 0$ . La normalisation  $\|v_1(\epsilon)\|_2^2 = 1$  donne (en la dérivant)  $v_1^* v_1'(0) = 0$  et on en déduit que  $\alpha_1 = -\sum_{i=2}^n \alpha_i v_1^* v_i$ . Si l'on insère les formules pour  $\alpha_i$  dans (1.8), on obtient pour  $v_1(\epsilon) = v_1 + \epsilon v_1'(0) + \mathcal{O}(\epsilon^2)$  la relation

$$v_1(\epsilon) = v_1 + \epsilon \sum_{i=2}^n \frac{u_i^* Cv_1}{(\lambda_1 - \lambda_i) u_i^* v_i} (v_i - v_1 v_1^* v_i) + \mathcal{O}(\epsilon^2). \quad (1.10)$$

De cette formule, on voit que la condition du calcul du vecteur propre  $v_1$  dépend de la grandeur  $u_i^* v_i$  (comme c'est le cas pour la valeur propre; voir la formule (1.3)) et aussi de la distance entre  $\lambda_1$  et les autres valeurs propres de  $A$ .

### Un algorithme dangereux

La première méthode (déjà utilisée par Lagrange) pour calculer les valeurs propres d'une matrice  $A$  est la suivante: *calculer d'abord les coefficients du polynôme caractéristique  $\chi_A(\lambda)$  et déterminer ensuite les zéros de ce polynôme.* Si la dimension de  $A$  est très petite (disons  $n \leq 3$ ) ou si l'on fait le calcul en arithmétique exacte, cet algorithme peut être très utile. Par contre, si l'on fait le calcul en virgule flottante, cet algorithme peut donner des mauvaises surprises.

Considérons, par exemple, le problème de calculer les valeurs propres de la matrice diagonale

$$A = \text{diag}(1, 2, 3, \dots, n) \quad (1.11)$$

dont le polynôme caractéristique est

$$\begin{aligned} \chi_A(\lambda) &= (1 - \lambda)(2 - \lambda)(3 - \lambda) \cdots (n - \lambda) \\ &= (-1)^n \lambda^n + a_{n-1} \lambda^{n-1} + a_{n-2} \lambda^{n-2} + \dots + a_1 \lambda + a_0. \end{aligned} \quad (1.12)$$

Les coefficients calculés satisfont  $\hat{a}_i = a_i(1 + \epsilon_i)$  avec  $|\epsilon_i| \leq \text{eps}$ . Cette perturbation dans les coefficients provoque une grande erreur dans les zéros de (1.12). Les résultats numériques pour  $n = 9, 11, 13, 15$  (avec  $\text{eps} \approx 6 \cdot 10^{-8}$ , simple précision) sont dessinés dans la figure V.2.

**Conclusion.** Eviter le calcul des coefficients du polynôme caractéristique. Un tel algorithme est numériquement instable.

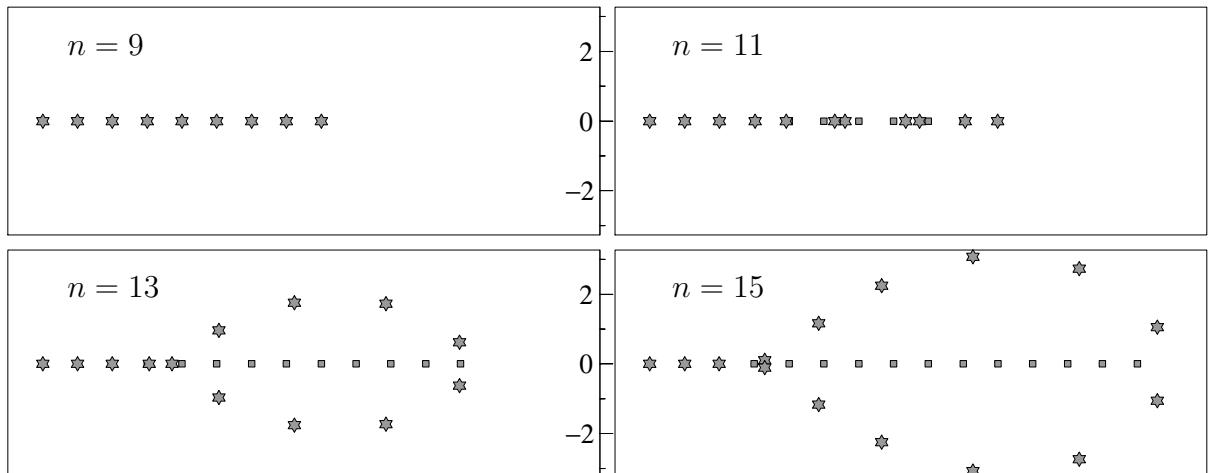


FIG. V.2: Zéros de (1.12) avec des coefficients perturbés

## V.2 La méthode de la puissance

Un algorithme simple pour calculer les valeurs propres d'une matrice  $A$  est basé sur l'itération

$$y_{k+1} = Ay_k \quad (2.1)$$

où  $y_0$  est un vecteur arbitraire. Dans le théorème suivant, on démontre que  $y_k = A^k y_0$  (méthode de la puissance) tend vers un vecteur propre de  $A$  et que le *quotient de Rayleigh*  $y_k^* A y_k / y_k^* y_k$  est une approximation d'une valeur propre de  $A$ .

**Théorème 2.1** Soit  $A$  une matrice diagonalisable de valeurs propres  $\lambda_1, \dots, \lambda_n$  et de vecteurs propres  $v_1, \dots, v_n$  (normalisés par  $\|v_i\|_2 = 1$ ).

Si  $|\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq \dots \geq |\lambda_n|$ , les vecteurs  $y_k$  de l'itération (2.1) vérifient

$$y_k = \lambda_1^k (a_1 v_1 + \mathcal{O}(|\lambda_2/\lambda_1|^k)) \quad (2.2)$$

(le nombre  $a_1$  est défini par  $y_0 = \sum_i a_i v_i$ ). Le quotient de Rayleigh satisfait (si  $a_1 \neq 0$ )

$$\frac{y_k^* A y_k}{y_k^* y_k} = \lambda_1 + \mathcal{O}\left(\left|\frac{\lambda_2}{\lambda_1}\right|^k\right). \quad (2.3)$$

Si  $A$  est une matrice normale (c.-à-d. que les vecteurs propres sont orthogonaux), l'erreur dans (2.3) est  $\mathcal{O}(|\lambda_2/\lambda_1|^{2k})$ .

*Démonstration.* Exprimons le vecteur de départ  $y_0$  dans la base des vecteur propres, c.-à-d.,  $y_0 = \sum_{i=1}^n a_i v_i$ . Par récurrence, on voit que

$$y_k = A^k y_0 = \sum_{i=1}^n a_i \lambda_i^k v_i = \lambda_1^k \left( a_1 v_1 + \sum_{i=2}^n a_i \left( \frac{\lambda_i}{\lambda_1} \right)^k v_i \right), \quad (2.4)$$

ce qui démontre la formule (2.2). De cette relation, on déduit que

$$y_k^* A y_k = y_k^* y_{k+1} = \sum_{i=1}^n |a_i|^2 |\lambda_i|^{2k} \lambda_i + \sum_{i \neq j} \bar{a}_i a_j \overline{\lambda_i}^k \lambda_j^{k+1} v_i^* v_j \quad (2.5)$$

$$y_k^* y_k = \sum_{i=1}^n |a_i|^2 |\lambda_i|^{2k} + \sum_{i \neq j} \bar{a}_i a_j \overline{\lambda_i}^k \lambda_j^k v_i^* v_j. \quad (2.6)$$

Si  $a_1 \neq 0$ , la formule (2.3) est une conséquence de

$$\frac{y_k^* A y_k}{y_k^* y_k} = \frac{|a_1|^2 \cdot |\lambda_1|^{2k} \cdot \lambda_1 \cdot (1 + \mathcal{O}(|\lambda_2/\lambda_1|^k))}{|a_1|^2 \cdot |\lambda_1|^{2k} \cdot (1 + \mathcal{O}(|\lambda_2/\lambda_1|^k))}. \quad (2.7)$$

Pour une matrice normale, le deuxième terme dans les formules (2.5) et (2.6) est absent et l'expression  $\mathcal{O}(|\lambda_2/\lambda_1|^k)$  peut être remplacée par  $\mathcal{O}(|\lambda_2/\lambda_1|^{2k})$  dans (2.7) et dans (2.3).  $\square$

**Exemple 2.2** Considérons la matrice  $A = \begin{pmatrix} 2 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 2 \end{pmatrix}$  dont la valeur propre la plus grande est  $\lambda_1 = 2(1 + \cos(\pi/4)) \approx 3.414213562373095$ . Quelques itérations de la méthode de la puissance nous donnent

$$y_0 = (1, 1, 1)^T, \quad y_1 = (3, 4, 3)^T, \quad y_2 = (10, 14, 10)^T$$

et une première approximation de  $\lambda_1$  est obtenue par

$$\frac{y_1^* A y_1}{y_1^* y_1} = \frac{y_1^* y_2}{y_1^* y_1} = \frac{116}{34} \approx 3.41176.$$

*Remarques.* Les éléments du vecteur  $y_k$  croissent exponentiellement avec  $k$ . Il est alors recommandé de normaliser  $y_k$  après chaque itération, c.-à-d. de remplacer  $y_k$  par  $y_k/\|y_k\|$ . Sinon, on risque un “overflow”. Si  $|\lambda_2/\lambda_1|$  est proche de 1, la convergence est très lente. Pour accélérer la convergence, on utilise la modification suivante.

### Méthode de la puissance inverse de Wielandt

Supposons qu'on connaisse une approximation  $\mu$  de la valeur propre cherchée  $\lambda_1$  (il n'est pas nécessaire de supposer que  $\lambda_1$  soit la plus grande valeur propre de  $A$ ). L'idée est d'appliquer l'itération (2.1) à la matrice  $(A - \mu I)^{-1}$ . Les valeurs propres de cette matrice sont  $(\lambda_i - \mu)^{-1}$ . Si  $\mu$  est proche de  $\lambda_1$ , on a

$$\frac{1}{|\lambda_1 - \mu|} \gg \frac{1}{|\lambda_i - \mu|} \quad \text{pour } i \geq 2$$

et la convergence va être très rapide. L'itération devient alors  $y_{k+1} = (A - \mu I)^{-1}y_k$  ou

$$(A - \mu I)y_{k+1} = y_k. \quad (2.8)$$

Après avoir calculé la décomposition LR de la matrice  $A - \mu I$ , une itération de (2.8) ne coûte pas plus cher qu'une de (2.1).

**Exemple 2.3** Pour la matrice  $A$  de l'exemple précédent, choisissons  $\mu = 3.41$  et  $y_0 = (1, 1.4, 1)^T$ . Deux itérations de (2.8) nous donnent

$$y_1 = \begin{pmatrix} 236.134453781513 \\ 333.949579831933 \\ 236.134453781513 \end{pmatrix}, \quad y_2 = \begin{pmatrix} 56041.9461902408 \\ 79255.2785820210 \\ 56041.9461902408 \end{pmatrix}$$

et on obtient

$$\frac{1}{\lambda_1 - 3.41} \approx \frac{y_1^*(A - \mu I)^{-1}y_1}{y_1^*y_1} = \frac{y_1^*y_2}{y_1^*y_1} \approx 237.328870774159.$$

De cette relation, on calcule  $\lambda_1$  et on obtient l'approximation 3.41421356237333. Les 13 premiers chiffres sont corrects.

La méthode de la puissance (et celle de Wielandt) est importante pour la compréhension d'autres algorithmes. Si l'on veut calculer toutes les valeurs propres d'une matrice, on utilise des méthodes encore plus sophistiquées. En pratique, on procède de la manière suivante:

- on distingue les cas:  $A$  symétrique ou  $A$  quelconque.
- on cherche  $T$  telle que  $T^{-1}AT = H$  devienne une matrice de Hessenberg (ou une matrice tridiagonale, si  $A$  est symétrique); voir V.3.
- on applique l'algorithme QR à la matrice  $H$  (voir V.6).
- si  $H$  est une matrice tridiagonale et symétrique, on peut également appliquer la méthode de bisection (voir V.4).

## V.3 Transformation sous forme de Hessenberg (ou tridiagonale)

Avec la transformation  $v = Tu$  (où  $T$  est une matrice inversible) le problème

$$Av = \lambda v \quad \text{devient} \quad T^{-1}ATu = \lambda u.$$

Donc, les valeurs propres de  $A$  et de  $T^{-1}AT$  sont les mêmes et les vecteurs propres  $v_i$  de  $A$  se transforment par  $v_i = Tu_i$ . Le but de ce paragraphe est de trouver une matrice  $T$  telle que  $T^{-1}AT$  devienne “plus simple”. La situation idéale serait trouvée si  $T^{-1}AT$  devenait diagonale ou triangulaire – mais une telle transformation nécessiterait déjà la connaissance des valeurs propres.

Alors, on cherche  $T$  tel que  $T^{-1}AT$  soit sous *forme de Hessenberg*

$$T^{-1}AT = H = \begin{pmatrix} * & * & \ddots & \cdots & * \\ * & * & \ddots & \ddots & \vdots \\ * & \ddots & \ddots & \ddots & * \\ & \ddots & \ddots & \ddots & * \\ & & * & * \end{pmatrix}, \quad (3.1)$$

c.-à-d.,  $h_{ij} = 0$  pour  $i > j + 1$ . Pour arriver à ce but, nous considérons deux algorithmes.

### a) A l'aide des transformations élémentaires

Comme pour l'élimination de Gauss, nous utilisons les transformations  $L_i$  pour faire apparaître les zéros – colonne par colonne – dans (3.1).

Dans un premier pas, nous choisissons  $k \geq 2$  tel que  $|a_{k1}| \geq |a_{j1}|$  pour  $j \geq 2$  et nous permutions les lignes 2 et  $k$ , c.-à-d., nous formons  $PA$  où  $P$  est une matrice de permutation convenable. Pour ne pas changer les valeurs propres, il faut également permuter les colonnes 2 et  $k$  (ceci correspond au calcul de  $A' = PAP^{-1}$  car  $P^2 = I$  et donc  $P = P^{-1}$ ). Si  $a'_{21} = 0$ , on a aussi  $a'_{i1} = 0$  pour  $i \geq 3$  et le premier pas est terminé. Sinon, nous déterminons

$$L_2 = \begin{pmatrix} 1 & & & & \\ 0 & 1 & & & \\ 0 & -\ell_{32} & 1 & & \\ \vdots & \vdots & \ddots & \ddots & \\ 0 & -\ell_{n2} & \dots & 0 & 1 \end{pmatrix} \quad \text{telle que} \quad L_2 A' = \begin{pmatrix} a'_{11} & a'_{12} & \dots & a'_{1n} \\ a'_{21} & a'_{22} & \dots & a'_{2n} \\ 0 & * & \dots & * \\ \vdots & \vdots & & \vdots \\ 0 & * & \dots & * \end{pmatrix}.$$

Pour ceci, on définit  $\ell_{i2} = a'_{i1}/a'_{21}$ . Une multiplication à droite avec

$$L_2^{-1} = \begin{pmatrix} 1 & & & & \\ 0 & 1 & & & \\ 0 & \ell_{32} & 1 & & \\ \vdots & \vdots & \ddots & \ddots & \\ 0 & \ell_{n2} & \dots & 0 & 1 \end{pmatrix}$$

ne change pas la première colonne de  $L_2 A'$ .

On répète la même procédure avec la sous-matrice de  $L_2 A' L_2^{-1}$  de dimension  $n - 1$ , et ainsi de suite. À cause des multiplications à droite avec  $L_i^{-1}$ , cet algorithme coûte deux fois plus cher que l'élimination de Gauss (donc  $\approx 2n^3/3$  opérations).

*Exemple.* Pour la matrice

$$A = \begin{pmatrix} 3 & 2 & 1 \\ 2 & 1 & 3 \\ 1 & 3 & 1 \end{pmatrix} \quad \text{on prend} \quad L_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -1/2 & 1 \end{pmatrix}$$

et on obtient

$$L_2 A = \begin{pmatrix} 3 & 2 & 1 \\ 2 & 1 & 3 \\ 0 & 5/2 & -1/2 \end{pmatrix}, \quad \text{puis} \quad L_2 A L_2^{-1} = \begin{pmatrix} 3 & 5/2 & 1 \\ 2 & 5/2 & 3 \\ 0 & 9/4 & -1/2 \end{pmatrix} = H.$$

Cet exemple montre un désavantage de cet algorithme : si l'on part avec une matrice symétrique  $A$ , la matrice de Hessenberg  $H$ , obtenue par cet algorithme, n'est plus symétrique en général.

### b) A l'aide des transformations orthogonales

Il est souvent préférable de travailler avec des réflexions de Householder (voir le paragraphe IV.7). Commençons par une réflexion pour les coordonnées  $2, \dots, n$  laissant fixe la première coordonnée :  $\bar{Q}_2 = I - 2\bar{u}_2\bar{u}_2^T$  ( $\|\bar{u}_2\|_2 = 1$ ) tel que  $\bar{Q}_2\bar{A}_1 = \alpha_2 e_1$  où  $\bar{A}_1 = (a_{21}, \dots, a_{n1})^T$ . En posant  $u_2 = (0, \bar{u}_2)^T$  et  $Q_2 = I - 2u_2u_2^T$ , la matrice  $Q_2A$  contient des zéros dans la première colonne à partir du troisième élément. La multiplication à droite avec  $Q_2^{-1} = Q_2^T = Q_2$  ne change pas cette colonne :

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \xrightarrow{Q_2 \cdot A} \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ \boxed{\alpha_2} & * & * \\ 0 & * & * \end{pmatrix} \xrightarrow{Q_2 \cdot A \cdot Q_2} \begin{pmatrix} a_{11} & * & * \\ \alpha_2 & * & * \\ 0 & * & * \end{pmatrix}.$$

Dans le pas suivant, on applique la même procédure à la sous-matrice de dimension  $n - 1$ , etc. Finalement, on arrive à la forme de Hessenberg (3.1) avec la transformation  $T^{-1} = Q_{n-1} \cdots Q_2$ , qui est une matrice orthogonale (c.-à-d.,  $T^{-1} = T^T$ ).

Nous avons un double avantage avec cet algorithme :

- il ne faut pas faire une recherche de pivot ;
- si  $A$  est symétrique, alors  $T^{-1}AT$  est aussi symétrique, et donc *tridiagonale*.

## V.4 Méthode de bisection pour des matrices tridiagonales

Considérons une matrice symétrique tridiagonale

$$A = \begin{pmatrix} d_1 & e_2 & & & & \\ e_2 & d_2 & e_3 & & & \\ & e_3 & \ddots & \ddots & & \\ & & \ddots & \ddots & & \\ & & & e_n & & \\ & & & & d_n & \end{pmatrix}. \quad (4.1)$$

On observe tout d'abord que si un élément  $e_i$  est nul, la matrice  $A$  est déjà décomposée en deux sous-matrices du même type, qui ensemble fournissent les valeurs propres de  $A$ .

On peut donc supposer, sans restreindre la généralité, que

$$e_i \neq 0 \quad \text{pour} \quad i = 2, \dots, n. \quad (4.2)$$

Pour cette matrice, il est possible de calculer la valeur  $\chi_A(\lambda)$  du polynôme caractéristique sans connaître ses coefficients. En effet, si l'on pose

$$A_1 = (d_1), \quad A_2 = \begin{pmatrix} d_1 & e_2 \\ e_2 & d_2 \end{pmatrix}, \quad A_3 = \begin{pmatrix} d_1 & e_2 & \\ e_2 & d_2 & e_3 \\ & e_3 & d_3 \end{pmatrix}, \quad \dots$$

et si l'on définit  $p_i(\lambda) := \det(A_i - \lambda I)$ , on obtient

$$\begin{aligned} p_0(\lambda) &= 1 \\ p_1(\lambda) &= d_1 - \lambda \\ p_i(\lambda) &= (d_i - \lambda)p_{i-1}(\lambda) - e_i^2 p_{i-2}(\lambda), \quad i = 2, \dots, n. \end{aligned} \quad (4.3)$$

La formule de récurrence dans (4.3) est obtenue en développant le déterminant de la matrice  $A_i - \lambda I$  par rapport à la dernière ligne (ou colonne).

En principe, on peut maintenant calculer les valeurs propres de  $A$  (c.-à-d. les zéros de  $p_n(\lambda)$ ) de la manière suivante : chercher un intervalle où  $p_n(\lambda)$  change de signe et localiser une racine de  $p_n(\lambda) = 0$  par bisection. Les évaluations de  $p_n(\lambda)$  sont faites à l'aide de la formule (4.3). Mais il existe une astuce intéressante qui permet d'améliorer cet algorithme.

**Théorème 4.1** Si (4.2) est vérifié, les polynômes  $p_i(\lambda)$  définis par (4.3) satisfont

- a)  $p'_n(\hat{\lambda}) p_{n-1}(\hat{\lambda}) < 0$  si  $p_n(\hat{\lambda}) = 0$  ( $\hat{\lambda} \in \mathbb{R}$ );
- b)  $p_{i-1}(\hat{\lambda}) p_{i+1}(\hat{\lambda}) < 0$  si  $p_i(\hat{\lambda}) = 0$  pour un  $i \in \{1, \dots, n-1\}$ ;
- c)  $p_0(\lambda)$  ne change pas de signe sur  $\mathbb{R}$ .

*Démonstration.* L'affirmation (c) est triviale.

Si  $p_i(\hat{\lambda}) = 0$  pour un  $i \in \{1, \dots, n-1\}$ , la formule de récurrence (4.3) donne l'inégalité  $p_{i+1}(\hat{\lambda}) p_{i-1}(\hat{\lambda}) \leq 0$ . Pour démontrer (b), il suffit d'exclure le cas  $p_{i+1}(\hat{\lambda}) p_{i-1}(\hat{\lambda}) = 0$ . Si deux valeurs consécutives de la suite  $\{p_i(\hat{\lambda})\}$  sont nulles, la formule de récurrence montre que  $p_i(\hat{\lambda}) = 0$  pour tout  $i$ , ce qui contredit  $p_0(\hat{\lambda}) = 1$ .

Nous démontrons par récurrence que

$$\text{toutes les racines de } p_i(\lambda) \text{ sont réelles, simples et séparées par celles de } p_{i-1}(\lambda). \quad (4.4)$$

Il n'y a rien à démontrer pour  $i = 1$ . Supposons la propriété vraie pour  $i$  et montrons qu'elle est encore vraie pour  $i + 1$ . Comme les zéros  $\lambda_1 < \dots < \lambda_i$  de  $p_i(\lambda)$  sont séparés par ceux de  $p_{i-1}(\lambda)$  et comme  $p_{i-1}(-\infty) = +\infty$ , nous avons  $\text{sign } p_{i-1}(\lambda_j) = (-1)^{j+1}$ . Alors, on déduit de (b) que  $\text{sign } p_{i+1}(\lambda_j) = (-1)^j$ . Ceci et le fait que  $p_{i+1}(\lambda) = (-1)^{i+1}\lambda^{i+1} + \dots$  montrent que  $p_{i+1}(\lambda)$  possède un zéro réel dans chacun des intervalles ouverts  $(-\infty, \lambda_1), (\lambda_1, \lambda_2), \dots, (\lambda_i, \infty)$ .

L'affirmation (a) est maintenant une conséquence de (b) et de (4.4); voir la figure V.3.  $\square$

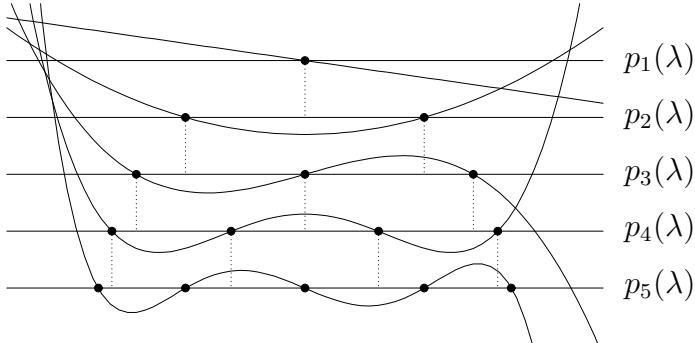


FIG. V.3: Suite de Sturm

**Définition 4.2 (suite de Sturm)** Une suite  $(p_0, p_1, \dots, p_n)$  de polynômes à coefficients réelles s'appelle une suite de Sturm<sup>1</sup>, si elle vérifie les conditions (a), (b), (c) du Théorème 4.1.

**Théorème 4.3** Considérons une suite de Sturm  $(p_0, p_1, \dots, p_n)$ . Si l'on définit

$$\omega(\lambda) = \text{nombre de changements de signes de } \{p_0(\lambda), p_1(\lambda), \dots, p_n(\lambda)\}, \quad (4.5)$$

alors le polynôme  $p_n(\lambda)$  possède exactement

$$\omega(b) - \omega(a) \quad (4.6)$$

zéros dans l'intervalle  $[a, b]$  (si  $p_i(\lambda) = 0$ , on définit  $\text{sign } p_i(\lambda) = \text{sign } p_{i-1}(\lambda)$ ).

*Démonstration.* Par continuité, l'entier  $\omega(\lambda)$  peut changer sa valeur seulement si une valeur des fonctions  $p_i(\lambda)$  devient nulle. La fonction  $p_0(\lambda)$  ne change pas de signe. Supposons alors que  $p_i(\hat{\lambda}) = 0$  pour un  $i \in \{1, \dots, n-1\}$ . La condition (b) et la continuité de  $p_j(\lambda)$  montrent que seulement les deux situations suivantes sont possibles ( $\epsilon$  petit):

<sup>1</sup>Jacques Charles François Sturm (1803 – 1855), né le 29 septembre 1803 à Genève.

	$\hat{\lambda} - \epsilon$	$\hat{\lambda}$	$\hat{\lambda} + \epsilon$		$\hat{\lambda} - \epsilon$	$\hat{\lambda}$	$\hat{\lambda} + \epsilon$
$p_{i-1}(\lambda)$	+	+	+		-	-	-
$p_i(\lambda)$	$\pm$	0	$\mp$		$\pm$	0	$\mp$
$p_{i+1}(\lambda)$	-	-	-		+	+	+

Chaque fois, on a  $\omega(\hat{\lambda} + \epsilon) = \omega(\hat{\lambda}) = \omega(\hat{\lambda} - \epsilon)$  et la valeur de  $\omega(\lambda)$  ne change pas si  $\lambda$  traverse un zéro de  $p_i(\lambda)$  pour  $i \in \{1, \dots, n-1\}$ .

Il reste à étudier la fonction  $\omega(\lambda)$  dans un voisinage d'un zéro  $\hat{\lambda}$  de  $p_n(\lambda)$ . La propriété (a) implique que pour les signes de  $p_j(\lambda)$  on a seulement les deux possibilités suivantes:

	$\hat{\lambda} - \epsilon$	$\hat{\lambda}$	$\hat{\lambda} + \epsilon$		$\hat{\lambda} - \epsilon$	$\hat{\lambda}$	$\hat{\lambda} + \epsilon$
$p_{n-1}(\lambda)$	+	+	+		-	-	-
$p_n(\lambda)$	+	0	-		-	0	+

c.-à-d.,  $\omega(\hat{\lambda} + \epsilon) = \omega(\hat{\lambda} - \epsilon) + 1$ . Ceci démontre que la fonction  $\omega(\lambda)$  est constante par morceaux et augmente de 1 sa valeur si  $\lambda$  traverse un zéro de  $p_n(\lambda)$ .  $\square$

**Méthode de bisection.** Si l'on applique ce théorème à la suite (4.3), la différence  $\omega(b) - \omega(a)$  est égale au nombre de valeurs propres de (4.1) dans l'intervalle  $[a, b]$ . On obtient toutes les valeurs propres de  $A$  de la manière suivante:

- on cherche un intervalle  $[a, b]$  qui contienne toutes les valeurs propres de  $A$  (p.ex., en appliquant le théorème de Gershgorin). On a donc que  $\omega(a) = 0$  et  $\omega(b) = n$ .
- on pose  $c = (a + b)/2$  et on calcule  $\omega(c)$ . Les différences  $\omega(c) - \omega(a)$  et  $\omega(b) - \omega(c)$  indiquent combien de valeurs propres de  $A$  sont dans  $[a, c]$  et combien sont dans  $[c, b]$ .
- on continue à diviser les intervalles qui contiennent au moins une valeur propre de  $A$ .

On peut facilement modifier cet algorithme pour calculer la valeur propre la plus petite ou la 3ème plus grande valeur propre, etc.

Pour éviter un “overflow” dans le calcul de  $p_n(\lambda)$  (si  $n$  et  $\lambda$  sont grands), il vaut mieux travailler avec

$$f_i(\lambda) := p_i(\lambda)/p_{i-1}(\lambda) \quad i = 1, \dots, n \quad (4.7)$$

et utiliser le fait que

$$\omega(\lambda) = \text{nombre d'éléments négatifs parmi } \{f_1(\lambda), \dots, f_n(\lambda)\} \quad (4.8)$$

(attention: si  $p_{i-1}(\lambda)$  est zéro, on pose  $f_i(\lambda) = -\infty$ ; cette valeur compte pour un élément négatif). Pour une programmation de l'algorithme, on utilise la récurrence

$$\begin{aligned} f_1(\lambda) &= d_1 - \lambda \\ f_i(\lambda) &= d_i - \lambda - \begin{cases} e_i^2/f_{i-1}(\lambda) & \text{si } f_{i-1}(\lambda) \neq 0 \\ |e_i|/\text{eps} & \text{si } f_{i-1}(\lambda) = 0. \end{cases} \end{aligned} \quad (4.9)$$

La formule pour le cas  $f_{i-1}(\lambda) \neq 0$  est une conséquence de (4.3). Si  $f_{i-1}(\lambda) = 0$  (c.-à-d.,  $p_{i-1}(\lambda) = 0$ ), on remplace cette valeur par  $|e_i| \cdot \text{eps}$ . Ceci correspond à ajouter la perturbation  $|e_i| \cdot \text{eps}$  à  $d_{i-1}$ .

## V.5 L'itération orthogonale

Dans ce paragraphe, nous allons généraliser la méthode de la puissance (voir le paragraphe V.2) afin de pouvoir calculer les deux (trois, ...) valeurs propres dominantes en même temps. Cette généralisation motivera l'itération QR qui constitue l'algorithme le plus important pour le calcul des valeurs propres d'une matrice.

**Généralisation de la méthode de la puissance (pour calculer les deux valeurs propres dominantes).** Considérons une matrice  $A$  dont les valeurs propres satisfont

$$|\lambda_1| > |\lambda_2| > \dots > |\lambda_n|. \quad (5.1)$$

La méthode de la puissance est basée sur l'itération  $y_{k+1} = Ay_k$  (voir (2.1)) et nous permet d'obtenir une approximation de  $\lambda_1$  à l'aide du quotient de Rayleigh. Pour calculer (en même temps) la deuxième valeur propre  $\lambda_2$ , nous prenons deux vecteurs  $y_0$  et  $z_0$  satisfaisant  $y_0^* z_0 = 0$  et nous considérons l'itération

$$\begin{aligned} y_{k+1} &= Ay_k \\ z_{k+1} &= Az_k - \beta_{k+1} y_{k+1} \end{aligned} \quad (5.2)$$

où  $\beta_{k+1}$  est déterminé par la condition  $y_{k+1}^* z_{k+1} = 0$ . Par induction, on voit que

$$\begin{aligned} y_k &= A^k y_0 \\ z_k &= A^k z_0 - \gamma_k y_k \end{aligned} \quad (5.3)$$

où  $\gamma_k$  est tel que

$$y_k^* z_k = 0. \quad (5.4)$$

Ceci signifie que le calcul de  $\{z_k\}$  correspond à la méthode de la puissance appliquée à  $z_0$ , combinée avec une orthogonalisation (projection de  $A^k z_0$  sur le complément orthogonal de  $y_k$ ).

En exprimant les vecteurs initiaux dans la base de vecteurs propres  $v_1, \dots, v_n$  de la matrice  $A$  (on suppose  $\|v_i\|_2 = 1$ ),

$$y_0 = \sum_{i=1}^n a_i v_i, \quad z_0 = \sum_{i=1}^n b_i v_i,$$

les vecteurs  $y_k, z_k$  deviennent

$$y_k = \sum_{i=1}^n a_i \lambda_i^k v_i, \quad z_k = \sum_{i=1}^n (b_i - \gamma_k a_i) \lambda_i^k v_i. \quad (5.5)$$

Comme nous l'avons constaté dans le paragraphe V.2, pour  $k \rightarrow \infty$ , le terme  $a_1 \lambda_1^k v_1$  est dominant dans  $y_k$  (si  $a_1 \neq 0$ ) et on obtient une approximation du premier vecteur propre  $v_1$ . Que peut-on dire pour la suite  $z_k$ ?

La condition (5.4) d'orthogonalité implique que

$$\sum_{i=1}^n \sum_{j=1}^n \bar{a}_i (b_j - \gamma_k a_j) \bar{\lambda}_i^k \lambda_j^k v_i^* v_j = 0. \quad (5.6)$$

Cette relation définit  $\gamma_k$ . Comme le terme avec  $i = j = 1$  est dominant, on voit que  $\gamma_k \approx b_1/a_1$ . Par la suite, nous allons supposer que  $a_1 \neq 0$  et  $a_1 b_2 - a_2 b_1 \neq 0$ . En divisant (5.6) par  $\bar{\lambda}_1^k$ , on obtient

$$\bar{a}_1 (b_1 - \gamma_k a_1) \lambda_1^k (1 + \mathcal{O}(|\lambda_2/\lambda_1|^k)) = -\bar{a}_1 (b_2 - \gamma_k a_2) \lambda_2^k (v_1^* v_2 + \mathcal{O}(|\lambda_2/\lambda_1|^k) + \mathcal{O}(|\lambda_3/\lambda_2|^k)).$$

Maintenant, on peut insérer cette formule dans (5.5) et on en déduit

$$z_k = \lambda_2^k (b_2 - \gamma_k a_2) \left( v_2 - v_1^* v_2 \cdot v_1 + \mathcal{O}(|\lambda_2/\lambda_1|^k) + \mathcal{O}(|\lambda_3/\lambda_2|^k) \right). \quad (5.7)$$

Visiblement, le vecteur  $z_k$  s'approche (pour  $k \rightarrow \infty$ ) d'un multiple de  $v_2 - v_1^* v_2 \cdot v_1$ , qui est la projection orthogonale de  $v_2$  à l'hyperplan  $v_1^\perp$ . Concernant les valeurs propres, on a le résultat suivant.

**Théorème 5.1** *Considérons les vecteurs  $y_k, z_k$  donnés par (5.2) et notons*

$$U_k = (y_k / \|y_k\|_2, z_k / \|z_k\|_2) \quad (5.8)$$

(observer que  $U_k^* U_k = I$ ). Si (5.1) est vérifié, on a que

$$U_k^* A U_k \rightarrow \begin{pmatrix} \lambda_1 & * \\ 0 & \lambda_2 \end{pmatrix} \quad \text{pour } k \rightarrow \infty. \quad (5.9)$$

*Démonstration.* L'élément (1,1) de la matrice  $U_k^* A U_k$  est le quotient de Rayleigh (2.3) qui converge vers  $\lambda_1$ . En utilisant (5.7), on voit que l'élément (2,2) satisfait

$$\frac{z_k^* A z_k}{z_k^* z_k} \rightarrow \frac{(v_2 - v_1^* v_2 \cdot v_1)^* (\lambda_2 v_2 - \lambda_1 v_1^* v_2 \cdot v_1)}{(v_2 - v_1^* v_2 \cdot v_1)^* (v_2 - v_1^* v_2 \cdot v_1)} = \frac{\lambda_2 (1 - |v_1^* v_2|^2)}{1 - |v_1^* v_2|^2} = \lambda_2.$$

De façon similaire, on obtient pour l'élément (2,1)

$$\frac{z_k^* A y_k}{\|z_k\|_2 \cdot \|y_k\|_2} \rightarrow \frac{(v_2 - v_1^* v_2 \cdot v_1)^* \lambda_1 v_1}{\|v_2 - v_1^* v_2 \cdot v_1\|_2 \cdot \|v_1\|_2} = 0.$$

Finalement, l'élément (1,2) de  $U_k^* A U_k$  satisfait

$$\frac{y_k^* A z_k}{\|y_k\|_2 \cdot \|z_k\|_2} \rightarrow \frac{v_1^* (\lambda_2 v_2 - \lambda_1 v_1^* v_2 \cdot v_1)}{\|v_1\|_2 \cdot \|v_2 - v_1^* v_2 \cdot v_1\|_2} = \frac{(\lambda_2 - \lambda_1) v_1^* v_2}{\sqrt{1 - |v_1^* v_2|^2}}.$$

Cette expression est en général non nulle. □

*Remarque.* Avec la notation (5.8), l'itération (5.2) peut être écrite sous la forme

$$A U_k = U_{k+1} R_{k+1} \quad (5.10)$$

où  $R_{k+1}$  est une matrice  $2 \times 2$  qui est triangulaire supérieure.

**Méthode de la puissance (pour le calcul de toutes les valeurs propres)** ou simplement **itération orthogonale**. La généralisation de l'algorithme précédent au cas où l'on veut calculer toutes les valeurs propres d'une matrice est évidente: on choisit une matrice orthogonale  $U_0$ , c.-à-d., on choisit  $n$  vecteurs orthogonaux (les colonnes de  $U_0$ ) qui jouent le rôle de  $y_0, z_0$ , etc. Puis, on effectue l'itération

```

for  $k = 1, 2, \dots$ 
     $Z_k = A U_{k-1}$ 
     $U_k R_k = Z_k$       (décomposition QR)
end

```

Si (5.1) est vérifié et si la matrice  $U_0$  est bien choisie ( $a_1 \neq 0, a_1 b_2 - a_2 b_1 \neq 0$ , etc), une généralisation du théorème précédent donne la convergence de

$$T_k := U_k^* A U_k \quad (5.11)$$

vers une matrice triangulaire dont les éléments de la diagonale sont les valeurs propres de  $A$ . On a donc transformé  $A$  en forme triangulaire à l'aide d'une matrice orthogonale (*décomposition de Schur*).

Il y a une possibilité intéressante pour calculer  $T_k$  de (5.11) directement à partir de  $T_{k-1}$ . D'une part, on déduit de (5.10) que

$$T_{k-1} = U_{k-1}^* A U_{k-1} = (U_{k-1}^* U_k) R_k. \quad (5.12a)$$

D'autre part, on a

$$T_k = U_k^* A U_k = U_k^* A U_{k-1} U_{k-1}^* U_k = R_k (U_{k-1}^* U_k). \quad (5.12b)$$

On calcule la décomposition QR de la matrice  $T_{k-1}$  et on échange les deux matrices de cette décomposition pour obtenir  $T_k$ .

## V.6 L' algorithme QR

La méthode QR, due à J.C.F. Francis et à V.N. Kublanovskaya, est la méthode la plus couramment utilisée pour le calcul de l'ensemble des valeurs propres . . . (P.G. Ciarlet 1982)  
 . . . the QR iteration, and it forms the backbone of the most effective algorithm for computing the Schur decomposition. (G.H. Golub & C.F. van Loan 1989)

La version simple du célèbre algorithme QR n'est rien d'autre que la méthode du paragraphe précédent. En effet, si l'on pose  $Q_k = U_{k-1}^* U_k$  et si l'on commence l'itération avec  $U_0 = I$ , les formules (5.11) et (5.12) nous permettent d'écrire l'algorithme précédent comme suit:

```

 $T_0 = A$ 
for  $k = 1, 2, \dots$ 
     $Q_k R_k = T_{k-1}$            (décomposition QR)
     $T_k = R_k Q_k$ 
end
```

Les  $T_k$  qui sont les mêmes que dans le paragraphe V.5, convergent (en général) vers une matrice triangulaire. Ceci nous permet d'obtenir toutes les valeurs propres de la matrice  $A$  car les  $T_k$  ont les mêmes valeurs propres que  $A$  (voir (5.11)).

Cet algorithme important a été développé indépendamment par J.G.F. Francis (1961) et par V.N. Kublanovskaya (1961). Un algorithme similaire, qui utilise la décomposition LR à la place de la décomposition QR, a été introduit par H. Rutishauser (1958).

**Exemple numérique.** Appliquons la méthode QR à la matrice

$$A = \begin{pmatrix} 10 & 2 & 3 & 5 \\ 3 & 6 & 8 & 4 \\ 0 & 5 & 4 & 3 \\ 0 & 0 & 4 & 3 \end{pmatrix}. \quad (6.1)$$

On peut montrer (voir exercice 14) que, pour une matrice de Hessenberg  $A$ , toutes les matrices  $T_k$  sont aussi sous forme de Hessenberg. Pour étudier la convergence vers une matrice triangulaire, il suffit alors de considérer les éléments  $t_{i+1,i}^{(k)}$  ( $i = 1, \dots, n-1$ ) de la sous-diagonale. On constate que

$$\frac{t_{i+1,i}^{(k+1)}}{t_{i+1,i}^{(k)}} \approx \frac{\lambda_{i+1}}{\lambda_i} \quad (6.2)$$

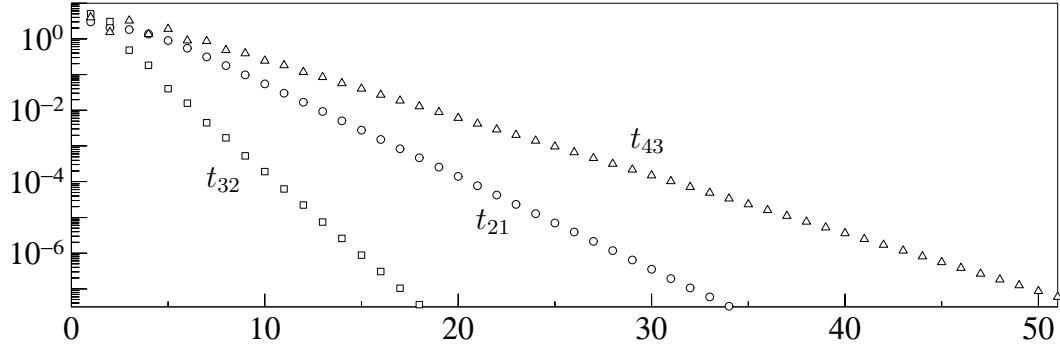


FIG. V.4: Convergence de la méthode QR (sans shift)

$(\lambda_1 \approx 14.3, \lambda_2 \approx 7.86, \lambda_3 \approx 2.70, \lambda_4 \approx -1.86)$ . Comme  $|\lambda_{i+1}/\lambda_i| < 1$ , les éléments  $t_{i+1,i}^{(k)}$  convergent, pour  $k \rightarrow \infty$ , linéairement vers 0 (voir la figure V.4, où les valeurs  $|t_{i+1,i}^{(k)}|$  sont dessinées en fonction du nombre  $k$  de l’itération).

**Remarques.** (a) Comme le calcul de la décomposition QR d’une matrice pleine est très coûteux ( $\mathcal{O}(n^3)$  opérations), on applique l’algorithme QR uniquement aux matrices de Hessenberg. Dans cette situation une itération nécessite seulement  $\mathcal{O}(n^2)$  opérations.

(b) La convergence est très lente en général (seulement *linéaire*). Pour rendre efficace cet algorithme, il faut absolument trouver un moyen pour accélérer la convergence.

(c) Considérons la situation où  $A$  est une matrice réelle qui possède des valeurs propres complexes (l’hypothèse (5.1) est violée). L’algorithme QR produit une suite de matrices  $T_k$  qui sont toutes réelles. Dans cette situation, les  $T_k$  ne convergent pas vers une matrice triangulaire, mais deviennent *triangulaires par blocs* (sans démonstration). Comme la dimension des blocs dans la diagonale vaut en général 1 ou 2, on obtient également des approximations des valeurs propres.

## Accélération de la convergence

D’après l’observation (6.2), nous savons que

$$t_{n,n-1}^{(k)} = \mathcal{O}(|\lambda_n/\lambda_{n-1}|^k). \quad (6.3)$$

La convergence vers zéro de cet élément ne va être rapide que si  $|\lambda_n| \ll |\lambda_{n-1}|$ . Une idée géniale est d’appliquer l’algorithme QR à la matrice  $A - pI$  où  $p \approx \lambda_n$ . Comme les valeurs propres de  $A - pI$  sont  $\lambda_i - p$ , on a la propriété  $|\lambda_n - p| \ll |\lambda_i - p|$  pour  $i = 1, \dots, n-1$  et l’élément  $t_{n,n-1}^{(k)}$  va converger rapidement vers zéro. Rien ne nous empêche d’améliorer l’approximation  $p$  après chaque itération. L’algorithme QR avec “shift” devient alors:

```

 $T_0 = A$ 
for  $k = 1, 2, \dots$ 
    déterminer le paramètre  $p_{k-1}$ 
     $Q_k R_k = T_{k-1} - p_{k-1} I$            (décomposition QR)
     $T_k = R_k Q_k + p_{k-1} I$ 
end

```

Les matrices  $T_k$  de cette itération satisfont

$$Q_k^* T_{k-1} Q_k = Q_k^* (Q_k R_k + p_{k-1} I) Q_k = R_k Q_k + p_{k-1} I = T_k. \quad (6.4)$$

Ceci implique que, indépendamment de la suite  $p_k$ , les matrices  $T_k$  ont toutes les mêmes valeurs propres que  $T_0 = A$ .

Pour décrire complètement l’algorithme QR avec shift, il faut encore discuter le choix du paramètre  $p_k$  et il faut donner un critère pour arrêter l’itération.

**Choix du “shift”-paramètre.** On a plusieurs possibilités:

- $p_k = t_{nn}^{(k)}$ : ce choix marche très bien si les valeurs propres de la matrice sont réelles.
- on considère la matrice

$$\begin{pmatrix} t_{n-1,n-1}^{(k)} & t_{n-1,n}^{(k)} \\ t_{n,n-1}^{(k)} & t_{n,n}^{(k)} \end{pmatrix}. \quad (6.5)$$

Si les valeurs propres de (6.5) sont réelles, on choisit pour  $p_k$  celle qui est la plus proche de  $t_{n,n}^{(k)}$ . Si elles sont de la forme  $\alpha \pm i\beta$  avec  $\beta \neq 0$  (donc complexes), on prend d’abord  $p_k = \alpha + i\beta$  et pour l’itération suivante  $p_{k+1} = \alpha - i\beta$ .

**Critère pour arrêter l’itération.** L’idée est d’itérer jusqu’à ce que  $t_{n,n-1}^{(k)}$  ou  $t_{n-1,n-2}^{(k)}$  soit suffisamment petit. Plus précisément, on arrête l’itération quand

$$|t_{\ell,\ell-1}^{(k)}| \leq \text{eps} \cdot (|t_{\ell-1,\ell-1}^{(k)}| + |t_{\ell,\ell}^{(k)}|) \quad \text{pour } \ell = n \quad \text{ou } \ell = n-1. \quad (6.6)$$

- Si (6.6) est vérifié pour  $\ell = n$ , on accepte  $t_{n,n}^{(k)}$  comme approximation de  $\lambda_n$  et on continue l’itération avec la matrice  $(t_{ij}^{(k)})_{i,j \leq n-1}$ .
- Si (6.6) est vérifié pour  $\ell = n-1$ , on accepte les deux valeurs propres de (6.5) comme approximations de  $\lambda_n$  et  $\lambda_{n-1}$  et on continue l’itération avec la matrice  $(t_{ij}^{(k)})_{i,j \leq n-2}$ .

**Exemple numérique.** Nous avons appliqué l’algorithme QR à la matrice (6.1) avec le shift  $p_k = t_{nn}^{(k)}$ . La convergence de  $t_{i+1,i}^{(k)}$  vers zéro est illustrée dans la figure V.5. Une comparaison avec la figure V.4 nous montre que la convergence est beaucoup plus rapide (convergence quadratique). Après 5 itérations, on a  $|t_{43}^{(k)}| \leq 10^{-15}$ . Encore 4 itérations pour la matrice de dimension 3 donnent  $|t_{32}^{(k)}| \leq 10^{-15}$ . Il ne reste plus que 3 itérations à faire pour la matrice de dimension 2 pour avoir  $|t_{21}^{(k)}| \leq 10^{-15}$ . En tout, 12 itérations ont donné toutes les valeurs propres avec une précision de 15 chiffres.

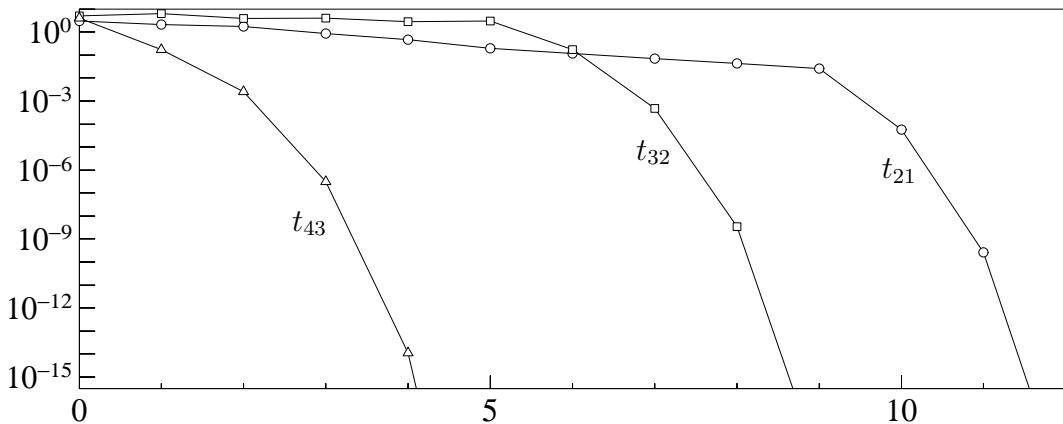


FIG. V.5: Convergence de la méthode QR (avec shift)

## Le “double shift” de Francis

Dans la situation où  $A$  est une matrice réelle ayant des valeurs propres complexes, il est recommandé de choisir un shift-paramètre  $p_k$  qui soit complexe. Une application directe de l’algorithme précédent nécessite un calcul avec des matrices complexes. L’observation suivante permet d’éviter ceci.

**Lemme 6.1** Soit  $T_k$  une matrice réelle,  $p_k = \alpha + i\beta$  et  $p_{k+1} = \alpha - i\beta$ . Alors, on peut choisir les décompositions dans l’algorithme QR de manière à ce que  $T_{k+2}$  soit réelle.

*Remarque.* La décomposition QR d'une matrice est unique sauf qu'on peut remplacer  $QR$  par  $(QD)(D^{-1}R)$  où  $D = \text{diag}(d_1, \dots, d_n)$  avec  $|d_i| = 1$ .

*Démonstration.* La formule (6.4) montre que

$$T_{k+2} = (Q_{k+1}Q_{k+2})^*T_k(Q_{k+1}Q_{k+2}). \quad (6.7)$$

Il suffit alors de démontrer que le produit  $Q_{k+1}Q_{k+2}$  est réel. Une manipulation à l'aide de formules pour  $T_k$  donne

$$\begin{aligned} Q_{k+1}Q_{k+2}R_{k+2}R_{k+1} &= Q_{k+1}(T_{k+1} - p_{k+1}I)R_{k+1} = Q_{k+1}(R_{k+1}Q_{k+1} + p_kI - p_{k+1}I)R_{k+1} \\ &= (Q_{k+1}R_{k+1})^2 + (p_k - p_{k+1})Q_{k+1}R_{k+1} = (T_k - p_kI)^2 + (p_k - p_{k+1})(T_k - p_kI) \\ &= T_k^2 - (p_k + p_{k+1})T_k + p_k p_{k+1}I =: M. \end{aligned} \quad (6.8)$$

On a donc trouvé une décomposition QR de la matrice  $M$  qui, en conséquence des hypothèses du lemme, est une matrice réelle. Si, dans l'algorithme QR, la décomposition est choisie de manière à ce que les éléments diagonaux de  $R_{k+1}$  et  $R_{k+2}$  soient réels, alors, à cause de l'unicité de la décomposition QR, les matrices  $Q_{k+1}Q_{k+2}$  et  $R_{k+2}R_{k+1}$  sont réelles.  $\square$

Une possibilité de calculer  $T_{k+2}$  à partir de  $T_k$  est de calculer  $M$  de (6.8), de faire une décomposition QR (réelle) de  $M$  et de calculer  $T_{k+2}$  à l'aide de (6.7). Cet algorithme n'est pas pratique car le calcul de  $T_k^2$  nécessite  $\mathcal{O}(n^3)$  opérations, même si  $T_k$  est sous forme de Hessenberg.

Il y a une astuce intéressante pour obtenir  $T_{k+2}$  à partir de  $T_k$  en  $\mathcal{O}(n^2)$  opérations. Elle est basée sur la propriété suivante.

**Théorème 6.2** Soit  $T$  une matrice donnée et supposons que

$$Q^*TQ = S \quad (6.9)$$

où  $Q$  est orthogonale et  $S$  est sous forme de Hessenberg satisfaisant  $s_{i,i-1} \neq 0$  pour  $i = 2, \dots, n$ . Alors,  $Q$  et  $S$  sont déterminées de manière “unique” par la première colonne de  $Q$ .

*Remarque.* On a “unicité” dans le sens suivant: si  $\widehat{Q}^*T\widehat{Q}$  est de type Hessenberg avec une matrice orthogonale  $\widehat{Q}$  satisfaisant  $\widehat{Q}e_1 = Qe_1$ , alors  $\widehat{Q} = QD$  où  $D = \text{diag}(d_1, \dots, d_n)$  avec  $|d_i| = 1$ .

*Démonstration.* Notons les colonnes de  $Q$  par  $q_i$ . Alors, la relation (6.9) implique

$$Tq_i = \sum_{j=1}^{i+1} s_{ji}q_j, \quad q_j^*Tq_i = s_{ji}. \quad (6.10)$$

Si  $q_1$  est fixé, la valeur  $s_{11}$  est donnée par la deuxième formule de (6.10). Avec cette valeur, on obtient de la première formule de (6.10) que  $q_2$  est un multiple de  $Tq_1 - s_{11}q_1$ . Ceci détermine  $q_2$  à une unité  $d_2$  près. Maintenant, les valeurs  $s_{21}, s_{12}, s_{22}$  sont déterminées et  $q_3$  est un multiple de  $Tq_2 - s_{21}q_1 - s_{22}q_2$ , etc.  $\square$

Si les hypothèses du lemme précédent sont vérifiées, on peut calculer la matrice réelle  $T_{k+2}$  en  $\mathcal{O}(n^2)$  opérations de la manière suivante:

- calculer  $Me_1$ , la première colonne de  $M$  (formule (6.8));
- déterminer une matrice de Householder  $H_1$  telle que  $H_1(Me_1) = \alpha e_1$ ;
- transformer  $H_1^T T_k H_1$  sous forme de Hessenberg à l'aide de matrices de Householder  $H_2, \dots, H_{n-1}$  (voir le paragraphe V.3); c.-à-d., calculer  $H^T T_k H$  où  $H = H_1 H_2 \cdot \dots \cdot H_{n-1}$ .

Comme  $H_i e_1 = e_1$  pour  $i = 2, \dots, n - 1$ , la première colonne de  $H$  est un multiple de celle de  $M$  (observer que  $H_1^T = H_1$ ). Par la formule (6.8), la première colonne de  $Q_{k+1} Q_{k+2}$  est aussi un multiple de  $M e_1$ . Par conséquent, pour un bon choix des décompositions  $Q_{k+1} R_{k+1}$  et  $Q_{k+2} R_{k+2}$ , on a  $H = Q_{k+1} Q_{k+2}$  et la matrice obtenue par cet algorithme est égale à  $T_{k+2}$  (voir (6.7)).

### Etude de la convergence

Supposons d'être déjà proche de la limite et considérons, par exemple, la matrice

$$T_0 = A = \begin{pmatrix} 2 & a \\ \epsilon & 1 \end{pmatrix}$$

où  $\epsilon$  est un nombre petit. Avec le choix  $p_0 = 1$  pour le shift-paramètre, on obtient

$$T_0 - p_0 I = \begin{pmatrix} 1 & a \\ \epsilon & 0 \end{pmatrix} = \begin{pmatrix} \frac{1}{\sqrt{1+\epsilon^2}} & -\frac{\epsilon}{\sqrt{1+\epsilon^2}} \\ \frac{\epsilon}{\sqrt{1+\epsilon^2}} & \frac{1}{\sqrt{1+\epsilon^2}} \end{pmatrix} \begin{pmatrix} \sqrt{1+\epsilon^2} & \frac{a}{\sqrt{1+\epsilon^2}} \\ 0 & -\frac{a\epsilon}{\sqrt{1+\epsilon^2}} \end{pmatrix} = Q_1 R_1$$

et

$$T_1 - p_0 I = R_1 Q_1 = \begin{pmatrix} * & * \\ -\frac{a\epsilon^2}{1+\epsilon^2} & * \end{pmatrix}.$$

- si  $A$  est symétrique (c.-à-d.,  $a = \epsilon$ ) on a  $t_{n,n-1}^{(1)} = \mathcal{O}(\epsilon^3)$ , donc convergence *cubique*.
- si  $A$  n'est pas symétrique (p.ex.  $a = 1$ ) on a  $t_{n,n-1}^{(1)} = \mathcal{O}(\epsilon^2)$ , donc convergence *quadratique*.

Ces propriétés restent vraies pour des matrices générales (sans démonstration).

## V.7 Exercices

1. Calculer les valeurs propres de la matrice tridiagonale (dimension  $n, b \cdot c > 0$ )

$$A = \begin{pmatrix} a & c & & & \\ b & a & c & & \\ & b & a & c & \\ & & b & \ddots & \ddots \end{pmatrix}.$$

*Indication.* Les composants du vecteur propre  $(v_1, v_2, \dots, v_n)^T$  satisfont une équation aux différences finies avec  $v_0 = v_{n+1} = 0$ . Vérifier que  $v_j = Const \cdot (\alpha_1^j - \alpha_2^j)$  où

$$\alpha_1 + \alpha_2 = \frac{\lambda - a}{c}, \quad \alpha_1 \cdot \alpha_2 = \frac{b}{c}, \quad \left(\frac{\alpha_1}{\alpha_2}\right)^{n+1} = 1.$$

*Résultat.*  $\lambda_j = a - 2\sqrt{bc} \cdot \cos\left(\frac{j\pi}{n+1}\right)$ ,  $j = 1, 2, \dots, n$ .

2. Considérer la matrice

$$A(\varepsilon) = \begin{pmatrix} 1 & \varepsilon & 0 \\ -1 & 0 & 1 \\ 1 & -1 + \varepsilon & -\varepsilon \end{pmatrix}$$

D'après le théorème 1.3 cette matrice possède une valeur propre de la forme

$$\lambda(\varepsilon) = i + \varepsilon \cdot d + \mathcal{O}(\varepsilon^2).$$

Calculer  $d$  et dessiner la tangente à la courbe  $\lambda(\varepsilon)$  au point  $\lambda(0)$ .

3. (a) Calculer par la méthode de la puissance, la plus grande valeur propre de la matrice

$$A = \begin{pmatrix} 99 & 1 & 0 \\ 1 & 100 & 1 \\ 0 & 1 & 98 \end{pmatrix}.$$

- (b) Pour accélérer considérablement la vitesse de convergence, appliquer la méthode de la puissance à la matrice  $A - pI$  avec un choix intelligent de  $p$ .  
(c) Avec quel choix de  $p$  obtient-on la valeur propre la plus petite?

4. Considérons la matrice tridiagonale

$$A = \begin{pmatrix} b_1 & c_1 & & \\ a_1 & b_2 & c_2 & \\ & a_2 & \ddots & \end{pmatrix}.$$

Montrer que, si  $a_i \cdot c_i > 0$  pour  $i = 1, \dots, n - 1$ , toutes les valeurs propres de  $A$  sont réelles.

*Indication.* Trouver  $D = \text{diag}(d_1, \dots, d_n)$  telle que  $DAD^{-1}$  soit symétrique.

5. Soit  $A$  une matrice symétrique et  $B$  quelconque. Montrer que pour chaque valeur propre  $\lambda_B$  de  $B$  il existe une valeur propre  $\lambda_A$  de  $A$  telle que

$$|\lambda_A - \lambda_B| \leq \|A - B\|_2.$$

*Indication.* Montrer l'existence d'un vecteur  $v$  tel que  $v = (A - \lambda_B)^{-1}(A - B)v$ . En déduire que  $1 \leq \|(A - \lambda_B)^{-1}(A - B)\| \leq \|(A - \lambda_B)^{-1}\| \|(A - B)\|$ .

6. (Schur, 1909). Soit  $A$  une matrice symétrique. Montrer que pour chaque indice  $i$  il existe une valeur propre  $\lambda$  de  $A$  telle que

$$|\lambda - a_{ii}| \leq \sqrt{\sum_{j \neq i} |a_{ij}|^2}.$$

*Indication.* Appliquer l'exercice 5 avec une  $B$  convenable.

7. Soit  $A$  une matrice réelle avec pour valeur propre  $\alpha + i\beta$ . Montrer que l'itération

$$\begin{pmatrix} \bar{\alpha}I - A & -\bar{\beta}I \\ \bar{\beta}I & \bar{\alpha}I - A \end{pmatrix} \begin{pmatrix} u_{k+1} \\ v_{k+1} \end{pmatrix} = \begin{pmatrix} u_k \\ v_k \end{pmatrix}$$

(où  $\bar{\alpha} \approx \alpha$  et  $\bar{\beta} \approx \beta$ ) permet de calculer la valeur propre  $\alpha + i\beta$  et le vecteur propre correspondant.

*Indication.* Considérer les parties réelles et complexes de l'itération de Wielandt. On obtient alors

$$\frac{u_k^T A u_k + v_k^T A v_k}{u_k^T u_k + v_k^T v_k} \rightarrow \alpha, \quad \frac{u_k^T A v_k - v_k^T A u_k}{u_k^T u_k + v_k^T v_k} \rightarrow \beta.$$

8. Considérons la matrice de Hilbert,

$$A = \begin{pmatrix} 1/2 & 1/3 & 1/4 \\ 1/3 & 1/4 & 1/5 \\ 1/4 & 1/5 & 1/6 \end{pmatrix}$$

- (a) Transformer  $A$  en une matrice tridiagonale ayant les mêmes valeurs propres.  
(b) En utilisant une suite de Sturm, montrer que toutes les valeurs propres sont positives et qu'une valeur propre est plus petite que 0.001.  
(c) Calculer approximativement la condition de  $A$  pour la norme Euclidienne.

9. La formule de récurrence

$$(k+1)P_{k+1}(x) = (2k+1)xP_k(x) - kP_{k-1}(x)$$

pour les *polynômes de Legendre* (voir aussi (I.4.5)) ressemble à

$$p_i(\lambda) = (d_i - \lambda)p_{i-1}(\lambda) - e_i^2 p_{i-2}(\lambda), \quad i = 2, \dots, n,$$

pour les polynômes  $\det(A_i - \lambda I)$ . Trouver une matrice tridiagonale  $A$  de dimension  $n$  telle que les valeurs propres de  $A$  sont les racines de  $P_n(x)$ .

10. Soit  $p(x)$  un polynôme de degré  $n$  et supposons que toutes les racines soient simples. Démontrer que la suite définie par l'algorithme d'Euclid,

$$\begin{aligned} p_n(x) &= p(x), & p_{n-1}(x) &= -p'(x) \\ p_i(x) &= q_i(x)p_{i-1}(x) - \gamma_i^2 p_{i-2}(x), & i &= n, \dots, 2, \end{aligned}$$

est une suite de Sturm.

Pour le polynôme  $p(x) = x^5 - 5x^4 + 3x^3 + 3x^2 + 2x + 8$ .

- (a) déterminer le nombre de racines réelles.
- (b) Combien de racines sont complexes?
- (c) Combien de racines sont réelles et positives?

11. Pour un  $\varphi$  donné notons  $c = \cos \varphi$  et  $s = \sin \varphi$ . La matrice  $\Omega_{k\ell}$ , définie par

$$(\Omega_{k\ell})_{ij} = \begin{cases} 1 & \text{si } i = j, j \neq k, j \neq \ell \\ c & \text{si } i = j = k \text{ où } i = j = \ell \\ s & \text{si } i = k \text{ et } j = \ell \\ -s & \text{si } i = \ell \text{ et } j = k \\ 0 & \text{sinon,} \end{cases}$$

s'appelle *rotation de Givens*.

a) Montrer qu'elle est orthogonale.

b) Soit  $A$  une matrice symétrique. Déterminer  $\varphi$  tel que le  $(k, \ell)$ -ième élément de  $A' = \Omega_{k\ell} A \Omega_{k\ell}^T$  s'annule.

*Résultat.*  $\operatorname{ctg} 2\varphi = (a_{kk} - a_{\ell\ell})/(2a_{k\ell})$ .

12. La *méthode de Jacobi* (1846) pour le calcul des valeurs propres d'une matrice symétrique:

- i) on choisit  $a_{k\ell}$  ( $k > \ell$ ) tel que  $|a_{k\ell}| = \max_{i>j} |a_{ij}|$ ;
- ii) on détermine  $A'$  comme dans l'exercice 11.

Montrer que, si on répète cette procédure, on a convergence vers une matrice diagonale, dont les éléments sont les valeurs propres de  $A$ .

*Indication.* Montrer que  $\sum_{i>j} |a'_{ij}|^2 = \sum_{i>j} |a_{ij}|^2 - |a_{k\ell}|^2$ .

13. On considère la matrice

$$A = \begin{pmatrix} 7 & 0.5 \\ 0.0001 & 8 \end{pmatrix}$$

dont on cherche à calculer les valeurs propres.

- (a) Faire une itération de l'algorithme QR sans shift.
- (b) Faire une itération de l'algorithme QR avec shift.

- (c) Estimer la position des valeurs propres de  $A$  à l'aide du Théorème de Gershgorine.
- (d) Calculer les valeurs propres de  $A$  à l'aide du polynôme caractéristique.
14. Montrer que si la matrice  $T_0 = A$  est une matrice de Hessenberg (ou tridiagonale et symétrique), alors les matrices  $T_k$ ,  $k \geq 1$ , construites par l'algorithme QR sont également des matrices de Hessenberg (tridiagonales et symétriques).
15. Donner une estimation grossière du nombre d'opérations qui sont nécessaires pour effectuer la décomposition QR d'une matrice de Hessenberg et pour calculer ensuite le produit RQ.
16. Soit  $T_0$  une matrice de Hessenberg dont tous les éléments de la sous-diagonale sont non-nuls. Montrer que, si  $p_0$  est une valeur propre de  $T_0$ , une itération de l'algorithme QR avec shift  $p_0$  donne  $t_{n,n-1}^{(1)} = 0$ .
17. Expliquer, comment le calcul de  $T_k$  à partir de  $T_{k-1}$

$$Q_k R_k = T_{k-1} - p_{k-1} I, \quad T_k = R_k Q_k + p_{k-1} I$$

peut être effectué sans soustraire (et additionner) explicitement la matrice  $p_{k-1} I$ .

*Indication.* Laissez-vous inspirer par le “double shift” algorithme de Francis.

# Chapitre VI

## Méthodes Itératives – 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  $f : \mathbb{R}^n \rightarrow \mathbb{R}^n$  donnée, on cherche un point  $x \in \mathbb{R}^n$  tel que

$$f(x) = 0. \quad (0.1)$$

En général, il n'y a pas d'algorithme fini pour trouver une solution. On est donc obligé d'utiliser des méthodes itératives.

Sans hypothèses supplémentaires, on ne sait rien sur l'existence d'une solution de (0.1). Par exemple, pour  $f(x) = e^x$  il n'y a pas de solution, pour  $f(x) = \sin x$  il y en a une infinité. Mais, le *théorème d'inversion locale* nous fournit un résultat sur l'unicité locale: si  $f(a) = 0$  et si la matrice jacobienne  $f'(a)$  est inversible, il existe un voisinage  $U$  de  $a$  et un voisinage  $V$  de 0 tels que  $f : U \rightarrow V$  soit bijective. Ceci implique que  $a$  est la seule solution de (0.1) dans le voisinage  $U$  de  $a$ .

### Bibliographie sur ce chapitre

- P. Deuflhard (2004): *Newton Methods for Nonlinear Problems. Affine Invariance and Adaptive Algorithms*. Springer Ser. Comp. Math. 35, Springer, Berlin.
- J.M. Ortega & W.C. Rheinboldt (1970): *Iterative Solution of Nonlinear Equations in Several Variables*. Academic Press, New York.
- A.M. Ostrowski (1966): *Solution of Equations and Systems of Equations*. Academic Press, New York, 2nd edition. [MA 65/27]

### VI.1 Méthode des approximations successives

On considère le problème du calcul d'un *point fixe* de l'application  $\Phi : \mathbb{R}^n \rightarrow \mathbb{R}^n$ ; c.-à-d., on cherche  $x \in \mathbb{R}^n$  tel que

$$x = \Phi(x). \quad (1.1)$$

Les problèmes (0.1) et (1.1) sont équivalents et il y a beaucoup de possibilités pour écrire (0.1) sous la forme (1.1). Par exemple, on peut définir  $\Phi$  comme  $\Phi(x) = x - f(x)$  ou  $\Phi(x) = x - Bf(x)$  (ici  $B$  est soit un nombre non-nul, soit une matrice bien choisie).

Pour résoudre (1.1), on se donne une approximation initiale  $x_0$  (arbitraire) et on considère la méthode itérative

$$x_{k+1} = \Phi(x_k). \quad (1.2)$$

Si la suite  $\{x_k\}$  converge, disons vers  $a \in \mathbb{R}^n$ , et si la fonction  $\Phi(x)$  est continue en  $a$ , la limite  $a$  est une solution de (1.1). Mais que peut-on dire sur l'erreur?

**Théorème 1.1** *Si  $\Phi(x)$  est 2 fois continûment différentiable et si  $a \in \mathbb{R}^n$  est une solution de (1.1), l'erreur  $e_k = x_k - a$  satisfait*

$$e_{k+1} = \Phi'(a) e_k + \mathcal{O}(\|e_k\|^2). \quad (1.3)$$

*Démonstration.* Comme  $a = \Phi(a)$ , on obtient

$$e_{k+1} = x_{k+1} - a = \Phi(x_k) - \Phi(a) = \Phi'(a)e_k + \mathcal{O}(\|e_k\|^2). \quad \square$$

On peut tirer plusieurs conclusions de la formule (1.3):

- si  $\Phi'(a)$  possède une valeur propre  $\lambda_1$  satisfaisant  $|\lambda_1| > 1$ , la composante de  $e_k$  dans la direction du vecteur propre  $v_1$  va être agrandie – l'itération *ne converge pas* vers  $a$ .
- si toutes les valeurs propres de  $\Phi'(a)$  satisfont  $|\lambda_i| < 1$ , on peut choisir une norme dans  $\mathbb{R}^n$  telle que pour la norme matricielle correspondante  $\|\Phi'(a)\| < 1$ . Ceci et (1.3) impliquent que, pour  $\|e_k\|$  suffisamment petit, on a  $\|e_{k+1}\| \leq \alpha \|e_k\|$  où  $\alpha$  est un nombre entre  $\|\Phi'(a)\|$  et 1. L'erreur  $e_k$  converge donc vers zéro.

**Exemple.** Pour résoudre numériquement  $y' = f(y)$ , on peut appliquer la méthode d'Euler implicite

$$y_1 = y_0 + h f(y_1) \quad (1.4)$$

qui définit implicitement l'approximation  $y_1$  de la solution après un pas de longueur  $h$ . L'équation (1.4) est déjà sous la forme (1.1) avec  $\Phi(x) = y_0 + h f(x)$ . Si  $h$  est suffisamment petit, les valeurs propres de  $\Phi'(y_1) = h f'(y_1)$  sont petites et l'itération (1.2) converge.

**Critère pour arrêter l'itération.** En pratique, on s'intéresse à une approximation  $x_k$  qui satisfasse  $\|x_k - a\| \leq tol$ . Une possibilité est d'accepter  $x_k$  comme approximation de la solution dès que  $\|x_k - x_{k-1}\| \leq tol$ .

Le critère qui va suivre est basé sur l'hypothèse que  $\Phi$  soit une contraction et que

$$\|x_k - x_{k-1}\| \leq \alpha \|x_{k-1} - x_{k-2}\| \quad \text{avec} \quad \alpha < 1.$$

En appliquant l'inégalité du triangle à l'identité

$$x_k - a = (x_k - x_{k+1}) + (x_{k+1} - x_{k+2}) + \dots$$

(en cas de convergence), on obtient

$$\|x_k - a\| \leq \frac{\alpha}{1 - \alpha} \|x_k - x_{k-1}\|. \quad (1.5)$$

Le facteur de contractivité peut être estimé par

$$\alpha_k = \|x_k - x_{k-1}\| / \|x_{k-1} - x_{k-2}\|, \quad k \geq 2.$$

L'idée est d'arrêter l'itération quand

$$\frac{\alpha_k}{1 - \alpha_k} \|x_k - x_{k-1}\| \leq tol \quad (1.6)$$

et d'accepter  $x_k$  comme approximation de la solution.

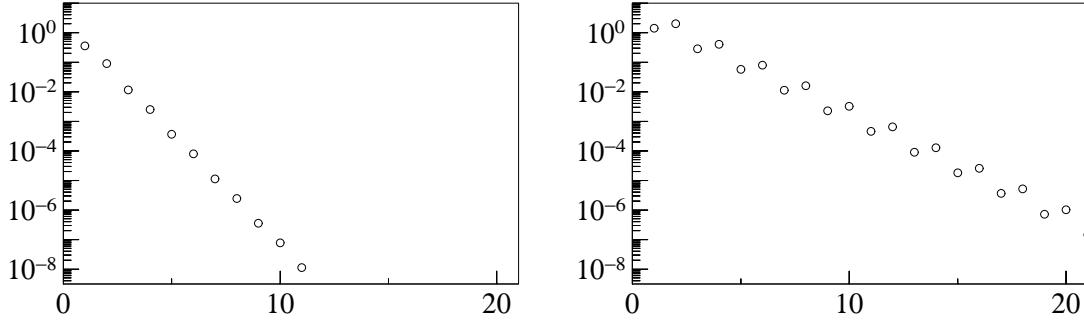


FIG. VI.1: Convergence des itérations (1.7)

**Exemples.** Pour les deux itérations

$$\begin{pmatrix} x_{k+1} \\ y_{k+1} \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix} + 0.3 \begin{pmatrix} y_k \\ -\sin x_k \end{pmatrix}, \quad \begin{pmatrix} x_{k+1} \\ y_{k+1} \end{pmatrix} = \begin{pmatrix} 0 & -0.1 \\ 2 & 0 \end{pmatrix} \begin{pmatrix} x_k \\ y_k \end{pmatrix} \quad (1.7)$$

la norme euclidienne de l'erreur de  $(x_k, y_k)$  est dessinée dans la figure VI.1 (à gauche pour la première itération et à droite pour la deuxième). On peut bien observer la convergence linéaire. Le rayon spectral de  $\Phi'(a)$  vaut  $\rho \approx 0.177$  et  $\rho \approx 0.447$  respectivement. C'est la raison pour laquelle la première itération converge plus rapidement que la seconde. Le dessin à droite montre aussi que la convergence n'est pas nécessairement monotone (pour la norme  $\|\cdot\|_2$ ). Donc, le coefficient  $\alpha_k$  de (1.6) peut être plus grand que 1 même si l'itération converge.

## VI.2 Méthodes itératives pour systèmes linéaires

Pour la résolution des systèmes linéaires

$$Ax = b, \quad (2.1)$$

il y a des situations où les méthodes itératives sont très utiles. Par exemple, si la matrice  $A$  possède une très grande dimension et si beaucoup d'éléments de  $A$  sont nuls (matrice creuse), ce qui est le cas pour les discrétisations des équations aux dérivées partielles.

Pour se ramener à un problème de point fixe, on considère une décomposition  $A = M - N$  (“splitting”) et on définit l'itération

$$Mx_{k+1} = Nx_k + b, \quad A = M - N. \quad (2.2)$$

Le choix de la décomposition  $A = M - N$  est important pour la performance de la méthode. D'une part,  $M$  doit être choisie telle que le système (2.2) soit beaucoup plus facile à résoudre que le système (2.1). D'autre part, les valeurs propres de la matrice  $M^{-1}N$  doivent satisfaire  $|\lambda_i| < 1$  pour que l'itération (2.2) converge.

Il y a beaucoup de possibilités de définir la décomposition  $A = M - N$ . Si l'on dénote

$$L = \begin{pmatrix} 0 & & & & \\ a_{21} & \ddots & & & \\ \vdots & \ddots & \ddots & & \\ a_{n1} & \dots & a_{n,n-1} & 0 & \end{pmatrix}, \quad U = \begin{pmatrix} 0 & a_{12} & \dots & a_{1n} \\ & \ddots & \ddots & \vdots \\ & & \ddots & a_{n-1,n} \\ & & & 0 \end{pmatrix}$$

et  $D = \text{diag}(a_{11}, \dots, a_{nn})$  (afin que  $A = L + D + U$ ) les itérations les plus connues sont:

- *Jacobi*:  $M = D, N = -L - U$ ;
- *Gauss-Seidel*:  $M = D + L, N = -U$ .

Pour ces deux méthodes, la matrice  $M$  est choisie de manière à ce que le système (2.2) soit très facile à résoudre. Un avantage de la méthode de Gauss-Seidel est le fait que pour le calcul de la composante  $x_i^{k+1}$  du vecteur  $x_{k+1}$  on n'a besoin que des composantes  $x_{i+1}^k, \dots, x_n^k$  du vecteur  $x_k$ . Alors, on peut utiliser la même variable pour  $x_i^{k+1}$  que pour  $x_i^k$ . Une itération devient donc simplement:

$$\begin{aligned} \textbf{for } i &= 1, \dots, n \\ x_i &= (b_i - \sum_{j=1}^{i-1} a_{ij}x_j - \sum_{j=i+1}^n a_{ij}x_j)/a_{ii}. \end{aligned}$$

Une modification intéressante de cet algorithme est la *méthode SOR* (“successive over-relaxation”):

$$\begin{aligned} x_{k+1} &= (1 - \omega)x_k + \omega\hat{x} \\ D\hat{x} &= b - Lx_{k+1} - Ux_k \end{aligned} \tag{2.3}$$

où  $\omega$  est un paramètre donné (pour  $\omega = 1$ , SOR se réduit à Gauss-Seidel). Elle est aussi simple à programmer que la précédente.

Les résultats suivants démontrent la convergence dans quelques situations particulières.

**Théorème 2.1** *Si la matrice  $A$  est “diagonale dominante”, c.-à-d.*

$$|a_{ii}| > \sum_{j \neq i} |a_{ij}| \quad \text{pour } i = 1, \dots, n, \tag{2.4}$$

*alors l’itération de Jacobi converge.*

*Démonstration.* Pour la méthode de Jacobi, on a  $M^{-1}N = -D^{-1}(L + U)$ . La condition (2.4) implique que  $\|M^{-1}N\|_\infty < 1$  (voir la formule (4.5) du chapitre IV).  $\square$

Pour étudier la convergence de la méthode SOR (en particulier pour Gauss-Seidel), on l’écrit sous la forme équivalente

$$(D + \omega L)x_{k+1} = ((1 - \omega)D - \omega U)x_k + \omega b$$

ou encore  $x_{k+1} = H(\omega)x_k + \omega(D + \omega L)^{-1}b$  avec

$$H(\omega) = (D + \omega L)^{-1}((1 - \omega)D - \omega U). \tag{2.5}$$

**Théorème 2.2** *Si  $A$  est symétrique et définie positive et si  $\omega \in (0, 2)$ , l’itération SOR, définie par (2.3), converge.*

*Démonstration.* Il faut démontrer que toutes les valeurs propres de  $H(\omega)$  satisfont  $|\lambda| < 1$ . Soient  $\lambda$  une valeur propre et  $x \neq 0$  le vecteur propre correspondant:

$$((1 - \omega)D - \omega U)x = \lambda(D + \omega L)x. \tag{2.6}$$

Comme  $(1 - \omega)D - \omega U = D - \omega A + \omega L$ , la formule (2.6) devient

$$\omega Ax = (1 - \lambda)(D + \omega L)x. \tag{2.7}$$

En multipliant (2.6) et (2.7) avec  $x^*$ , on obtient pour  $\alpha := x^*(D + \omega L)x$  (observer que  $U = L^T$ )

$$\lambda\alpha + \bar{\alpha} = (2 - \omega)x^*Dx > 0, \quad (1 - \lambda)\alpha = \omega x^*Ax =: \beta > 0. \tag{2.8}$$

On en déduit

$$0 < \lambda\alpha + \bar{\alpha} = \beta \left( \frac{\lambda}{1 - \lambda} + \frac{1}{1 - \bar{\lambda}} \right) = \beta \cdot \frac{1 - |\lambda|^2}{|1 - \lambda|^2},$$

ce qui implique  $|\lambda| < 1$ .  $\square$

Pour quelques matrices, on connaît la valeur de  $\omega$  qui minimise le rayon spectral de  $H(\omega)$  et, par conséquent, qui accélère la convergence par rapport à l’itération de Gauss-Seidel. D’autres méthodes itératives pour des systèmes linéaires sont SSOR (“symmetric successive over-relaxation”) et la méthode du gradient conjugué (avec préconditionnement). Voir le livre de Golub & Van Loan, déjà mentionné au chapitre IV, et les livres classiques

L.A. Hageman & D.M. Young (1981): *Applied Iterative Methods*. Academic Press, New York.

R.S. Varga (1962): *Matrix Iterative Methods*. Prentice Hall, Englewood Cliffs, New Jersey.

### VI.3 Méthode de Newton

Considérons le problème de la résolution d’un système d’équations non linéaires

$$f(x) = 0 \quad (3.1)$$

où la fonction  $f : \mathbb{R}^n \rightarrow \mathbb{R}^n$  est supposée être au moins une fois différentiable. Si  $x_0 \in \mathbb{R}^n$  est une approximation de la solution cherchée, on linéarise  $f(x)$  autour de  $x_0$

$$f(x) \approx f(x_0) + f'(x_0)(x - x_0)$$

et on calcule le zéro de cette linéarisation. Si l’on répète cette procédure avec la nouvelle approximation, on obtient l’algorithme suivant:

```
for  $k = 0, 1, 2, \dots$ 
    calculer  $f(x_k)$  et  $f'(x_k)$ 
     $f'(x_k)\Delta x_k = -f(x_k)$       (système linéaire à résoudre)
     $x_{k+1} = x_k + \Delta x_k$ 
end for
```

**Exemple 3.1** La méthode d’Euler implicite (voir (1.4)), appliquée à l’équation différentielle  $x' = y$ ,  $y' = 10(1 - x^2)y - x$  avec pour valeurs initiales  $\xi = 2$ ,  $\eta = -0.66$ , nous conduit à l’équation non linéaire

$$\begin{aligned} x &= \xi + h \cdot y \\ y &= \eta + h \cdot (10(1 - x^2)y - x). \end{aligned} \quad (3.2)$$

L’itération (1.2) ne converge que pour  $h$  suffisamment petit. Par exemple, pour  $h = 0.3$  elle diverge et on est obligé d’utiliser un autre algorithme. La méthode de Newton

$$\begin{pmatrix} 1 & -h \\ h(20x_k y_k + 1) & 1 - 10h(1 - x_k^2) \end{pmatrix} \begin{pmatrix} x_{k+1} - x_k \\ y_{k+1} - y_k \end{pmatrix} = - \begin{pmatrix} x_k - \xi - hy_k \\ y_k - \eta - h(10(1 - x_k^2)y_k - x_k) \end{pmatrix}$$

converge sans difficulté avec  $h = 0.3$  si l’on commence l’itération avec  $x_0 = \xi$  et  $y_0 = \eta$  (voir le tableau VI.1 pour les valeurs de  $x_k$ ,  $y_k$  et les erreurs, mesurées dans la norme euclidienne).

TAB. VI.1: Convergence de la méthode de Newton

$k$	$x_k$	$y_k$	erreur
0	2.000000000000000	-0.660000000000000	$5.31 \cdot 10^{-1}$
1	1.95099818511797	-0.163339382940109	$3.38 \cdot 10^{-2}$
2	1.96084279415163	-0.130524019494582	$4.27 \cdot 10^{-4}$
3	1.96072023704926	-0.130932543169149	$6.65 \cdot 10^{-8}$
4	1.96072021795300	-0.130932606823320	$1.79 \cdot 10^{-15}$

Pour étudier la convergence de la méthode de Newton, considérons un terme de plus dans la série de Taylor:

$$f(x) = f(x_k) + f'(x_k)(x - x_k) + \frac{1}{2}f''(x_k)(x - x_k, x - x_k) + \mathcal{O}(\|x - x_k\|^3). \quad (3.3)$$

Si l'on pose  $x = a$  dans cette formule (avec  $a$  comme solution de (3.1)) et si l'on soustrait

$$0 = f(x_k) + f'(x_k)(x_{k+1} - x_k)$$

(définition de  $x_{k+1}$ ), on obtient pour l'erreur  $e_k = x_k - a$  la formule

$$0 = f'(x_k)(-e_{k+1}) + \frac{1}{2}f''(x_k)(e_k, e_k) + \mathcal{O}(\|e_k\|^3).$$

On vient de démontrer le résultat suivant:

**Théorème 3.2** *Supposons que  $f(x)$  soit 3 fois continûment différentiable et que  $f'(x)$  soit inversible dans un voisinage de  $a$  (solution de (3.1)). Alors, pour  $x_k$  suffisamment proche de  $a$ , l'erreur  $e_k = x_k - a$  de la méthode de Newton satisfait*

$$e_{k+1} = \frac{1}{2}(f'(x_k))^{-1}f''(x_k)(e_k, e_k) + \mathcal{O}(\|e_k\|^3). \quad (3.4)$$

Donc, la convergence de cette méthode est quadratique.  $\square$

Ce théorème montre la convergence *locale* de la méthode de Newton, c.-à-d., si  $x_0$  est proche d'une solution  $a$  de (3.1), la suite  $\{x_k\}$  converge vers  $a$ . Concernant la convergence *globale*, on en sait très peu et on ne sait analyser seulement que quelques cas de fonctions simples. L'exemple le plus connu est

$$f(z) = z^3 - 1 \quad \text{ou} \quad f(x, y) = \begin{pmatrix} x^3 - 3xy^2 - 1 \\ 3x^2y - y^3 \end{pmatrix} \quad (3.5)$$

( $z = x + iy$ ) pour lequel l'itération devient

$$z_{k+1} = z_k - \frac{z_k^3 - 1}{3z_k^2} = \frac{1}{3}\left(2z_k + \frac{1}{z_k^2}\right). \quad (3.6)$$

Il est intéressant de déterminer les ensembles (bassins d'attraction)

$$A(a) = \{z_0 \in \mathbb{C} \mid \{z_k\} \text{ converge vers } a\} \quad (3.7)$$

pour les trois solutions  $1, (-1 \pm i\sqrt{3})/2$  de  $f(z) = 0$ . Un calcul par ordinateur donne la figure VI.2. Les  $z_0$  du domaine blanc entraînent une convergence vers  $a = 1$ , ceux du domaine gris vers  $a = (-1 - i\sqrt{3})/2$  et ceux du domaine noir vers  $a = (-1 + i\sqrt{3})/2$ . On observe que la suite  $\{z_k\}$  ne converge pas nécessairement vers la solution la plus proche de  $z_0$ .

## Calcul numérique de la matrice jacobienne

En pratique, il arrive souvent que la forme analytique de la matrice  $f'(x)$  est inconnue. Dans cette situation on approche les éléments  $\partial f_i / \partial x_j$  de la matrice jacobienne par

$$\frac{\partial f_i}{\partial x_j}(x_1, \dots, x_n) \approx \frac{f_i(x_1, \dots, x_j + \delta, \dots, x_n) - f_i(x_1, \dots, x_n)}{\delta}. \quad (3.8)$$

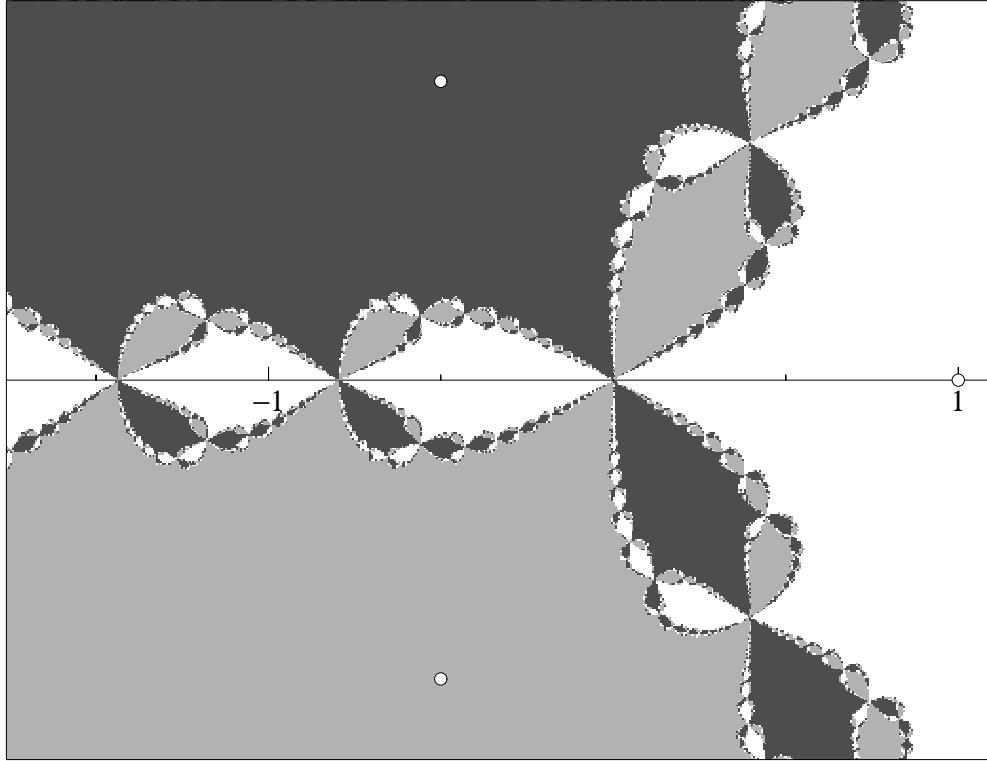


FIG. VI.2: Bassins d'attraction pour l'itération (3.6)

Mais comment doit-on choisir le  $\delta$ ? Pour simplifier la notation, considérons une fonction à une variable et notons-la par  $g(x)$ .

Un développement en série de Taylor montre que

$$\frac{g(x + \delta) - g(x)}{\delta} = g'(x) + \frac{\delta}{2}g''(x) + \mathcal{O}(\delta^2). \quad (3.9)$$

En conséquence,  $\delta$  doit être petit pour que *l'erreur de la discréétisation* ne soit pas trop grande. D'autre part, la soustraction dans (3.9) est très mal conditionnée. Une étude des *erreurs d'arrondi* montre que l'expression obtenue par un calcul en virgule flottante vaut

$$\begin{aligned} & \frac{g((x + \delta)(1 + \epsilon_1))(1 + \epsilon_2) - g(x)(1 + \epsilon_3)}{\delta} \\ & \approx \frac{g(x + \delta) - g(x)}{\delta} + \frac{1}{\delta} \left( g'(x + \delta)(x + \delta)\epsilon_1 + g(x + \delta)\epsilon_2 - g(x)\epsilon_3 \right) \end{aligned}$$

où  $|\epsilon_i| \leq \text{eps}$  ( $\text{eps}$  est la précision de l'ordinateur, voir section II.3). L'idée est de choisir  $\delta$  afin que les deux erreurs soient de la même grandeur :

$$\frac{\delta}{2}(\dots) \approx \frac{1}{\delta}(\dots)\text{eps}. \quad (3.10)$$

Donc, on prend par exemple

$$\delta = \sqrt{\text{eps}} \quad \text{ou} \quad \delta = \sqrt{\text{eps}(1 + |x|)}. \quad (3.11)$$

## Modifications de la méthode de Newton

Si la dimension du problème (3.1) est très grande et/ou l'évaluation de la matrice  $f'(x)$  est très coûteuse, on peut remplacer la définition de  $\Delta x_k$  par

$$f'(x_0)\Delta x_k = -f(x_k). \quad (3.12)$$

Dans ce cas, il suffit de calculer une fois pour toutes la décomposition LR de  $f'(x_0)$ , ce qui facilite grandement la résolution des systèmes linéaires successifs. Mais on perd la convergence quadratique car (3.12) n'est rien d'autre qu'une itération (1.2) avec  $\Phi(x) = x - (f'(x_0))^{-1}f(x)$  et  $\Phi'(a)$  est non nul en général.

En cas de mauvaise convergence, on remplace la définition de  $x_{k+1}$  par

$$x_{k+1} = x_k + \lambda_k \Delta x_k \quad (3.13)$$

et on détermine  $\lambda_k$  de façon à ce que  $\|f(x_k + \lambda \Delta x_k)\|$  soit minimale.

## VI.4 Méthode de Gauss-Newton

Dans ce paragraphe, on va s'intéresser à des systèmes non linéaires surdéterminés. On considère une fonction  $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$  où  $m > n$  et on cherche une solution de  $f(x) = 0$ . 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  $x \in \mathbb{R}^n$  tel que

$$\|f(x)\|_2 \rightarrow \min. \quad (4.1)$$

Si  $f(x)$  est différentiable, la fonction  $F(x) = \|f(x)\|_2^2$  est aussi différentiable et une condition nécessaire pour que  $x$  soit un minimum local est d'avoir  $F'(x) = 0$ , c.-à-d.

$$f'(x)^T f(x) = 0. \quad (4.2)$$

Une possibilité pour résoudre (4.1) est d'appliquer une méthode itérative, par exemple la méthode de Newton, au système (4.2). Ceci nécessite le calcul de la deuxième dérivée de  $f(x)$ .

Une autre possibilité est de linéariser  $f(x)$  dans (4.1) autour d'une approximation  $x_0$  de la solution et de calculer  $x_1$  de

$$\|f(x_0) + f'(x_0)(x_1 - x_0)\|_2 \rightarrow \min. \quad (4.3)$$

Une répétition de cette idée donne l'algorithme suivant (méthode de Gauss-Newton)

```

for  $k = 0, 1, 2, \dots$ 
    calculer  $f(x_k)$  et  $f'(x_k)$ 
    déterminer  $\Delta x_k$  de  $\|f(x_k) + f'(x_k)\Delta x_k\|_2 \rightarrow \min$  (moindres carrés)
     $x_{k+1} = x_k + \Delta x_k$ 
end for
```

Pour calculer  $\Delta x_k$  on peut soit résoudre les équations normales (section IV.6)

$$(f'(x_k))^T f'(x_k) \Delta x_k = -(f'(x_k))^T f(x_k) \quad (4.4)$$

soit calculer la décomposition QR de  $f'(x_k)$  et appliquer l'algorithme de la section IV.7.

## Etude de la convergence

Pour étudier la convergence de la méthode de Gauss-Newton, développons la fonction

$$g(x) = (f'(x))^T f(x), \quad g_i(x) = \sum_{j=1}^m \frac{\partial f_j}{\partial x_i}(x) f_j(x) \quad (4.5)$$

en série de Taylor. Pour ceci, calculons

$$\frac{\partial g_i}{\partial x_\ell} = \sum_{j=1}^m \frac{\partial^2 f_j}{\partial x_\ell \partial x_i}(x) f_j(x) + \sum_{j=1}^m \frac{\partial f_j}{\partial x_i}(x) \frac{\partial f_j}{\partial x_\ell}(x). \quad (4.6)$$

Matriciellement, la formule (4.6) s'écrit

$$g'(x) = B(x)(f(x), \cdot) + (f'(x))^T f'(x) \quad (4.7)$$

où  $B(x) : \mathbb{R}^m \times \mathbb{R}^n \rightarrow \mathbb{R}^n$  est l'application bilinéaire définie par

$$(B(x)(u, v))_i = \sum_{\ell=1}^n \sum_{j=1}^m \frac{\partial^2 f_j}{\partial x_\ell \partial x_i}(x) \cdot u_j \cdot v_\ell.$$

Avec cette notation, la formule de Taylor donne

$$g(x) = (f'(x_k))^T f(x_k) + (f'(x_k))^T f'(x_k)(x - x_k) + B(x_k)(f(x_k), x - x_k) + \mathcal{O}(\|x - x_k\|^2). \quad (4.8)$$

Soit maintenant  $a$  une solution de (4.1). Elle satisfait  $g(a) = 0$ . En posant  $x = a$  dans (4.8) et en soustrayant (4.4), nous obtenons ainsi le résultat suivant.

**Théorème 4.1** *Supposons que  $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$  (avec  $m > n$ ) soit 3 fois continûment différentiable, que le rang de  $f'(x)$  soit maximal et que  $a$  soit une solution de (4.1). Alors, pour  $x_k$  suffisamment proche de  $a$ , l'erreur  $e_k = x_k - a$  de la méthode de Gauss-Newton satisfait*

$$e_{k+1} = -((f'(x_k))^T f'(x_k))^{-1} B(x_k)(f(x_k), e_k) + \mathcal{O}(\|e_k\|^2).$$

□

Une conséquence de cette formule est :

- en général, la convergence est linéaire;
- on a convergence quadratique si  $f(a) = 0$ ;
- si  $f(a)$  est trop grand, la méthode diverge.

Terminons ce chapitre avec une application typique de cet algorithme.

**Exemple (Identification de paramètres).** La figure VI.3 montre une photographie de la Vallée Blanche (prise par G. Wanner). On y reconnaît le Col des Grandes Jorasses, l'Aiguille du Géant, l'Aiguille Blanche de Peterey, l'Aiguille du Tacul, le Petit Rognon et l'Aiguille du Moine. La figure VI.4 est une copie d'une carte géographique de cette région. Le problème consiste à trouver la position de la caméra, ses caractéristiques (foyer) et les angles d'inclinaison.

Pour la formulation mathématique de ce problème, nous avons choisi des coordonnées  $(u, v)$  sur la photographie (figure VI.3, l'origine est au centre) et des coordonnées  $(x, y, z)$  sur la carte ( $z$  représente l'altitude). Les valeurs mesurées pour les 6 points reconnus sont données dans le tableau VI.2.

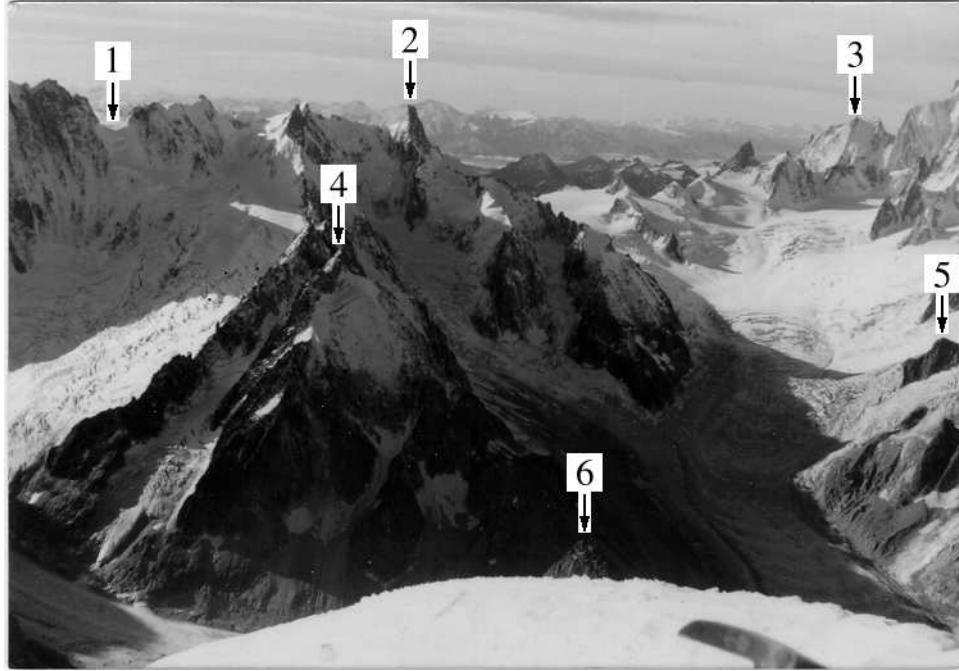


FIG. VI.3: Photographie de la Vallée Blanche

TAB. VI.2: Les données pour le problème “Vallée Blanche”

$k$		$u_k$	$v_k$	$x_k$	$y_k$	$z_k$
1.	Col des Grandes Jorasses	-0.0480	0.0290	9855	5680	3825
2.	Aiguille du Géant	-0.0100	0.0305	8170	5020	4013
3.	Aig. Blanche de Peterey	0.0490	0.0285	2885	730	4107
4.	Aiguille de Tacul	-0.0190	0.0115	8900	7530	3444
5.	Petit Rognon	0.0600	-0.0005	5700	7025	3008
6.	Aiguille du Moine	0.0125	-0.0270	8980	11120	3412

Pour fixer les *inconnues*, notons par  $(\hat{x}, \hat{y}, \hat{z})$  la position du foyer de la caméra, par le vecteur  $(a, b, c)$  la direction de vue avec norme correspondant à la distance du plan du film et par  $\theta$  l'angle de rotation de la caméra autour du vecteur  $(a, b, c)$ . On a donc 7 paramètres à trouver. De plus, considérons la base orthogonale

$$\begin{pmatrix} a \\ b \\ c \end{pmatrix}, \quad h = \frac{1}{\sqrt{a^2 + b^2}} \begin{pmatrix} b \\ -a \\ 0 \end{pmatrix}, \quad g = \frac{1}{\sqrt{(a^2 + b^2)(a^2 + b^2 + c^2)}} \begin{pmatrix} -ac \\ -bc \\ a^2 + b^2 \end{pmatrix} \quad (4.9)$$

dont les deux vecteurs  $h$  et  $g$  engendrent le plan du film,  $h$  étant horizontal et  $g$  vertical. Alors, le vecteur dans l'espace qui montre du centre  $(\hat{x}, \hat{y}, \hat{z})$  de la lentille vers le point  $(u_k, v_k)$  sur le film est donné par

$$\begin{pmatrix} w_{1k} \\ w_{2k} \\ w_{3k} \end{pmatrix} = \begin{pmatrix} a \\ b \\ c \end{pmatrix} + \alpha_k \cdot h + \beta_k \cdot g \quad \text{où} \quad \begin{pmatrix} \alpha_k \\ \beta_k \end{pmatrix} = \begin{pmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{pmatrix} \begin{pmatrix} u_k \\ v_k \end{pmatrix}.$$

Les conditions à satisfaire sont que les vecteurs  $(w_{1k}, w_{2k}, w_{3k})$  et  $(x_k - \hat{x}, y_k - \hat{y}, z_k - \hat{z})$  soient

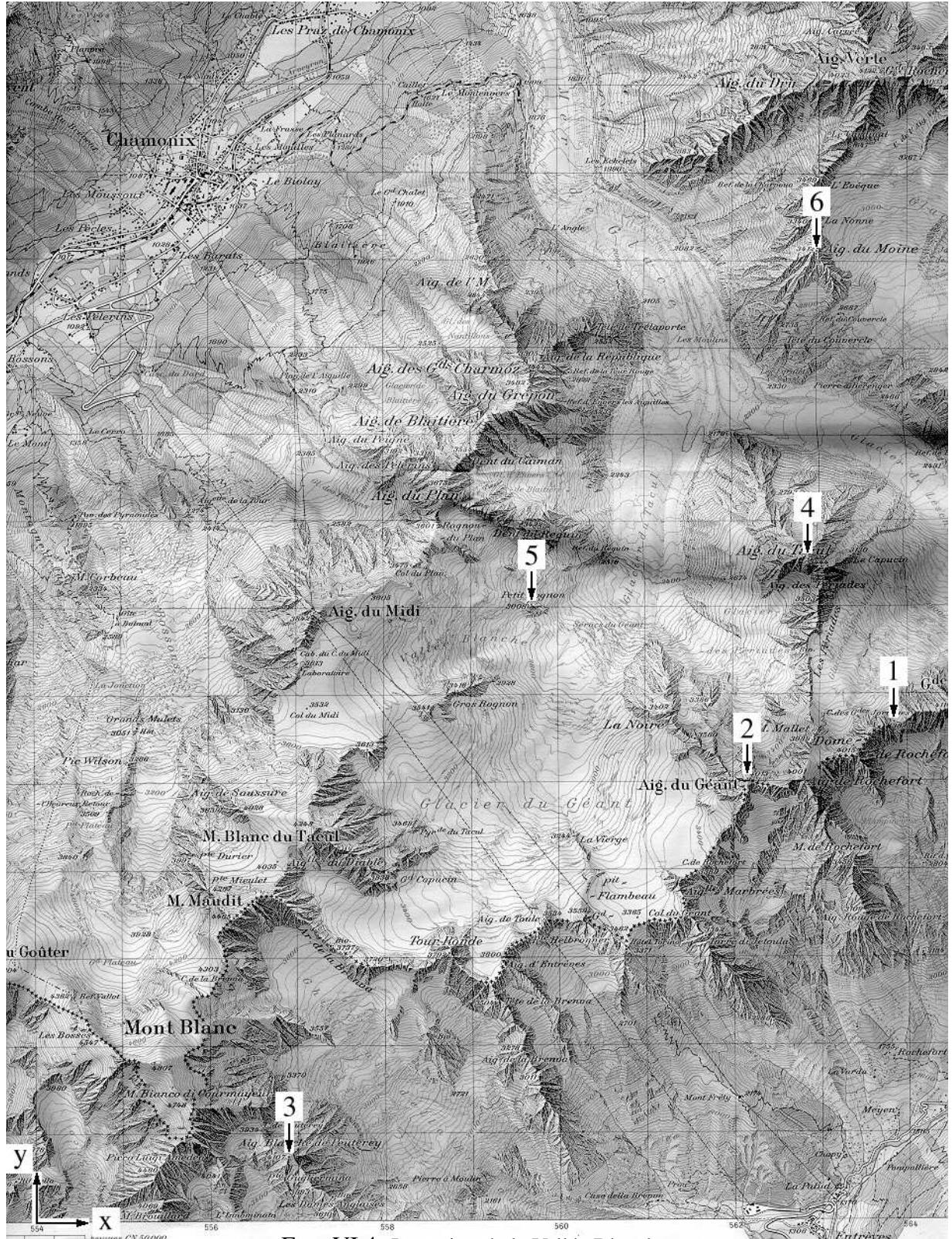


FIG. VI.4: La région de la Vallée Blanche

colinéaires. Ceci donne deux conditions pour chaque  $k$ . Mais, par symétrie, il est plus naturel de considérer les trois conditions (pour chaque  $k$ )

$$0 = \begin{pmatrix} w_{1k} \\ w_{2k} \\ w_{3k} \end{pmatrix} \times \begin{pmatrix} x_k - \hat{x} \\ y_k - \hat{y} \\ z_k - \hat{z} \end{pmatrix} = \begin{pmatrix} w_{2k}(z_k - \hat{z}) - w_{3k}(y_k - \hat{y}) \\ w_{3k}(x_k - \hat{x}) - w_{1k}(z_k - \hat{z}) \\ w_{1k}(y_k - \hat{y}) - w_{2k}(x_k - \hat{x}) \end{pmatrix}. \quad (4.10)$$

En tout, ceci donne  $3 \cdot 6 = 18$  conditions pour 7 inconnues. Voici le programme FORTRAN pour la fonction  $f : \mathbb{R}^7 \rightarrow \mathbb{R}^{18}$ .

```

SUBROUTINE FCN(N,XSOL,M,F)
IMPLICIT REAL*8 (A-H,O-Z)
PARAMETER(ND=50)
DIMENSION XSOL(N),F(M)
COMMON/DAT/NDAT,XDAT(ND),YDAT(ND),ZDAT(ND),UDAT(ND),VDAT(ND)
C ----- NOTATION LOCALE -----
X0=XSOL(1)
Y0=XSOL(2)
Z0=XSOL(3)
A=XSOL(4)
B=XSOL(5)
C=XSOL(6)
THETA=XSOL(7)
C ----- VECTEUR HORIZONTAL -----
RAC=SQRT(A*A+B*B)
H1=B/RAC
H2=-A/RAC
H3=0.
C ----- VECTEUR VERTICAL -----
RAC=SQRT((A*C)**2+(B*C)**2+(A**2+B**2)**2)
G1=-A*C/RAC
G2=-B*C/RAC
G3=(A**2+B**2)/RAC
C ----- LES POINTS -----
DO I=1,NDAT
C ----- ROTATION INITIALE -----
U= UDAT(I)*COS(THETA)+VDAT(I)*SIN(THETA)
V=-UDAT(I)*SIN(THETA)+VDAT(I)*COS(THETA)
C ----- LES VECTEURS A ALIGNER -----
W1=A+H1*U+G1*V
W2=B+H2*U+G2*V
W3=C+H3*U+G3*V
Q1=XDAT(I)-X0
Q2=YDAT(I)-Y0
Q3=ZDAT(I)-Z0
C ----- LES FONCTIONS A ANNULER -----
F(3*(I-1)+1)=W1*Q2-W2*Q1
F(3*(I-1)+2)=W2*Q3-W3*Q2
F(3*(I-1)+3)=W3*Q1-W1*Q3
END DO
RETURN
END

```

En appliquant la méthode de Gauss-Newton à ce système avec comme valeurs initiales  $\hat{x} = 8000$ ,  $\hat{y} = 15000$ ,  $\hat{z} = 1000$ ,  $a = 0$ ,  $b = -1$ ,  $c = 0$ ,  $\theta = 0$ , après peu d’itérations on obtient la solution  $\hat{x} = 9664$ ,  $\hat{y} = 13115$ ,  $\hat{z} = 4116$  avec une précision de 5 chiffres (voir le tableau VI.3). On a donc bien trouvé la position de la caméra. C’est à l’Aiguille Verte (altitude 4122) que Gerhard a pris cette photo.

TAB. VI.3: Convergence de la méthode de Gauss-Newton

$k$	$\hat{x}_k$	$\hat{y}_k$	$\hat{z}_k$	$a$	$b$	$c$	$\theta$
0	8000	15000	1000	0.000	-1.000	0.000	0.000
1	8030	9339	1169	-0.003	-0.085	-0.003	0.047
2	8680	11163	4017	-0.014	-0.114	-0.021	0.017
3	9577	13034	3993	-0.040	-0.167	-0.032	-0.094
4	9660	13107	4116	-0.043	-0.169	-0.032	-0.074
5	9664	13115	4116	-0.043	-0.169	-0.032	-0.074
6	9664	13115	4116	-0.043	-0.169	-0.032	-0.074

## VI.5 Exercices

1. Pour une fonction  $f : \mathbf{R} \rightarrow \mathbf{R}$ , considérons l’itération  $(x_k, x_{k-1}) \rightarrow x_{k+1}$  définie par

$$0 = f(x_k) + \frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}}(x_{k+1} - x_k). \quad (5.1)$$

- (a) Donner une interprétation géométrique.
- (b) Démontrer que l’erreur  $e_k = |x_k - \zeta|$ , où  $\zeta$  est une racine simple de  $f(x) = 0$ , satisfait

$$e_{k+1} \approx Ce_k e_{k-1}, \quad \text{avec} \quad C = \left| \frac{f''(\zeta)}{2f'(\zeta)} \right|.$$

- (c) Déduire de (b) que

$$e_{k+1} \approx De_k^p, \quad \text{avec} \quad p = \frac{1}{2}(1 + \sqrt{5}) \approx 1.618.$$

- (d) Soit  $f(x)$  un polynôme, nous avons que le travail pour évaluer  $f(x)$  et  $f'(x)$  est approximativement le mme. Supposons que le coût des opérations d’additions, de soustractions, de multiplications et de divisions sont négligeables par rapport à l’évaluation de  $f(x)$ . Quelle méthode choisiriez-vous entre la méthode de Newton et la méthode de la sécante (5.1) pour calculer une racine de  $f(x)$  à travail égal?
2. Soit  $D \subset \mathbf{R}$  et  $f : D \rightarrow \mathbf{R}^n$  continûment différentiable. Pour un  $x_0 \in D$  supposons que  $f(x_0) \neq 0$  et que  $f'(x_0)$  soit inversible. Montrer que

$$p_0 = -f'(x_0)^{-1}f(x_0)$$

est la seule direction ayant la propriété suivante:

Pour toute matrice inversible  $A$ , il existe  $\lambda_0 > 0$  tel que pour  $0 < \lambda < \lambda_0$

$$\|Af(x_0 + \lambda p_0)\|_2 < \|Af(x_0)\|_2.$$

# Chapitre VII

## Travaux Pratiques

Cet appendice contient une collection de travaux pratiques, dont les thèmes reprennent (dans l'ordre chronologique) les sujets abordés dans le cours. La dernière partie de cet appendice est consacrée à une introduction au language FORTAN90/95 qui n'a pas, et de loin, la prétention d'être complète.

### VII.1 Introduction au FORTRAN 90

1. Le programme ci-dessous évalue la fonction  $\sin(x) \cdot x$  en des points équidistants  $h = \frac{i}{n}\pi$ , où  $i = 1, 2, \dots, 20$  et  $n = 20$ . Essayer de le comprendre, puis dessiner au moyen de **gnuplot** le graphe  $(x_i, f(x_i))$ , pour  $n = 20, 50$  dans les intervalles  $[0, \pi]$  et  $[0, 2\pi]$ .

```
program eval ! ce programme \'evalue une fonction
    implicit none
    integer, parameter :: n=20
    integer :: i
    integer, parameter :: dp=kind(1.d0)
    real(kind=dp), parameter :: pi=3.14159265358979324_dp
    real(kind=dp) :: v,x,rn
    real(kind=dp) :: f
    open (8,file='sleval.out') ! fichier qui stocke les donn\'ees
    rn=n
    do i=1,n
        x=(i/rn)*pi
        v=f(x)
        write(8,100) x,v ! \'ecrit (x_i,f(x_i)) sleval.out
        100 format(2f18.15) ! format des donn\'ees
    end do
end program eval

function f(x) ! fonction \'a \'evaluer
    implicit none
    integer, parameter :: dp=kind(1.d0)
    real(kind=dp) :: x,f
    f=sin(x)*x
end function
```

2. Ecrire (sur le modèle du programme ci-dessus) un programme qui évalue la fonction

$$f(x) = \begin{cases} 2x & \text{si } 0 \leq x \leq \frac{1}{2} \\ 2 - 2x & \text{si } \frac{1}{2} < x \leq 1 \end{cases}$$

en  $n$  points équidistants de  $(0, 1)$  ( $n$  étant choisi par l'utilisateur). Dessiner les résultats.

3. Ecrire un programme qui détermine le plus petit  $n$  tel que  $\sum_{k=1}^n \frac{1}{k} \geq M$  où  $M$  est donné par l'utilisateur. Combien de termes faut-il si  $M = 3$  ? Même question en remplaçant  $\frac{1}{k}$  par  $\frac{1}{k^2}$  dans la fonction.

## VII.2 Intégration par la règle du Trapèze et Simpson

Le programme ci-dessous évalue l'intégrale  $\int_0^3 \cos(x)e^{\sin(x)}$  avec la règle du trapèze. Donner le résultat pour  $n = 2, 4, 8, 16, 32$ , et dessiner sur une échelle logarithmique l'erreur en fonction du nombre d'évaluations de la fonction. Adapter ensuite ce programme pour la règle de Simpson.

```
program integration ! int\'egration avec la r\'egle du trap\`eze
implicit none
integer, parameter :: dp=kind(1.d0)
integer :: i,n, ...
real(kind=dp) :: a,b,res,err, ...
open (7,file='trapeze.out')
a=0._dp
b=3._dp
n=1
call trapeze(a,b,n,res)
write(7,*) 'n= ',n,' res= ',res ! \'ecrit dans trapeze.out
subroutine trapeze(a,b,n,res) ! m\'ethode du trap\`eze
implicit none
integer, parameter :: dp=kind(1.d0)
integer :: n,i
real(kind=dp) :: a,b,res,f,h
h=(b-a)/n
res=0.5_dp*(f(a)+f(b))
do i=1,n-1
    res=res+f(i*h)
end do
res=res*h
end subroutine trapeze

function f(x) ! fonction \`a integrer
implicit none
integer, parameter :: dp=kind(1.d0)
real(kind=dp) :: x,f
f=cos(x)*exp(sin(x))
end function
```

### VII.3 Calcul de racines par la méthode de la bissection

1. Ecrire un programme qui pour  $n$  donné calcule les racines de  $P_n(x)$ , le  $n$ ième polynôme de Legendre, en utilisant la méthode de bissection.

Indications:

- (a) Localiser les racines de  $P_n(x)$  en utilisant celles de  $P_{n-1}(x)$ .
- (b) Si  $[a, b]$  est un intervalle avec  $P_n(a) \cdot P_n(b) < 0$ , poser  $\text{centre} = \frac{a+b}{2}$ ,  $\text{pa} = P_n(a)$ ,  $\text{pb} = P_n(b)$ ,  $\text{pc} = P_n(\text{centre})$ .
- (c) Tant que  $P_n(\text{centre}) \neq 0$  et  $a \neq \text{centre}$  et  $b \neq \text{centre}$ , itérer
  - i. si  $P_n(a) \cdot P_n(\text{centre}) < 0$  poser  $b = \text{centre}$ ,  $\text{pb} = \text{pc}$
  - ii. sinon poser  $a = \text{centre}$ ,  $\text{pa} = \text{pc}$

Pour écrire la fonction `p(n, x)` qui calcule les polynômes de Legendre  $P_n(x)$ , utiliser la formule de récurrence pour ces polynômes qui s'écrit

$$(n+1)P_{n+1}(x) = (2n+1)xP_n(x) - nP_{n-1}(x)$$

2. A l'aide du programme obtenu en 1, calculer les poids  $b_i$  pour une formule de Gauss d'ordre 30.

### VII.4 Wegral: programme d'intégration d'ordre 8

1. Le but de ce tp est d'écrire un programme adaptatif d'intégration numérique basé sur l'algorithme expliqué au cours (I.6). On vous propose d'utiliser la formule de Weddle d'ordre 8 (voir polycopié p.3). Il faut d'abord trouver une méthode emboîtée convenable, par exemple la méthode de Newton d'ordre 4, puis programmer l'algorithme de division. Pour l'erreur sur un sous-intervalle utiliser

$$\text{abserr} = \text{abs}(\text{res} - \text{resem}),$$

où `res` est le résultat obtenu avec la formule de Weddle et `resem` celui obtenu avec la formule emboîtée. Le programme pourrait avoir la structure suivante:

```
program wegral ! programme adaptatif d'integration
implicit none
integer, parameter :: dp=kind(1.d0)
integer, parameter :: maxdiv=1000 ! borne les subdivisions
integer :: ndiv, ...
real (kind=dp), parameter :: a=..., b=..., tol=10._dp**(-14)
real(kind=dp) :: centre,errtot,weedabs, ...
...
! ----- premiere integration h=b-a
...
call weddle(..,..,..)
! ----- l'algorithme de subdivision
do ndiv=2,maxdiv
! ----- on teste si la precision est suffisante
  if (errtot <= tol*weedabs) then
```

```

    ...
end if
! ----- on cherche l'erreur maximale
...
! ----- on divise l'intervalle o\`u l'erreur est maximale
centre=
call weddle(.,centre,...)
call weddle(centre,...)
...
end do
...
end program wegral

subroutine weddle(a,b,res,abserr,resabs)
implicit none
integer, parameter :: dp=kind(1.d0)
integer :: i, ...
real(kind=dp) :: a,b,res,ensem,abserr,resabs,f,h, ...
real (kind=dp),dimension(4):: poidw
real (kind=dp),dimension(2):: poidn
data (poidw(i),i=1,4)/ 4.88095238095238096E-2_dp, \&
0.25714285714285712_dp, 3.21428571428571397E-2_dp, \&
0.32380952380952382_dp/
data (poidn(i),i=1,2)/ 0.125_dp, 0.375_dp/
...
end subroutine weddle

function f(x)
...
end function

```

## 2. Calculer à l'aide de votre programme les intégrales suivantes

$$\int_0^{10} e^{-x^2} dx, \quad \int_0^1 \sin x^2 dx, \quad \int_0^1 \cos x^2 dx.$$

## 3. On sait que

$$\int_{-1}^3 (1 + ce^{-c^2 x^2}) dx \longrightarrow 4 + \sqrt{\pi} \quad \text{pour } c \rightarrow \infty.$$

Appliquer votre programme à l'intégrale ci-dessus avec plusieurs valeurs de  $c$  croissantes ( $c = 1, 2, \dots, 10, \dots$ ). Observer et expliquer ce qui se passe.

## VII.5 Interpolation

- Le but de ce tp est d'écrire un programme newton qui interpole une fonction donnée grâce au polynôme d'interpolation de Newton. Ensuite, on va évaluer ce polynôme à l'aide de la méthode de Horner (voir exercices théoriques) et calculer l'erreur entre une fonction donnée et son polynôme d'interpolation.

**Remarque:** Une façon plus élégante pour définir la précision des variables est d'utiliser un module.

```
module accuracy
    integer, parameter :: dp=kind(1.d0)
end module accuracy
```

Il suffit d'insérer au début du programme principal ainsi que dans les sous-routines l'instruction `use accuracy` et l'extension `dp` peut être utilisée.

On vous propose la démarche suivante pour écrire votre programme:

- Ecrire une sous-routine `equidist` ainsi qu'une sous-routine `chebychev` pour calculer des noeuds équidistants et des noeuds de Chebyshev respectivement. La sous-routine `equidist` pourrait avoir la structure suivante:

```
subroutine equidist(a,b,n,grid)
    use accuracy ! l'extension dp est definie par le module
    implicit none
    real (kind=dp), dimension(0:n) :: grid
    real (kind=dp) :: a, b
    integer :: n
```

- Ecrire une sous-routine `diffdiv` qui calcule les différences divisées. Pour cela on a seulement besoin d'un vecteur `dd` pour le résultat. La sous-routine `diffdiv` pourrait avoir la structure suivante:

```
subroutine diffdiv(n,noeuds,dd)
    use accuracy
    implicit none
    real (kind=dp), dimension(0:n) :: dd
    real (kind=dp), dimension(0:n) :: noeuds
    integer :: n
```

- Après avoir écrit le programme `newton`, écrire une fonction `horner` qui évalue ce polynôme à l'aide de la méthode de Horner. La sous-routine `horner` pourrait avoir la structure suivante:

```
function horner(n,dd,noeuds,x)
    use accuracy
    implicit none
    real (kind=dp) :: horner
    real (kind=dp) :: x
    real (kind=dp), dimension(0:n) :: dd, noeuds
    integer :: n
```

3. En utilisant gnuplot vérifier votre programme en dessinant les fonctions ci-dessous ainsi que leurs polynômes d'interpolation.

$$\sin x, \quad x \in [0, 5\pi],$$

$$\frac{1}{1+25x^2}, \quad x \in [-1, 1],$$

$$e^{-1/(1+x^2)}, \quad x \in [-4, 4],$$

$$e^{-x^2}, \quad x \in [-4, 4].$$

4. Ecrire une sous-routine qui calcule l'erreur maximale

$$\max_{x_i} |f(x_i) - p(x_i)| \quad i = 0, 1, \dots, m,$$

sur une grille de  $m$  points donnés. Tester cette sous-routine sur les fonctions ci-dessus.

## VII.6 Interpolation et erreur

Ecrire un programme qui, pour une fonction donnée, calcule ses polynômes d'interpolations (pour des points équidistants et pour les points de Chebyshev), ainsi que l'erreur dans les différentes normes suivantes:

$$\begin{aligned} \|f - p\|_\infty &= \max_{x \in [a,b]} |f(x) - p(x)| \text{ norme maximale,} \\ \|f - p\|_{L^1} &= \int_a^b |f(x) - p(x)| \, dx \text{ norme } L^1, \\ \|f - p\|_{L^2} &= \left( \int_a^b |f(x) - p(x)|^2 \, dx \right)^{1/2} \text{ norme } L^2. \end{aligned}$$

Pour cela il faudra utiliser le programme d'intégration `wegral` ainsi que les programmes d'interpolations du tp précédent.

1. Il sera utile de pouvoir passer une sous-routine comme argument d'une fonction en utilisant l'instruction `external`. Par exemple la fonction `fmax`, qui calcule le maximum de `fcn` (sous-routine) sur un intervalle  $[a, b]$

```
function fmax(fcn,a,b,m)
  use accuracy
  implicit none
  integer :: m
  real (kind=dp) :: a,b
  external fcn
  real (kind=dp) :: fmax, ...
  ...
  call fcn(fmax,a)
  ...
end function fmax
```

a comme argument la sous-routine `fcn`

```
subroutine fcn(x,fx)    ! fx=f(x)
  ...

```

Compléter cette fonction `fmax`, nécessaire pour le calcul de l'erreur dans la norme maximale.

2. Ecrire des sous-routines `errequ` et `errch` qui déterminent l'erreur  $|f(x) - p(x)|$  (au point  $x$ ) entre  $f$  et ses polynômes d'interpolation  $p$  (basé respectivement sur des points équidistants et sur les points de Chebyshev).

```
subroutine errequ(x,e) ! evaluate e=|f(x)-p(x)|
    use accuracy
    use equi
    implicit none
    real (kind=dp) :: e
    real (kind=dp) :: x
    ...
end subroutine errequ
```

Cette sous-routine va faire appel à la fonction `horner`. Pour pouvoir utiliser les paramètres de la fonction `horner` sans les passer comme arguments, on peut utiliser un module. En définissant

```
module equi
    use accuracy
    real (kind=dp),dimension(:),allocatable :: noeuds, dd
    integer :: n
end module equi
```

et par l'instruction `use equi` dans la sous-routine `errequ`, les variables contenues dans le module `equi` sont visibles et utilisables dans cette sous-routine. Ces variables sont les mêmes que dans le programme `newton` si l'on ajoute dans ce dernier l'instruction `use equi` (elles n'ont alors plus besoin d'être définies dans le programme `newton`). De même, on utilisera pour la sous-routine `errch` un module similaire.

3. Modifier le programme `wegral` (VII.4) pour en faire une fonction

```
function wegral(fcn,a,b,tol)
    implicit none
    integer, parameter :: dp=kind(1.d0)
    real (kind=dp) :: a,b,tol
    real (kind=dp) :: wintegral
    external fcn
```

4. Finalement, modifier le programme `newton` pour en faire un programme qui, pour une fonction donnée, calcule ses polynômes d'interpolation (pour des points équidistants et pour les points de Chebyshev), ainsi que l'erreur dans les différentes normes du début de l'énoncé. Appliquer votre programme aux fonctions de la série 4 et donner les différentes erreurs pour des polynômes jusqu'au degré 20.

## VII.7 Résolution d'équations différentielles et calcul de la trajectoire des planètes

- On aimerait résoudre l'équation différentielle

$$\begin{aligned} y'_1 &= y_2 & y_1(0) &= 0 \\ y'_2 &= -y_1 & y_2(0) &= 1 \end{aligned}$$

dans  $[0, \frac{\pi}{4}]$  avec les méthodes d'Euler et de Runge (voir le cours). Donner les résultats numériques et calculer l'erreur en  $x = \frac{\pi}{4}$  (on connaît la solution exacte de cette équation différentielle). Pour cela il faudra écrire des sous-routines Euler et Runge.

```
subroutine euler(n, fcn, y, x, xend, npas)
...
subroutine runge(n, fcn, y, x, xend, npas)
...
! n dimension du syst\`eme
! npas le nombre de pas dans l'intervalle [x, xend]
! fcn la sous-routine qui contient la fonction y'=fcn(x, y)
! y vecteur qui contient les valeurs initiales \`a l'entr\'ee
!                               les valeurs y(xend) \`a la sortie
```

Résoudre ensuite à l'aide de votre programme l'équation différentielle de Riccati (1712)

$$y' = x^2 + y^2, \quad y(0) = 0, \quad y\left(\frac{1}{2}\right) = ?$$

Combien d'évaluations de la fonction sont nécessaires pour arriver à une précision de 6 chiffres, respectivement pour la méthode d'Euler et de Runge ?

Remarque: Cette équation différentielle ne possède pas de solution élémentaire.

Résultat (solution de référence):  $y\left(\frac{1}{2}\right) = 0.04179114615468186322076$ .

- Ecrire une sous-routine

```
subroutine rk4(n, fcn, y, x, xend, h, tol)
```

qui permet la résolution d'un système d'équations différentielles à l'aide de pas variables. On veut que le programme puisse choisir le pas d'intégration  $h$  afin que l'erreur locale soit partout environ égale à la tolérance  $tol$ , fournie par l'utilisateur. Prenez la méthode Runge-Kutta 3/8 avec la méthode emboîtée vue au cours.

Tester votre programme sur les équations différentielles ci-dessus.

- Utiliser vos programmes pour calculer la position autour du soleil des 5 planètes Jupiter, Saturne, Uranus, Neptune, Pluton en septembre 2004. Effectuer des calculs avec différents pas et différentes tolérances et observer la convergence.

La loi de Newton  $f = ma$  conduit aux équations du mouvement suivantes

$$q''_k = -G \sum_{j=0, j \neq k}^5 \frac{m_j(q_k - q_j)}{\|q_k - q_j\|^3}$$

où les  $q_j \in \mathbb{R}^3$  représentent la position des planètes et du soleil,  $q''_j$  leurs accélérations et  $G = 2.95912208286 \cdot 10^{-4}$  la constante de gravitation. Les valeurs initiales du problème calculées

TAB. VII.1: Valeurs initiales pour le système solaire

planète	masse	position initiale	vitesse initiale
Jupiter	$m_1 = 0.000954786104043$	-3.5023653	0.00565429
		-3.8169847	-0.00412490
		-1.5507963	-0.00190589
Saturne	$m_2 = 0.000285583733151$	9.0755314	0.00168318
		-3.0458353	0.00483525
		-1.6483708	0.00192462
Uranus	$m_3 = 0.0000437273164546$	8.3101420	0.00354178
		-16.2901086	0.00137102
		-7.2521278	0.00055029
Neptune	$m_4 = 0.0000517759138449$	11.4707666	0.00288930
		-25.7294829	0.00114527
		-10.8169456	0.00039677
Pluton	$m_5 = 1/(1.3 \cdot 10^8)$	-15.5387357	0.00276725
		-25.2225594	-0.00170702
		-3.1902382	-0.00136504

le 5 septembre 1994 sont données par le tableau ci-dessous, pour le soleil  $q_0(t_0) = (0, 0, 0)$ ,  $q'_0(t_0) = (0, 0, 0)$ . Les unités de masse choisies sont des unités relatives au soleil. Pour celui-ci on a encore rajouté les masses des planètes proches, ainsi  $m_0 = 1.00000597682$ . Les distances sont en unités astronomiques  $1 [\text{A.U.}] = 149\,597\,870 [\text{km}]$  et le temps est compté en jours terrestres. Calculer la solution pour  $t_{\text{end}} = 3652$ .

Remarque: Pour appliquer votre programme à cet exemple il faudra transformer l'équation différentielle en une équation du premier ordre.

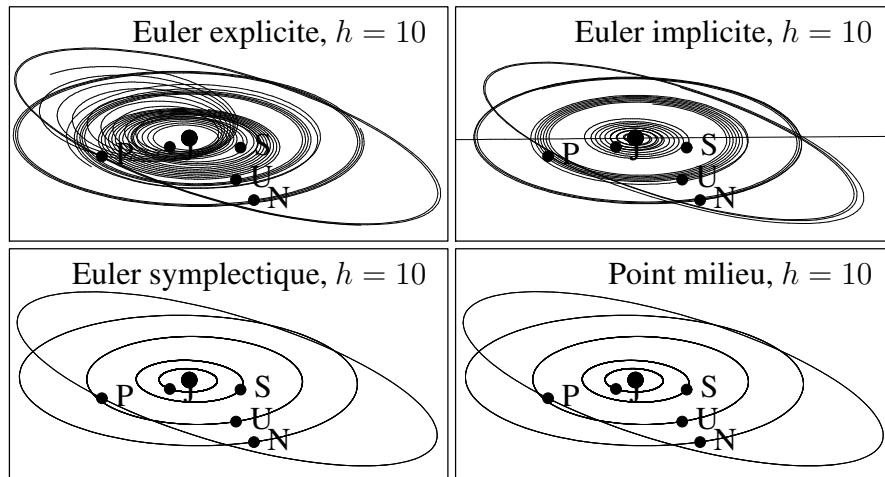


FIG. VII.1: Solutions du mouvement des planètes par différentes méthodes pour de longs intervalles de temps

## VII.8 Décomposition LR

- En utilisant l'algorithme d'élimination de Gauss avec recherche de pivot résoudre le système linéaire

$$Ax = b, \quad (8.1)$$

où  $A$  est une matrice carrée.

Pour cela, écrire une sous-routine de la forme

```
subroutine dec(n,A,ip)
  use accuracy
  ...
  integer, dimension(n) :: ip
  real (kind=dp), dimension(n,n) :: A
```

qui effectue la décomposition  $PA = LR$ , où  $P$  est une matrice de permutation,  $L$  une matrice triangulaire inférieure et  $R$  une matrice triangulaire supérieure.

Les paramètres sont :  $n$  = dimension de la matrice;  $A$  = la matrice en entrée/sortie;  $ip$  = le vecteur de permutation ( $ip(k)$  = index de la ligne du  $k$ -ième pivot).

Ecrivez ensuite une sous-routine de la forme

```
subroutine sol(n,A,b,ip)
  use accuracy
  ...
  integer, dimension(n) :: ip
  real (kind=dp), dimension(n) :: b
  real (kind=dp), dimension(n,n) :: A
```

qui étant donné la matrice  $A$  obtenue par `dec`, résoud le système (1). La solution du problème se trouvera dans le vecteur  $b$ .

- Ecrire un programme qui permet de résoudre  $Ax = b$  et testez-le sur les exemples ci-dessous.

$$A = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 2^{-1} & 2^{-2} & 2^{-3} \\ 1 & 3^{-1} & 3^{-2} & 3^{-3} \\ 1 & 4^{-1} & 4^{-2} & 4^{-3} \end{pmatrix} \quad B = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \end{pmatrix} \quad \text{résultat: } \begin{pmatrix} 10 \\ -35 \\ 50 \\ -24 \end{pmatrix}$$

$$A = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix} \quad B = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}.$$

**Remarque:** En utilisant les instructions `allocate(A(ndim,ndim),...)` on peut utiliser les mêmes tableaux pour des matrices (vecteurs) de dimensions différentes. Il suffit d'écrire `deallocate(A,...)` puis `allocate(A(ndim2,ndim2),...)`.

## VII.9 Décomposition QR et trajectoire d'un astéroïde

Pour résoudre un système surdéterminé

$$Ax = b, \quad x \in \mathbb{R}^n, \quad b \in \text{real}^m, \quad m \geq n,$$

une méthode efficace (voir le cours) est d'utiliser la décomposition  $QR$  de la matrice  $A$ . On vous propose dans la première partie de ce tp d'écrire un programme effectuant cette décomposition. Dans une deuxième partie on traitera un problème surdéterminé de façon complète (estimation de l'erreur, test de confiance du modèle).

1. Pour effectuer la décomposition  $QR$ , on utilise (voir cours) l'algorithme de Householder - Businger - Golub. On commence par écrire une sous-routine `decqr`

```
subroutine decqr(m, n, A, alpha)
  use accuracy
  implicit none
  integer :: m, n, ...
  real (kind=dp), dimension(n) :: alpha
  real (kind=dp), dimension(m, n) :: A
  ...
```

qui effectue la décomposition  $QR$  d'une matrice  $A$  possédant  $m$  lignes et  $n$  colonnes. Cette sous-routine doit retourner la matrice  $A$  modifiée comme suit:

- (a) les vecteurs  $v_i \in \mathbb{R}^{m-i+1}$   $i = 1, \dots, n$  (voir cours), dans la partie "triangulaire" inférieure de la matrice  $A$ ,
- (b) la matrice triangulaire  $R$  (**sans** les  $\alpha_i$ ) dans la partie "triangulaire" supérieure de la matrice  $A$ .

La diagonale de la matrice triangulaire  $R$  (les  $\alpha_i$ ) est stockée dans le vecteur `alpha`. On vous conseille de faire un schéma de la matrice  $A$  à la sortie, pour illustrer la description ci-dessus.

2. Ecrivez ensuite une sous-routine `solqr` qui calcule  $Q^T b$  ainsi que la solution du système triangulaire  $Rx = c$ .

```
subroutine solqr(m, n, a, alpha, b)
  use accuracy
  implicit none
  integer :: m, n, ...
  real (kind=dp), dimension(n) :: alpha
  real (kind=dp), dimension(m) :: b
  real (kind=dp), dimension(m, n) :: A
  ...
```

Pour réaliser la multiplication  $Q^T b$ , on effectue les multiplications successives  $H_i b$   $i = 1, \dots, n$  de la même façon que l'on a effectué les multiplications  $H_i A$  pour la décomposition  $QR$  ( $H_n \cdot \dots \cdot H_2 H_1 = Q^T$ ).

*Remarque.* On ne demande pas de traiter le cas où les colonnes de  $A$  seraient linéairement dépendantes.

3. Ecrire un programme qui permet de résoudre  $Ax = b$  à l'aide de la décomposition  $QR$  et testez-le sur l'exemple ci-dessous pour lequel la solution se calcule facilement (faire un dessin). Vous pouvez aussi tester votre programme sur les exemples de la série 7.

$$\begin{array}{lcl} x + 2y & = 1 \\ 3x + 4y & = 2 \\ 5x + 6y & = 3 \end{array}$$

4. Afin d'effectuer une estimation de l'erreur (pour des observations donnant lieu à un système surdéterminé) on doit calculer  $\text{diag}(A^T A)^{-1}$  (voir cours). On remarque que

$$(A^T A) = (R^T R),$$

où  $R$  est la matrice triangulaire supérieure obtenue par la décomposition  $QR$  (à laquelle on a rajouté la diagonale “alpha (i)”). On peut alors calculer  $\text{diag}(A^T A)^{-1}$  en résolvant successivement les systèmes linéaires (pour  $i = 1, \dots, n$ )

$$\begin{aligned} R^T c &= e_i, \quad \text{où } e_i = (0, \dots, 1, \dots, 0)^T \text{ ième vecteur de base de } \mathbb{R}^n, \\ Rx &= c. \end{aligned}$$

##### 5. (Sur les traces de Gauss)

Un astéroïde en orbite autour du soleil a pu être observé pendant quelques jours avant de disparaître. Voici 10 observations:

$x_{i=1,\dots,5}$	-1.024940	-0.949898	-0.866114	-0.773392	-0.671372
$x_{i=6,\dots,10}$	-0.559524	-0.437067	-0.302909	-0.155493	-0.007464
$y_{i=1,\dots,5}$	-0.389269	-0.322894	-0.265256	-0.216557	-0.177152
$y_{i=6,\dots,10}$	-0.147582	-0.128618	-0.121353	-0.127348	-0.148885

On veut calculer sa trajectoire à partir de ces observations, afin de pouvoir prédire l'instant où son orbite sera à nouveau visible. On suppose un modèle ellipsoïdal pour l'orbite

$$x^2 = ay^2 + bxy + cx + dy + e.$$

Cela conduit à un système surdéterminé que l'on résoud par les “moindres carrés” pour déterminer  $a, b, c, d, e$ . Faire ensuite une estimation de l'erreur ainsi qu'un test de confiance du modèle.

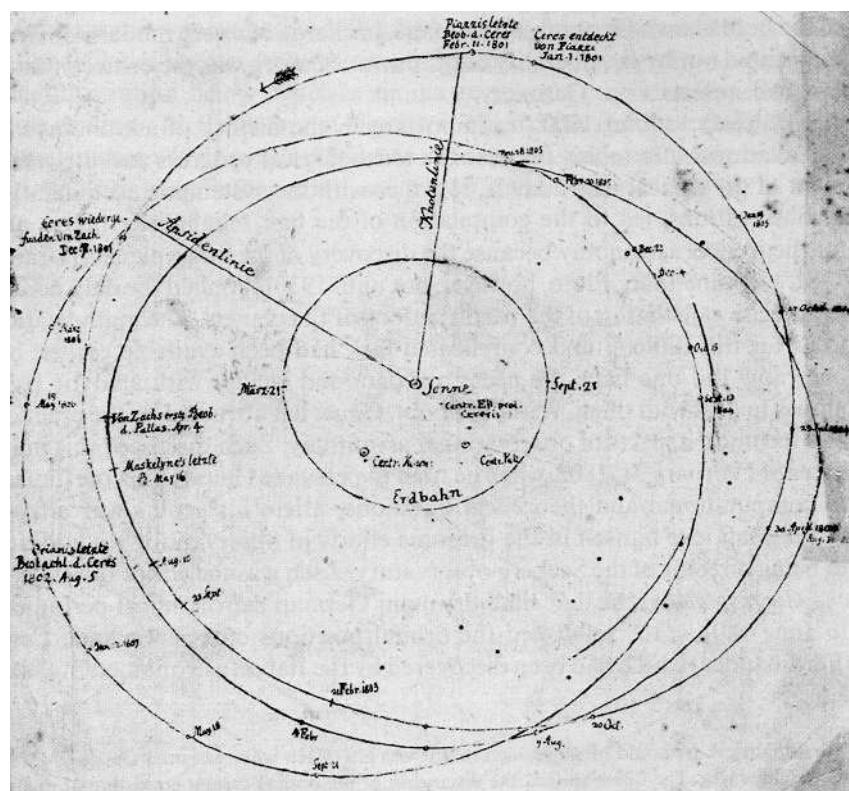
Faire la même étude pour le modèle parabolique

$$x^2 = ay + e.$$

Quelle est la trajectoire la plus probable ?

En janvier 1801, l'astronome italien G.Piazzi découvre une nouvelle planète *Ceres* (entre Mars et Jupiter), et observe son orbite pendant quelques jours, avant que celle-ci ne disparaîsse. Tous les astronomes de l'époque s'attendent à ce que cette planète réapparaisse à la fin de l'année 1801 où au début de l'année 1802. Une grande effervescence habite le monde scientifique européen et plusieurs prédictions quant à la trajectoire de l'orbite sont effectuées.

Le 29 septembre 1801, un jeune mathématicien allemand (alors peu connu), du nom de C.F. Gauss, publie une trajectoire qui diffère sensiblement des autres prévisions, utilisant notamment sa méthode des "moindres carrés". Le 7 décembre 1801, Zach, un des astronomes les plus connus d'Allemagne, redécouvre la planète sur la trajectoire prévue par Gauss. Ce résultat publié en février 1802 dans les *Monatliche Correspondenz* fait de Gauss une célébrité européenne.



*Esquisse de Gauss des orbites de Ceres et Pallas (planète découverte en 1802).* Tiré de W.K. Bühler: *Gauss, a biographical study.*

## VII.10 FORTRAN 90/95

Le Fortran90 est la nouvelle version du language de programmation Fortran77, largement utilisé pour les calculs scientifiques. Les instructions de Fortran77 sont pour la plupart encore valables en Fortran90.

### Compiler

L'extension pour un programme en Fortran 90 est ".f90". Pour compiler un programme taper f90 nom.f90 et ensuite a.out pour l'exécuter.

### Découpage des lignes en zones

La longueur de la ligne est de 132 caractères, et commence dès la première colonne. Tout ce qui est à droite d'un "!" est en commentaire. Pour continuer une ligne sur la ligne suivante, terminer la première par un "&". Pour écrire plusieurs instructions sur une même ligne, les séparer par des ";".

### Structure d'un programme

```
program exemple
    zone de déclaration des variables
    ...
    instructions exécutables
    ...
end program exemple

subroutine et fonction
    ...
```

Un programme peut appeler des sous-programmes qui sont des subroutine ou des function. Voir la série 1 pour des exemples de la structure et de l'utilisation des sous-routines et des fonctions.

### Types de données

Fortran90 contient 5 types intrinsèques:

1. **Types numériques:** integer, real et complex.
2. **Types non numériques:** character et logical.

Il est possible de définir de nouveaux types de données à partir des types intrinsèques. Aux 5 types intrinsèques sont associés un entier non négatif appelé le "**kind type parameter**" (dépend du système et indique en général le nombre de bytes utilisés pour stocker une valeur). Quand celui-ci n'est pas spécifié, c'est la valeur par défaut qui est utilisée.

Exemples:

1. real (kind=8) :: x ! x est de type réel double précision sur sun (ultra), simple précision sur Cray. La double précision correspond à un type prédéfini qui donne environ 15 chiffres significatifs, la simple précision où précision réelle par défaut en donne environ 7.

2. integer, parameter :: dp=kind(1.d0) ! le paramètre dp (constante littérale) prend la valeur de kind qui correspond à la double précision.  
real (kind=dp) :: x ! x est de type réel double précision (défini ci-dessus)
3. real :: x ! x est de type réel par défaut (simple précision)
4. integer, parameter :: q=select\_real\_kind(15,60) ! choisi q de sorte que real (kind=q) possède au moins 15 chiffres significatifs et une étendue égale où supérieure à  $10^{-60}, 10^{60}$  (pour autant que le compilateur en question supporte cette précision). C'est donc une façon indépendante du compilateur de définir des sous-types numériques.
5. 1234 ! constante de type "entier"  
1234. ! constante de type "r'eel"  
1234.\_dp! constante de type "dp" (pr'ealablement d'efini)

## Entrées et Sorties

1. read(5,\*) x ! lit un caractère et l'assigne à la variable x. L'étiquette 5 indique que le caractère est lu du clavier.
2. read(8,\*) x ! comme ci-dessus, sauf que l'étiquette 8 indique que le caractère est lu dans le fichier avec l'étiquette 8 (la lecture se fait de façon séquentielle en commençant par la première valeur du fichier).
3. write(6,\*) x ! écrit le contenu de la variable x à l'écran
4. write(8,\*) x ! écrit le contenu de la variable x dans le fichier avec l'étiquette 8
5. write(6,\*) 'valeur de x' ! écrit les caractères entre apostrophes à l'écran

**Remarque:** L'étoile \* (format par défaut) dans la description ci-dessus peut-être remplacée par un descripteur de format (voir les sorties formatées).

## Affectation des variables

L'opération d'affectation des variables est la suivante:  
variable réceptrice=expression source

## Déclaration de paramètres et initialisations

1. integer, parameter :: n=20 ! défini un paramètre entier et lui assigne la valeur 20. Cette assignation est définitive et ne peut plus être changée dans le programme.
2. real(kind=dp) :: rn=20 ! défini une variable rn réelle de type dp et lui donne 20 comme valeur initiale. Cette assignation peut-être changée dans le programme. On peut aussi déclarer rn comme variable real (kind=dp) et l'initialiser dans le programme.

## Opérateurs et fonctions

+ , - , \* , / , \*\* (exponentiation)  
abs(x) (valeur absolue), sin(x), cos(x), sqrt(x) (racine carrée), etc.

## Opérations arithmétiques

Lors de calculs arithmétiques avec des constantes ou des variables, il est souvent dangereux de mélanger dans une même expression plusieurs types numériques. Pour l'assignation

```
variable=expression,
```

où `variable` est une variable numérique et `expression` est une expression numérique, si `expression` n'est pas du même type que `variable`, `expression` va être converti dans le type de `variable`. Cette conversion peut donner lieu à des pertes de précision. Lors d'opérations arithmétiques mélangeant plusieurs types numériques, les "types moins précis" sont convertis dans le "type le plus précis" de l'expression.

### Exemples:

```
1. integer :: i,j
   integer, parameter :: dp=kind(1.d0)
   real(kind=dp) :: a,b
   a=1;b=3;i=1;j=3
   write(6,*) a/b,a/j,i/j
   real(kind=dp) :: a,b
```

résultats: 0.3333333333333331, 0.3333333333333331, 0

Pour `a/b` les entiers sont convertis en `dp` (de façon exacte) et le calcul est effectué en précision `dp`. Pour `a/j` le contenu de la variable entière `j` est converti (de façon exacte) en `dp` et le calcul est effectué en précision `dp`. Pour `i/j` le calcul est fait en division entière et le résultat est faux.

```
2. integer, parameter :: dp=kind(1.d0)
   real(kind=dp) :: a,b
   a=1.123456789;b=1.123456789_dp
   write(6,*) a,b
   write(6,*) a/b,a-b
   a=1.123456789_dp;b=1.123456789
   write(6,*) a/b,a-b
   a=1.123456789_dp;b=1.123456789_dp
   write(6,*) a/b,a-b
```

résultats: 1.1234568357467651, 1.123456789  
 1.0000000416097581, 4.67467651255049077E-8  
 0.99999995839024369, -4.67467651255049077E-8  
 1., 0.E+0

L'expression `a=1.123456789` mélange le mode réel `dp` pour `a` et le mode réel simple précision (précision réelle par défaut) pour la constante `1.123456789`. Lors de la conversion on perd 3 chiffres significatifs. Ceci explique pourquoi toutes les opérations arithmétiques ci-dessus mélangeant plusieurs types numériques (excepté la dernière) donnent des résultats faux.

## Les sorties formatées

Exemples:

1. `write(6,i5) 123456` ! écrit l'entier 123456 (“i” pour “integer”) dans un champ de 5 caractères
2. `write(6,f10.5) 1.23456` ! écrit le réel 1.23456 dans un champ de 10 caractères dont 5 sont réservés pour la partie décimale.

Ces instructions peuvent également s'écrire

1. `write(6,100) 123456  
100 format(i5)`
2. `write(6,100) 1.23456  
100 format(f10.5)`

**Remarque:** L'étiquette 6 dans la description ci-dessus peut-être remplacée par un label indiquant une autre sortie, par exemple une sortie fichier (voir “fichiers”).

## Les fichiers

Ecriture des résultats dans un fichier:

```
open(8,file='fich1')
```

ouvre un fichier fich1 et lui assigne l'étiquette 8

```
write(8,*) a
```

écrit dans le fichier avec l'étiquette 8 la valeur de a

```
read(8,*) a
```

lit dans le fichier avec l'étiquette 8 la valeur de a.

**Remarque:** Le format par défaut dans la description ci-dessus peut-être remplacé par un format spécifié par l'utilisateur (voir “sorties formatées”).

## Structures de contrôles

1. `if (expression-logique) then`  
    traitements  
`else`  
    traitements  
`end if`
2. `do variable=i,n,m`  
    traitements  
`end do`  
où i,n,m sont des entiers. La boucle se fait de i à n avec un pas m.
3. `do while (expression-logique)`  
    traitements  
`end do`

## Les opérateurs de comparaison

$>$  : plus grand que;  
 $\geq$  : plus grand ou égal à;  
 $==$  : égal;  
 $/=$  : différent de;  
 $\leq$  : plus petit ou égal à;  
 $<$  : plus petit que.

## Les tableaux

Exemple de déclaration:

```
real,dimension(2,3):: tabr1,tabr2
```

déclare un tableau  $2 \times 3$  à valeurs réelles.

**Remarque:** Il est permis de créer des nouveaux tableaux à l'intérieur des fonctions et des sous-routines.

### Allocation dynamique:

Pour déclarer un tableau à deux dimensions, mais dont la taille n'est pas connue au début du programme:

```
real,dimension(:,:,),allocatable:: tabdyn1
```

et plus loin dans le programme:

```
n1=4;n2=8  
allocate(tabdyn(n1,n2))
```

## Références

1. *Programmer en Fortran 90* de Claude Delannoy.
2. *Fortran 90 explained* de M.Metcalf et J.Reid.
3. Et sur l'internet:
4. [www-curri.u-strasbg.fr/DOCtechnique/fortran.htm](http://www-curri.u-strasbg.fr/DOCtechnique/fortran.htm)
5. <http://perso.wanadoo.fr/jean-pierre.moreau/fortran.html>
6. <http://csep1.phy.ornl.gov:80/CSEP/PL/PL.html>
7. [http://www.psc.edu/burkardt/src/f\\_src.html](http://www.psc.edu/burkardt/src/f_src.html)