

Analyse numérique

notes rédigées par ARTEM NAPOV

Avant-propos

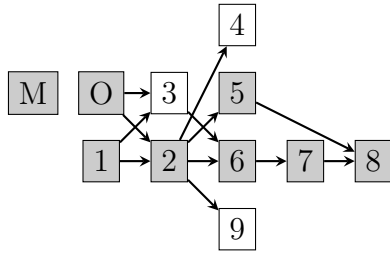
Ces notes couvrent la matière cumulée des cours d'analyse numérique donnés aux étudiants de deux écoles de l'Université libre de Bruxelles : l'École polytechnique et l'École interfacultaire de Bioingénieurs. Elles représentent un support complémentaire aux transparents des cours respectifs et ont pour but de proposer un exposé exhaustif de la matière vue dans ces cours.

Les notes sont structurées autour de problèmes d'analyse numérique. Les chapitres correspondants commencent avec la description d'un problème particulier, viennent ensuite les propriétés qui permettent de caractériser ce problème, les méthodes classiques qui permettent de le résoudre, et les propriétés de ces méthodes.

Les deux premiers chapitres – Chapitre [M](#) et Chapitre [O](#) – jouent un rôle particulier. Le Chapitre [M](#) motive la matière de ces notes avec quelques exemples concrets d'ingénieur. Pour ce qui est du Chapitre [O](#), il introduit les éléments de syntaxe d'Octave – un outil particulièrement commode pour la résolution des problèmes d'analyse numérique – que nous utilisons abondamment à travers le texte.

Le restant des notes est subdivisé en 9 chapitres. Parmi ces chapitres, le premier porte sur les erreurs d'arrondi et leur propagation dans un calcul numérique. Les chapitres suivants couvrent un sujet clé de l'algèbre linéaire numérique - la résolution des systèmes linéaires. On y aborde la résolution des systèmes réguliers (Chapitre [2](#)) et surdéterminés (Chapitre [3](#)), ainsi que les méthodes itératives pour les systèmes réguliers (Chapitre [4](#)). La résolution des équations et systèmes d'équations non linéaires est présentée juste après (Chapitre [5](#)) ; elle est basée en partie sur les notions vues aux chapitres antérieurs. Viennent ensuite les chapitres qui abordent l'interpolation et l'approximation polynomiales (Chapitre [6](#)), ainsi que l'intégration numérique (Chapitre [7](#)). Ces deux derniers chapitres forment la base pour aborder les équations différentielles, utilisées soit avec des conditions initiales (Chapitre [8](#)), soit avec des conditions aux limites (Chapitre [9](#)).

Même si en grande partie les Chapitres 2–9 peuvent être lus indépendamment les uns des autres, ils ne sont pas totalement indépendants. Les dépendances principales sont reprises dans le graphe suivant, où les flèches indiquent quel chapitre doit idéalement être vu avant quel autre. Tous les chapitres sont vus par les ingénieurs ; les chapitres dans les cellules en gris sont (partiellement) vus par les bioingénieurs.



Un exposé alternatif et/ou plus complet de la matière peut être trouvé dans les ouvrages suivants :

- [1] ALFIO QUARTERONI, RICCARDO SACCO, FAUSTO SALERI, *Méthodes Numériques*, Springer-Verlag Italia, Milano, 2004.
- [2] JACQUES RAPPAZ, MARCO PICASSO, *Introduction à l'analyse numérique*, Presses polytechniques et universitaires romandes, Lausanne, 1998.
- [3] LLOYD N. TREFETHEN, DAVID BAU III, *Numerical Linear Algebra*, SIAM, Philadelphia, 2004.
- [4] URI ASCHER AND CHEN GREIF, *A First Course in Numerical Methods*, SIAM, Philadelphia, 2011.
- [5] G. W. STEWART, *Afternotes on Numerical Analysis*, SIAM, Philadelphia, 1996.

Il va de soi que ces ouvrages complètent mais ne remplacent pas les notes de cours.

Pour finir, je tiens à remercier tout particulièrement Engin Kumanova qui a pris le soin de relire plus d'une fois l'ensemble de ces notes et a suggéré de nombreuses améliorations. Je remercie également Prof. Jérémy Dohet-Eraly pour ses remarques, ainsi que pour le texte de l'Annexe 2.B.

Notations

Dans ces notes on fait un usage récurrent des vecteurs et des matrices. Les vecteurs sont notés ici avec des lettres latines minuscules grasses (e.g., \mathbf{x} , \mathbf{y}), alors que les matrices sont représentées avec des lettres majuscules (e.g., A , B). Les éléments des vecteurs et des matrices sont supposés réels. Pour référencer l'élément a_{ij} d'une matrice $A = (a_{ij})$, on indique d'abord son indice de ligne (ici i) et ensuite son indice de colonne (ici j). Dans la même logique, une matrice est de dimensions $m \times n$ si elle a m lignes et n colonnes. Un vecteur \mathbf{a} est par défaut un vecteur colonne (ou, de manière équivalente, une matrice avec une seule colonne), le vecteur ligne associé est le transposé \mathbf{a}^T de \mathbf{a} . En particulier, pour deux vecteurs $\mathbf{x} = (x_i)$ et $\mathbf{y} = (y_i)$ de dimension n le produit scalaire usuel est donné par

$$\mathbf{x}^T \mathbf{y} = \sum_{i=1}^n x_i y_i ;$$

pour une matrice $A = (a_{ij})$ de dimensions $m \times n$ et un vecteur $\mathbf{x} = (x_i)$ de dimension n , le produit $A\mathbf{x}$ est un vecteur dont la i -ème composante, $1 \leq i \leq m$, est donnée par

$$\sum_{j=1}^n a_{ij}x_j ;$$

pour deux matrices $A = (a_{ij})$ et $B = (b_{ij})$ de dimensions $m \times k$ et $k \times n$, respectivement, le produit AB est une matrice $m \times n$ dont la composante à la ligne i et la colonne j est

$$\sum_{s=1}^k a_{is}b_{sj} .$$

En général, les éléments structurellement nuls d'une matrice ne sont pas représentés. Par exemple, une matrice triangulaire inférieure $A = (a_{ij})$ de dimensions 2×2 sera en général représentée comme

$$\begin{pmatrix} a_{11} & \\ a_{21} & a_{22} \end{pmatrix} \quad (1)$$

pour mieux mettre en évidence ses éléments non nuls.

Par ailleurs, la notation $F \subset G$ signifie que l'ensemble F est compris dans l'ensemble G , le cas $F = G$ n'étant pas exclu.

Table des matières

M Motivation	11
M.1 Définition	11
M.2 Exemples d'applications	12
M.3 Exemples supplémentaires	15
O Éléments de syntaxe d'Octave	21
O.1 Avant de commencer	21
O.2 Conditions	22
O.3 Boucles	24
O.4 Fichiers <code>.m</code>	25
O.5 Fonctions	25
1 Représentation en virgule flottante, stabilité et conditionnement	29
1.1 Représentation en virgule flottante	29
1.1.1 Motivation	29
1.1.2 Représentation en virgule flottante	30
1.1.3 Erreurs d'arrondi	31
1.1.4 Standard IEEE	33
1.1.5 Modèle d'arithmétique	34
1.1.6 Annulation	36
1.1.7 Compléments	37
1.2 Stabilité et conditionnement	38
1.2.1 Stabilité directe	38
1.2.2 Conditionnement	39
1.2.3 Stabilité inverse	41
2 Systèmes d'équations linéaires : méthodes directes	43
2.1 Problème	43
2.2 Conditionnement	44

2.3	Méthodes directes	46
2.3.1	«Mauvaises» méthodes directes	46
2.3.2	Systèmes triangulaires	47
2.3.3	Méthode d'élimination de Gauss	49
2.3.4	Factorisation LU	51
2.3.5	Stabilité et pivotage	56
2.4	Applications	61
2.4.1	Calcul de déterminant	61
2.4.2	Inversion matricielle	61
Annexe 2.A	Rappels	63
2.A.1	Normes	63
2.A.2	Calcul de déterminant à l'aide de mineurs	63
2.A.3	Matrices de permutation	64
Annexe 2.B	Somme des puissances des n premiers entiers positifs	65
2.B.1	Calcul approché	65
2.B.2	Calcul exact	66
3	Factorisation QR et moindres carrés	69
3.1	Factorisation QR	69
3.1.1	Définitions	69
3.1.2	Méthode de Householder	70
3.2	Moindres carrés	78
3.2.1	Problème	78
3.2.2	Équations normales	78
3.2.3	Interprétation géométrique	80
3.2.4	Conditionnement	80
3.2.5	Méthode des équations normales (version LU)	81
3.2.6	Méthode de la factorisation QR	83
3.2.7	Stabilité : exemple	83
Annexe 3.A	Propriétés de la norme matricielle euclidienne	86
Annexe 3.B	Conditionnement : dérivation	88
4	Méthodes itératives pour systèmes linéaires	89
4.1	Définition et contexte	89
4.2	Méthodes stationnaires	92
4.2.1	Principe	92
4.2.2	Convergence	93
4.2.3	Méthodes stationnaires classiques	94
4.3	Méthodes basées sur la minimisation d'énergie	98
4.3.1	Energie du système	98
4.3.2	Minimisation d'énergie	99
4.3.3	Méthode du gradient	100
4.3.4	Méthode du gradient conjugué	101
4.3.5	Préconditionnement	102

4.4	Contrôle de convergence et critères d'arrêt	103
	Annexe 4.A Méthode du gradient conjugué : direction optimale	105
5	Equations et systèmes d'équations non linéaires	107
5.1	Problème	107
5.2	Méthodes d'encadrement	108
5.2.1	Généralités	108
5.2.2	Méthode de dichotomie	109
5.2.3	Méthode de la fausse position	110
5.3	Méthodes de Newton	110
5.3.1	Méthode de Newton-Raphson	110
5.3.2	Newton avec recherche linéaire	113
5.4	Méthodes pour systèmes non linéaires	115
5.4.1	Newton-Raphson pour systèmes non linéaires	115
5.4.2	Méthode de Newton-corde	116
5.5	Critères d'arrêt	116
6	Interpolation et approximation	119
6.1	Problèmes d'interpolation et d'approximation	119
6.2	Interpolation et approximation polynomiales	120
6.2.1	Résolution à l'aide de systèmes linéaires	120
6.2.2	Interpolation avec polynômes de Lagrange	122
6.2.3	Conditionnement	123
6.3	Interpolation polynomiale par morceaux	124
6.4	Interpolation par splines	124
7	Intégration numérique	127
7.1	Fonctions d'une variable	127
7.1.1	Problème	127
7.1.2	Formule des trapèzes	128
7.1.3	Formule de Simpson	129
7.1.4	Formules de Newton-Cotes	130
7.1.5	Méthode de Romberg	132
7.2	Fonctions de plusieurs variables	133
	Annexe 7.A Formule des trapèzes : erreur locale	134
8	Équations différentielles avec conditions initiales	137
8.1	Généralités	137
8.1.1	Équations différentielles du premier ordre	137
8.1.2	Systèmes d'équations différentielles du premier ordre	138
8.1.3	Équations d'ordre deux et plus	139
8.1.4	Existence, unicité	140
8.1.5	Principe de résolution numérique	141
8.2	Méthodes d'Euler	141

8.2.1	Méthode d'Euler progressive	141
8.2.2	Méthode d'Euler rétrograde	142
8.2.3	Caractère explicite/implicite	142
8.2.4	Convergence	143
8.2.5	Cas vectoriel	144
8.3	Stabilité	145
8.3.1	Stabilité absolue	145
8.3.2	Stabilité des méthodes d'Euler	145
8.3.3	Cas plus généraux	147
8.4	Méthodes du second ordre	152
8.4.1	Méthode de Crank-Nicolson	152
8.4.2	Méthode de Heun	155
8.5	Méthodes multi-pas	156
	Annexe 8.A Régions de stabilité	158
	Annexe 8.B Compléments sur la stabilité	159
	Annexe 8.C Qualité d'approximation en fonction de $h\beta$	162
9	Équations différentielles avec conditions aux limites	165
9.1	Généralités	165
9.1.1	Problème aux limites	165
9.1.2	Existence, unicité et cas particuliers	166
9.1.3	Principe de résolution numérique	167
9.2	Problème de Poisson	168
9.2.1	Applications	168
9.2.2	Approximation aux points intérieurs	169
9.2.3	Condition de Dirichlet	169
9.2.4	Condition de Neumann	171
9.3	Problème de convection-diffusion	173
9.3.1	Différence centrée	173
9.3.2	Schéma upwind	174
	Annexe 9.A Formules de différences finies : terme d'erreur	176



Motivation

Ce chapitre a pour but de motiver la présence d'un cours d'analyse numérique dans un cursus d'ingénieur. Il commence avec la discussion d'une définition communément acceptée de l'analyse numérique. On présente ensuite quelques exemples de problèmes d'ingénieur qui font appel à l'analyse numérique ; ces exemples seront en partie réutilisés dans les chapitres suivants. Finalement, on aborde quelques exemples qui montrent qu'il est important de non seulement connaître les outils de l'analyse numérique, mais également de comprendre les propriétés et/ou la portée de ces outils.

M.1 Définition

Nous considérons la définition suivante de l'analyse numérique

*Numerical analysis is the study of algorithms
for the problems of continuous mathematics.*

Lloyd N. Trefethen¹

Deux aspects de cette définition sont à souligner. Tout d'abord, il s'agit d'une science qui étudie les algorithmes pour des problèmes mathématiques, et donc fait appel aussi bien aux mathématiques qu'à l'informatique. Ensuite, seuls les problèmes à variables continues (c'est-à-dire à variables dans \mathbb{R} , \mathbb{C}) sont considérés ; ainsi, par exemple, les problèmes d'ordonnancement ou de tri ne sont pas abordés dans ce cours, car leur formulation ne fait pas nécessairement intervenir des variables continues. Le caractère continu des problèmes est une source intrinsèque d'imprécision, car il n'est possible de représenter qu'un nombre fini de réels différents sur une machine de calcul moderne.

1. «The definition of numerical analysis» de Lloyd N. Trefethen, disponible via metronu.ulb.ac.be/MATH-H-202/trefethen-def-na.pdf

M.2 Exemples d'applications

Les trois exemples suivants présentent quelques problèmes d'analyse numérique utiles aux ingénieurs. La taille des problèmes considérés est limitée pour des raisons didactiques.

Pour résoudre numériquement ces problèmes, on utilise un programme appelé Octave. Octave est une sorte de «calculatrice polyvalente» dont plusieurs fonctionnalités nous sont utiles dans la suite de ce cours. Plus de détails sur Octave sont donnés dans le Chapitre O ; dans le présent chapitre on se limite donc à des programmes Octave simples dont le fonctionnement est clarifié à l'aide de commentaires (texte commençant par %).

EXEMPLE 1.

Soit un système mécanique schématisé sur la Figure M.1. Ce système est composé de trois masses – m_1 , m_2 et m_3 – qui sont reliées à l'aide de trois ressorts dont les constantes de rappel sont k_1 , k_2 et k_3 . Si le système est en équilibre, les déplacements x_1 , x_2 et x_3 des ressorts par rapport à leur position en absence de la force de pesanteur sont déterminés par le système de trois équations à trois inconnues suivant

$$\begin{cases} k_2(x_2 - x_1) - k_1x_1 + m_1g = 0, \\ k_3(x_3 - x_2) - k_2(x_2 - x_1) + m_2g = 0, \\ -k_3(x_3 - x_2) + m_3g = 0. \end{cases}$$

Sous forme matricielle cela s'écrit

$$\begin{pmatrix} k_1 + k_2 & -k_2 & 0 \\ -k_2 & k_2 + k_3 & -k_3 \\ 0 & -k_3 & k_3 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} m_1g \\ m_2g \\ m_3g \end{pmatrix},$$

ou encore, avec des notations génériques,

$$A\mathbf{x} = \mathbf{b},$$

où A est la matrice du système, \mathbf{b} est le vecteur des seconds membres, et \mathbf{x} est le vecteur des inconnues. Pour $k_i = m_i g = 1$, le système devient

$$\begin{pmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}.$$

La résolution avec Octave se fait alors comme suit :

```
b = [1; 1; 1]; % créer le second membre b
A = [2 -1 0; -1 2 -1; 0 -1 1]; % créer la matrice A
x = A\b; % résoudre Ax=b
x % afficher x
```

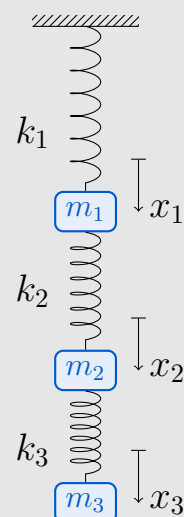


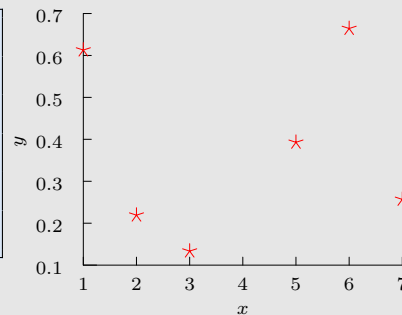
FIGURE M.1 – Système mécanique.

Notons que le système pour 3 masses peut aussi être résolu à la main. Par contre, pour un grand nombre d'équations (par exemple, 10^6), un ordinateur est nécessaire !

EXEMPLE 2. Soit un ensemble donné de 6 points (x_i, y_i) , $i = 1, \dots, 6$. On considère le problème de détermination d'un polynôme de degré au plus 3 qui passe aussi «près» que possible de ces 6 points. Notez qu'il est généralement impossible d'avoir un polynôme de degré inférieur à 5 qui passe par 6 points quelconques.

Plus spécifiquement, les 6 points suivants sont obtenus en fixant les abscisses à 1, 2, 3, 5, 6, 7, respectivement, et en générant les ordonnées de manière aléatoire. Le code Octave correspondant est donné à gauche, un résultat possible généré avec ce code est représenté à droite.

```
% choisir les abscisses
x = [1; 2; 3; 5; 6; 7];
% choisir les ordonnées
y = rand(6,1);
% afficher le résultat
plot(x, y, '*r')
```

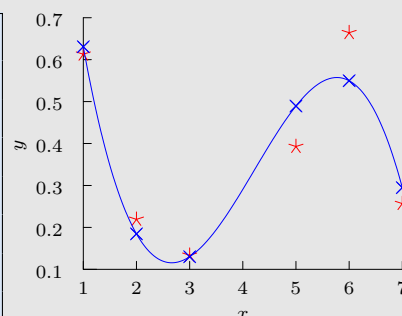


Les points étant choisis, le code suivant utilise la fonction `polyfit(.,.,3)` pour déterminer un polynôme $p(x)$ de degré au plus 3 qui approche ces points au sens des moindres carrés, c'est-à-dire qui minimise

$$\sum_{k=1}^6 (y_i - p(x_i))^2.$$

Une figure possible générée par un tel code est à droite.

```
% approcher par polynôme
% de degré <= 3 au sens
% des moindres carrés
pol = polyfit(x,y,3);
% garder graphe précédent
hold on
% afficher polynôme en x
plot(x, polyval(pol,x), 'x')
% raffiner x & afficher
x = [1:0.01:7];
plot(x, polyval(pol,x), '-')
```



EXEMPLE 3.

Le transitoire électrique dans le circuit schématisé sur la Figure M.2 est décrit par les équations suivantes, dont la première est la loi des mailles, la seconde est la définition du courant, et les deux autres définissent les conditions initiales

$$\begin{cases} L \frac{dI}{dt} + RI + \frac{Q}{C} = 0, \\ \frac{dQ}{dt} = I, \\ I(0) = I_0, \quad Q(0) = Q_0. \end{cases}$$

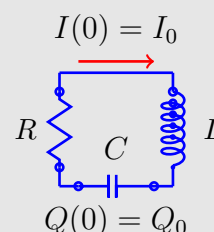


FIGURE M.2 – Circuit RLC.

L'ensemble des équations obtenu est un système d'équations différentielles avec conditions initiales, qui peut aussi se mettre sous la forme

$$\begin{cases} \frac{d}{dt} \begin{pmatrix} I \\ Q \end{pmatrix} = \begin{pmatrix} -RL^{-1} & -C^{-1}L^{-1} \\ 1 & \end{pmatrix} \begin{pmatrix} I \\ Q \end{pmatrix}, \\ \begin{pmatrix} I \\ Q \end{pmatrix} \Big|_{t=0} = \begin{pmatrix} I_0 \\ Q_0 \end{pmatrix}, \end{cases}$$

ou encore, en utilisant des notations génériques,

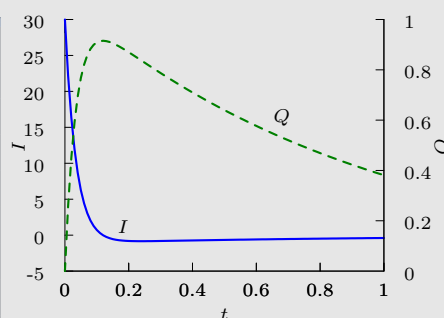
$$\begin{cases} \frac{d}{dt} \mathbf{u} = \mathbf{F}(\mathbf{u}, t), \\ \mathbf{u}(0) = \mathbf{u}_0, \end{cases} \quad \text{avec} \quad \mathbf{u} = \begin{pmatrix} I \\ Q \end{pmatrix}.$$

Pour considérer un cas plus spécifique, posons $RL^{-1} = C^{-1}L^{-1} = I_0 = 30$ et $Q_0 = 0$; on a alors

$$\mathbf{F}(\mathbf{u}, t) = \begin{pmatrix} -30u_1 - 30u_2 \\ u_1 \end{pmatrix} \quad \text{et} \quad \mathbf{u}_0 = \begin{pmatrix} 30 \\ 0 \end{pmatrix}. \quad (\text{M.1})$$

Ce problème est résolu à l'aide de la fonction `lsode` d'Octave, comme montré dans le code suivant ; la figure produite par le code est donnée à droite.

```
% définir F(u,t)
F = @(u,t) ( [-30*u(1)-30*u(2); u(1)] );
% conditions initiales
u0 = [30; 0];
% t = 0, 0.01, ..., 1
T = 0:0.01:1;
% résoudre
sol = lsode(F, u0, T);
% graphe de I & Q
plotyy(T, sol(:,1), T, sol(:,2))
```



M.3 Exemples supplémentaires

Les trois exemples précédents illustrent l'utilité des outils d'analyse numérique disponibles en Octave. On peut néanmoins se demander s'il est réellement nécessaire de comprendre le fonctionnement des algorithmes sous-jacents ? La réponse est «oui», et les quelques arguments qui justifient cette réponse sont donnés dans les sections suivantes, chaque argument étant illustré par un exemple concret. Notons que les exemples suivants sont choisis surtout pour leur simplicité et leur facilité de mise en œuvre ; ils ne seront donc pas nécessairement approfondis dans la suite de ce cours.

Argument 1 : choisir la bonne méthode

Une méthode est rarement la meilleure dans tous les cas, et pour pouvoir choisir l'approche la plus adéquate il est souhaitable de comprendre au préalable le fonctionnement des différentes méthodes.

EXEMPLE 4. L'efficacité de la résolution d'un système linéaire dépend, entre autres, du format de la matrice du système. En particulier, si la matrice a plusieurs éléments nuls, un format creux (*sparse* en anglais) peut non seulement permettre de ne pas mémoriser les éléments nuls, mais aussi d'accélérer la résolution du système correspondant.

Pour illustrer ce cas, nous utilisons la fonction `solvetime`, dont le code est donné ci-dessous ; elle mesure le temps de résolution d'un système (opération $\mathbf{x} = \mathbf{A} \backslash \mathbf{b}$ en Octave, avec \mathbf{A} la matrice et \mathbf{b} le vecteur des seconds membres) représenté en format

- *dense* (par défaut) ;
- *creux* disponible en Octave (la conversion est faite via l'instruction `sparse`).

```
function solvetime(A,b)
    % solveur dense
    tic
    x = A\b;
    toc
    % solveur creux
    A = sparse(A);
    tic
    x = A\b;
    toc
```

Si la matrice a effectivement beaucoup d'éléments nuls, le passage vers le format creux peut s'avérer bénéfique. A titre d'exemple, prenons le système de l'Exemple 1 qui, pour un nombre n de ressorts quelconque, devient

$$\begin{pmatrix} 2 & -1 & & \\ -1 & 2 & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix}.$$

Considérons ce système dans le cas où le nombre n de masses est fixé à 1000, et fixons, comme avant, $k_i = m_i g = 1$. Le code qui génère le système et mesure son temps de résolution dans les deux formats est donné ci-dessous ; les temps de résolution obtenus (pour une certaine configuration informatique) par le code sont donnés à droite de celui-ci.

```
% matrice plutôt creuse
n=1000;
A = -diag(ones(n-1,1),1) + ...
    diag(ones(n,1));
A = A + A';
A(n,n) = 1;
b = ones(n,1);
solvetime(A,b)
```

résultats :

Elapsed time is 0.14 seconds.

Elapsed time is 0.00051 seconds.

fonction pour matrices denses

275× plus lente !

On peut alors se dire qu'il est toujours préférable de passer par le format creux. Néanmoins, pour les systèmes dont la matrice a peu (ou pas) d'éléments nuls, c'est loin d'être le cas. A titre d'exemple, les résultats pour les matrices dont tous les éléments sont générés de manière aléatoire, et donc vraisemblablement non nuls, sont présentés dans les lignes suivantes (comme avant, le code est à gauche et les résultats sont à droite) :

```
% matrice plutôt dense
n=1000;
A=rand(n);
b = ones(n,1);
solvetime(A,b)
```

résultats :

Elapsed time is 0.16 seconds.

Elapsed time is 1.2 seconds.

fonction pour matrices denses

7.5× plus rapide !

Argument 2 : identifier la source des difficultés

Parfois ce qui pose problème n'est pas la méthode mais bien le problème lui-même. Et sans identifier la source des difficultés on ne peut pas avancer vers leur résolution.

EXEMPLE 5. A titre d'exemple, le système d'équations différentielles

$$\begin{cases} \frac{du_1}{dt} = -u_1 + 2u_2 \\ \frac{du_2}{dt} = u_1 \end{cases}, \quad \text{avec} \quad \begin{cases} u_1(0) = -2 \\ u_2(0) = 1 \end{cases} \quad (\text{M.2})$$

a comme (seule!) solution exacte

$$\begin{pmatrix} u_1(t) \\ u_2(t) \end{pmatrix} = \begin{pmatrix} -2 \\ 1 \end{pmatrix} e^{-2t};$$

en particulier $u_1(t)$ et $u_2(t)$ tendent vers 0 pour $t \rightarrow \infty$.

Les résultats obtenus avec la fonction `lsode` d'Octave (déjà vue dans l'Exemple 3) pour la résolution de ce problème sont donnés sur la Figure M.3. Le comportement des solutions jusqu'à environ $t = 30$ (Figure M.3 (gauche)) semble conforme au comportement attendu. Par contre, la situation change sensiblement à partir de $t = 30$ (Figure M.3 (droite)) – on observe un comportement instable de la solution numérique.

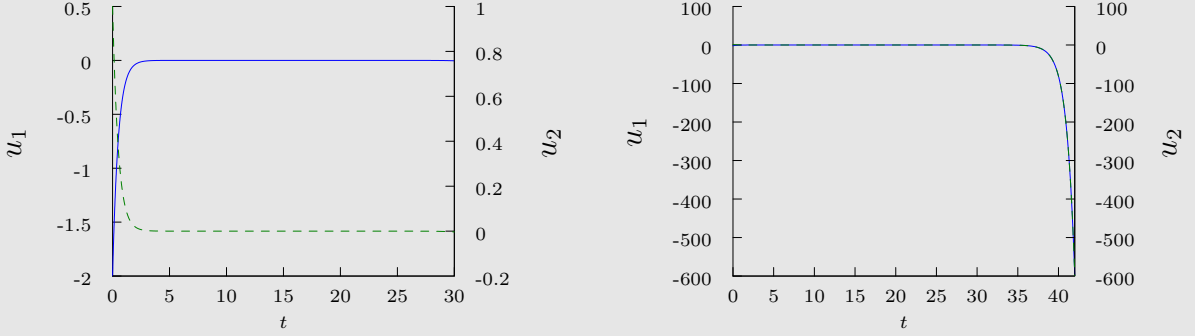


FIGURE M.3 – Les solutions numériques $u_1(t)$ et $u_2(t)$ du système (M.2) en fonction du temps sur un intervalle allant de 0 à 30 (gauche) et de 0 à 42 (droite).

A priori, on pourrait penser que c'est la fonction `lsode` qui pose problème. Néanmoins, toute méthode similaire risque de développer une telle instabilité. L'explication du phénomène réside dans le fait que la solution pour $u_1(0)$ et $u_2(0)$ quelconques est donnée par

$$\begin{pmatrix} u_1(t) \\ u_2(t) \end{pmatrix} = \begin{pmatrix} -2 \\ 1 \end{pmatrix} \frac{u_2(0) - u_1(0)}{3} e^{-2t} + \begin{pmatrix} 1 \\ 1 \end{pmatrix} \frac{2u_2(0) + u_1(0)}{3} e^t;$$

cette solution a donc une composante qui croît de manière exponentielle. Les valeurs $u_2(0)$, $u_1(0)$ du système (M.2) ci-dessus sont telles que le coefficient de e^t est nul en arithmétique exacte, mais les erreurs d'arrondi «ressuscitent» la composante instable lors d'un calcul numérique. Notez que le moment où l'instabilité apparaît dépend, entre autres, de la configuration informatique.

Dans le cas considéré, la difficulté vient du problème plutôt que de la méthode utilisée. On parle ici plus précisément d'un problème mal conditionné². Pour un tel problème, de petites erreurs d'arrondi présentes dans les données, ou introduites lors de la résolution numérique du problème, mènent à des erreurs sensiblement plus importantes dans la solution, et ce indépendamment de la méthode de résolution utilisée.

2. Il est aussi mal posé si on considère $t \rightarrow \infty$, car deux conditions initiales infiniment proches peuvent donner deux solutions très différentes.

Argument 3 : développer vos propres outils

Vous serez amenés à développer vos propres outils, il est donc important que vous soyez sensibilisés aux différents aspects importants.

EXEMPLE 6. Pour calculer la variance d'un échantillon $\{x_1, \dots, x_n\}$ on peut utiliser une des deux formules suivantes

$$\text{var}(x_1, \dots, x_n) = \frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^2 \quad (\text{M.3})$$

$$= \frac{1}{n} \sum_{i=1}^n x_i^2 - \bar{x}^2. \quad (\text{M.4})$$

Ces deux formules sont mathématiquement équivalentes, et peuvent donc a priori toutes les deux être utilisées dans un code numérique. Les codes qui correspondent aux deux formules sont

```
function var = var_frm1(x,n)
x_moy = sum(x)/n;
var = sum((x-x_moy).^2)/n;
```

```
function var = var_frm2(x,n)
x_moy = sum(x)/n;
var = sum(x.^2)/n - x_moy^2;
```

Pour l'échantillon $x=[200000000 \ 200000001 \ 200000002]$ les résultats sont (pour une certaine configuration informatique, et avec chaque résultat aligné sous le code correspondant)

```
ans = 0.66667
```

```
ans = 0
```

alors que pour $x=[100000000 \ 100000001 \ 100000002]$ on a

```
ans = 0.66667
```

```
ans = 2
```

Étant donné que les deux échantillons sont les mêmes à une constante additive près, leur variance est également la même en arithmétique exacte. On en conclut que le deuxième code est problématique. Et le premier ?

Le lecteur motivé peut vérifier que, pour le cas considéré, le résultat de la première formule est correct. Plus de détails sur les différents algorithmes de calcul de variance, qui ne sont pas abordés dans ce cours, peuvent être trouvés, entre autres, dans

TONY F. CHAN, GENE H. GOLUB, RANDALL J. LEVEQUE, *Algorithms for Computing the Sample Variance : Analysis and Recommendations*, The American Statistician, Vol. 37, No. 3, 1983, pp. 242–247.

En résumé

Les exemples précédents justifient deux observations importantes.

1. Octave (et il n'est pas le seul) comporte de nombreuses fonctions utiles pour résoudre des problèmes d'analyse numérique ; ces fonctions sont basées sur des algorithmes de qualité.
2. Et malgré cela, il est déconseillé d'utiliser ces outils sans avoir une *compréhension* des problèmes traités, ainsi que des algorithmes correspondants, de leurs avantages, inconvénients et limitations.

Ces observations expliquent à leur tour pourquoi le présent cours n'est pas (uniquement) une introduction à Octave, mais (aussi, et surtout !) une introduction à l'analyse numérique.



Éléments de syntaxe d'Octave

Ce chapitre reprend les quelques éléments de syntaxe d'Octave qui sont utiles pour la bonne compréhension de ces notes. Il correspond au document *Éléments de syntaxe Octave* (version 0.0.6) disponible sur le site web du cours.

O.1 Avant de commencer

Octave est un environnement de calcul scientifique qui dispose de multiples fonctions utiles en analyse numérique. Ce chapitre a pour but d'introduire les quelques éléments de syntaxe d'Octave qui sont d'intérêt pour le cours. Historiquement Octave a été développé comme une alternative libre et gratuite à Matlab[®], la syntaxe de ces deux logiciels est donc virtuellement identique.

Nous commençons avec quelques commentaires sur l'installation d'Octave. Le lien vers le (nouveau) site web d'Octave est

<https://www.gnu.org/software/octave/> ;

ce site web offre les exécutables et/ou les procédures d'installation d'Octave pour divers systèmes d'exploitation. Alternativement,

http://enacit1.epfl.ch/cours_matlab/

décrit étape par étape les différentes procédures d'installation. Pour la plupart des systèmes d'exploitation récentes de type Unix (Linux, BSD, etc.), Octave est typiquement inclus dans les distributions officielles, et peut donc être installé à l'aide du gestionnaire de programmes ad hoc. En cas de problème avec une des versions d'Octave, vous pouvez répéter la procédure d'installation avec une autre version du logiciel ; les versions à partir de 3.2.0 sont typiquement suffisantes pour l'ensemble des travaux pratiques.

Vous pouvez utiliser Octave à l'aide de son interface graphique (**gui**, une abréviation de *graphical user interface*), ou directement dans un terminal de commande. Pour passer d'une version à l'autre, les arguments `--no-gui` ou `--force-gui` doivent être utilisés lors du lancement d'Octave.

Les commandes fournies à Octave doivent être entrées dans la fenêtre de commandes ; il s'agit de l'onglet **Command Window** de la fenêtre centrale (si vous utilisez l'interface gra-

pique) ou de la fenêtre du terminal. Ces commandes correspondent soit aux instructions prédéfinies d'Octave, soit aux noms de fichiers avec une extension `.m` ; ces derniers fichiers doivent alors contenir des fonctions ou des scripts écrits avec des instructions prédéfinies.

Dans ce qui suit nous passons en revue les quelques instructions de base.

`help commande`

indique ce que fait une commande particulière. Par exemple

`help plot`

indique comment utiliser la commande `plot` pour afficher des graphiques en deux dimensions. Plus précisément, `help` ouvre un environnement d'affichage séparé de l'environnement de travail principal : pour se déplacer dans ce dernier, suivez l'indication en bas à gauche

```
-- less -- (f)orward, (b)ack, (q)uit
```

(utilisez les touches `b` et `f` pour vous déplacer dans le texte, `q` pour sortir).

Pour interrompre l'exécution d'un programme, utilisez la combinaison de touches `Ctrl+c`. Vous pouvez aussi utiliser `Ctrl+c` pour sortir d'Octave, même si l'usage des instructions `exit` ou `quit` est plus conventionnel et recommandé.

La dernière instruction utilisée peut être obtenue en appuyant sur la touche `↑` du clavier. La commande `history n` permet d'afficher les `n` dernières instructions utilisées.

Lors de son lancement, Octave choisit son répertoire de travail (celui dans lequel il cherche les fichiers `.m`, y compris ceux que vous avez écrits). Pour déterminer ce répertoire, utilisez `pwd` (une abréviation de *print working directory*). Pour afficher son contenu, utilisez `ls`, et pour changer de répertoire – `cd nouveau-répertoire`.

Alternativement, pour ceux qui utilisent l'interface graphique d'Octave, il est possible de changer le répertoire de travail dans le champs intitulé *Current Directory* et situé dans la partie supérieure de la fenêtre d'Octave. Le contenu du répertoire est alors affiché dans la partie *File Browser* en haut à gauche de la fenêtre.

Le restant de ce chapitre est organisé comme suit. La section [0.2](#) porte sur les conditions, la section [0.3](#) détaille l'usage des boucles, la section [0.4](#) concerne les fichiers `.m`, et la section [0.5](#) explique l'utilisation des fonctions. Finalement, les quelques instructions qui ne sont pas accessibles via la commande `help` sont détaillées dans le *Petit Guide d'Octave* à la fin de ce chapitre.

O.2 Conditions

Les conditions sont spécifiées à l'aide des structures basées sur l'instruction `if`. La structure la plus simple qui permet de spécifier une condition est

```
if (condition)
    code pour if
end
```

Dans ce cas, le *code pour if* sera exécuté si la *condition* est vérifiée.

Une condition simple typiquement vérifie si une (in)égalité est satisfaite ; la syntaxe

relation	instruction
$<$ (ou $>$)	$<$ (ou $>$)
\leq (ou \geq)	$>=$ (ou $<=$)
$=$	$==$
\neq	$\sim=$

TABLE O.1 – Syntaxe des (in)égalités.

des (in)égalités est donnée dans la Table [O.1](#). Pour combiner les conditions logiques on utilise les opérateurs logiques ; ils sont indiqués dans la Table [O.2](#).

Une structure plus complète est donnée par

```
if (condition)
    code pour if
else
    code pour else
end
```

ou encore

```
if (condition 1)
    code pour if
elseif (condition 2)
    code pour premier elseif
else
    code pour else
end
```

Plusieurs **elseif** dans une même construction sont également permis. Notons que **elseif** doit être «en un seul morceau», l’instruction **else if** étant interprétée comme un **else** suivi d’un **if** (ce qui n’est pas la même chose). Finalement, toutes ces constructions se terminent par un **end**, lequel délimite donc la portée de la condition.

Voici un exemple d’utilisation d’une structure **if** plus complète.

```
if (x<=2 && x>=0)
    disp('x est entre 0 et 2 (inclus)')
elseif (x > 0)
    disp('x est supérieur à 2')
else
    disp('x est négatif')
end
```

opérations	instruction
and	$(cond\ 1)\&\&(cond\ 2)$ <code>and(cond 1, cond 2)</code>
or	$(cond\ 1)\ \ (cond\ 2)$ <code>or(cond 1, cond 2)</code>
not	$\sim(cond\ 1)$ <code>not(cond 1)</code>
xor	<code>xor(cond 1, cond 2)</code>

TABLE O.2 – Opérations logiques.

O.3 Boucles

La structure d'une boucle qui parcourt un ensemble prédéfini de valeurs est :

```
for i = vect
    code pour for
end
```

Ici, `vect` est un vecteur et `i` prend successivement la valeur de tous ses éléments (du premier au dernier). Pour parcourir les valeurs de `imin` à `imax` avec un pas de 1 (avec `imin < imax`) on utilise `imin:imax` à la place de `vect`. Par exemple,

```
for i = 1:10
    disp(i)
end
```

De même, pour parcourir les valeurs de `i1` à `i2` avec un pas de `d`, on utilise `i1:d:i2` ; en particulier, `imax:-1:imin` permet de parcourir les valeurs de `imax` à `imin` dans le sens décroissant avec un pas de 1.

Une boucle peut être interrompue à tout moment avec l'instruction `break`. En cas de plusieurs boucles imbriquées l'instruction `break` interrompt seulement la boucle la plus interne dans laquelle elle se trouve.

Notons que des boucles `while` et `do-until` existent également. Un exemple avec `while` (qui produit le même résultat que l'exemple précédent pour la boucle `for`) est donné ci-dessous

```
i=1;
while (i <= 10)
    disp(i); i++;
end
```

et avec `do-until` :


```
i=1;
do
    disp(i); i++;
until (i > 10)
```

Notez qu'en Octave l'instruction `i++` est équivalente à l'instruction `i=i+1` ; néanmoins, seulement cette dernière est considérée comme valide dans Matlab®.

O.4 Fichiers .m

Un fichier `.m` contient des instructions ou des fonctions Octave sous forme de texte non formaté. Il peut être édité à l'aide d'un programme de traitement de texte standard, tel que **notepad** sous Windows ou **TextEdit** sous Mac OS X. Les éditeurs de code qui reconnaissent l'extension `.m` (comme **gedit** ou **emacs**, disponibles dans un environnement Unix, ou comme **notepad++** sous Windows) permettent une édition plus aisée.

Il y a deux types de fichiers `.m` : des scripts et des fonctions. Si un fichier `.m` ne contient qu'une séquence d'instructions on parle d'un *script*. Lorsque le répertoire de travail d'Octave contient un script `nom1.m`, le fait d'exécuter l'instruction `nom1` est équivalent à effectuer un copier-coller du contenu de ce fichier dans le terminal d'Octave.

Les fichiers `.m` peuvent également contenir des *fonctions*. (voir section [O.5](#) pour plus de détails). Dans ce cas le nom du fichier doit idéalement coïncider avec le nom de la fonction pour qu'Octave sache dans quel fichier il doit chercher la définition de celle-ci.

Dans les deux cas, l'instruction **help** suivie du nom du fichier (sans l'extension `.m`) permet d'afficher le premier commentaire du fichier en question. Par exemple, `help nom1` affiche le premier commentaire du fichier `nom1.m` pour autant que ce fichier se trouve dans le répertoire de travail.

O.5 Fonctions

Les fonctions peuvent être définies de deux manières. Celles qui ont une expression simple et retournent une sortie peuvent être définies via

```
nomf1 = @(x1,...,xn) expression
```

où `x1, ..., xn` (qui peuvent être des vecteurs ou des matrices) sont les entrées (arguments, variables) de la fonction et l'*expression* définit la sortie. Par exemple, la fonction qui pour une entrée renvoie $\sin(2x)$ est définie via

```
f1 = @(x) sin(2*x)
```

Alternativement, on peut définir une fonction avec la syntaxe

```
function [y1,...,ym] = nomf2 (x1,...,xn)
    code de fonction nomf2
```

et enregistrer cette fonction dans un fichier `nomf2.m`. Cette dernière option permet plus

de flexibilité, et en particulier elle permet de renvoyer plusieurs sorties y_1, \dots, y_m . Par ailleurs, notez que le `end` n'est pas nécessaire pour indiquer la fin du code de la fonction.

Par exemple, la fonction suivante (enregistrée dans un fichier `nomf2.m` du répertoire de travail) est mathématiquement équivalente à la fonction `f1` de l'exemple précédent

```
function y = f2 (x)
    y = sin(2*x);
```

Notez ici que la fonction se termine après l'exécution de la dernière ligne, la variable `y` étant spécifiée comme la variable de retour dans la définition de la fonction.

Une fonction peut être passée à une autre fonction comme argument. Dans ce cas, la syntaxe diffère selon que l'on choisisse l'une ou l'autre option décrite plus haut. Nous expliquons les détails à l'aide de l'exemple suivant. La fonction suivante retourne le carré de la valeur de la fonction `f` au point `x` (implicitement, `f` doit être une fonction à une entrée et à une sortie) :

```
function sqf = squaref (f,x)
    sqf = f(x)^2;
```

Dans le répertoire où `squaref.m` est défini, la fonction `f1` – telle que définie plus haut dans cette section – doit être passée par argument de manière suivante

```
squaref (f1,pi/8)    % retourne 0.5
```

Par contre, si on considère la fonction `f2` telle que définie plus haut dans cette section, elle doit être passée par argument de manière suivante

```
squaref (@f2,pi/8)  % retourne 0.5
```

L'exécution de la fonction définie via un fichier `.m` s'arrête après l'exécution de la dernière ligne de cette fonction. On peut aussi arrêter une fonction en utilisant l'instruction `return` (dans ce cas Octave continue l'exécution de la fonction/script appelant) ou `error (message)` (dans ce cas Octave affiche le *message* d'erreur et arrête l'exécution de toutes les fonctions et scripts appelants).

Les variables définies dans le code de la fonction ne sont pas accessibles dans l'environnement d'Octave. Il s'agit de variables locales à la fonction et leur contenu est perdu à la fin de l'exécution. La communication entre l'environnement d'Octave et une fonction

```
function [y1,...,ym] = nomf2 (x1,...,xn)
```

s'effectue via les arguments d'entrée x_1, \dots, x_n et ceux de sortie y_1, \dots, y_m . Réciproquement, les variables de l'environnement d'Octave ne sont pas accessibles dans le code de la fonction.

Petit Guide d'Octave

Ce guide contient des opérations accessibles en Octave et qui typiquement ne sont pas documentées via la commande `help`. Un guide (sensiblement) plus complet est aussi disponible via

http://enacit1.epfl.ch/octave_doc/refcard/refcard-a4.pdf.

Le manuel d'Octave fourni par ses concepteurs est accessible (en anglais) via

<https://www.gnu.org/software/octave/doc/interpreter/>.

Un autre manuel (en français) est disponible ici

https://enacit1.epfl.ch/cours_matlab/.

<code>a:b</code>	si $b \geq a$, crée un vecteur ligne $[a \ a+1 \ \dots \ a+n]$, où n est le plus grand entier tel que $a + n \leq b$; sinon, renvoie un vecteur vide.
<code>a:d:b</code>	si $b \geq a$ et $d > 0$, crée un vecteur ligne $[a \ a+d \ \dots \ a+nd]$, où n est le plus grand entier tel que $a + dn \leq b$; idem si $b \leq a$ et $d < 0$, avec n le plus grand entier satisfaisant $a + dn \geq b$; renvoie un vecteur vide dans les autres cas.
<code>A+B</code> , <code>A-B</code> , <code>A*B</code>	si A et B sont deux matrices (vecteurs et scalaires étant des cas particuliers), effectue l'opération $+$, $-$ ou $*$ matricielle correspondante (bien entendu, si les dimensions sont concordantes);
<code>A\B</code>	si A une matrice carrée régulière, l'instruction donne la solution X du système $AX = B$, pour autant que les dimensions correspondent; notez que B peut être un vecteur colonne ou une matrice (dans ce dernier cas $X = A^{-1}B$ est aussi une matrice). Si A est une matrice surdéterminée, $AX = B$ est résolu au sens des moindres carrés;
<code>A.*B</code> , <code>A./B</code> , <code>A.\B</code>	si $A = (a_{ij})$ et $B = (b_{ij})$ sont deux matrices de mêmes dimensions $m \times n$, le résultat est une matrice $C = (c_{ij})$ de dimensions $m \times n$ telle que $c_{ij} = a_{ij} * b_{ij}$, $c_{ij} = a_{ij}/b_{ij}$ ou $c_{ij} = a_{ij}\backslash b_{ij} = b_{ij}/a_{ij}$; l'opération est donc effectuée élément par élément;
<code>A^p</code> , <code>A.^p</code>	la première instruction produit l'exposant p de la matrice $A = (a_{ij})$, c'est-à-dire A^p , alors que la seconde fournit la matrice d'éléments $(a_{ij})^p$;
<code>A'</code>	transposition de A (si A est une matrice complexe, transposition et conjugaison complexe);
<code>@(x1,...,xn) expr</code>	renvoie une fonction qui dépend des variables x_1, \dots, x_n (dont certaines peuvent être vectorielles ou matricielles) définie par l'expression <code>expr</code> (pouvant également être un vecteur ou une matrice);
<code>i</code> , <code>e</code>	voir <code>help i</code> , <code>help e</code> ; attention, si une variable portant le même nom est créée, la valeur par défaut est automatiquement réécrite avec la nouvelle variable; essayez <code>disp(e)</code> ; <code>e=1</code> ; <code>disp(e)</code> .

Représentation en virgule flottante, stabilité et conditionnement

Ce chapitre aborde les problématiques liées à la représentation des nombres réels dans les ordinateurs modernes. La *représentation en virgule flottante* – le standard actuel – est présentée dans la première section. On y voit que ce choix de représentation a une conséquence sur les erreurs d'arrondi : il met l'accent sur les erreurs relatives. La deuxième section est consacrée aux notions de *stabilité* et de *conditionnement*. Bien que ces notions soient formellement indépendantes des erreurs d'arrondi, leur utilisation est motivée par la nécessité de comprendre l'effet global de ces erreurs.

1.1 Représentation en virgule flottante

1.1.1 Motivation

L'utilisation de nombres réels est inévitable en analyse numérique. Pourtant, leur manipulation dans un environnement numérique doit se faire avec précautions, au risque de ne pas obtenir le résultat voulu. Les effets négatifs – souvent limités, ce qui explique le peu d'attention qu'on leur prête parfois – peuvent aussi avoir des conséquences importantes, comme le montrent les exemples suivants¹.

EXEMPLE 7. (SYSTÈME PATRIOT)



Durant la Guerre du Golfe, le système d'interception des missiles Patriot n'a pas réussi à intercepter le Scud irakien, ce qui a coûté la vie à 28 soldats. La défaillance a été imputée aux erreurs d'arrondi dans l'estimation du temps écoulé depuis la mise en opération du système : de 0.34 secondes après 100 heures. Cette erreur était connue des concepteurs mais n'a été que partiellement corrigée, ce qui a

1. Vous trouverez plus d'exemples sur la page de K. Vuik <http://ta.twi.tudelft.nl/users/vuik/wi211/disasters.html>

rendu l'accident possible.²

EXEMPLE 8. (FUSÉE ARIANE 5)



Lors du premier lancement d'Ariane 5 en 1996 la fusée est devenue incontrôlable 30 secondes après le décollage et a dû être détruite. La perte de contrôle a été provoquée par le dépassement de la valeur maximale du registre qui contenait la vitesse horizontale.

1.1.2 Représentation en virgule flottante

L'utilisation de nombres réels sur un ordinateur va de pair avec les erreurs d'arrondi. Cela résulte du fait que seulement un ensemble fini de nombres réels peut être représenté exactement (c'est-à-dire sans aucune approximation) dans la mémoire d'une machine. En particulier, les nombres réels n'appartenant pas à l'ensemble des réels représentables doivent être approchés par un élément de cet ensemble, et une erreur d'arrondi est potentiellement commise. Il y a pourtant une autre source d'erreurs, plus récurrente en analyse numérique : à chaque fois qu'une opération arithmétique (addition, multiplication, etc.) impliquant des nombres réels est effectuée, son résultat *ne fait* en général *pas* partie de l'ensemble des réels représentables, et une erreur d'arrondi est donc potentiellement commise lors de chaque opération arithmétique.

Pour comprendre les effets des erreurs d'arrondi il faut introduire au préalable la représentation des nombres réels sur ordinateur. Les ordinateurs modernes utilisent une *représentation en virgule flottante*. Cette représentation est fournie dans l'équation (1.1), où le membre de gauche correspond à la notation d'un nombre réel dans cette représentation (notez le caractère surligné de la mantisse, qui la distingue de l'écriture décimale habituelle), alors que le membre de droite indique comment évaluer la valeur de ce nombre réel ; la représentation est donc donnée par

$$\pm \overline{0.d_1 d_2 \cdots d_t} \cdot \beta^e = \pm \beta^e \sum_{i=1}^t \frac{d_i}{\beta^i}, \quad (1.1)$$

avec

- β : la base ;
- t : le nombre de chiffres significatifs ;
- d_i : le i -ème chiffre significatif ($0 \leq d_i \leq \beta - 1$) ;
- e : l'exposant ($e_{\min} \leq e \leq e_{\max}$, où e_{\min} et e_{\max} définissent la plage de variation de l'exposant) ;

2. Vous trouverez plus de détails dans «Roundoff Error and the Patriot Missile» de R. Skeel, disponible via <http://metronu.ulb.ac.be/MATH-H-202/Patriot-dharan-skeel-siam.pdf>.

les chiffres significatifs $\overline{0.d_1d_2\cdots d_t}$ forment la *mantisse*. Les deux bases utilisées en pratique sont $\beta = 2$ (base binaire) et $\beta = 10$ (base décimale), cette dernière étant destinée à des calculatrices.

EXEMPLE 9. Considérons les représentations avec $t = 3$ chiffres significatifs. Le réel 2 en base décimale peut s'écrire $\overline{0.200} \cdot 10^1$ (car $2 = (\frac{2}{10} + \frac{0}{10^2} + \frac{0}{10^3})10^1$), mais aussi $\overline{0.020} \cdot 10^2$. En base binaire il s'écrit $\overline{0.100} \cdot 2^2$ (car $2 = (\frac{1}{2} + \frac{0}{2^2} + \frac{0}{2^3})2^2$). On constate en particulier que la mantisse dépend du choix de la base.

L'exemple précédent montre que certains nombres réels peuvent posséder des représentations multiples. Pour lever cette redondance on se restreint à une représentation *normalisée* pour laquelle

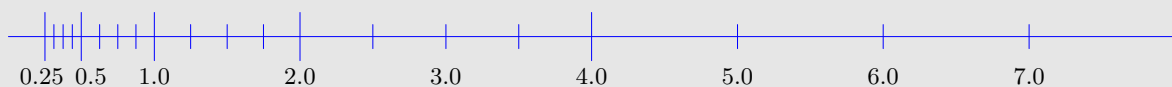
$$d_1 \neq 0.$$

Dans une base binaire ($\beta = 2$) cette représentation implique $d_1 = 1$, et donc le premier chiffre significatif ne doit pas être spécifié (ni gardé en mémoire). L'ensemble des réels possédant une représentation normalisée (1.1) est noté³ \mathbb{F} ; en d'autres termes

$$\mathbb{F} = \{ x \mid x = \pm \overline{0.d_1d_2\cdots d_t} \cdot \beta^e, e_{\min} \leq e \leq e_{\max} \}.$$

On notera x_{\max} l'élément positif le plus grand de \mathbb{F} et x_{\min} l'élément positif le plus petit; ce dernier est non nul car $0 \notin \mathbb{F}$ (pourquoi?).

EXEMPLE 10. La représentation de la partie positive de \mathbb{F} pour $\beta = 2$, $t = 3$, $e_{\min} = -1$ et $e_{\max} = 3$ est schématisée sur la figure suivante; la partie négative de cette représentation est obtenue en appliquant la symétrie par rapport au 0. Notons en particulier que $x_{\max} = \overline{0.111} \cdot 2^3 = 7.0$ et $x_{\min} = \overline{0.100} \cdot 2^{-1} = 0.25$. (Pouvez-vous retrouver les autres éléments de \mathbb{F} représentés sur la figure?)



1.1.3 Erreurs d'arrondi

Dans une représentation en virgule flottante, la distance entre deux éléments consécutifs de \mathbb{F} reste globalement proportionnelle à la valeur absolue de ces éléments. Cela a comme conséquence que l'erreur absolue maximale commise en approchant un réel x quelconque par un élément de \mathbb{F} est elle aussi globalement proportionnelle à $|x|$, et que dès lors l'erreur relative maximale reste uniformément bornée.

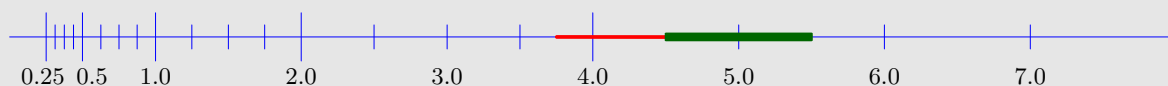
3. Bien que cet ensemble dépende de β , t , e_{\min} et e_{\max} , on n'explicite pas cette dépendance.

Pour être plus précis, on considère la fonction $\text{fl} : \mathbb{R} \mapsto \mathbb{F}$ qui approche tout réel x par un élément de la représentation $\text{fl}(x) \in \mathbb{F}$ comme suit

$$\text{fl}(x) \equiv \text{le réel dans } \mathbb{F} \text{ le plus proche}^4 \text{ de } x.$$

La fonction $\text{fl}(x)$ décrit la manière habituelle⁵ dont les réels sont arrondis en pratique, du moins pour autant que son argument x satisfasse $x_{\min} \leq |x| \leq x_{\max}$, c'est-à-dire pour autant que x ne dépasse pas les limites de la représentation. Les cas où x est au delà de ces limites sera discuté dans la sous-section 1.1.7.

EXEMPLE 10. (SUITE) : Pour la représentation en virgule flottante de l'Exemple 10 on a $\text{fl}(x) = 4.0$ pour tout x entre 3.75 et 4.5 (segment épais sur la figure ci-dessous), et $\text{fl}(x) = 5.0$ pour tout x entre 4.5 et 5.5 (segment très épais).



On peut maintenant spécifier ce qu'on entend par erreurs d'arrondi : $|\text{fl}(x) - x|$ est l'*erreur absolue d'arrondi* alors que $|\text{fl}(x) - x|/|x|$ est l'*erreur relative d'arrondi* (ou *erreur d'arrondi* tout court). Le comportement de ces erreurs est très différent, comme le montre l'exemple suivant.

EXEMPLE 10. (FIN) : On observe sur la figure de l'Exemple 10 (suite) que l'erreur absolue d'arrondi $|\text{fl}(x) - x|$ vaut au plus la moitié de la distance entre deux éléments de \mathbb{F} qui entourent x . En particulier

- pour $x \in [4, 7]$ on a $|\text{fl}(x) - x| \leq 0.5$
- pour $x \in [2, 4]$ on a $|\text{fl}(x) - x| \leq 0.25$
- ...
- pour $x \in [0.25, 0.5]$ on a $|\text{fl}(x) - x| \leq 2^{-5}$

Par conséquent, l'«amplitude» de l'erreur *absolue* d'arrondi varie selon la région dans laquelle x se situe. On constate par contre que l'erreur *relative* d'arrondi est bornée par

$$\frac{|\text{fl}(x) - x|}{|x|} \leq \frac{1}{8},$$

indépendamment de la région dans laquelle x se situe, mais pour autant que $|x| \in [0.25, 7]$. (Pourquoi cette condition est-elle importante?)

4. En cas d'ex aequo ($d_{t+1} = \beta/2$) on arrondi vers l'élément de \mathbb{F} dont le dernier chiffre significatif est pair.

5. D'autres manières d'arrondir sont également possibles, mais elles ne changent en rien les résultats de ce chapitre, mis à part la valeur de l'unité d'arrondi – une constante introduite plus tard.

Les observations de l'exemple précédent peuvent être généralisées. Pour cela on considère que le développement (potentiellement infini) d'un nombre réel x en base β est donnée par

$$x = \pm \overline{0.d_1 d_2 \cdots d_t d_{t+1} \cdots} \cdot \beta^e; \quad (1.2)$$

avec l'exposant e choisi pour satisfaire $d_1 \neq 0$. On suppose aussi que le nombre réel x satisfait $x_{\min} \leq |x| \leq x_{\max}$, c'est-à-dire qu'il ne sort pas des limites de la représentation en virgule flottante \mathbb{F} . Dès lors, x est toujours entouré par deux éléments consécutifs x_- , x_+ de \mathbb{F} , et l'erreur absolue d'arrondi satisfait alors

$$|\text{fl}(x) - x| \leq \frac{1}{2} \cdot |x_+ - x_-| = \frac{1}{2} \cdot \underbrace{0.00 \cdots 01}_{\beta^{-t}} \cdot \beta^e = \frac{1}{2} \beta^{e-t}, \quad (1.3)$$

où la première inégalité traduit l'observation, faite précédemment, que l'erreur absolue d'arrondi vaut au plus la moitié de la distance entre deux éléments de \mathbb{F} qui entourent x . Par ailleurs, le développement de x dans (1.2) et le fait que $d_1 \neq 0$ impliquent

$$|x| \geq \overline{0.1} \cdot \beta^e = \beta^{e-1}. \quad (1.4)$$

La borne sur l'erreur relative d'arrondi est alors obtenue en combinant les inégalités (1.3) et (1.4), ce qui donne

$$\frac{|\text{fl}(x) - x|}{|x|} \leq \frac{\frac{1}{2} \beta^{e-t}}{\beta^{e-1}} = \underbrace{\frac{1}{2} \beta^{1-t}}_{=: u}. \quad (1.5)$$

La quantité u , qui correspond au dernier terme de l'expression précédente, est appelée *unité d'arrondi*; elle ne dépend que de la base β considérée et du nombre t de chiffres significatifs.

La relation (1.5) peut être mise sous une forme plus exploitable. Pour ce faire notons que toute erreur relative $\epsilon_{\text{rel}} = |x - \hat{x}|/|x|$ de \hat{x} par rapport à $x \neq 0$ satisfait aussi la relation suivante (pourquoi?)

$$\hat{x} = x(1 + \epsilon), \quad |\epsilon| = \epsilon_{\text{rel}}.$$

En transposant cette observation à l'inégalité (1.5) on a alors

$$\text{fl}(x) = x(1 + \epsilon), \quad |\epsilon| \leq u. \quad (1.6)$$

De manière similaire à (1.6) on montre aussi que

$$\text{fl}(x) = \frac{x}{1 + \epsilon'}, \quad |\epsilon'| \leq u.$$

1.1.4 Standard IEEE

Le standard IEEE 754 porte sur la représentation des nombres réels et sur les opérations effectuées avec des réels. Ce standard a été proposé en 1985 pour permettre un traitement uniforme des erreurs d'arrondi et rendre ainsi possible le développement des

codes numériques portables pour les machines qui respectent le standard, c'est-à-dire virtuellement sur toutes les machines modernes.

Pour une base binaire, le standard spécifie deux principaux formats en virgule flottante : **single** (*simple précision*, 32 bits) et **double** (*double précision*, 64 bits). Les deux formats ont la représentation suivante en mémoire

signe	exposant	mantisse
-------	----------	----------

dont les différentes sections correspondent à l'emplacement du signe, de l'exposant et de la mantisse. Les spécifications principales de ces formats sont comme suit :

single			double		
1 bit	8 bits	23 bits	1 bit	11 bits	52 bits
$e_{\min} = -125$			$e_{\min} = -1021$		
$e_{\max} = 128$			$e_{\max} = 1024$		
$x_{\min} \approx 1.2 \cdot 10^{-38}$			$x_{\min} \approx 2.2 \cdot 10^{-308}$		
$x_{\max} \approx 3.4 \cdot 10^{38}$			$x_{\max} \approx 1.8 \cdot 10^{308}$		
$u \approx 6.0 \cdot 10^{-8}$			$u \approx 1.1 \cdot 10^{-16}$		

On constate que la plage de valeurs entre x_{\min} et x_{\max} est suffisante pour éviter autant que possible le dépassement de ces valeurs. Par ailleurs, l'unité d'arrondi en double précision (le format par défaut en Octave) vaut $u \approx 1.1 \cdot 10^{-16}$. Notons aussi que certaines de ces informations sont également accessibles en Octave, via les commandes **realmax**, **realmin** et **eps** ($= 2u$).

Parmi ces données, la valeur de l'unité d'arrondi u en double précision est celle qui nous sera utile par la suite. Elle est obtenue à partir des spécifications du format **double** de manière suivante :

$$u = \frac{1}{2} \beta^{1-t} = \frac{1}{2} 2^{-52} \approx 1.1 \cdot 10^{-16},$$

où $\beta = 2$ et $t = 53$ (car le premier chiffre significatif n'est pas conservé en mémoire).

1.1.5 Modèle d'arithmétique

Le résultat d'une opération arithmétique avec des éléments de \mathbb{F} appartient lui-même rarement à \mathbb{F} ; il est donc important de savoir comment ce résultat est «converti» vers un élément de cet ensemble. La réponse s'avère simple : même si le résultat de la conversion varie d'un ordinateur à l'autre, l'erreur relative commise reste de l'ordre de grandeurs de l'unité d'arrondi u .

Plus précisément, soient

x, y deux réels dans \mathbb{F} ;

◦ une opération en arithmétique exacte ($+$, $-$, \cdot ou $/$) ;

⊙ l'opération correspondante en virgule flottante (\oplus , \ominus , \odot ou \oslash) ; notez en particulier que $x \odot y \in \mathbb{F}$.

De plus, supposons que $x_{\min} \leq |x \circ y| \leq x_{\max}$. Alors le *modèle standard d'arithmétique en virgule flottante* (satisfait avec IEEE) spécifie que

$$x \odot y = (x \circ y)(1 + \epsilon), \quad \text{avec } |\epsilon| \leq u. \quad (1.7)$$

Pour interpréter cette relation il suffit de la comparer avec (1.6). On constate alors que l'opération en virgule flottante \odot mène à des erreurs d'arrondi comparables à celles produites par la conversion du résultat de l'opération exacte \circ dans \mathbb{F} .⁶

Ce modèle est à la base de notre compréhension des effets dus aux erreurs d'arrondi. Pour étudier la propagation de ces erreurs on utilise aussi des règles suivantes.

Règles (de propagation des erreurs d'arrondi) :

Pour l'ensemble des règles, on suppose $|\epsilon_i|, |\epsilon'_i| \leq u$ et $\alpha, \beta \in \mathbb{R}$

1. $\alpha\epsilon_1 \pm \beta\epsilon_2 = (|\alpha| + |\beta|)\epsilon_3$
2. $(1 + \alpha\epsilon_1)(1 + \beta\epsilon_2) = 1 + (|\alpha| + |\beta|)\epsilon_3 + |\alpha\beta|\mathcal{O}(u^2)$
3. $\frac{1}{1 + \alpha\epsilon_4 + \mathcal{O}(u^2)} = 1 + \alpha\epsilon'_4 + |\alpha|^2\mathcal{O}(u^2)$

On vérifie la règle 1 en notant que

$$|\epsilon_3| = \frac{|\alpha\epsilon_1 \pm \beta\epsilon_2|}{|\alpha| + |\beta|} \leq \frac{|\alpha| \overbrace{|\epsilon_1|}^{\leq u} + |\beta| \overbrace{|\epsilon_2|}^{\leq u}}{|\alpha| + |\beta|} \leq u.$$

La règle 2 est une conséquence directe de la règle 1 (pourquoi?) alors que la règle 3 découle du développement de Taylor de $1/(1+x)$ (pourquoi?).

Notons par ailleurs que les facteurs devant $\mathcal{O}(u^2)$ dans les règles 2 et 3 sont omis dans la suite sous l'hypothèse (souvent implicite) que $\alpha, \beta = \mathcal{O}(1)$.

EXEMPLE 11. On se propose d'estimer l'erreur d'arrondi sur le résultat du calcul suivant

$$(1 \oslash 3) \odot 3.$$

En supposant que tous les ϵ_i satisfont $|\epsilon_i| \leq u$, on a alors

$$\begin{aligned} (1 \oslash 3) \odot 3 &= 1/3(1 + \epsilon_1) \odot 3 && \text{(par (1.7))} \\ &= (1/3(1 + \epsilon_1) \cdot 3)(1 + \epsilon_2) && \text{(par (1.7))} \\ &= 1 \cdot (1 + \epsilon_1)(1 + \epsilon_2) \\ &= 1 \cdot (1 + 2\epsilon_3) + \mathcal{O}(u^2). && \text{(par règle 2)} \end{aligned}$$

6. Pour être complet notons que le standard IEEE est plus strict, et exige que $x \odot y = \text{fl}(x \circ y)$, c'est-à-dire que l'opération en virgule flottante donne le même résultat que l'arrondi de l'opération exacte. Néanmoins, pour comprendre le comportement des erreurs d'arrondi dans les algorithmes considérés ici la relation (1.7) est suffisante.

Par conséquent, l'erreur relative d'arrondi en double précision est bornée (à $\mathcal{O}(u^2)$ près) par $2u \approx 2.2 \cdot 10^{-16}$.

1.1.6 Annulation

L'exemple précédant décrit le cas où les erreurs d'arrondi s'accumulent de manière progressive, ce qui est à la fois inévitable et peu inquiétant, étant donné que l'unité d'arrondi u est suffisamment petite pour que la perte de précision soit considérée minime. Il y a pourtant des situations où une importante perte de précision peut survenir suite à quelques opérations seulement. C'est notamment ce qui arrive avec le phénomène d'*annulation*. Avant d'en présenter une description plus détaillée commençons par un exemple.

EXEMPLE 12. Soient $\beta = 10$ et $t = 3$. Évaluons la différence

$$(\overline{0.102} \cdot 10^1 \oplus \overline{0.6} \cdot 10^{-2}) \ominus (\overline{0.102} \cdot 10^1 \oplus \overline{0.4} \cdot 10^{-2}).$$

La première parenthèse donne $1.02 \oplus 0.006 = 1.03$, alors que la deuxième vaut $1.02 \oplus 0.004 = 1.02$; la différence vaut donc 0.01. Notons que si toutes les opérations sont exactes (ou si $t \geq 4$), le résultat du calcul devient 0.002; le résultat obtenu numériquement est donc 5 fois plus grand que celui en arithmétique exacte (et l'erreur relative est de 4, ou 400%!), alors que l'unité d'arrondi dans cette représentation ne vaut que $u = 0.005$. La raison principale de cette perte de précision est la soustraction de deux réels très proches qui ont été préalablement entachés par des erreurs d'arrondi.

Pour analyser ce phénomène, supposons que deux réels x, y ont été contaminés par des erreurs d'arrondi :

$$\begin{aligned} \tilde{x} &= x(1 + \epsilon_1) \in \mathbb{F}, & |\epsilon_1| &\leq u, \\ \tilde{y} &= y(1 + \epsilon_2) \in \mathbb{F}, & |\epsilon_2| &\leq u. \end{aligned}$$

On se propose d'estimer l'erreur relative d'arrondi de $\tilde{x} \ominus \tilde{y}$ comme approximation de $x - y$. Pour ce faire, notons (avec comme avant $|\epsilon_i| \leq u$) que

$$\tilde{x} \ominus \tilde{y} = (\tilde{x} - \tilde{y})(1 + \epsilon_3), \quad (\text{par (1.7)})$$

où

$$\tilde{x} - \tilde{y} = (x - y) + (x\epsilon_1 - y\epsilon_2) = (x - y)(1 + \delta u),$$

avec

$$|\delta| = \left| \frac{x\epsilon_1 - y\epsilon_2}{x - y} \right| \cdot \frac{1}{u} \leq \frac{|x| + |y|}{|x - y|}. \quad (1.8)$$

En regroupant ces trois expressions ensemble et en combinant les facteurs de propagation d'erreur on a

$$\tilde{x} \ominus \tilde{y} = (x - y)(1 + \delta u + \epsilon_3) + \mathcal{O}(\delta u^2).$$

En particulier, si $x \approx y$ on risque⁷ d'avoir $\delta \gg 1$, ce qui implique une perte de précision.

Pour éviter toute confusion, notons que le phénomène d'annulation ne survient que si les nombres réels qu'on soustrait sont déjà contaminés par des erreurs d'arrondi car, dans le cas contraire, le modèle d'arithmétique en virgule flottante nous indique que l'erreur relative commise est bornée par u . Cette constatation est confirmée par l'observation que dans le raisonnement précédent $\epsilon_1 = \epsilon_2 = 0$ implique aussi $\delta = 0$.

Notons pour finir que le phénomène de perte soudaine de précision n'arrive qu'avec la soustraction de deux nombres proches (ou l'addition de deux nombres de signe opposé et d'amplitude proche). L'addition de nombres de même signe, la multiplication et la division ne peuvent pas mener à une perte de précision similaire, du moins si le résultat de l'opération reste dans les limites de la représentation.

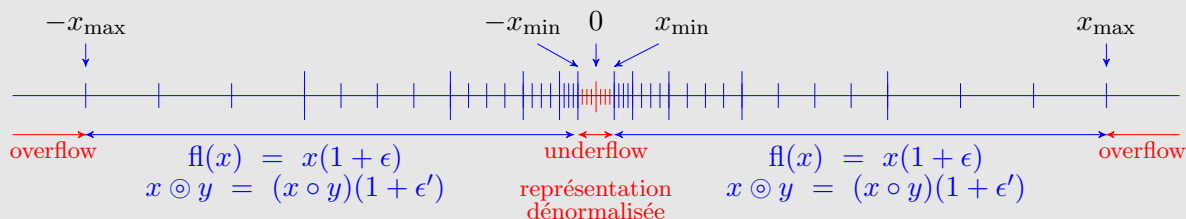
1.1.7 Compléments

Jusqu'à présent la discussion a porté sur le comportement des erreurs d'arrondi dans le cas où $x_{\min} \leq |x| \leq x_{\max}$. Bien que x_{\min} soit la plus petite valeur positive dans la représentation normalisée, la région de l'axe réel correspondant à $|x| < x_{\min}$ n'est en pratique pas déserte : une représentation dénormalisée qui inclut 0 est utilisée ici pour approcher les valeurs réelles (cf. Exemple 13). Cela implique potentiellement une importante perte de précision si le calcul «passe» par cette région (on parle alors d'*underflow*), mais cette perte est moindre que celle qu'on obtiendrait en arrondissant à $\pm x_{\min}$. Insistons sur le fait que 0 fait bien partie de la représentation dénormalisée ; plus précisément, il y a deux zéros : $+0$ et -0 .

À l'inverse, si $|x|$ dépasse⁸ x_{\max} , on parle d'*overflow*, et le résultat obtenu est converti en $\pm \text{Inf}$; $+\text{Inf}$ et $-\text{Inf}$ font donc en pratique partie de la représentation. On obtient aussi $\pm \text{Inf}$ avec la division d'un réel non nul par ± 0 , avec le signe $+$ ou $-$ choisi selon le signe des éléments de la division.

Finalement, le résultat d'une opération qui ne se situe pas dans $\mathbb{R} \cup \{\pm\infty\}$ est converti en NaN (Not-a-Number). Ceci arrive notamment pour $0/0$, $0 \cdot \infty$ ou encore $\sqrt{-1}$ (du moins si le traitement des nombres complexes n'est pas prévu).

EXEMPLE 13. L'ensemble des éléments de \mathbb{F} pour $\beta = 2$, $t = 3$, $e_{\min} = -1$ et $e_{\max} = 3$ complété par une représentation dénormalisée est représenté ci-dessous.



7. La borne supérieure (1.8) sur δ peut être atteinte pour $\epsilon_1 = -\text{signe}(x)\text{signe}(y)\epsilon_2 = u$.

8. Pour simplifier, on ne tient pas compte des valeurs de $|x|$ qui dépassent légèrement x_{\max} mais qui peuvent être arrondies à x_{\max} .

1.2 Stabilité et conditionnement

La section précédente fournit une description « microscopique » du traitement des erreurs d'arrondi. À l'inverse, les considérations de cette section apportent une caractérisation « macroscopique » des phénomènes correspondants.

1.2.1 Stabilité directe

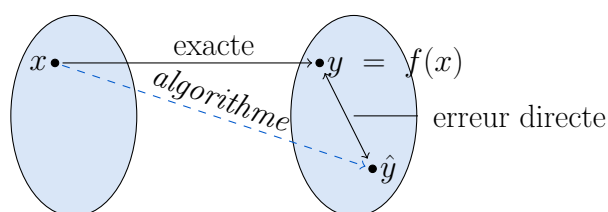


FIGURE 1.1 – Erreur directe.

Soient x la variable qui représente les données d'un problème particulier et y la variable qui représente sa solution, que l'on supposera existante et unique. Supposons qu'il existe une fonction f résolvante qui associe aux données x la solution y correspondante ; c'est-à-dire

$$y = f(x).$$

Tout algorithme qui fournit la solution du problème peut en *arithmétique exacte* jouer le rôle de la fonction $f(x)$. La situation est différente en *arithmétique machine*, car un algorithme ne fournit alors qu'une solution approchée \hat{y} qui de plus dépend de cet algorithme. La différence $\hat{y} - y$ décrit alors la propagation des erreurs d'arrondi dans l'algorithme ; elle est appelée *erreur directe* ; voir Figure 1.1.

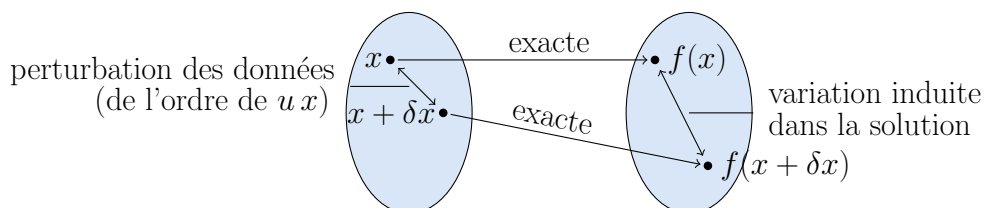


FIGURE 1.2 – Variation de la solution due à une perturbation dans les données.

On caractérise l'amplitude de l'erreur directe en la comparant aux variations induites dans la solution exacte par des perturbations – de l'ordre des erreurs d'arrondi – dans les données ; une telle variation est schématisée sur la Figure 1.2. Si l'erreur directe est inférieure ou comparable aux variations en question, on parle de la *stabilité directe*. En d'autres termes, on considère qu'un algorithme qui produit \hat{y} a la stabilité directe en x

s'il existent $C_1, C_2 \geq 1$ suffisamment petits⁹ tels que

$$\|\hat{y} - y\| \leq C_1 \|f(x + \delta x) - f(x)\| \quad (1.9)$$

pour au moins un δx tel que $\|\delta x\|/\|x\| \leq C_2 u$. Si $y = f(x) \neq 0$, la relation (1.9) peut également s'écrire sous forme d'erreurs relatives ; on a alors

$$\frac{\|\hat{y} - y\|}{\|y\|} \leq C_1 \frac{\|f(x + \delta x) - f(x)\|}{\|f(x)\|}. \quad (1.10)$$

La motivation de comparer l'effet des erreurs d'arrondi dans un algorithme à celui des erreurs d'arrondi dans les données est double.

1. Tout d'abord, les données de départ ne sont en règle générale pas représentables exactement, et une erreur d'approximation dans les données est alors commise. Un algorithme stable ne doit donc pas introduire des erreurs sensiblement plus importantes, et ce n'est pas grave s'il introduit des erreurs comparables.
2. Même lorsque les données de départ sont représentables exactement, les premières opérations avec ces données mènent à des effets comparables à ceux introduits par des erreurs d'arrondi dans les données.

1.2.2 Conditionnement

Notons que si le membre de gauche dans (1.10) dépend d'un algorithme concret, le membre de droite ne dépend lui que du problème considéré. Si on met en évidence l'erreur sur les données dans le membre de droite (toujours à condition que $f(x) \neq 0$), on a

$$\frac{\|f(x + \delta x) - f(x)\|}{\|f(x)\|} = \underbrace{\frac{\|f(x + \delta x) - f(x)\| \|x\|}{\|f(x)\| \|\delta x\|}}_{\text{facteur d'amplification}} \cdot \underbrace{\frac{\|\delta x\|}{\|x\|}}_{\leq C_2 u}, \quad (1.11)$$

ce qui permet d'isoler le facteur qui multiplie l'erreur relative (en norme) dans les données pour fournir l'erreur relative (en norme) dans la solution. Le pire des facteurs pour lequel des perturbations sont limitées avec $\|\delta x\| \leq \epsilon \|x\|$ est donné par

$$\kappa_\epsilon(x) := \sup_{\|\delta x\| \leq \epsilon \|x\|} \frac{\|f(x + \delta x) - f(x)\| / \|f(x)\|}{\|\delta x\| / \|x\|}.$$

Pour $\epsilon = C_2 u$, l'inégalité (1.11) implique alors

$$\frac{\|f(x + \delta x) - f(x)\|}{\|f(x)\|} \leq C_2 \kappa_\epsilon(x) u. \quad (1.12)$$

En pratique, on utilise plutôt le facteur

$$\kappa(x) = \lim_{\epsilon \rightarrow 0} \kappa_\epsilon(x),$$

9. En d'autres termes, $C_1, C_2 = \mathcal{O}(1)$ sont acceptables, même si les exigences concrètes peuvent varier d'un problème à l'autre.

car d'une part, $\kappa(x)$ et $\kappa_\epsilon(x)$ sont typiquement très proches pour un $\epsilon = C_2 u$ qui est alors très petit par rapport à 1 et, d'autre part, $\kappa(x)$ est plus facile à calculer ; le facteur $\kappa(x)$ est appelé le *conditionnement* du problème.

Le conditionnement caractérise la sensibilité (relative) de la solution du problème aux erreurs dans les données. Ainsi, si $\kappa(x) \gg 1$ le problème est sensible à ces erreurs ; on parle alors d'un problème *mal conditionné*. Dans le cas contraire, le problème est *bien conditionné*.

Le conditionnement d'un problème donné peut souvent être évalué. Par exemple, si $f(x)$ est non nul et différentiable en x , et si $f'(x)$ est sa matrice jacobienne, alors¹⁰

$$\kappa(x) = \frac{\|f'(x)\| \|x\|}{\|f(x)\|}. \quad (1.13)$$

EXEMPLE 14. L'opération d'évaluation de la racine carrée d'un réel $x > 0$ correspond à $f(x) = \sqrt{x}$. Le conditionnement est alors donné (via (1.13)) par

$$\kappa(x) = \frac{|1/(2\sqrt{x})| |x|}{|\sqrt{x}|} = \frac{1}{2}.$$

On conclut qu'il s'agit d'un problème bien conditionné.

EXEMPLE 15. L'opération de soustraction correspond à $f(x_1, x_2) = x_1 - x_2$. Comme $f'(x_1, x_2) = (1, -1)$ (il s'agit du gradient de f), le conditionnement vaut alors (en utilisant la norme euclidienne dans (1.13))

$$\kappa(x) = \frac{\sqrt{2} \sqrt{x_1^2 + x_2^2}}{|x_1 - x_2|}.$$

On risque ainsi une perte de précision si $x_1 \approx x_2$ (on le savait déjà!).

Pour revenir à la stabilité on notera que la définition de la stabilité directe en version relative (1.10) combiné avec la relation (1.12) impliquent aussi

$$\frac{\|\hat{y} - y\|}{\|y\|} \leq C_1 C_2 \kappa_\epsilon(x) u \approx C_1 C_2 \kappa(x) u. \quad (1.14)$$

Il est important de bien interpréter l'inégalité précédente. Le fait qu'elle soit vérifiée avec un $C_1 C_2 > 1$ petit pour un triplet de valeurs x , y et \hat{y} ne signifie pas pour autant que l'algorithme qui a fourni \hat{y} possède la stabilité directe (pourquoi?). Par contre, le fait que cette inégalité ne soit pas satisfaite pour un $C_1 C_2 > 1$ petit implique bien que l'algorithme considéré n'est pas stable.

10. La démonstration de cette formule découle du fait qu'une fonction différentiable en x satisfait $f(x + \delta x) = f(x) + f'(x)\delta x + o(\|\delta x\|)$ ainsi que du fait que $\max_{\delta x} \|f'(x)\delta x\|/\|\delta x\| = \|f'(x)\|$ si $f'(x)$ est une matrice (vecteur, scalaire).

1.2.3 Stabilité inverse

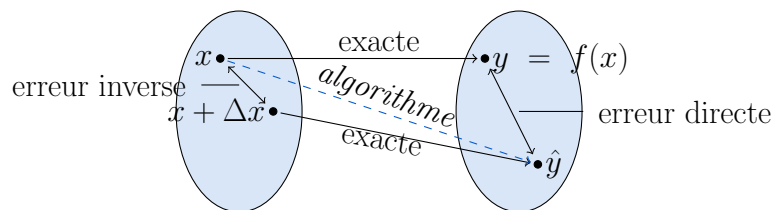


FIGURE 1.3 – Erreurs directe et inverse.

Il est parfois possible d'interpréter directement (et ne pas seulement comparer) les erreurs d'arrondi dans un algorithme en arithmétique machine comme des erreurs dans les données. Dans ce cas il est commode d'utiliser l'*erreur inverse*. Une erreur inverse d'un algorithme produisant la solution approchée \hat{y} est une perturbation Δx des données qui satisfait

$$f(x + \Delta x) = \hat{y}. \quad (1.15)$$

On dit qu'un *algorithme a la stabilité inverse* si au moins un tel Δx satisfaisant (1.15) existe et vérifie (pour un $C \geq 1$ petit)

$$\frac{\|\Delta x\|}{\|x\|} \leq Cu. \quad (1.16)$$

Notons qu'un Δx qui satisfait (1.15) n'existe pas nécessairement, la stabilité inverse est donc une notion plus restreinte. Elle fournit aussi une caractérisation plus forte que la stabilité directe. En effet, comme la stabilité inverse implique

$$\|\hat{y} - y\| = \|f(x + \Delta x) - f(x)\|$$

pour un Δx de l'ordre de grandeur des erreurs d'arrondi (car (1.16)), elle implique aussi la stabilité directe (1.9) avec $C_1 = 1$ et $C_2 = C$.

2.1 Problème

$$\left\{ \begin{array}{l} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2 \\ \qquad \qquad \qquad \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n = b_m \end{array} \right.$$
$$A\mathbf{x} = \mathbf{b}, \quad (2.1)$$
$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}, \quad \mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \text{ et } \mathbf{b} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix}$$

sont, respectivement, la matrice des coefficients, le vecteur des inconnues et le vecteur des seconds membres.

Dans ce chapitre on considère les systèmes *réguliers*, c'est-à-dire des systèmes dont la matrice A est carrée ($m = n$) et de rang maximal ($\text{rang}(A) = n$) ; pour une matrice A carrée la condition sur le rang maximal est satisfaite si et seulement si $\det(A) \neq 0$, ou encore si et seulement le système (2.1) possède une et une seule solution.

Dans un chapitre ultérieur on aborde également les systèmes surdéterminés de rang maximal, c'est-à-dire les systèmes qui satisfont $m \geq n = \text{rang}(A)$; la résolution de ces systèmes au sens des moindres carrés sera alors considérée.

2.2 Conditionnement

Ici on analyse la sensibilité relative de la solution d'un système linéaire régulier aux erreurs dans les données. Cette sensibilité est caractérisée par le conditionnement du problème qui, pour rappel, est défini comme le pire des facteurs qui amplifient les erreurs relatives dans les données pour obtenir les erreurs relatives dans la solution ; ce facteur est évalué pour les erreurs tendant vers zéro. Pour le système linéaire (2.1), les données du problème sont \mathbf{b} , A , et les perturbations correspondantes sont notées $\delta\mathbf{b}$, δA ; la solution du problème est \mathbf{x} et celle du problème perturbé est notée $\mathbf{x} + \delta\mathbf{x}$.

On considère les effets des perturbations $\delta\mathbf{b}$ et δA séparément. Les propriétés des normes utilisées sont rappelées en Annexe 2.A.1.

CAS 1. On étudie d'abord les perturbations $\delta\mathbf{b}$ de \mathbf{b} . Si $\mathbf{x} + \delta\mathbf{x}$ est la solution du système perturbé, on a

$$A(\mathbf{x} + \delta\mathbf{x}) = \mathbf{b} + \delta\mathbf{b}.$$

En soustrayant le système initial (2.1) de ce dernier on trouve

$$A\delta\mathbf{x} = \delta\mathbf{b}$$

et donc, en utilisant les normes matricielles (cf. Annexe 2.A.1),

$$\|\delta\mathbf{x}\| = \|A^{-1}\delta\mathbf{b}\| \leq \|A^{-1}\| \|\delta\mathbf{b}\|.$$

Le conditionnement est alors borné comme suit

$$\kappa = \lim_{\epsilon \rightarrow 0} \sup_{\|\delta\mathbf{b}\| < \epsilon \|\mathbf{b}\|} \frac{\|\delta\mathbf{x}\|/\|\mathbf{x}\|}{\|\delta\mathbf{b}\|/\|\mathbf{b}\|} \leq \frac{\|A^{-1}\| \|\mathbf{b}\|}{\|\mathbf{x}\|} = \frac{\|A^{-1}\| \|A\mathbf{x}\|}{\|\mathbf{x}\|} \leq \|A^{-1}\| \|A\|.$$

CAS 2. On étudie à présent les effets des perturbations δA de A . Si $\mathbf{x} + \delta\mathbf{x}$ est la solution du système perturbé, on a

$$(A + \delta A)(\mathbf{x} + \delta\mathbf{x}) = \mathbf{b}.$$

En soustrayant le système initial (2.1), on a

$$A\delta\mathbf{x} + \delta A \mathbf{x} + \underbrace{\delta A \delta\mathbf{x}}_{\text{second ordre}} = 0,$$

où le terme de second ordre est négligé car on considère des perturbations qui tendent vers zéro. Cela implique donc que

$$\|\delta \mathbf{x}\| = \|A^{-1} \delta A \mathbf{x}\| \leq \|A^{-1}\| \|\delta A\| \|\mathbf{x}\|$$

et le conditionnement est de nouveau borné par

$$\kappa = \lim_{\epsilon \rightarrow 0} \sup_{\|\delta A\| < \epsilon \|A\|} \frac{\|\delta \mathbf{x}\| / \|\mathbf{x}\|}{\|\delta A\| / \|A\|} \leq \|A^{-1}\| \|A\|.$$

On conclut dans les deux cas que les erreurs dans les données A , \mathbf{b} d'un système sont amplifiées par (au plus)

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

cette quantité étant appelée le conditionnement de la matrice A . Il importe néanmoins de ne pas confondre, dans les deux cas considérés plus haut, le conditionnement du système linéaire κ et sa borne supérieure donnée par le conditionnement matriciel $\kappa(A)$.

Notons qu'avec Octave le conditionnement matriciel $\kappa(A)$ peut être déterminé via l'instruction `cond`, ou encore en évaluant les normes matricielles de A et de A^{-1} à l'aide des instructions `norm` (pour évaluation des normes, y compris matricielles) et `inv` (pour inversion, y compris matricielle).

EXEMPLE 16. La matrice de Hilbert

$$A = \left(\frac{1}{i+j-1} \right)_{ij}$$

est connue pour avoir un conditionnement matriciel qui croît exponentiellement avec la dimension (cf. la Figure 2.1, courbe supérieure). Cela implique que même si une méthode stable est utilisée pour la résolution du système correspondant, la perte de précision sera vraisemblablement proportionnelle à $\kappa(A)$, c'est-à-dire exponentielle. Ceci est illustré sur la Figure 2.1 pour le solveur `\` d'Octave. Ce solveur est stable, et donc les membres de gauche et de droite de l'inégalité (1.14) qui sont représentés sur la figure sont en effet comparables.

Le programme qui a servi à générer la figure est à gauche de celle-ci. Notons en particulier que le vecteur \mathbf{x} est choisi constant avec ses composantes toutes égales à 1, et le vecteur des seconds membres \mathbf{b} est obtenu à partir de ce vecteur via $\mathbf{b} = A\mathbf{x}$. Comme le produit $A\mathbf{x}$ revient à sommer les éléments de A , et que ces derniers ont tous le même signe, l'erreur relative sur le vecteur \mathbf{b} obtenu de cette manière est d'au plus nu .

```

function hilb_cond
N = 10;
for n = 1:N
    % matrice de Hilbert
    A = hilb(n+1);
    % estimer le
    % conditionnement
    c(n) = cond(A);
    % vraie solution
    x0 = ones(n+1, 1);
    % a_{ij} > 0, x0_{i} > 0
    % => l'erreur relative
    % sur b ~ unit  
    % d'arrondi (cf. TP2)
    b = A*x0;
    x = A\b; % r  soudre
    r(n) = norm(x-x0)/norm(x0);
end
semilogy(2:N+1, c*eps/2, 'r')
hold on
semilogy(2:N+1, r, 'b')

```

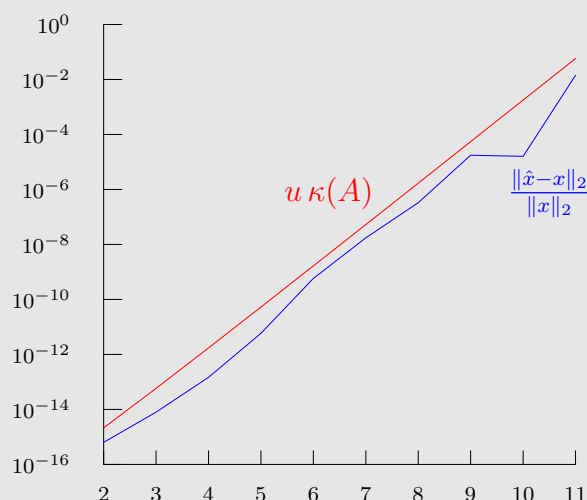


FIGURE 2.1 – Pr  cision relative sur la solution obtenue avec «\» (courbe inf  rieure) et une estimation de celle-ci pour un algorithme stable (courbe sup  rieure).

2.3 M  thodes directes

Les m  thodes de r  solution des syst  mes lin  aires sont commun  ment r  parties en deux cat  gories : m  thodes directes et m  thodes it  ratives. Une m  thode de r  solution est *directe* si en arithm  tique exacte elle fournit la solution apr  s un nombre fini (et connu d'avance) d'op  rations. Dans un chapitre ult  rieur on aborde aussi les m  thodes it  ratives qui g  n  rent une suite de solutions approch  es (et g  n  ralement diff  rentes de la solution exacte).

2.3.1 «Mauvaises» m  thodes directes

La m  thode de Cramer est une des m  thodes pr  f  r  es de r  solution    la main des petits syst  mes lin  aires. Elle permet de calculer l'inconnue x_i en   valuant un quotient de d  terminants

$$x_i = \det(A_i) / \det(A),$$

o   A_i est obtenue en rempla  ant la i -  me colonne de A par le second membre \mathbf{b} . N  anmoins, le co  t d'  valuation d'un d  terminant rend cette m  thode trop co  teuse en pratique. A titre d'exemple, le co  t d'  valuation d'un d  terminant via ses mineurs (le d  veloppement en mineurs est rappel   en Annexe 2.A.2) correspond au co  t d'  valuation

de n déterminants de taille $n - 1$, n multiplications, et $n - 1$ additions et soustractions. Même en négligeant ces deux dernières contributions et en estimant récursivement le coût des déterminants de taille $n - 1$ on constate que le coût d'un déterminant de taille n est au moins $n(n - 1) \cdots 3$ fois le coût d'un déterminant de taille 2, soit au total un coût $\mathcal{O}(n!)$.

Une autre méthode inefficace consiste à calculer le vecteur \mathbf{x} via $\mathbf{x} = A^{-1}\mathbf{b}$, ce qui nécessite le calcul de l'inverse de la matrice A . Comme les détails du coût de l'inversion ne sont vus qu'à la section 2.4, on se limite ici à une comparaison Octave de cette méthode avec \backslash .

EXEMPLE 17. On évalue le temps nécessaire pour

1. résoudre un système $A\mathbf{x} = \mathbf{b}$ via \backslash ;
2. calculer l'inverse d'une matrice A et la multiplier par un vecteur \mathbf{b} .

Le code correspondant est donné à gauche et les temps ainsi obtenus à droite.

```
A = rand(1000);
b = rand(1000,1);
tic; x1 = A\b; toc
tic; x1 = inv(A)*b; toc
```

Elapsed time is 0.1557 seconds.

Elapsed time is 0.3586 seconds.

l'inversion matricielle est 2.3 fois plus lente !

2.3.2 Systèmes triangulaires

On commence la discussion sur les méthodes directes en considérant les systèmes triangulaires. La résolution de tels systèmes est à la fois simple et importante pour les algorithmes plus généraux.

Systèmes triangulaires inférieurs

Pour résoudre

$$\begin{cases} a_{11}x_1 & = b_1 \\ a_{21}x_1 + a_{22}x_2 & = b_2 \\ & \dots \\ a_{i1}x_1 + a_{i2}x_2 + \dots + a_{ii}x_i & = b_i \\ & \dots \end{cases}$$

on détermine la première inconnue x_1 à partir de la première équation, ensuite on utilise x_1 pour déterminer x_2 dans l'équation suivante, et ainsi de suite. La solution finale est donnée par

$$\begin{cases} x_1 & = b_1/a_{11} \\ x_2 & = (b_2 - a_{21}x_1)/a_{22} \\ & \dots \\ x_i & = (b_i - \sum_{j=1}^{i-1} a_{ij}x_j)/a_{ii} \\ & \dots \end{cases}$$

Systèmes triangulaires supérieurs

La résolution d'un système triangulaire supérieur suit les mêmes principes. Pour résoudre

$$\begin{cases} \dots \\ a_{ii}x_i + \dots + a_{in-1}x_{n-1} + a_{in}x_n = b_i \\ \dots \\ a_{n-1n-1}x_{n-1} + a_{n-1n}x_n = b_{n-1} \\ a_{nn}x_n = b_n \end{cases}$$

on détermine l'inconnue x_n à partir de la dernière équation, ensuite on utilise x_n pour déterminer x_{n-1} dans l'équation précédente, et ainsi de suite. La solution finale est donnée par

$$\begin{cases} x_n = b_n/a_{nn} \\ x_{n-1} = (b_{n-1} - a_{n-1n}x_n)/a_{n-1n-1} \\ \dots \\ x_i = (b_i - \sum_{j=i+1}^n a_{ij}x_j)/a_{ii} \\ \dots \end{cases}$$

Coût

Considérons le cas du système triangulaire inférieur, celui du système triangulaire supérieur est identique. Le nombre d'opérations avec des réels (ou encore le nombre de *flops*, de l'anglais *floating point operations*) qui sont effectuées en manipulant la i -ème équation est de 1 division, $i - 1$ soustractions et $i - 1$ multiplications, soit $2i - 1$ flops. En sommant sur toutes les équations, on a donc

$$\sum_{i=1}^n (2i - 1) = n^2 \text{ flops.}$$

Dans le cas où les éléments diagonaux valent 1, les divisions ne sont pas nécessaires et le nombre d'opérations peut être réduit à $n(n - 1)$ flops.

Notons qu'il est impossible d'avoir un algorithme avec sensiblement moins d'opérations car le nombre d'éléments potentiellement différents d'une matrice triangulaire est de $n(n + 1)/2 = \mathcal{O}(n^2)$ et, comme la solution dépend de chacun de ces éléments, le nombre d'opérations doit aussi être d'au moins $\mathcal{O}(n^2)$.

Existence et stabilité

Notons pour commencer qu'une matrice triangulaire est régulière si et seulement si tous ses éléments diagonaux sont non nuls. En effet, comme rappelé en Annexe 2.A.2, le déterminant d'une matrice triangulaire $A = (a_{ij})$ est donné par $\det(A) = a_{11}a_{22} \cdots a_{nn}$, et, par ailleurs, on a $\det(A) \neq 0$ si et seulement si la matrice A est régulière.

Par ailleurs, l'algorithme fournit une et une seule solution pour tout système régulier. En particulier, le seul obstacle potentiel à sa bonne exécution – une division par zéro – n'est pas rencontré, car on divise toujours par un élément diagonal de A , qui est non nul.

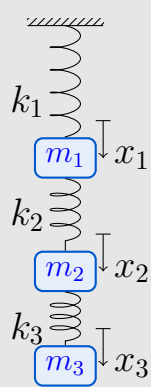
Concernant la stabilité, on observera sans le démontrer que l'algorithme a la stabilité inverse. Plus précisément, si $\tilde{\mathbf{x}}$ est la solution calculée, il existe une matrice δA triangulaire avec $|(\delta A)_{ij}| \leq n \cdot u \cdot |(A)_{ij}|$ telle que

$$(\delta A + A)\tilde{\mathbf{x}} = \mathbf{b}.$$

2.3.3 Méthode d'élimination de Gauss

On rappelle le principe de l'algorithme d'élimination de Gauss à l'aide de l'exemple suivant, où il est appliqué à un système de 3 équations à 3 inconnus. Dans ce cas, l'algorithme procède en deux étapes. A l'étape k , $k = 1, 2$, il consiste à soustraire des multiples de l'équation k aux équations suivantes de sorte à annuler le coefficient devant x_k . Le système après chaque étape est donné dans l'exemple. Suite à ces éliminations il devient triangulaire et est résolu comme expliqué dans la sous-section précédente.

EXEMPLE 18.



$$\Rightarrow \left\{ \begin{array}{l|l} 2x_1 & -x_2 & = & 1 \\ -x_1 & +2x_2 & -x_3 & = & 1 \\ & -x_2 & +x_3 & = & 1 \end{array} \right. \begin{array}{l} \text{(ligne du pivot)} \\ +\frac{1}{2} \text{ (ligne 1)} \\ +0 \text{ (ligne 1)} \end{array}$$

$$\Rightarrow \left\{ \begin{array}{l|l} 2x_1 & -x_2 & = & 1 \\ & \frac{3}{2}x_2 & -x_3 & = & \frac{3}{2} \\ & -x_2 & +x_3 & = & 1 \end{array} \right. \begin{array}{l} \text{(ligne du pivot)} \\ +\frac{2}{3} \text{ (ligne 2)} \end{array}$$

$$\Rightarrow \left\{ \begin{array}{l|l} 2x_1 & -x_2 & = & 1 \\ & \frac{3}{2}x_2 & -x_3 & = & \frac{3}{2} \\ & & \frac{1}{3}x_3 & = & 2 \end{array} \right.$$

\rightarrow résolution de ce système triangulaire

Notons que la même procédure peut également être effectuée dans le cadre matriciel, comme illustrée dans l'encadré suivant. Au lieu de manipuler les équations, on manipule alors les lignes de la matrice et du second membre.

EXEMPLE 18. (SUITE)

VERSION CLASSIQUE

$$\left\{ \begin{array}{l|l} 2x_1 & -x_2 & = & 1 \\ -x_1 & +2x_2 & -x_3 & = & 1 \\ & -x_2 & +x_3 & = & 1 \end{array} \right.$$

$$\Rightarrow \left\{ \begin{array}{l|l} 2x_1 & -x_2 & = & 1 \\ & \frac{3}{2}x_2 & -x_3 & = & \frac{3}{2} \\ & -x_2 & +x_3 & = & 1 \end{array} \right.$$

$$\Rightarrow \left\{ \begin{array}{l|l} 2x_1 & -x_2 & = & 1 \\ & \frac{3}{2}x_2 & -x_3 & = & \frac{3}{2} \\ & & \frac{1}{3}x_3 & = & 2 \end{array} \right.$$

VERSION MATRICIELLE

$$\left(\begin{array}{c|cc} 2 & -1 & \\ -1 & 2 & -1 \\ & -1 & 1 \end{array} \right) \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

$$\Rightarrow \left(\begin{array}{c|cc} 2 & -1 & \\ & \frac{3}{2} & -1 \\ & -1 & 1 \end{array} \right) \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 1 \\ \frac{3}{2} \\ 1 \end{pmatrix}$$

$$\Rightarrow \left(\begin{array}{c|cc} 2 & -1 & \\ & \frac{3}{2} & -1 \\ & & \frac{1}{3} \end{array} \right) \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 1 \\ \frac{3}{2} \\ 2 \end{pmatrix}$$

Revenons à présent au cas général. L'algorithme procède en $n - 1$ étapes. L'étape k consiste à soustraire des multiples de la ligne k aux lignes $k + 1, \dots, n$ de sorte à annuler leurs éléments dans la colonne k ; la soustraction correspondante doit aussi être effectuée dans le second membre. Étape par étape, cela donne

$$\begin{aligned}
 & \left(\begin{array}{c|ccc} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1n}^{(1)} \\ a_{21}^{(1)} & a_{22}^{(1)} & \cdots & a_{2n}^{(1)} \\ \vdots & \vdots & & \vdots \\ a_{n1}^{(1)} & a_{n2}^{(1)} & \cdots & a_{nn}^{(1)} \end{array} \right) \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1^{(1)} \\ b_2^{(1)} \\ \vdots \\ b_n^{(1)} \end{pmatrix} \begin{matrix} (ligne \text{ du pivot}) \\ -a_{21}^{(1)}/a_{11}^{(1)} \text{ (ligne 1)} \\ \vdots \\ -a_{n1}^{(1)}/a_{11}^{(1)} \text{ (ligne 1)} \end{matrix} \\
 \Rightarrow & \left(\begin{array}{c|ccc} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1n}^{(1)} \\ & a_{22}^{(2)} & \cdots & a_{2n}^{(2)} \\ & \vdots & & \vdots \\ & a_{n2}^{(2)} & \cdots & a_{nn}^{(2)} \end{array} \right) \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1^{(1)} \\ b_2^{(2)} \\ \vdots \\ b_n^{(2)} \end{pmatrix} \begin{matrix} (ligne \text{ du pivot}) \\ \vdots \\ -a_{n2}^{(2)}/a_{22}^{(2)} \text{ (ligne 2)} \end{matrix} \\
 \Rightarrow & \dots \\
 \Rightarrow & \left(\begin{array}{c|ccc} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1n}^{(1)} \\ & a_{22}^{(2)} & \cdots & a_{2n}^{(2)} \\ & & \ddots & \vdots \\ & & & a_{nn}^{(n)} \end{array} \right) \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1^{(1)} \\ b_2^{(2)} \\ \vdots \\ b_n^{(n)} \end{pmatrix} \\
 \Rightarrow & \text{résolution de ce système triangulaire}
 \end{aligned}$$

Et l'algorithme correspondant est comme suit. Notons que la première opération dans la boucle consiste à déterminer les facteurs qui pondèrent l'équation k lorsque celle-ci est soustraite des équations suivantes ; la deuxième opération effectuée la soustraction de la ligne k de la matrice aux lignes suivantes ; la troisième opération réalise la même soustraction pour les seconds membres.

ALGORITHME (ÉLIMINATION DE GAUSS)

entrées : A, \mathbf{b}

sortie : \mathbf{x}

pour $k = 1, \dots, n - 1$

$$\ell_{ik} = a_{ik}^{(k)} / a_{kk}^{(k)}, \quad i = k + 1, \dots, n$$

$$a_{ij}^{(k+1)} = a_{ij}^{(k)} - \ell_{ik} \cdot a_{kj}^{(k)}, \quad i, j = k + 1, \dots, n$$

$$b_i^{(k+1)} = b_i^{(k)} - \ell_{ik} \cdot b_k^{(k)}, \quad i = k + 1, \dots, n$$

résoudre le système triangulaire résultant

Notons pour la petite histoire que, bien que les valeurs des facteurs de pondération ℓ_{ik} à l'étape k de l'algorithme d'élimination de Gauss sont choisis précisément pour annuler les éléments $a_{ik}^{(k+1)}$, $i = k + 1, \dots, n$, ces éléments ne sont pas annulés explicitement dans l'algorithme. Par conséquent, la matrice à la fin de l'exécution de l'algorithme n'est pas formellement triangulaire supérieure. Néanmoins, la partie triangulaire supérieure de cette matrice est bien la matrice triangulaire supérieure qui résulte de l'élimination de Gauss.

Coût

A l'étape k de l'algorithme, la contribution des lignes 1 à 3 de la boucle sont de $n - k$, $2(n - k)^2$ et $2(n - k)$ flops respectivement. Clairement, la contribution de la deuxième ligne est dominante et, en sommant sur toutes les étapes, le coût total vaut

$$\sum_{k=1}^{n-1} 2(n - k)^2 + \mathcal{O}(n^2) = 2 \underbrace{((n - 1)^2 + \dots + 2^2 + 1^2)}_{\sum_{i=0}^{n-1} i^2} + \mathcal{O}(n^2) = \frac{2n^3}{3} + \mathcal{O}(n^2) \text{ flops.}$$

Une astuce qui permet d'obtenir une approximation pour la somme des carrés $\sum_{i=0}^{n-1} i^2$ est de l'approcher par $\int_0^{n-1} x^2 dx = (n - 1)^3/3 = n^3/3 + \mathcal{O}(n^2)$; la formule exacte ainsi que la justification de cette astuce sont fournies en Annexe 2.B.

2.3.4 Factorisation LU

La factorisation LU d'une matrice régulière A est un couple de matrices L et U telles que L est une matrice triangulaire inférieure dont les éléments diagonaux valent 1, U est une matrice triangulaire supérieure et

$$A = LU.$$

L'algorithme de calcul d'une factorisation LU présenté ici est une modification de la méthode d'élimination de Gauss, et plus précisément de sa partie qui opère sur la matrice du système. Cet algorithme résulte d'une interprétation matricielle des étapes d'élimination. Comme explicité plus loin, l'avantage principal de la factorisation LU dans la résolution des systèmes linéaires est le fait que la matrice du système peut être factorisée une seule fois, quel que soit le nombre de membres de droite considérés.

EXEMPLE 18. (FIN) Reprenons le système de l'Exemple 18 et effectuons l'élimination de Gauss en remplaçant la soustraction des lignes par des transformations matricielles. A l'étape k , $k = 1, 2$, la transformation consiste à multiplier les deux membres du système par une matrice qui a 1 sur la diagonale et 0 ailleurs, sauf pour la k -ème colonne sous la diagonale dont les éléments sont les facteurs $-a_{ik}^{(k)}/a_{kk}^{(k)}$ qui multiplient la ligne k en élimination de Gauss pour la soustraire aux autres. Notons par ailleurs que le calcul de

la factorisation LU nécessite d'appliquer ces transformations uniquement à la matrice A du système.

Pour la première étape, cela donne

$$\begin{aligned}
 & \underbrace{\begin{pmatrix} 2 & -1 & \\ -1 & 2 & -1 \\ & -1 & 1 \end{pmatrix}}_A \underbrace{\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}}_{\mathbf{b}} = \underbrace{\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}}_{\mathbf{b}} \\
 & \Rightarrow \underbrace{\begin{pmatrix} 1 & & \\ \frac{1}{2} & 1 & \\ 0 & & 1 \end{pmatrix}}_{L_1} \underbrace{\begin{pmatrix} 2 & -1 & \\ -1 & 2 & -1 \\ & -1 & 1 \end{pmatrix}}_A \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & & \\ \frac{1}{2} & 1 & \\ 0 & & 1 \end{pmatrix}}_{L_1} \underbrace{\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}}_{\mathbf{b}} \\
 & \Rightarrow \underbrace{\begin{pmatrix} 2 & -1 & \\ & \frac{3}{2} & -1 \\ & -1 & 1 \end{pmatrix}}_{L_1 A} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \underbrace{\begin{pmatrix} 1 \\ \frac{3}{2} \\ 1 \end{pmatrix}}_{L_1 \mathbf{b}}
 \end{aligned}$$

et pour la seconde étape

$$\begin{aligned}
 & \underbrace{\begin{pmatrix} 1 & & \\ & 1 & \\ & \frac{2}{3} & 1 \end{pmatrix}}_{L_2} \underbrace{\begin{pmatrix} 2 & -1 & \\ & \frac{3}{2} & -1 \\ & -1 & 1 \end{pmatrix}}_{L_1 A} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & & \\ & 1 & \\ & \frac{2}{3} & 1 \end{pmatrix}}_{L_2} \underbrace{\begin{pmatrix} 1 \\ \frac{3}{2} \\ 1 \end{pmatrix}}_{L_1 \mathbf{b}} \\
 & \Rightarrow \underbrace{\begin{pmatrix} 2 & -1 & \\ & \frac{3}{2} & -1 \\ & & \frac{1}{3} \end{pmatrix}}_{L_2 L_1 A} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \underbrace{\begin{pmatrix} 1 \\ \frac{3}{2} \\ 2 \end{pmatrix}}_{L_2 L_1 \mathbf{b}} \Rightarrow \text{résolution de ce système triangulaire}
 \end{aligned}$$

On constate que l'application des transformations successives mène, tout comme l'élimination de Gauss auparavant, à un système

$$U\mathbf{x} = L_2 L_1 \mathbf{b}$$

dont la matrice $U = L_2 L_1 A$ est triangulaire supérieure par construction. La matrice A initiale peut quant à elle s'écrire comme

$$A = L_1^{-1} L_2^{-1} U.$$

Sous cette forme la factorisation LU est à moitié achevée; il reste à montrer que $L = L_1^{-1} L_2^{-1}$ est, en effet, une matrice triangulaire inférieure dont les éléments diagonaux valent 1.

Le travail est donc, étant donné

$$L_1 = \begin{pmatrix} 1 & & \\ \frac{1}{2} & 1 & \\ 0 & & 1 \end{pmatrix}, \quad L_2 = \begin{pmatrix} 1 & & \\ & 1 & \\ & \frac{2}{3} & 1 \end{pmatrix},$$

de déterminer L_1^{-1} , L_2^{-1} et leur produit $L = L_1^{-1}L_2^{-1}$. Pour cela, les propriétés suivantes sont utiles.

PROPRIÉTÉ 1. Soient I_1 , I_2 deux matrices identités et C une matrice rectangulaire de dimensions compatibles. On a alors

$$\begin{pmatrix} I_1 & \\ C & I_2 \end{pmatrix}^{-1} = \begin{pmatrix} I_1 & \\ -C & I_2 \end{pmatrix}.$$

DÉMONSTRATION. Dans la multiplication matricielle, les blocs peuvent être traités comme des éléments matriciels si leurs dimensions sont compatibles. Cela implique en particulier

$$\begin{pmatrix} I_1 & \\ C & I_2 \end{pmatrix} \begin{pmatrix} I_1 & \\ -C & I_2 \end{pmatrix} = \begin{pmatrix} I_1 & \\ & I_2 \end{pmatrix}. \quad \blacksquare$$

PROPRIÉTÉ 2. Soient I_1 , I_2 deux matrices identités, C une matrice carrée et \mathbf{v} un vecteur, tous de dimensions compatibles. On a alors

$$\begin{pmatrix} 1 & \\ \mathbf{v} & I_2 \end{pmatrix} \begin{pmatrix} 1 & \\ & C \end{pmatrix} = \begin{pmatrix} 1 & \\ \mathbf{v} & C \end{pmatrix}$$

et

$$\begin{pmatrix} I_1 & & \\ & 1 & \\ & \mathbf{v} & I_2 \end{pmatrix} \begin{pmatrix} I_1 & & \\ & 1 & \\ & & C \end{pmatrix} = \begin{pmatrix} I_1 & & \\ & 1 & \\ & \mathbf{v} & C \end{pmatrix}.$$

DÉMONSTRATION. Comme pour la Propriété 1, il suffit d'effectuer le produit bloc par bloc. \blacksquare

Pour les matrices de l'exemple, on a donc

$$L_1^{-1} = \begin{pmatrix} 1 & & \\ -\frac{1}{2} & 1 & \\ 0 & & 1 \end{pmatrix}, \quad L_2^{-1} = \begin{pmatrix} 1 & & \\ & 1 & \\ & -\frac{2}{3} & 1 \end{pmatrix}$$

et

$$L = L_1^{-1}L_2^{-1} = \begin{pmatrix} 1 & & \\ -\frac{1}{2} & 1 & \\ 0 & -\frac{2}{3} & 1 \end{pmatrix},$$

ce qui confirme bien que $L_1^{-1}L_2^{-1}$ est une matrice triangulaire inférieure dont les éléments diagonaux valent 1. La factorisation LU est donc dans notre cas

$$\underbrace{\begin{pmatrix} 1 & & \\ -\frac{1}{2} & 1 & \\ 0 & -\frac{2}{3} & 1 \end{pmatrix}}_L \underbrace{\begin{pmatrix} 2 & -1 & \\ & \frac{3}{2} & -1 \\ & & \frac{1}{3} \end{pmatrix}}_U = \underbrace{\begin{pmatrix} 2 & -1 & \\ -1 & 2 & -1 \\ & -1 & 1 \end{pmatrix}}_A.$$

De manière générale on procède par transformations successives

$$\underbrace{\begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1n}^{(1)} \\ a_{21}^{(1)} & a_{22}^{(1)} & \cdots & a_{2n}^{(1)} \\ \vdots & \vdots & & \vdots \\ a_{n1}^{(1)} & a_{n2}^{(1)} & \cdots & a_{nn}^{(1)} \end{pmatrix}}_A \Rightarrow \underbrace{\begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1n}^{(1)} \\ & a_{22}^{(2)} & \cdots & a_{2n}^{(2)} \\ & \vdots & & \vdots \\ & a_{n2}^{(2)} & \cdots & a_{nn}^{(2)} \end{pmatrix}}_{L_1 A} \cdots \Rightarrow \cdots \underbrace{\begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1n}^{(1)} \\ & a_{22}^{(2)} & \cdots & a_{2n}^{(2)} \\ & & \ddots & \vdots \\ & & & a_{nn}^{(n)} \end{pmatrix}}_{U = L_{n-1} \cdots L_1 A}$$

avec les matrices de transformation

$$L_1 = \begin{pmatrix} 1 & & & \\ -a_{21}^{(1)}/a_{11}^{(1)} & 1 & & \\ \vdots & & \ddots & \\ -a_{n1}^{(1)}/a_{11}^{(1)} & & & 1 \end{pmatrix}, \quad L_2 = \begin{pmatrix} 1 & & & \\ & 1 & & \\ & -a_{32}^{(2)}/a_{22}^{(2)} & 1 & \\ & \vdots & & \ddots \\ & -a_{n2}^{(2)}/a_{22}^{(2)} & & 1 \end{pmatrix}, \quad \cdots$$

En utilisant les Propriétés 1 et 2 comme dans l'exemple précédent on obtient une factorisation

$$A = LU$$

avec U triangulaire supérieure et

$$L = L_1^{-1} \cdots L_{n-1}^{-1} = \begin{pmatrix} 1 & & & \\ a_{21}^{(1)}/a_{11}^{(1)} & 1 & & \\ \vdots & \vdots & \ddots & \\ a_{n1}^{(1)}/a_{11}^{(1)} & a_{n2}^{(2)}/a_{22}^{(2)} & \cdots & 1 \end{pmatrix}.$$

L'algorithme correspondant est alors comme suit. Notons que cet algorithme est identique à l'élimination de Gauss si ce n'est que la matrice triangulaire supérieure qui en résulte est maintenant sauvegardée dans U alors que les facteurs ℓ_{ik} sont conservés dans L . Par ailleurs, les opérations avec le second membre ne sont plus effectuées étant donné qu'on calcule une factorisation et non pas la solution d'un système linéaire. Finalement, notons que comme les éléments dans la partie supérieure de L et inférieure de U ne sont pas manipulés, ils doivent être soit initialisés à 0, soit ignorés.

ALGORITHME (FACTORISATION LU)

entrée : A

sorties : $L = (\ell_{ij})$, $U = (u_{ji})$ (avec $\ell_{ij} = u_{ji} = 0$ pour $j > i$)

pour $k = 1, \dots, n$

$$u_{kj} = a_{kj}^{(k)}, \quad j = k, \dots, n$$

$$\ell_{kk} = 1 \text{ et } \ell_{ik} = a_{ik}^{(k)} / a_{kk}^{(k)}, \quad i = k+1, \dots, n$$

$$a_{ij}^{(k+1)} = a_{ij}^{(k)} - \ell_{ik} \cdot a_{kj}^{(k)}, \quad i, j = k+1, \dots, n$$

Une fois qu'une factorisation LU est obtenue, la résolution du système linéaire

$$A\mathbf{x} = L \underbrace{U\mathbf{x}}_{\mathbf{y}} = \mathbf{b}, \quad (2.2)$$

se fait en deux étapes

1. résolution du système triangulaire $L\mathbf{y} = \mathbf{b}$;
2. résolution du système triangulaire $U\mathbf{x} = \mathbf{y}$.

Coût

Comme déjà noté, l'algorithme de factorisation LU est équivalent à celui de l'élimination de Gauss (excepté pour les opérations avec le second membre) ; son coût est donc aussi de

$$\frac{2}{3}n^3 + \mathcal{O}(n^2) \text{ flops.}$$

Par ailleurs, le coût de la résolution des deux systèmes triangulaires est de $2n^2$.

Ces deux éléments montrent l'utilité principale d'une factorisation LU : la séparation entre les opérations avec la matrice (factorisation) et avec le second membre (résolutions triangulaires). En particulier, lorsque *plusieurs* seconds membres sont utilisés avec une seule matrice il suffit d'effectuer la factorisation (coût $\approx 2/3n^3$) une seule fois et de répéter la résolution des deux systèmes triangulaires *pour chaque* second membre (coût $\approx 2n^2$ par second membre).

Existence et unicité

Si une factorisation LU existe, elle est nécessairement unique. Néanmoins, une factorisation LU **n'existe pas** nécessairement, même pour une matrice régulière. Par exemple,

la matrice

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

est régulière (car son déterminant vaut -1) mais n'a pas de factorisation LU (pourquoi ?).

2.3.5 Stabilité et pivotage

L'algorithme de factorisation LU, ainsi que la résolution des systèmes linéaires via cette factorisation LU, ne sont pas nécessairement des algorithmes stables, comme le montre l'exemple suivant.

EXEMPLE 19. Soit le système linéaire

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

dont la matrice est bien conditionnée : $\kappa(A) = 2.6180$. La factorisation LU obtenue numériquement est comme suit

$$L = \begin{pmatrix} 1 & 0 \\ 10^{20} & 1 \end{pmatrix}, \quad U = \begin{pmatrix} 10^{-20} & 1 \\ 0 & \mathbf{1} \ominus \mathbf{10}^{20} \end{pmatrix} = \begin{pmatrix} 10^{-20} & 1 \\ 0 & -\mathbf{10}^{20} \end{pmatrix}$$

(car $\mathbf{1} \ominus \mathbf{10}^{20} = -\mathbf{10}^{20}$ en double précision) et donc

$$LU = \begin{pmatrix} 10^{-20} & 1 \\ 1 & \mathbf{0} \end{pmatrix}.$$

On constate que l'erreur relative $\|LU - A\|_2 / \|A\|_2 \approx 0.618$ dépasse largement $\kappa(A)u$. De même, la solution obtenue par $\mathbf{U} \setminus (\mathbf{L} \setminus [1; 2])$ est $\begin{pmatrix} 2 & 1 \end{pmatrix}$, alors que la vraie solution (à $\mathcal{O}(10^{-20})$ près) est $\begin{pmatrix} 1 & 1 \end{pmatrix}$; l'erreur relative sur la solution en norme euclidienne est ici d'environ $1/\sqrt{2} \approx 0.707$ et de nouveau elle dépasse largement $\kappa(A)u$.

Notons que le problème de stabilité disparaît si on permute la première et la deuxième ligne entre elles (pivotage des lignes); en effet, l'algorithme de la factorisation LU appliqué à

$$PA = \begin{pmatrix} 1 & 1 \\ 10^{-20} & 1 \end{pmatrix} \quad \text{avec la matrice de permutation } P = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

est *stable*. De plus, la factorisation calculée permet de résoudre le système initial avec la première et la seconde équation permutées :

$$\underbrace{\begin{pmatrix} 1 & 1 \\ 10^{-20} & 1 \end{pmatrix}}_{PA} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \underbrace{\begin{pmatrix} 2 \\ 1 \end{pmatrix}}_{P\mathbf{b}}.$$

L'approche qui a permis de rendre l'algorithme de factorisation stable dans l'exemple précédent est appelé *pivotage des lignes*. Cette technique est aussi utilisée pour stabiliser

l'algorithme de factorisation dans le cas général. On considère dans ce qui suit une version particulière de pivotage.

PIVOTAGE PARTIEL DES LIGNES : on choisit d'éliminer (c'est-à-dire de choisir comme pivot) la ligne non éliminée qui possède l'élément *maximal en valeur absolue* dans la colonne du pivot.

Insistons sur le fait que les lignes déjà éliminées ne peuvent plus être choisies comme ligne de pivot ; en termes d'élimination de Gauss cela signifie qu'il n'est pas adéquat d'éliminer une même équation deux fois.

L'exemple suivant permet de voir comment le pivotage des lignes peut être intégré dans l'algorithme de factorisation LU. La factorisation résultante est de la forme

$$PA = LU$$

avec P une matrice de permutation, L une matrice triangulaire inférieure dont les éléments diagonaux valent 1 et U une matrice triangulaire supérieure. Les propriétés des matrices de permutation élémentaires utilisées plus bas sont rappelées en Annexe 2.A.3.

EXEMPLE 20. On calcule ici la factorisation LU avec pivotage partiel des lignes pour la matrice

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

La démarche suivie est similaire à celle de la factorisation sans pivotage, mais, avant d'éliminer la ligne k , on permute (si nécessaire) la ligne de pivot – la ligne non éliminée qui possède l'élément maximal en valeur absolue dans la colonne du pivot – et la ligne k .

En particulier, comme l'élément le plus grand en valeur absolue de la colonne 1 se trouve déjà à la première ligne, on ne permute pas cette ligne ; cela équivaut à appliquer à gauche une transformation P_1 donnée par une matrice identité. L'élimination de la première colonne donne alors

$$L_1 P_1 A = \begin{pmatrix} 1 & 1 & \\ \boxed{0} & 1 & \\ 1 & 2 & \end{pmatrix} \quad \text{avec } L_1 = \begin{pmatrix} 1 & & \\ -1 & 1 & \\ 0 & & 1 \end{pmatrix}.$$

L'élément le plus grand en valeur absolue de la colonne 2 (excepté la ligne 1, déjà éliminée) est cette fois à ligne 3, il faut donc permuter les lignes 2 et 3.

$$P_2 L_1 P_1 A = \begin{pmatrix} 1 & 1 & \\ & 1 & 2 \\ & \boxed{0} & 1 \end{pmatrix} \quad \text{avec } P_2 = \begin{pmatrix} 1 & & \\ & & 1 \\ & 1 & \end{pmatrix}$$

Finalement, la dernière étape de l'élimination n'est pas nécessaire, car la matrice résultante est déjà triangulaire (L_2 est l'identité); la factorisation est donc donnée par

$$L_2 P_2 L_1 P_1 A = U.$$

Comme on a maintenant un produit alterné de matrices L_i et P_i , il n'est pas a priori clair comment on peut récupérer une matrice triangulaire inférieure, comme ce fut le cas en absence de pivotage. Cela est néanmoins possible grâce à la propriété suivante.

PROPRIÉTÉ 3. Soit P_i la matrice de permutation élémentaire pour les indices i et $j > i$, et soit

$$L_k = \begin{pmatrix} I_1 & & \\ & 1 & \\ & \mathbf{v} & I_2 \end{pmatrix}$$

avec I_1 la matrice identité de dimensions $(k-1) \times (k-1)$ et avec \mathbf{v} , I_2 , respectivement, un vecteur et la matrice identité de dimensions compatibles (lorsque $k = 1$, le résultat reste valable avec I_1 «supprimée»).

Si $k < i$, alors

$$P_i L_k P_i = \begin{pmatrix} I_1 & & \\ & 1 & \\ & \mathbf{w} & I_2 \end{pmatrix}$$

avec le vecteur \mathbf{w} obtenu à partir du vecteur \mathbf{v} en permutant les éléments $i - k$ et $j - k$ de celui-ci.

DÉMONSTRATION.

$$\underbrace{\begin{pmatrix} I_1 & & & & \\ & 1 & & & \\ & \vdots & \ddots & & \\ & \mathbf{v}_{i-k} & & 1 & \cdots & 0 \\ & \vdots & & \vdots & \ddots & \vdots \\ & \mathbf{v}_{j-k} & & 0 & \cdots & 1 \\ & \vdots & & & & \ddots \end{pmatrix}}_{L_k} \rightarrow \underbrace{\begin{pmatrix} I_1 & & & & \\ & 1 & & & \\ & \vdots & \ddots & & \\ & \mathbf{v}_{j-k} & & 0 & \cdots & 1 \\ & \vdots & & \vdots & \ddots & \vdots \\ & \mathbf{v}_{i-k} & & 1 & \cdots & 0 \\ & \vdots & & & & \ddots \end{pmatrix}}_{P_i L_k} \rightarrow \underbrace{\begin{pmatrix} I_1 & & & & \\ & 1 & & & \\ & \vdots & \ddots & & \\ & \mathbf{v}_{j-k} & & 1 & \cdots & 0 \\ & \vdots & & \vdots & \ddots & \vdots \\ & \mathbf{v}_{i-k} & & 0 & \cdots & 1 \\ & \vdots & & & & \ddots \end{pmatrix}}_{P_i L_k P_i}$$

■

Pour l'Exemple 20 on a ainsi

$$P_2 L_1 P_2 = \begin{pmatrix} 1 & & \\ 0 & 1 & \\ -1 & & 1 \end{pmatrix}.$$

L'expression $L_2 P_2 L_1 P_1 A = U$ peut alors être mise sous la forme (en utilisant $P_2 P_2 = I$)

$$L_2 \underbrace{P_2 L_1 P_2}_{L'_1} P_2 P_1 A = U$$

et on constate, à l'aide de la Propriété 3, que la matrice L'_1 a exactement la même forme que la matrice L_1 , et qu'elle joue exactement le même rôle que celui joué par L_1 en absence de pivotage. Ainsi, en définissant $P = P_2 P_1$ et $L = L'^{-1}_1 L^{-1}_2$ (qui est bien triangulaire inférieure et dont les éléments diagonaux valent bien 1), on a

$$PA = LU.$$

La démarche précédente est applicable pour toute matrice de dimensions 3×3 . Pour fixer les idées, effectuons cette démarche pour une matrice de dimensions 4×4 .

EXEMPLE 21. L'algorithme de la factorisation LU avec pivotage partiel des lignes appliqué à une matrice A de dimensions 4×4 se ramène à

$$L_3 P_3 L_2 P_2 L_1 P_1 A = U,$$

où U est une matrice triangulaire supérieure, P_1, \dots, P_3 sont des matrices de permutation élémentaire, et L_1, \dots, L_3 sont des matrices d'élimination des lignes 1, ..., 3. En tenant compte de $P_3 P_3 = I$ et $P_2(P_3 P_3)P_2 = P_2 P_2 = I$, on obtient

$$L_3 \underbrace{P_3 L_2 (P_3 P_3)}_{L'_2} \underbrace{P_2 L_1 (P_2 P_3 P_3 P_2)}_{L'_1} \underbrace{P_1}_{P} A = U,$$

$\underbrace{\hspace{10em}}_{L''_1}$

où $L'_2 = P_3 L_2 P_3$ et $L''_1 = P_3 L'_1 P_3 = P_3 P_2 L_1 P_2 P_3$ ont la même structure (par la Propriété 3) que L_2 et L_1 , respectivement. Dès lors, on a bien

$$PA = LU$$

où

$$L = L''^{-1}_1 L'^{-1}_2 L^{-1}_3$$

est une matrice triangulaire inférieure dont les éléments diagonaux valent 1 (par les Propriétés 1 et 2).

De manière générale, l'algorithme de la factorisation LU avec pivotage partiel des lignes mène à une matrice triangulaire supérieure de la forme

$$L_{n-1} P_{n-1} L_{n-2} \cdots P_2 L_1 P_1 A = U$$

qu'on peut réécrire en utilisant $P_i P_i = I$ comme suit

$$\underbrace{L_{n-1}}_{L'_{n-1}} \underbrace{P_{n-1} L_{n-2} P_{n-1}}_{L'_{n-2}} \cdots \underbrace{P_{n-1} \cdots P_2 L_1 P_2 \cdots P_{n-1}}_{L'_1} \underbrace{P_{n-1} \cdots P_2 P_1}_P A = U. \quad (2.3)$$

Étant donné que les matrices L'_i ont (par la Propriété 3) la même forme que les matrices L_i respectives et qu'elles jouent le même rôle que celui que jouent les matrices L_i respectives en absence de pivotage, le produit de leurs inverses

$$L = L'_1{}^{-1} \cdots L'_{n-1}{}^{-1}$$

est bien (par les Propriétés 1 et 2) une matrice triangulaire inférieure dont les éléments diagonaux valent 1. Par conséquent, (2.3) devient

$$PA = LU,$$

où L est une matrice triangulaire inférieure dont les éléments diagonaux valent 1, U une matrice triangulaire supérieure et P une matrice de permutation.

Coût

L'algorithme de factorisation LU avec pivotage partiel des lignes effectue exactement le même nombre d'opérations (flops) que celui sans pivotage, à savoir

$$\frac{2}{3}n^3 + \mathcal{O}(n^2);$$

il effectue aussi au plus $(n-1)(n-2)/2$ permutations d'éléments matriciels étant donné que la permutation P_i s'applique au plus à L_1, \dots, L_{i-1} , et que chaque application mène au plus à une permutation de deux éléments.

Existence et stabilité

La factorisation LU avec pivotage existe pour toute matrice régulière. L'algorithme présenté ici pour calculer cette factorisation est aussi considérée comme *stable en pratique*, même si des matrices peuvent être construites pour lesquelles il devient progressivement instable pour des valeurs de n croissantes¹.

Résolution de systèmes

La résolution d'un système (2.1) en utilisant la factorisation $PA = LU$ est basée sur le fait que

$$A\mathbf{x} = \mathbf{b} \quad \Leftrightarrow \quad L \underbrace{U\mathbf{x}}_{\mathbf{y}} = PA\mathbf{x} = P\mathbf{b}.$$

Elle se fait dès lors en trois étapes suivantes

1. Une discussion plus complète et accessible peut être trouvée dans «The Smart Money's on Numerical Analysts» de L. N. Trefethen, disponible via <http://people.maths.ox.ac.uk/trefethen/nov12.pdf>

1. permuter les éléments de \mathbf{b} : $\mathbf{b}_p = P\mathbf{b}$ (0 flops, au plus $n - 1$ permutations)
2. résoudre $L\mathbf{y} = \mathbf{b}_p$ (n^2 flops)
3. résoudre $U\mathbf{x} = \mathbf{y}$ (n^2 flops)

2.4 Applications

2.4.1 Calcul de déterminant

La factorisation $PA = LU$ peut être utilisée pour calculer le déterminant $\det(A)$ de A . L'approche est basée sur le fait que, pour deux matrices carrées B et C de mêmes dimensions, on a $\det(BC) = \det(B)\det(C)$. Dans notre cas, cela signifie que

$$\det(P)\det(A) = \det(L)\det(U). \quad (2.4)$$

Or, comme L et U sont des matrices triangulaires, leur déterminants sont donnés par le produit de leurs éléments diagonaux respectifs (cf. Annexe 2.A.2); soit $\det(U) = a_{11}^{(1)} \cdots a_{nn}^{(n)}$ et $\det(L) = 1$, car les éléments diagonaux de L valent 1. Par ailleurs, on a aussi $\det(P) = \det(P_{n-1}) \cdots \det(P_1)$ avec $\det(P_i) = -1$ si la permutation a eu lieu (cf. Annexe 2.A.3) et $\det(P_i) = \det(I) = 1$ sinon; par conséquent $\det(P) = (-1)^p$ où p est le nombre de permutations effectuées. En rassemblant ces observations ensemble dans (2.4), on a finalement (en utilisant $1/(-1)^p = (-1)^p$)

$$\det(A) = \det(U)/\det(P) = (-1)^p a_{11}^{(1)} \cdots a_{nn}^{(n)}.$$

2.4.2 Inversion matricielle

La factorisation $PA = LU$ est également utile pour calculer l'inverse d'une matrice régulière. Pour cela notons que si \mathbf{x}_i est la i -ème colonne de A^{-1} , alors ce vecteur est aussi la solution du système

$$A\mathbf{x}_i = \mathbf{e}_i \quad (2.5)$$

avec \mathbf{e}_i le i -ème vecteur de la base canonique (son élément i vaut 1 et ses autres éléments valent 0). Le calcul de A^{-1} se fait alors en résolvant n systèmes (2.5), $i = 1, \dots, n$, et en assemblant la matrice inverse à partir de ses colonnes \mathbf{x}_i , $i = 1, \dots, n$. En particulier, comme la matrice A est identique pour toutes les systèmes (2.5), sa factorisation n'est calculée qu'une seule fois.

Le calcul de A^{-1} se fait dès lors en quatre étapes

1. calcul de la factorisation $PA = LU$ de A ($\frac{2}{3}n^3 + \mathcal{O}(n^2)$ flops)
2. (pour $i = 1, \dots, n$) $\mathbf{e}_{pi} = P\mathbf{e}_i$ (0 flops, au plus $n(n - 1)$ permutations)
3. résolution (pour $i = 1, \dots, n$) de $L\mathbf{y}_i = \mathbf{e}_{pi}$ (n^3 flops)
4. résolution (pour $i = 1, \dots, n$) de $U\mathbf{x}_i = \mathbf{y}_i$ (n^3 flops)

Le coût total de cette approche est donc de $\frac{8}{3}n^3 + \mathcal{O}(n^2)$ flops.

NOTE : il est possible d'économiser des opérations lors de la première résolution (point 3) car les \mathbf{e}_i et donc les \mathbf{e}_{p_i} ont presque tous leurs éléments égaux à 0. Le coût total peut dans ce cas être réduit à $2n^3 + \mathcal{O}(n^2)$ flops.

Annexe 2.A Rappels

2.A.1 Normes

Une norme $\|\cdot\|$ dans un espace vectoriel réel \mathbb{R}^n est une application de \mathbb{R}^n dans \mathbb{R}^+ (avec donc $\|\mathbf{v}\| \geq 0$) qui satisfait

- (1) $\|\mathbf{v}\| = 0 \Rightarrow \mathbf{v} = \mathbf{0}$ (séparation)
- (2) $\|\lambda \mathbf{v}\| = |\lambda| \|\mathbf{v}\|$, $\lambda \in \mathbb{R}$, $\mathbf{v} \in \mathbb{R}^n$ (absolue homogénéité)
- (3) $\|\mathbf{v} + \mathbf{w}\| \leq \|\mathbf{v}\| + \|\mathbf{w}\|$, $\mathbf{v}, \mathbf{w} \in \mathbb{R}^n$ (inégalité triangulaire)

Pour toute norme vectorielle $\|\cdot\|$ la norme matricielle *induite* d'une matrice $A \in \mathbb{R}^{m \times n}$ (c'est-à-dire une matrice $m \times n$) est

$$\|A\| = \max_{\mathbf{v} \in \mathbb{R}^n, \mathbf{v} \neq \mathbf{0}} \frac{\|A\mathbf{v}\|}{\|\mathbf{v}\|}. \quad (2.A.1)$$

Cette norme satisfait aussi les propriétés (1)-(3) de séparation, d'homogénéité et l'inégalité triangulaire. Par ailleurs, la définition (2.A.1) implique directement

$$\|A\mathbf{v}\| \leq \|A\| \|\mathbf{v}\|.$$

Plus généralement, pour toute matrice $B \in \mathbb{R}^{m \times k}$ elle implique

$$\|AB\| \leq \|A\| \|B\|,$$

car

$$\|AB\| = \max_{\mathbf{v} \in \mathbb{R}^n, \mathbf{v} \neq \mathbf{0}} \frac{\|AB\mathbf{v}\|}{\|\mathbf{v}\|} \leq \max_{\mathbf{v} \in \mathbb{R}^n, \mathbf{v} \neq \mathbf{0}} \frac{\|A\| \|B\mathbf{v}\|}{\|\mathbf{v}\|} = \|A\| \|B\|.$$

Les quelques exemples de normes vectorielles et matricielles utilisées dans ce cours sont :

<u>Normes vectorielles</u>	<u>Normes matricielles induites</u>	
$\ \mathbf{v}\ _1 = \sum_{i=1}^n v_i $	$\ A\ _1 = \max_{j=1, \dots, n} \sum_{i=1}^m a_{ij} $	(norme 1)
$\ \mathbf{v}\ _2 = \left(\sum_{i=1}^n v_i ^2 \right)^{1/2}$	$\ A\ _2 = \sigma_{\max}(A) = \left(\lambda_{\max}(A^T A) \right)^{1/2}$	(norme euclidienne)
$\ \mathbf{v}\ _{\infty} = \max_{i=1, \dots, n} v_i $	$\ A\ _{\infty} = \max_{i=1, \dots, m} \sum_{j=1}^n a_{ij} $	(norme sup)

NOTE : $\sigma_{\max}(A)$ représente la valeur singulière maximale de A et $\lambda_{\max}(A^T A)$ représente la valeur propre maximale de $A^T A$.

2.A.2 Calcul de déterminant à l'aide de mineurs

Soient $A = (a_{ij})$ une matrice carrée et M_{ij} une matrice obtenue en supprimant la ligne i et la colonne j dans A . L'égalité suivante, exprimée ici par rapport à la première ligne de la matrice, est alors vérifiée

$$\det(A) = a_{11} \det(M_{11}) - a_{12} \det(M_{12}) + \dots + (-1)^{n+1} a_{1n} \det(M_{1n}). \quad (2.A.2)$$

Une égalité similaire peut aussi s'écrire par rapport à la première colonne :

$$\det(A) = a_{11} \det(M_{11}) - a_{21} \det(M_{21}) + \cdots + (-1)^{n+1} a_{n1} \det(M_{n1}). \quad (2.A.3)$$

En particulier, si A est une matrice triangulaire, le développement (2.A.2) implique que

$$\det(A) = a_{11} \cdots a_{nn}.$$

En effet, si A est triangulaire inférieure, tous les termes excepté le premier de (2.A.2) sont nuls et on obtient $\det(A) = a_{11} \det(M_{11})$; or, comme M_{11} est elle-même triangulaire inférieure avec les éléments a_{22}, \dots, a_{nn} sur la diagonale, l'application récursive de $\det(A) = a_{11} \det(M_{11})$ termine le raisonnement. Le résultat s'obtient de manière identique pour une matrice triangulaire supérieure en utilisant cette fois (2.A.3).

2.A.3 Matrices de permutation

La matrice de permutation *élémentaire* pour les indices i et $j > i$ est

$$P = \begin{pmatrix} 1 & & & & \\ & \ddots & & & \\ & & 0 & \cdots & 1 \\ & & \vdots & \ddots & \vdots \\ & & 1 & \cdots & 0 \\ & & & & \ddots & \\ & & & & & 1 \end{pmatrix} \begin{matrix} \\ \\ \leftarrow i \\ \\ \leftarrow j \\ \\ \end{matrix}$$

$\begin{matrix} \uparrow & \uparrow \\ i & j \end{matrix}$

Cette matrice a les propriétés suivantes :

- PA correspond à A dont les lignes i et j sont permutées ;
on a donc aussi $\det(PA) = -\det(A)$;
- AP correspond à A dont les colonnes i et j sont permutées ;
on a donc aussi $\det(AP) = -\det(A)$;
- $PP = I$ (la deuxième permutation élémentaire annule la première) .

Notons que le produit d'un nombre quelconque de matrices de permutation élémentaires est également appelé une matrice de permutation, cette dernière n'étant en général pas élémentaire.

Annexe 2.B Somme des puissances des n premiers entiers positifs

Dans cette annexe, on explique comment obtenir des formules compactes approchées et exactes pour la somme

$$S_k = 1^k + \dots + n^k = \sum_{i=1}^n i^k$$

des puissances k des n premiers nombres entiers positifs.

2.B.1 Calcul approché

On approche la somme S_k à l'aide de l'intégrale $\int_0^n x^k dx$. On commence par diviser l'intervalle d'intégration en n intervalles de longueur unité :

$$\int_0^n x^k dx = \int_0^1 x^k dx + \dots + \int_{n-1}^n x^k dx = \sum_{i=1}^n \int_{i-1}^i x^k dx.$$

Dans l' i -ème intervalle ($i = 1, \dots, n$), on a $x \in [i-1, i]$, ce qui implique

$$(i-1)^k \leq x^k \leq i^k$$

et donc, en intégrant chaque membre des inégalités sur l' i -ème intervalle,

$$(i-1)^k \leq \int_{i-1}^i x^k dx \leq i^k.$$

En sommant membre à membre ces inégalités pour tout $i = 1, \dots, n$, on obtient

$$\sum_{i=1}^n (i-1)^k \leq \int_0^n x^k dx \leq \sum_{i=1}^n i^k. \quad (2.B.1)$$

Une interprétation graphique de ces inégalités est donnée à la Figure 2.2 pour $k = 2$ et $n = 3$.

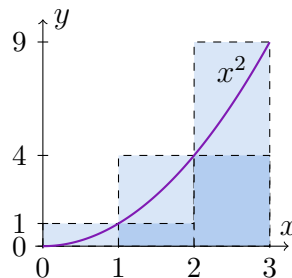


FIGURE 2.2 – Illustration de l'inégalité (2.B.1) pour $k = 2$ et $n = 3$. Plus précisément, l'inégalité gauche (2.B.1) compare l'aire des régions rectangulaires foncées avec l'aire sous la courbe, et celle de droite compare l'aire sous la courbe avec celle des zones rectangulaires foncées **et** claires.

En utilisant la relation $\int_0^n x^k dx = n^{k+1}/(k+1)$ et la définition de S_k , ces inégalités se réécrivent

$$S_k - n^k \leq \frac{n^{k+1}}{k+1} \leq S_k$$

ou, de manière équivalente,

$$\frac{n^{k+1}}{k+1} \leq S_k \leq \frac{n^{k+1}}{k+1} + n^k.$$

La différence en valeur absolue entre S_k et $n^{k+1}/(k+1)$ est donc bornée par n^k , ce qui implique

$$S_k = \frac{n^{k+1}}{k+1} + \mathcal{O}(n^k).$$

2.B.2 Calcul exact

Pour commencer, notons que $S_0 = n$ et que

$$S_1 = \frac{n(n+1)}{2}, \quad (2.B.2)$$

comme on peut le déduire des égalités suivantes

$$\begin{aligned} 2S_1 &= \underbrace{1 + 2 + \cdots + (n-1) + n}_{=S_1} + \underbrace{n + (n-1) + \cdots + 2 + 1}_{=S_1} \\ &= \sum_{i=1}^n i + \sum_{i=1}^n (n+1-i) \\ &= \sum_{i=1}^n (n+1) \\ &= n(n+1). \end{aligned}$$

Les autres formules compactes pour S_k peuvent être obtenues via la méthode générique suivante qui permet d'exprimer S_k en fonction de S_0, \dots, S_{k-1} pour tout $k \geq 1$. Notons d'abord que S_k satisfait

$$S_k = \sum_{i=1}^n i^k = 1 - (n+1)^k + \sum_{i=1}^n (i+1)^k. \quad (2.B.3)$$

Ensuite, la formule du binôme de Newton appliquée à $(i+1)^k$ donne

$$(i+1)^k = i^k + k i^{k-1} + \cdots + k i + 1 = \sum_{j=0}^k \binom{k}{j} i^j, \quad (2.B.4)$$

où

$$\binom{k}{j} = \frac{k!}{(k-j)!j!}$$

est le coefficient binomial. En combinant (2.B.3) et (2.B.4) on a

$$S_k = 1 - (n+1)^k + \sum_{i=1}^n \sum_{j=0}^k \binom{k}{j} i^j.$$

Après quelques manipulations algébriques élémentaires, données par

$$\begin{aligned} S_k &= 1 - (n+1)^k + \sum_{i=1}^n \sum_{j=0}^k \binom{k}{j} i^j = 1 - (n+1)^k + \sum_{j=0}^k \binom{k}{j} \sum_{i=1}^n i^j \\ &= 1 - (n+1)^k + \sum_{j=0}^k \binom{k}{j} S_j \\ &= 1 - (n+1)^k + S_k + kS_{k-1} + \sum_{j=0}^{k-2} \binom{k}{j} S_j, \end{aligned}$$

on aboutit finalement à

$$S_{k-1} = \frac{1}{k} \left((n+1)^k - 1 - \sum_{j=0}^{k-2} \binom{k}{j} S_j \right). \quad (2.B.5)$$

L'équation (2.B.5) permet d'obtenir les formules compactes pour S_{k-1} pour tout $k \geq 1$. En particulier, pour $k = 2$, on retrouve la formule (2.B.2) pour S_1 . Pour $k = 3$ et $k = 4$, on obtient, respectivement,

$$S_2 = \frac{1}{3} \left((n+1)^3 - 1 - S_0 - 3S_1 \right) = \frac{n(n+1)(2n+1)}{6}$$

et

$$S_3 = \frac{1}{4} \left((n+1)^4 - 1 - S_0 - 4S_1 - 6S_2 \right) = \frac{n^2(n+1)^2}{4}.$$

Factorisation QR et moindres carrés

Ce chapitre est composé de deux sections : *Factorisation QR* et *Moindres carrés*. La première section introduit la factorisation QR et la factorisation QR réduite, met en évidence le lien entre ces deux factorisations, et présente un algorithme de calcul d'une factorisation QR complète à l'aide des transformations de Householder. La seconde section introduit la résolution des systèmes linéaires surdéterminés au sens des moindres carrés, présente le conditionnement de ce problème, et introduit les différentes méthodes de résolution, y compris la méthode basée sur la factorisation QR réduite.

En toute rigueur, les deux sections susmentionnées auraient pu constituer deux chapitres distincts. Néanmoins, compte tenu de leur lien étroit et comme les autres applications de la factorisation QR ne sont pas abordées, ces deux sujets sont regroupés dans un seul chapitre. La présentation de la factorisation QR est alors simplifiée car spécifique aux matrices des systèmes surdéterminés, lesquelles ont au moins autant de lignes que de colonnes.

Dans ce qui suit on fait un usage répété du terme *transformation*. Dans notre contexte il s'agit d'une transformation matricielle, c'est-à-dire d'une «banale» opération de multiplication matrice-vecteur. Par abus de langage, on utilise aussi ce terme pour désigner la matrice de la transformation. A titre d'exemple, «transformation orthogonale» peut signifier, selon le contexte, matrice de transformation orthogonale ou multiplication de celle-ci avec le vecteur donné.

3.1 Factorisation QR

Rappelons que la factorisation QR est présentée ici pour des matrices dont le nombre de lignes est supérieur ou égal au nombre de colonnes.

3.1.1 Définitions

Pour définir la factorisation QR on doit définir au préalable ce qu'on entend par matrice orthogonale et matrice trapézoïdale supérieure.

Une matrice Q est *orthogonale* si elle est carrée et satisfait $Q^T Q = I$. Le caractère carré est important car la matrice Q est alors inversible, et a comme inverse Q^T . Le fait que Q^T soit l'inverse de Q implique aussi que $Q Q^T = I$.

Une matrice $R = (r_{ij})$ est *trapézoïdale supérieure* si $r_{ij} = 0$ pour tout $i > j$. Une matrice trapézoïdale supérieure carrée est aussi connue comme triangulaire supérieure.

Une factorisation QR d'une matrice rectangulaire A de dimensions $m \times n$ est une paire de matrices Q de dimensions $m \times m$ et R de dimensions $m \times n$ telles que Q est orthogonale, R trapézoïdale supérieure et

$$A = QR.$$

Il s'agit bien d'une factorisation car celle-ci n'est généralement pas unique.

Dans le cas où $m > n$, les $m - n$ dernières lignes de R sont nulles, et donc

$$R = \begin{pmatrix} \hat{R} \\ 0 \end{pmatrix}, \quad \text{avec } \hat{R} \text{ triangulaire supérieure de dimensions } n \times n.$$

Ainsi, en subdivisant la matrice Q comme

$$Q = \begin{pmatrix} \hat{Q} & \hat{Q}_\perp \end{pmatrix}$$

avec \hat{Q} de dimensions $m \times n$, on a aussi une *factorisation QR réduite*

$$A = \hat{Q} \hat{R}.$$

Si $m > n$, la matrice \hat{Q} n'est pas orthogonale (pourquoi?), mais ses colonnes restent orthogonales entre elles, et donc satisfont $\hat{Q}^T \hat{Q} = I$.

En Octave, une factorisation QR de A s'obtient avec l'instruction `qr(A)`, alors que `qr(A,0)` génère une factorisation QR réduite.

EXEMPLE 22. La représentation schématique des factorisations QR et QR réduite pour $m = 10$ et $n = 5$ est la suivante.

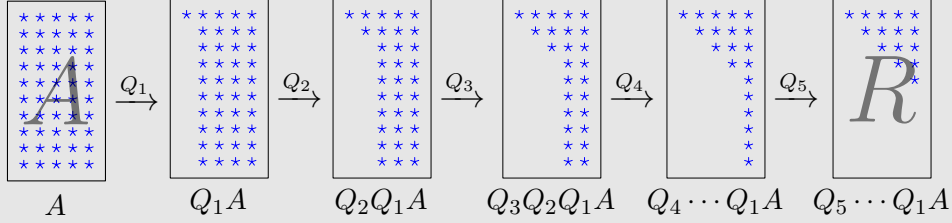
$$A = \begin{bmatrix} \text{10x10 grid of blue stars} \end{bmatrix} = \begin{bmatrix} \text{10x10 grid of blue stars, first 5 columns orange} \end{bmatrix} \begin{bmatrix} \text{10x5 grid of blue stars, first 5 rows orange} \end{bmatrix} = \begin{bmatrix} \text{10x5 grid of blue stars, first 5 columns orange} \end{bmatrix} \begin{bmatrix} \text{5x5 grid of blue stars, first 5 rows orange} \end{bmatrix}$$

3.1.2 Méthode de Householder

Principe de base

Une manière de construire une factorisation QR est d'appliquer une suite de transformations à la fois orthogonales et symétriques qui annulent progressivement les éléments de la matrice sous la diagonale. L'exemple suivant illustre le principe d'une telle approche, alors que les détails de construction des transformations utilisées sont abordés plus bas.

EXEMPLE 23. La représentation schématique des transformations qui annulent les éléments sous la diagonale d'une matrice A à $n = 5$ colonnes est comme suit.



Les matrices Q_i , $i = 1, \dots, 5$, dans cette représentation satisfont donc

$$Q_5 \cdots Q_1 A = R \quad (3.1)$$

avec R trapézoïdale supérieure. De plus, si les matrices Q_i , $i = 1, \dots, 5$, sont orthogonales et symétriques, alors

$$Q = Q_1^{-1} \cdots Q_5^{-1} = Q_1 \cdots Q_5$$

et

$$Q^T = Q_5 \cdots Q_1 \quad (3.2)$$

sont également orthogonales (pourquoi? et symétriques?). Finalement, en combinant (3.1), (3.2) et le fait que Q soit orthogonale on retrouve bien une factorisation

$$A = QR.$$

Transformation de Householder

Jusqu'à présent, on n'a pas spécifié comment obtenir la transformation Q_i qui est à la fois orthogonale, symétrique, et qui annule les éléments sous la diagonale d'une colonne. Cette matrice sera construite à l'aide de la transformation de Householder. Avant d'aborder les détails, on définit la matrice H de cette dernière transformation et examine ses quelques propriétés utiles.

La *transformation de Householder* pour un vecteur $\mathbf{v} = (v_i)$ donné est définie comme

$$H = I - 2 \frac{\mathbf{v}\mathbf{v}^T}{\|\mathbf{v}\|_2^2}. \quad (3.3)$$

PROPRIÉTÉ 1. La matrice H définie pour un vecteur \mathbf{v} donné par (3.3) est orthogonale et symétrique. De plus, si

$$\mathbf{v} = \mathbf{x} \pm \|\mathbf{x}\|_2 \mathbf{e}_j \quad (3.4)$$

avec \mathbf{e}_j le j -ème vecteur de la base canonique, alors

$$H\mathbf{x} = \mp \|\mathbf{x}\|_2 \mathbf{e}_j.$$

DÉMONSTRATION. Notons que $(\mathbf{v}\mathbf{v}^T)^T = \mathbf{v}\mathbf{v}^T$, et donc $\mathbf{v}\mathbf{v}^T$ est symétrique. H est une combinaison linéaire des deux matrices symétriques I et $\mathbf{v}\mathbf{v}^T$, et donc elle aussi est symétrique.

H est orthogonale car

$$H^T H = H H = I - 4 \frac{\mathbf{v}\mathbf{v}^T}{\|\mathbf{v}\|_2^2} + 4 \frac{\overbrace{\mathbf{v}(\mathbf{v}^T \mathbf{v})}^{\|\mathbf{v}\|_2^2}}{\|\mathbf{v}\|_2^4} \mathbf{v}^T = I.$$

Finalement, en utilisant (3.4) on vérifie que

$$2\mathbf{v}^T \mathbf{x} = 2\|\mathbf{x}\|_2^2 \pm 2\|\mathbf{x}\|_2 x_j = \|\mathbf{v}\|_2^2,$$

et donc

$$H\mathbf{x} = \mathbf{x} - 2 \frac{\mathbf{v}(\mathbf{v}^T \mathbf{x})}{\|\mathbf{v}\|_2^2} = \mathbf{x} - \mathbf{v} = \mp \|\mathbf{x}\|_2 \mathbf{e}_j. \quad \blacksquare$$

NOTE : Pour obtenir le vecteur \mathbf{v} à partir du vecteur \mathbf{x} sur base de (3.4) il suffit de rajouter $\pm \|\mathbf{x}\|_2$ à la j -ème composante de ce dernier. Pour éviter le phénomène d'annulation lors de cette opération on choisit le signe qui mène à l'addition de deux nombres de même signe ; en l'occurrence

$$\mathbf{v} = \mathbf{x} + \text{signe}(x_j) \|\mathbf{x}\|_2 \mathbf{e}_j. \quad (3.5)$$

Exemple

L'exemple suivant permet d'introduire et de justifier les différentes étapes de la méthode de Householder.

EXEMPLE 24. On se propose de calculer une factorisation QR de la matrice

$$A = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 8 \\ 1 & 3 & 9 & 27 \\ 1 & 5 & 25 & 125 \\ 1 & 6 & 36 & 216 \\ 1 & 7 & 49 & 343 \end{pmatrix}.$$

La première matrice Q_1 a pour but d'annuler les éléments sous la diagonale dans la première colonne de A . On la définit comme la transformation de Householder

$$Q_1 = I - 2 \frac{\mathbf{v}^{(1)} \mathbf{v}^{(1)T}}{\|\mathbf{v}^{(1)}\|_2^2} \quad (3.6)$$

construite pour satisfaire $Q_1 \mathbf{x} = -\text{signe}(x_1) \|\mathbf{x}\|_2 \mathbf{e}_1$ avec le vecteur \mathbf{x} correspondant à la première colonne de A . En effet, en notant avec $*$ le «reste» de la matrice, on a

$$Q_1 A = Q_1 (\mathbf{x} \ *) = (Q_1 \mathbf{x} \ *) = (-\text{signe}(x_1) \|\mathbf{x}\|_2 \mathbf{e}_1 \ *),$$

et les éléments de la première colonne de Q_1A , excepté le premier élément, deviennent nuls. Plus spécifiquement, $\mathbf{v}^{(1)}$ est défini via (3.4) à partir de la première colonne de A et vaut

$$\mathbf{v}^{(1)} = \mathbf{x} + \|\mathbf{x}\|_2 \mathbf{e}_1 = \begin{pmatrix} 1 + \sqrt{6} \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix};$$

l'application de la transformation Q_1 définie par (3.6) donne alors

$$Q_1A = \begin{pmatrix} -\sqrt{6} & -9.7... & -50.6... & -293.9... \\ & -1.1... & -10.9... & -77.5... \\ & -0.1... & -5.9... & -58.5... \\ & 1.8... & 10.0... & -39.4... \\ & 2.8... & 21.0... & 130.5... \\ & 3.8... & 34.0... & 257.5... \end{pmatrix}.$$

Pour annuler les éléments sous la diagonale dans la deuxième colonne on pourrait procéder de manière similaire : choisir \mathbf{x} égal à la deuxième colonne de Q_1A , ce qui donnerait

$$Q_2 := I - 2 \frac{\mathbf{v}^{(2)} \mathbf{v}^{(2)T}}{\|\mathbf{v}^{(2)}\|_2^2} \quad \text{avec} \quad \mathbf{v}^{(2)} = \mathbf{x} - \|\mathbf{x}\|_2 \mathbf{e}_2 = \begin{pmatrix} -9.7... \\ -1.1... - 11.1... \\ -0.1... \\ 1.8... \\ 2.8... \\ 3.8... \end{pmatrix}$$

et finalement

$$Q_2Q_1A = \begin{pmatrix} -0.7... & 9.7... & 85.0... \\ 2.1... & 11.1... & 64.6... & 396.9... \\ 0.02... & -5.1... & -53.4... \\ -0.3... & -1.4... & -32.8... \\ -0.5... & 3.3... & 19.5... \\ -0.6... & 10.1... & 107.8... \end{pmatrix}.$$

Les «zéros» de la deuxième colonne seraient ainsi introduits au détriment de ceux de la première colonne, ce qui n'est pas le résultat attendu. La propriété suivante aide à contourner cette difficulté.

PROPRIÉTÉ 2. Pour des matrices B , C , D et H de dimensions compatibles on a

$$\begin{pmatrix} I & \\ & H \end{pmatrix} \begin{pmatrix} B & C \\ & D \end{pmatrix} = \begin{pmatrix} B & C \\ & HD \end{pmatrix}.$$

DÉMONSTRATION. Il suffit d'effectuer le produit bloc par bloc. ■

EXEMPLE 24. (SUITE) La Propriété 2 indique que si la matrice Q_2 est choisie de la forme

$$Q_2 := \begin{pmatrix} 1 & \\ & I - 2 \frac{\mathbf{v}^{(2)} \mathbf{v}^{(2)T}}{\|\mathbf{v}^{(2)}\|_2^2} \end{pmatrix} \quad (3.7)$$

alors la première colonne de $Q_1 A$ garde ses éléments nuls sous la diagonale. De plus, le raisonnement utilisé pour la première colonne peut être réutilisé ici en construisant $\mathbf{v}^{(2)}$ sur base de la sous-matrice inférieure droite de $Q_1 A$ (qui correspond à D dans la Propriété 2), et pas sur base la matrice $Q_1 A$ toute entière. Ainsi, on forme $\mathbf{v}^{(2)}$ sur base de la deuxième colonne de

$$Q_1 A = \begin{pmatrix} -\sqrt{6} & -9.7... & -50.6... & -293.9... \\ & \boxed{\begin{matrix} -1.1... \\ -0.1... \\ 1.8... \\ 2.8... \\ 3.8... \end{matrix}} & \begin{matrix} -10.9... \\ -5.9... \\ 10.0... \\ 21.0... \\ 34.0... \end{matrix} & \begin{matrix} -77.5... \\ -58.5... \\ -39.4... \\ 130.5... \\ 257.5... \end{matrix} \end{pmatrix}$$

en excluant le premier élément, ce qui donne

$$\mathbf{v}^{(2)} = \mathbf{x} - \|\mathbf{x}\|_2 \mathbf{e}_1 = \begin{pmatrix} -1.1... - 5.2... \\ -0.1... \\ 1.8... \\ 2.8... \\ 3.8... \end{pmatrix}.$$

En combinaison avec (3.7) on a finalement

$$Q_2 Q_1 A = \begin{pmatrix} -\sqrt{6} & -9.7... & -50.6... & -293.9... \\ & 5.2... & 42.3... & 291.0... \\ & & -4.8... & -51.0... \\ & & -5.4... & -67.8... \\ & & -2.7... & -34.1... \\ & & 1.9... & 35.4... \end{pmatrix}.$$

Notons en passant que la matrice Q_2 définie via (3.7) reste bien symétrique et orthogonale.

Finalement, l'annulation des éléments sous la diagonale dans les colonnes restantes suit le même principe.

Algorithme

Les étapes de l'exemple précédent peuvent être écrites sous forme de l'algorithme suivant. Cet algorithme retourne une matrice R trapézoïdale supérieure et une séquence de vecteurs $\mathbf{v}^{(1)}, \dots, \mathbf{v}^{(n)}$ qui définissent implicitement les transformations élémentaires Q_i pour $i = 1, \dots, n$, et donc la matrice Q . À l'étape k l'algorithme annule les éléments sous la diagonale de la k -ème colonne de la matrice R (qui vaut A au début et devient progressivement trapézoïdale supérieure) en :

- définissant $\mathbf{v}^{(k)}$ via (3.5) à partir des éléments k à m de cette colonne, stockés dans \mathbf{x} ; on choisit de normaliser le vecteur $\mathbf{v}^{(k)}$ dès que celui-ci est construit ;
- appliquant la transformation de Householder (3.3) basée sur ce $\mathbf{v}^{(k)}$ à la sous-matrice $R(k:m, k:n)$ de R qui correspond aux lignes k à m et aux colonnes k à n ; ceci équivaut à appliquer (avec $\mathbf{v}^{(k)}$ normalisé) la transformation

$$Q_k := \begin{pmatrix} I_1 & \\ & I_2 - 2\mathbf{v}^{(k)} \mathbf{v}^{(k)T} \end{pmatrix} \quad (3.8)$$

à la matrice R entière.

ALGORITHME (QR DE HOUSEHOLDER)

entrée : A

sorties : R trapézoïdale supérieure, $\mathbf{v}^{(k)}$ ($k = 1, \dots, n$).

$R = A$

pour $k = 1, \dots, n$

$\mathbf{x} = R(k:m, k)$ % vecteur des composantes k à m de la colonne k de R

$\mathbf{v}^{(k)} = \mathbf{x} + \text{sign}(x_1) \|\mathbf{x}\|_2 \mathbf{e}_1$

$\mathbf{v}^{(k)} = \mathbf{v}^{(k)} / \|\mathbf{v}^{(k)}\|_2$ % normaliser $\mathbf{v}^{(k)}$

 % multiplier $I - 2\mathbf{v}^{(k)} \mathbf{v}^{(k)T}$ avec $R(k:m, k:n)$

$R(k:m, k:n) = R(k:m, k:n) - \mathbf{v}^{(k)}(2(\mathbf{v}^{(k)})^T R(k:m, k:n))$

Coût

On détaille ici le calcul du coût pour l'algorithme précédent, en expliquant au passage pourquoi les transformations de Householder doivent être effectuées comme indiqué dans cet algorithme.

Avant tout, notons que le nombre de flops nécessaires pour calculer le produit de deux matrices B et C de dimensions respectives $s \times \ell$ et $\ell \times p$ est donné par $sp(2\ell - 1)$; il

correspond à un calcul de sp éléments de la matrice résultante dont chacun requiert $2\ell - 1$ flops (ℓ multiplications et $\ell - 1$ additions).

On s'intéresse à présent au coût d'une application de la transformation de Householder $H = I - 2\mathbf{v}\mathbf{v}^T$ (avec \mathbf{v} normalisé) à un vecteur \mathbf{w} de dimension t . Si la matrice $\mathbf{v}\mathbf{v}^T$ doit être formée explicitement, le coût du calcul du produit $\mathbf{v}\mathbf{v}^T$ est de t^2 flops. En restant dans cette optique, notons que le coût de multiplication de la matrice H de dimensions $t \times t$ avec \mathbf{w} est de $2t^2 - t$, et donc aussi quadratique. Par contre, en écrivant $H\mathbf{w}$ comme

$$H\mathbf{w} = \mathbf{w} - \mathbf{v}(\underbrace{2(\mathbf{v}^T \mathbf{w})}_{\text{coût: } 2t \text{ flops}}),$$

le coût de l'évaluation de cette expression n'est plus que de $4t$ flops. De même, le coût de l'application d'une transformation de Householder à une matrice $(m - k + 1) \times (n - k + 1)$ (ou à $n - k + 1$ vecteurs de dimension $m - k + 1$), comme à la dernière ligne de l'algorithme précédent, est $4(m - k + 1)(n - k + 1)$ flops. Comme ce coût est dominant par rapport au coût des autres opérations, le coût global de l'algorithme est de

$$4 \sum_{k=1}^n (m - k + 1)(n - k + 1) + \mathcal{O}(mn) = 2n^2(m - n/3) + \mathcal{O}(mn) \text{ flops}.$$

La dernière égalité peut être retrouvée si on constate (avec $k' = k - 1$, et en notant que la contribution correspondant à $k' = n$ dans la deuxième somme est nulle) que

$$\sum_{k=1}^n (m - k + 1)(n - k + 1) = \sum_{k'=0}^{n-1} (m - k')(n - k') = \sum_{k'=0}^n (m - k')(n - k'),$$

et si on approche la deuxième somme par $\int_0^n (m - x)(n - x)dx = n^2/2(m - n/3)$.

Matrices Q et Q^T

La matrice Q n'est pas formée explicitement pour des raisons de coût qu'engendrerait une telle opération. De plus, il n'est en général pas nécessaire de connaître Q explicitement mais seulement de calculer son produit (ou celui de sa transposée) avec un vecteur. C'est pourquoi on examine d'abord la multiplication matrice-vecteur pour ces matrices.

Pour rappel, la matrice

$$Q = Q_1 \cdots Q_n$$

est obtenue à partir des transformations Q_k , qui elles-mêmes sont définies par le vecteur normalisé $\mathbf{v}^{(k)}$ via (3.8), $k = 1, \dots, n$. Le produit de Q avec un vecteur de dimension m , disons un vecteur \mathbf{w} , est alors obtenu comme suit.

ALGORITHME ($\mathbf{z} = Q\mathbf{w}$)
entrées : vecteurs \mathbf{w} et $\mathbf{v}^{(k)}$, $k = 1, \dots, n$
sortie : vecteur \mathbf{z} .

$\mathbf{z} = \mathbf{w}$
pour $k = n, \dots, 1$
 $\mathbf{z}(k:m) := \mathbf{z}(k:m) - \mathbf{v}^{(k)}(2 \mathbf{v}^{(k)T} \mathbf{z}(k:m))$

L'algorithme pour le produit de

$$Q^T = Q_n \cdots Q_1$$

avec un vecteur \mathbf{w} est similaire ; le seul changement est l'ordre dans lequel les transformations Q_i sont appliquées.

ALGORITHME ($\mathbf{z} = Q^T \mathbf{w}$)
entrées : vecteurs \mathbf{w} et $\mathbf{v}^{(k)}$, $k = 1, \dots, n$
sortie : vecteur \mathbf{z} .

$\mathbf{z} = \mathbf{w}$
pour $k = 1, \dots, n$
 $\mathbf{z}(k:m) := \mathbf{z}(k:m) - \mathbf{v}^{(k)}(2 \mathbf{v}^{(k)T} \mathbf{z}(k:m))$

Dans les deux cas, le coût de l'algorithme est

$$\sum_{k=1}^n (4(m-k) + \mathcal{O}(1)) = 2n(2m-n) + \mathcal{O}(n).$$

La matrice Q peut aussi être évaluée explicitement car sa j -ème colonne s'obtient avec un produit matrice-vecteur $Q\mathbf{e}_j$; alternativement, sa j -ème ligne vaut $(Q^T \mathbf{e}_j)^T$.

Factorisation QR réduite

Comme expliqué au début de ce chapitre (sous-section 3.1.1), on peut obtenir la version réduite d'une factorisation QR à partir de sa version habituelle. En particulier, la matrice triangulaire supérieure \hat{R} dans la factorisation réduite correspond aux premières lignes de la matrice trapézoïdale supérieure R dans la factorisation habituelle. De même, la matrice \hat{Q} dans la factorisation réduite correspond aux premières colonnes de la matrice orthogonale Q dans la factorisation habituelle, c'est-à-dire

$$Q = \begin{pmatrix} \hat{Q} & * \end{pmatrix},$$

l'étoile $*$ représentant le «reste» de la matrice.

Comme Q n'est pas disponible explicitement pour une factorisation QR obtenue avec la méthode de Householder, il convient de déterminer le produit de \hat{Q} et de \hat{Q}^T avec un vecteur à l'aide des algorithmes de la sous-section précédente formulés pour Q . En particulier, pour évaluer le produit de \hat{Q} avec un vecteur $\hat{\mathbf{w}}$ de dimension n on utilisera la transformation Q comme suit

$$\hat{Q}\hat{\mathbf{w}} = Q \begin{pmatrix} \hat{\mathbf{w}} \\ * \end{pmatrix}.$$

De manière similaire, le produit \hat{Q}^T avec un vecteur \mathbf{w} de dimension m s'obtient avec

$$\begin{pmatrix} \hat{Q}^T \mathbf{w} \\ * \end{pmatrix} = Q^T \mathbf{w}.$$

Finalement, la matrice \hat{Q} peut aussi être formée explicitement, sa j -ème colonne étant $\hat{Q}\mathbf{e}_j$.

3.2 Moindres carrés

3.2.1 Problème

Dans cette section on considère la résolution des systèmes

$$A\mathbf{x} = \mathbf{b} \tag{3.9}$$

qui sont *surdéterminés* et de rang maximal, c'est-à-dire des systèmes de dimensions $m \times n$ avec au moins autant d'équations que d'inconnues ($m \geq n$) et dont le rang est maximal ($\text{rang}(A) = n$). En règle générale, un système surdéterminé n'a pas de solution au sens habituel, et donc son *résidu*

$$\mathbf{r} = \mathbf{b} - A\mathbf{x}$$

n'est pas nul. Néanmoins, comme montré plus loin, on peut toujours résoudre un tel système au sens des moindres carrés. Une solution \mathbf{x} est dite *au sens des moindres carrés* si elle minimise la norme euclidienne du résidu ; c'est-à-dire si

$$\|\mathbf{b} - A\mathbf{x}\|_2 = \min_{\mathbf{y}} \|\mathbf{b} - A\mathbf{y}\|_2.$$

Le problème correspondant est un *problème des moindres carrés linéaires*. Le restant de ce chapitre est consacré à la résolution de ce problème.

3.2.2 Équations normales

Pour déterminer la solution \mathbf{x} du système (3.9) au sens des moindres carrés notons que cette solution minimise la fonction

$$f(\mathbf{x}) = \|\mathbf{b} - A\mathbf{x}\|_2^2. \tag{3.10}$$

En utilisant l'égalité

$$\|\mathbf{v} - \mathbf{w}\|_2^2 = (\mathbf{v} - \mathbf{w})^T(\mathbf{v} - \mathbf{w}) = \mathbf{v}^T\mathbf{v} - 2\mathbf{w}^T\mathbf{v} + \mathbf{w}^T\mathbf{w}.$$

valable pour deux vecteurs \mathbf{v} , \mathbf{w} quelconques, l'expression (3.10) de $f(\mathbf{x})$ peut encore être mise sous la forme suivante

$$f(\mathbf{x}) = \mathbf{b}^T\mathbf{b} - 2\mathbf{x}^T A^T \mathbf{b} + \mathbf{x}^T A^T A \mathbf{x}. \quad (3.11)$$

Cette expression met en évidence le fait que $f(\mathbf{x})$ est de classe $C^\infty(\mathbb{R}^n)$, et donc que la solution \mathbf{x} fait partie de ses points critiques, c'est-à-dire des points pour lesquels $\nabla f(\mathbf{x}) = 0$. Pour calculer le gradient de cette fonction par rapport au vecteur \mathbf{x} on utilise les règles suivantes.

PROPRIÉTÉ 3. Pour un vecteur \mathbf{c} et une matrice carrée C de dimensions compatibles on a

$$\nabla(\mathbf{x}^T \mathbf{c}) = \mathbf{c}, \quad \nabla(\mathbf{x}^T C \mathbf{x}) = (C^T + C)\mathbf{x},$$

où ∇ est le gradient par rapport au vecteur \mathbf{x} .

DÉMONSTRATION. Les membres de gauche et de droite de ces deux égalités sont des vecteurs ; il suffit dès lors de montrer l'égalité entre les composantes respectives de ces vecteurs. Pour $\mathbf{c} = (c_i)$ de dimension n et $C = (c_{ij})$ de dimensions $n \times n$ l'égalité entre les composantes ℓ des vecteurs se ramènent à

$$\frac{\partial}{\partial x_\ell} \left(\sum_{i=1}^n c_i x_i \right) = c_\ell, \quad \frac{\partial}{\partial x_\ell} \left(\sum_{i=1}^n \sum_{j=1}^n c_{ij} x_i x_j \right) = \sum_{j=1}^n c_{\ell j} x_j + \sum_{i=1}^n c_{i\ell} x_i. \quad \blacksquare$$

Les règles de dérivation de la Propriété 3 donnent dans notre cas

$$\nabla f(\mathbf{x}) = -2A^T \mathbf{b} + 2A^T A \mathbf{x},$$

et la dérivée s'annule si et seulement si

$$A^T A \mathbf{x} = A^T \mathbf{b}. \quad (3.12)$$

Ce système d'équations linéaires est appelé *système d'équations normales*. Si le rang de A est maximal (égal à n), alors le rang de $A^T A$ est maximal (égal à n) et $A^T A$ est inversible. Dans ce cas le système d'équations normales (3.12) possède une et une seule solution \mathbf{x} , et donc $f(\mathbf{x})$ définie par (3.10) a un et un seul point critique.

On montre que la solution \mathbf{x} du système d'équations normales minimise bien $f(\mathbf{x})$ (et minimise globalement !) en vérifiant que pour tout \mathbf{y} on a

$$\begin{aligned} \|\mathbf{b} - A\mathbf{y}\|_2^2 &= \|\mathbf{b} - A\mathbf{x} + A(\mathbf{x} - \mathbf{y})\|_2^2 \\ &= \|\mathbf{b} - A\mathbf{x}\|_2^2 + 2(\mathbf{x} - \mathbf{y})^T \underbrace{A^T(\mathbf{b} - A\mathbf{x})}_{= \mathbf{0} \text{ car (3.12)}} + \underbrace{\|A(\mathbf{x} - \mathbf{y})\|_2^2}_{\geq 0} \\ &\geq \|\mathbf{b} - A\mathbf{x}\|_2^2. \end{aligned}$$

3.2.3 Interprétation géométrique

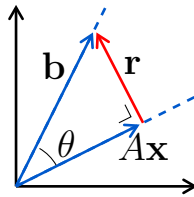


FIGURE 3.1 – Moindres carrés : représentation géométrique.

Une autre manière de retrouver la condition (3.12) est d'utiliser des arguments géométriques. On observe sur la Figure 3.1 que la norme du résidu $\mathbf{r} = \mathbf{b} - A\mathbf{x}$ est minimisée si ce vecteur est orthogonal à l'hyperplan engendré par les vecteurs de forme $A\mathbf{w}$, $\mathbf{w} \in \mathbb{R}^n$; c'est-à-dire si le résidu \mathbf{r} est orthogonal à l'hyperplan engendré par les colonnes de A . Algébriquement, cette condition d'orthogonalité s'écrit comme

$$A^T \mathbf{r} = A^T (\mathbf{b} - A\mathbf{x}) = \mathbf{0},$$

ce qui n'est rien d'autre que le système d'équations normales (3.12).

L'interprétation géométrique permet également d'introduire l'angle θ entre les vecteurs \mathbf{b} et $A\mathbf{x}$; cet angle mesure la capacité de $A\mathbf{x}$ à approcher \mathbf{b} . En particulier, le sinus de cet angle vaut

$$\sin(\theta) = \frac{\|\mathbf{r}\|_2}{\|\mathbf{b}\|_2}.$$

3.2.4 Conditionnement

Conditionnement d'une matrice rectangulaire

Pour une matrice A régulière, le conditionnement (en norme euclidienne, comme le rappelle l'indice 2) est donné par

$$\kappa(A) = \|A^{-1}\|_2 \|A\|_2.$$

Si le nombre de lignes est supérieur au nombre de colonnes, la matrice A n'est pas inversible. Néanmoins, en utilisant le système (3.12) d'équations normales on peut toujours écrire la solution \mathbf{x} d'un système surdéterminé $A\mathbf{x} = \mathbf{b}$ comme

$$\mathbf{x} = A^\dagger \mathbf{b},$$

avec la matrice

$$A^\dagger = (A^T A)^{-1} A^T$$

étant alors un *pseudo-inverse* de A . En suivant cette idée, on généralise la définition du conditionnement d'une matrice rectangulaire A en norme euclidienne en le définissant comme

$$\kappa(A) = \|A^\dagger\|_2 \|A\|_2.$$

Conditionnement des moindres carrés

Avant d'aborder les aspects algorithmiques on considère la sensibilité relative (et en norme euclidienne) de la solution \mathbf{x} aux perturbations des données. Comme dans le cas des systèmes réguliers, cette sensibilité est donnée par le conditionnement du problème, et peut être évaluée séparément pour les perturbations de A et celles de \mathbf{b} . Contrairement aux systèmes réguliers on arrive à des expressions différentes dans les deux cas : la sensibilité aux perturbations dans la matrice A est bornée par

$$\kappa_{MC,A} \leq \kappa(A) + \frac{\kappa(A)^2 \|\mathbf{r}\|_2}{\|A\|_2 \|\mathbf{x}\|_2} \leq \kappa(A) + \kappa(A)^2 \tan(\theta)$$

alors que celle pour les perturbations dans le second membre est majorée par

$$\kappa_{MC,\mathbf{b}} \leq \kappa(A) / \cos(\theta).$$

La dérivation de ces formules est donnée en Annexe 3.A.

Il est à noter que le conditionnement de la solution dépend non seulement du conditionnement $\kappa(A)$ de la matrice du système, mais aussi de l'angle θ entre les vecteurs \mathbf{b} et $A\mathbf{x}$. En fonction de la valeur de θ trois cas de figure sont possibles.

— Si $\theta \approx 0$, alors

$$\kappa_{MC,A} \approx \kappa_{MC,\mathbf{b}} \approx \kappa(A)$$

et les deux conditionnements sont essentiellement déterminés par la matrice du système. Notons que le cas des systèmes réguliers est aussi couvert par l'analyse et correspond à $\theta = 0$; on a alors $\kappa_{MC,A} = \kappa_{MC,\mathbf{b}} = \kappa(A)$, ce qui correspond aux observations du chapitre précédent.

— Si $0 \ll \theta \ll \frac{\pi}{2}$, alors

$$\kappa_{MC,A} \approx \kappa(A)^2 \quad \text{et} \quad \kappa_{MC,\mathbf{b}} \approx \kappa(A).$$

Dans ce cas le pire des conditionnements $\kappa_{MC,A}$ et $\kappa_{MC,\mathbf{b}}$ est plutôt de l'ordre de grandeur de $\kappa(A)^2$.

— Si $\theta \rightarrow \frac{\pi}{2}$, alors

$$\kappa_{MC,A}, \kappa_{MC,\mathbf{b}} \rightarrow \infty.$$

Ce comportement concorde avec l'interprétation géométrique inspirée par la Figure 3.1 : pour $\theta \rightarrow \frac{\pi}{2}$ on a aussi $\mathbf{x} \rightarrow \mathbf{0}$ et donc l'erreur relative $\|\mathbf{x} - \tilde{\mathbf{x}}\|/\|\mathbf{x}\|$ tend vers l'infini pour tout $\tilde{\mathbf{x}} \neq \mathbf{0}$.

3.2.5 Méthode des équations normales (version LU)

La *méthode des équations normales* consiste à former explicitement le système d'équations normales (3.12) et de le résoudre ensuite en utilisant, par exemple, la factorisation LU avec pivotage partiel. Les différents étapes sont alors comme suit.

ALGORITHME (MÉTHODE DES ÉQUATIONS NORMALES)

entrées : matrice A , vecteur \mathbf{b}

sortie : vecteur \mathbf{x}

1. former la matrice $A^T A$ et le second membre $A^T \mathbf{b}$
2. calculer la factorisation LU avec pivotage partiel de $A^T A : LU = P(A^T A)$
3. résoudre le système (3.12) en résolvant le système triangulaire inférieur $L\mathbf{y} = P(A^T \mathbf{b})$ suivi de la résolution du système triangulaire supérieur $U\mathbf{x} = \mathbf{y}$

A titre informatif, évaluons le coût de la méthode dans le cas (important en pratique) où $m \gg n$. Dans ce cas, le coût est dominé par la contribution d'environ mn^2 flops¹ de l'étape 1 qui correspond à la multiplication matricielle.

Pour ce qui est de la propagation des erreurs d'arrondi dans l'algorithme, elle est déterminée, entre autres, par la précision avec laquelle on peut résoudre le système (3.12). Pour toute méthode stable de résolution de systèmes linéaires, l'erreur relative (en norme euclidienne) sur la solution est typiquement de l'ordre de $\kappa(A^T A)u$. Or, comme le montre la propriété suivante, on a aussi $\kappa(A^T A)u = \kappa(A)^2 u$. D'autre part, dans le cas où l'angle θ entre \mathbf{b} et $A\mathbf{x}$ est proche de 0, on a $\kappa_{MC,A} \approx \kappa_{MC,\mathbf{b}} \approx \kappa(A)$, et donc un algorithme stable pour la résolution du système surdéterminé au sens des moindres carrés doit fournir une solution avec une précision relative (en norme euclidienne) de $\kappa(A)^2 u$. Dans ce cas, l'erreur relative sur la solution fournie par la méthode des équations normales risque d'être $\kappa(A)$ fois plus grande que celle d'une méthode stable ; en d'autres termes, la méthode est alors *instable* pour les matrices mal conditionnées (c'est-à-dire pour $\kappa(A) \gg 1$).

PROPRIÉTÉ 4. *Pour toute matrice A on a*

$$\kappa(A^T A) = \kappa(A)^2.$$

DÉMONSTRATION. Dans la démonstration on utilise les Propriétés A.2 et A.3 de l'Annexe 3.B qui se résument au fait que

$$\|B^T B\|_2 = \|B\|_2^2 = \|B^T\|_2^2 \quad (3.13)$$

pour toute matrice B . Notons pour commencer que, comme le pseudo-inverse est donné par $A^\dagger = (A^T A)^{-1} A^T$, la relation (3.13) avec $B = A^\dagger$ donne

$$\|(A^T A)^{-1}\|_2 = \|A^\dagger A^{\dagger T}\|_2 = \|A^\dagger\|_2^2. \quad (3.14)$$

1. La multiplication de deux matrices de dimensions $n \times m$ et $m \times n$ coûte $2mn^2$ flops, mais comme $A^T A$ est une matrice symétrique, ce coût peut être réduit de moitié, car approximativement la moitié des éléments doivent alors être calculés.

Par ailleurs, la relation (3.13) avec $B = A$ donne

$$\|A^T A\|_2 = \|A\|_2^2$$

et donc

$$\kappa(A^T A) = \|(A^T A)^{-1}\|_2 \|A^T A\|_2 = \|A^\dagger\|_2^2 \|A\|_2^2 = \kappa(A)^2. \quad \blacksquare$$

3.2.6 Méthode de la factorisation QR

La *méthode de la factorisation QR* évite de former le système d'équations normales (3.12) explicitement. Elle utilise une factorisation QR réduite $A = \hat{Q}\hat{R}$ pour simplifier analytiquement l'expression de la solution. Plus précisément, la solution \mathbf{x} du système d'équations normales (3.12) satisfait aussi

$$\mathbf{x} = (A^T A)^{-1} A^T \mathbf{b} = (\hat{R}^T \hat{Q}^T \hat{Q} \hat{R})^{-1} \hat{R}^T \hat{Q}^T \mathbf{b} = \hat{R}^{-1} \hat{R}^{-T} \hat{R}^T \hat{Q}^T \mathbf{b} = \hat{R}^{-1} \hat{Q}^T \mathbf{b};$$

L'algorithme correspondant est donc comme suit

ALGORITHME (MÉTHODE DE LA FACTORISATION QR)

entrées : matrice A , vecteur \mathbf{b}

sortie : vecteur \mathbf{x}

1. calculer une factorisation QR réduite $\hat{Q}\hat{R} = A$
2. former $\hat{Q}^T \mathbf{b}$
3. résoudre le système triangulaire supérieur $\hat{R}\mathbf{x} = \hat{Q}^T \mathbf{b}$

Notons que la première étape de l'algorithme est la plus coûteuse. Si $m \gg n$ et si on détermine la factorisation QR avec l'algorithme de Householder, le coût de cette étape correspond à environ $2mn^2$ flops ; c'est deux fois plus que la méthode des équations normales.

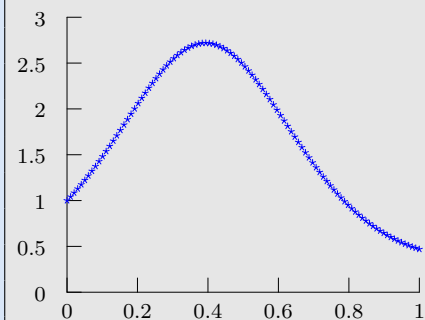
L'avantage principal de cette méthode réside dans le fait qu'elle possède la stabilité inverse, du moins si on détermine la factorisation QR réduite avec l'algorithme de Householder.

3.2.7 Stabilité : exemple

L'exemple suivant illustre un cas pratique où la méthode des équations normales ne permet pas d'obtenir la précision d'une méthode stable. Cet exemple est repris de *Numerical Linear Algebra* de L. N. Trefethen et D. Bau III.

EXEMPLE 25. On considère le problème d'approximation (au sens des moindres carrés) de la fonction $\exp(\sin(4*t))$ aux 100 points repartis uniformément sur l'intervalle $[0, 1]$ par un polynôme de degré 14. On verra dans le Chapitre 6 que ce problème correspond à un système surdéterminé dont le code suivant permet de générer les données A et b de départ. Le code fournit aussi la variable `x15ex` qui représente (aussi fidèlement que la double précision le permet) la valeur exacte de la composante x_{15} de la solution ; elle sert de référence pour comparer les résultats des différentes méthodes. A droite du code, on présente l'ensemble des points à interpoler.

```
m = 100; n = 15;
t = 0:1/(m-1):1; t = t';
A = [];
for i=1:n
    A = [A t.^(i-1)];
end
b = exp(sin(4*t));
% valeur exacte de x(15) obtenue avec
% des outils de précision étendue
x15ex = 2006.787453080206
```



Les résultats pour les différentes méthodes (y compris le $A \backslash b$ d'Octave, qui résout le système $Ax = b$ au sens des moindres carrés si la matrice A n'est pas carrée) est comme suit. Dans chaque cas, le code Octave de la méthode est placé à gauche et le résultat obtenu avec ce code à droite.

Instruction Octave

```
x = A \ b;
x(15) / x15ex
```

ans = 1.00000007318865

Méthode QR (Householder)

```
[Q,R] = qr(A,0);
x = R \ (Q' * b);
x(15) / x15ex
```

ans = 1.00000007318102

Méthode équations normales (version LU)

```
[L U P] = lu(A' * A);
x = U \ (L \ (P * (A' * b)));
x(15) / x15ex
```

ans = 1.56194368637586

L'explication d'un tel comportement est la suivante. On se trouve dans la situation où $\theta = 3.7 \cdot 10^{-6} \approx 0$ et avec une matrice A mal conditionnée ($\kappa(A) = 2.3 \cdot 10^{10}$). Le calcul direct (voir le code qui suit) nous indique que $\kappa_{MC,A} \approx \kappa_{MC,b} \approx \kappa(A)$, ce qui est en accord avec la discussion de la sous-section 3.2.4. L'erreur relative sur la solution (en norme euclidienne) d'une méthode stable est dès lors de l'ordre de $u\kappa(A) \approx 2.5 \cdot 10^{-6}$, en accord avec les observations pour les deux premières approches. Par contre, l'erreur relative sur

la solution due à la méthode des équations normales est de l'ordre de $u\kappa(A)^2 \approx 5.9 \cdot 10^4$; nos observations sont donc même assez optimistes par rapport à ce qu'on aurait pu avoir avec cette méthode.

```
kA = cond(A)
x=A\b; r = A*x-b;
theta = asin(norm(r)/norm(b));
c = norm(r)/norm(A)/norm(x);
K_MCA_max = kA + kA^2 * c
K_MCB_max = kA/cos(theta)
```

```
K_MCA_max = 3.1909e+10
K_MCB_max = 2.2718e+10
```

Annexe 3.A Propriétés de la norme matricielle euclidienne

Cette annexe regroupe quelques propriétés de la norme matricielle euclidienne.

PROPRIÉTÉ A.1. Soient A une matrice *symétrique* et λ_i , $i = 1, \dots, n$, ses valeurs propres. Alors

$$\|A\|_2 = \max_i |\lambda_i| = \max_{\mathbf{v}} \frac{|\mathbf{v}^T A \mathbf{v}|}{\mathbf{v}^T \mathbf{v}}.$$

DÉMONSTRATION. Soit \mathbf{p}_i le vecteur propre normalisé associé à λ_i , $i = 1, \dots, n$. Comme la matrice A est symétrique, l'ensemble de ces vecteurs forme une base orthonormée et tout vecteur \mathbf{v} a une représentation $\mathbf{v} = \sum_{i=1}^n \alpha_i \mathbf{p}_i$ dans cette base. On a alors

$$\|A\|_2^2 = \max_{\mathbf{v}} \frac{\|A\mathbf{v}\|_2^2}{\|\mathbf{v}\|_2^2} = \max_{\mathbf{v}} \frac{\mathbf{v}^T A^2 \mathbf{v}}{\mathbf{v}^T \mathbf{v}} = \max_{\alpha_1, \dots, \alpha_n} \frac{\sum_{i=1}^n \lambda_i^2 \alpha_i^2}{\sum_{i=1}^n \alpha_i^2} = \max_i |\lambda_i|^2.$$

Par analogie avec les deux dernières égalités on a aussi

$$\max_{\mathbf{v}} \frac{|\mathbf{v}^T A \mathbf{v}|}{\mathbf{v}^T \mathbf{v}} = \max_{\alpha_1, \dots, \alpha_n} \frac{|\sum_{i=1}^n \lambda_i \alpha_i^2|}{\sum_{i=1}^n \alpha_i^2} = \max_i |\lambda_i|. \quad \blacksquare$$

PROPRIÉTÉ A.2. Pour toute matrice A rectangulaire

$$\|A^T A\|_2 = \|A\|_2^2.$$

DÉMONSTRATION. La matrice $A^T A$ est symétrique, et donc

$$\|A^T A\|_2 = \max_{\mathbf{v}} \frac{|\mathbf{v}^T A^T A \mathbf{v}|}{\mathbf{v}^T \mathbf{v}} = \max_{\mathbf{v}} \frac{\|A\mathbf{v}\|_2^2}{\|\mathbf{v}\|_2^2}. \quad \blacksquare$$

PROPRIÉTÉ A.3. Pour toute matrice A rectangulaire

$$\|A^T\|_2 = \|A\|_2.$$

DÉMONSTRATION. Considérons d'abord le cas particulier d'un vecteur \mathbf{w} . En notant avec θ l'angle entre les vecteurs \mathbf{w} et \mathbf{v} , on a

$$\|\mathbf{w}^T\|_2 = \max_{\mathbf{v}} \frac{|\mathbf{w}^T \mathbf{v}|}{\|\mathbf{v}\|_2} = \max_{\theta} \frac{\|\mathbf{w}\|_2 \|\mathbf{v}\|_2 \cos(\theta)}{\|\mathbf{v}\|_2} = \|\mathbf{w}\|_2.$$

Maintenant, comme pour deux matrices A , B la norme du produit satisfait

$$\|AB\|_2 \leq \|A\|_2 \|B\|_2,$$

on a

$$\|A^T\|_2 = \max_{\mathbf{v}} \frac{\|A^T \mathbf{v}\|_2}{\|\mathbf{v}\|_2} = \max_{\mathbf{v}} \frac{\|\mathbf{v}^T A\|_2}{\|\mathbf{v}^T\|_2} \leq \max_{\mathbf{v}} \frac{\|\mathbf{v}^T\|_2 \|A\|_2}{\|\mathbf{v}^T\|_2} = \|A\|_2.$$

Puisque $A = A^{T^T}$ on a aussi $\|A\|_2 \leq \|A^T\|_2$, d'où l'égalité. ■

Annexe 3.B Conditionnement : dérivation

On justifie ici les formules du conditionnement (en norme euclidienne) telles qu'introduites dans la sous-section 3.2.4 pour la solution du système (3.9) au sens des moindres carrés. Rappelons que cette solution satisfait aussi le système d'équations normales (3.12), c'est-à-dire

$$A^T A \mathbf{x} = A^T \mathbf{b}. \quad (3.B.1)$$

Rappelons aussi que le conditionnement d'un problème est le pire des facteurs qui amplifient les erreurs relatives dans les données pour obtenir les erreurs relatives dans la solution, et ce dans le cas où les erreurs dans les données tendent vers zéro. On étudiera séparément l'effet des erreurs dans le second membre et dans la matrice du système.

CAS 1. On étudie d'abord les effets des erreurs $\delta \mathbf{b}$ du second membre \mathbf{b} . Soit $\mathbf{x} + \delta \mathbf{x}$ la solution du système perturbé

$$A^T A (\mathbf{x} + \delta \mathbf{x}) = A^T (\mathbf{b} + \delta \mathbf{b}).$$

En y soustrayant le système non perturbé (3.B.1) on a

$$A^T A \delta \mathbf{x} = A^T \delta \mathbf{b}$$

et donc, en utilisant les propriétés de la norme matricielle ainsi que la définition $A^\dagger = (A^T A)^{-1} A^T$ du pseude-inverse, on a

$$\|\delta \mathbf{x}\|_2 = \|A^\dagger \delta \mathbf{b}\|_2 \leq \|A^\dagger\|_2 \|\delta \mathbf{b}\|_2,$$

ce qui donne finalement, avec $\kappa(A) = \|A^\dagger\|_2 \|A\|_2$ et $\|A \mathbf{x}\|_2 / \|\mathbf{b}\|_2 = \cos(\theta)$ (voir Figure 3.1),

$$\kappa_{MC, \mathbf{b}} = \lim_{\epsilon \rightarrow 0} \sup_{\|\delta \mathbf{b}\|_2 \leq \epsilon \|\mathbf{b}\|_2} \frac{\|\delta \mathbf{x}\|_2 / \|\mathbf{x}\|_2}{\|\delta \mathbf{b}\|_2 / \|\mathbf{b}\|_2} \leq \kappa(A) \frac{\|\mathbf{b}\|_2}{\|A\|_2 \|\mathbf{x}\|_2} \leq \kappa(A) \frac{\|\mathbf{b}\|_2}{\|A \mathbf{x}\|_2} = \frac{\kappa(A)}{\cos(\theta)}.$$

CAS 2. Considérons à présent des perturbations δA de la matrice A . Si $\mathbf{x} + \delta \mathbf{x}$ est la solution du système perturbé

$$(A + \delta A)^T (A + \delta A) (\mathbf{x} + \delta \mathbf{x}) = (A + \delta A)^T \mathbf{b}$$

on a, en soustrayant le système non perturbé (3.B.1),

$$A^T A \delta \mathbf{x} + A^T \delta A \mathbf{x} + \delta A^T A \mathbf{x} + \underbrace{\text{le reste}}_{\text{ordre supérieur}} = \delta A^T \mathbf{b},$$

où le terme d'ordre supérieur est négligé car on considère des erreurs relatives qui tendent vers zéro. En utilisant de nouveau les propriétés de la norme matricielle, la définition $A^\dagger = (A^T A)^{-1} A^T$ ainsi que la relation (3.14) on a

$$\|\delta \mathbf{x}\|_2 \leq \|A^\dagger\|_2 \|\delta A\|_2 \|\mathbf{x}\|_2 + \underbrace{\|(A^T A)^{-1}\|_2}_{\|A^\dagger\|_2^2} \underbrace{\|\delta A^T\|_2}_{\|\delta A\|_2} \underbrace{\|\mathbf{b} - A \mathbf{x}\|_2}_{\|\mathbf{r}\|_2},$$

et finalement, avec de nouveau $\kappa(A) = \|A^\dagger\|_2 \|A\|_2$ et $\|\mathbf{r}\|_2 / \|A \mathbf{x}\|_2 = \tan(\theta)$ (voir Figure 3.1),

$$\kappa_{MC, A} = \lim_{\epsilon \rightarrow 0} \sup_{\|\delta A\|_2 \leq \epsilon \|A\|_2} \frac{\|\delta \mathbf{x}\|_2 / \|\mathbf{x}\|_2}{\|\delta A\|_2 / \|A\|_2} \leq \kappa(A) + \kappa(A)^2 \frac{\|\mathbf{r}\|_2}{\|A\|_2 \|\mathbf{x}\|_2} \leq \kappa(A) + \kappa(A)^2 \overbrace{\frac{\|\mathbf{r}\|_2}{\|A \mathbf{x}\|_2}}^{\tan(\theta)}.$$

Méthodes itératives pour systèmes linéaires

Ce chapitre est consacré aux méthodes itératives pour la résolution de systèmes linéaires réguliers. Il constitue une suite logique du Chapitre 2, qui introduit les systèmes linéaires réguliers et présente les méthodes directes. Le présent chapitre aborde donc la résolution des mêmes problèmes mais par une autre famille de méthodes.

Le principe des méthodes itératives consiste à générer une séquence de solutions approchées qu'on arrête quand la solution approchée la plus récente est jugée satisfaisante. Plus précisément, on considère ici deux groupes de méthodes : les méthodes stationnaires et les méthodes basées sur la minimisation de l'énergie. La fin du chapitre est consacrée aux critères utilisés pour arrêter une méthode itérative.

4.1 Définition et contexte

D'une manière générale, on parle d'une *méthode itérative* si elle génère une séquence potentiellement infinie de solutions approchées du problème considéré. Dans le cas d'un système régulier

$$A\mathbf{x} = \mathbf{b}, \quad (4.1)$$

une méthode itérative produit donc une séquence de solutions approchées de ce système ; ici cette séquence est notée $\mathbf{x}^{(0)}, \mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots$. Typiquement, la solution approchée initiale $\mathbf{x}^{(0)}$ est choisie¹ avant l'application de la méthode ; le vecteur $\mathbf{x}^{(1)}$ est ensuite déterminé sur base de $\mathbf{x}^{(0)}$ et de l'équation (4.1) ; le vecteur $\mathbf{x}^{(2)}$ est déterminé par après sur base de $\mathbf{x}^{(1)}$ (et éventuellement de $\mathbf{x}^{(0)}$) et de l'équation (4.1) ; et ainsi de suite. Cette dépendance est alors schématisée comme suit :

$$\mathbf{x}^{(0)} \xrightarrow{\text{itération 1}} \mathbf{x}^{(1)} \xrightarrow{\text{itération 2}} \dots \xrightarrow{\text{itération } k} \mathbf{x}^{(k)} \xrightarrow{\text{itération } k+1} \dots$$

Les méthodes itératives peuvent être attrayantes dans le cas où la solution recherchée doit être déterminée avec une précision limitée. C'est notamment le cas lorsque le système

1. Le choix de $\mathbf{x}^{(0)}$ est effectué soit par l'utilisateur, soit en prenant un vecteur par défaut, habituellement $\mathbf{0}$.

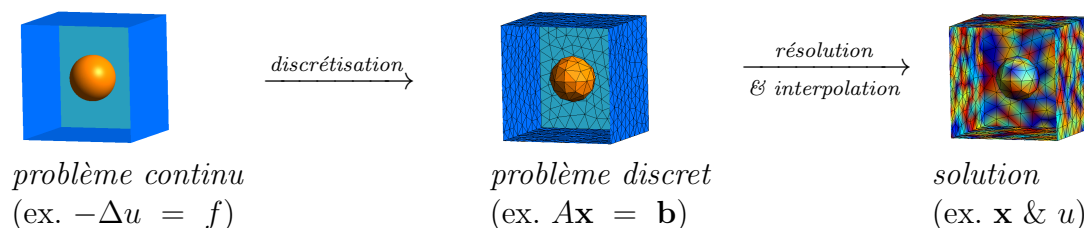


FIGURE 4.1 – Etapes de résolution d'un problème continu.

(4.1) provient de la discrétisation d'un *problème continu* (voir Figure 4.1). Dans ce cas, lors de l'approximation du problème continu par un problème discret (par exemple, un système linéaire), on commet une *erreur de discrétisation*, qui est souvent quantifiable. Par conséquent on peut souvent se contenter d'une *solution approchée* du problème discret, pour autant que l'erreur ainsi commise reste inférieure à l'erreur de discrétisation.

EXEMPLE 26. A titre d'exemple, considérons le problème aux limites de Poisson. Il décrit, entre autres, la distribution de la température dans un solide homogène en fonction des sources de chaleur à l'intérieur du solide et de la température sur ses parois. Le domaine représentant le solide peut être un intervalle (1D), un carré (2D) ou un cube (3D). Le problème 1D sera abordé plus en détails dans le Chapitre 9. Néanmoins, il est bien de savoir à ce stade-ci qu'un problème de Poisson continu 1D, 2D ou 3D peut être approché par un système linéaire.

Notons que la matrice du système approché est dite creuse car beaucoup d'éléments de cette matrice sont nuls. De plus, il est possible d'exploiter son caractère creux pour avoir une factorisation LU avec des facteurs creux (voir Figure 4.2). La structure du facteur détermine alors le coût (en flops) de la factorisation LU, qui est par ailleurs moindre que celui de la factorisation d'une matrice dense de même dimension.

Malgré le fait qu'il soit possible d'exploiter le caractère creux de la matrice, l'analyse de la Table 4.1 suggère que la méthode directe n'est pas toujours la plus rapide. Plus précisément, comparée à la méthode du gradient conjugué (une méthode itérative), elle est la seule à fournir la solution à 1D (la méthode du gradient conjugué n'a pas convergé en 1000 itérations), est comparable en temps à 2D, et est moins rapide à 3D. Le fait que les problèmes 3D soient les plus intéressants en pratique explique l'attention qu'on prête aux méthodes itératives. Pour ce qui est de l'occupation mémoire, la méthode du gradient conjugué (qui ne doit conserver en mémoire qu'essentiellement la matrice A elle-même) est toujours moins coûteuse que la factorisation LU (dont l'occupation mémoire est, à un facteur près, celle de la matrice L). Pour finir, notons que la discrétisation des problèmes 1D, 2D et 3D est choisie en sorte d'avoir des systèmes linéaires de même dimension. Dans ce cas, les différences entre les méthodes (dans un sens comme dans l'autre) s'accroissent quand cette dimension augmente.

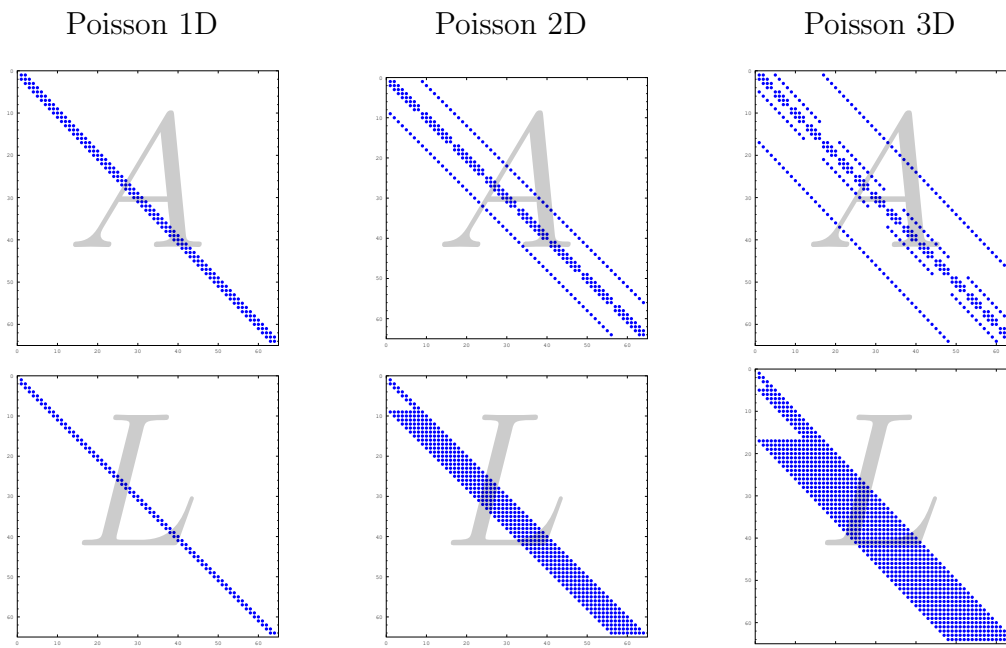


FIGURE 4.2 – L’emplacement des éléments non nuls de la matrice A pour un système issu de la discrétisation du problème de Poisson (à gauche – 1D, au centre – 2D, à droite – 3D), ainsi que ceux de la matrice L dans la factorisation LU de A (les éléments non nuls de la matrice U dans la factorisation LU sont ceux de L^T). Les éléments non nuls correspondent aux points bleus (gris) ; les éléments nuls sont blancs.

Poisson 1D	Poisson 2D	Poisson 3D
TEMPS		
	<code>lu(A)</code>	
0.04 s.	0.49 s.	3.7 s.
	<code>pcg(A,b,1e-4,1000)</code>	
— ^a	0.43 s. (234 itér.)	0.12 s. (56 itér.)
MÉMOIRE		
	élém. non nuls de A	
$4.7 \cdot 10^4$	$7.7 \cdot 10^4$	$1.1 \cdot 10^5$
	élém. non nuls de L	
$3.1 \cdot 10^4$	$1.9 \cdot 10^6$	$9.4 \cdot 10^6$

TABLE 4.1 – Principales caractéristiques de la factorisation LU creuse (`lu` étant une étape principale d’une méthode directe) et de la méthode du gradient conjugué (`pcg`, méthode itérative) appliquées aux systèmes issus de la discrétisation du problème de Poisson 1D, 2D et 3D avec un second membre aléatoire. La partie supérieure de la table fournit le temps de calcul (en secondes), la partie inférieure donne l’occupation mémoire de la matrice A et du facteur L . La matrice A du système est de dimensions 15625×15625 dans tous les cas (1D, 2D et 3D). —^a signifie que la méthode du gradient conjugué n’a pas trouvé une solution acceptable en 1000 itérations.

4.2 Méthodes stationnaires

La présentation des méthodes stationnaires fait intervenir la notion de *correction*. Une correction est ce qu'on ajoute à la solution approchée pour obtenir une nouvelle approximation. Déterminer la correction idéale – celle qui donne la solution exacte comme approximation suivante – peut s'avérer aussi coûteux que de résoudre le système de départ. L'utilisation d'une correction approchée peut par contre être rentable. Les méthodes stationnaires déterminent une correction approchée en résolvant un système auxiliaire avec une matrice qui ne change pas d'une itération à l'autre. Dans cette section nous présentons d'abord les aspects formels avant d'aborder deux méthodes simples : la méthode de Jacobi et la méthode de Gauss-Seidel.

4.2.1 Principe

Soient $\mathbf{x}^{(k)}$ une solution approchée du système

$$A\mathbf{x} = \mathbf{b} \quad (4.2)$$

et \mathbf{x} sa solution exacte. La correction idéale pour $\mathbf{x}^{(k)}$ est

$$\mathbf{e} = \mathbf{x} - \mathbf{x}^{(k)}, \quad (4.3)$$

puisque la nouvelle solution approchée $\mathbf{x}^{(k)} + \mathbf{e}$ correspond alors à la solution exacte. Il n'est pourtant pas possible d'évaluer cette correction directement, car \mathbf{x} n'est pas connu. Par contre, il est possible d'évaluer directement son produit avec A , à savoir

$$A\mathbf{e} = \underbrace{A\mathbf{x}}_{\mathbf{b}} - A\mathbf{x}^{(k)}, \quad (4.4)$$

en calculant le membre de droite

$$\mathbf{r}^{(k)} = \mathbf{b} - A\mathbf{x}^{(k)}; \quad (4.5)$$

ce dernier vecteur est appelé *résidu*. Une méthode *stationnaire* approche le système (4.4) par

$$B\mathbf{e}^{(k)} = \mathbf{b} - A\mathbf{x}^{(k)}, \quad (4.6)$$

où

$$B \approx A$$

est une approximation de la matrice du système. La correction approchée est alors calculée en résolvant le système approché (4.6) et la nouvelle approximation est déterminée via

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \mathbf{e}^{(k)}. \quad (4.7)$$

L'algorithme générique pour une méthode stationnaire est donc comme suit :

ALGORITHME (MÉTHODE STATIONNAIRE GÉNÉRIQUE) :

entrées : $A, \mathbf{b}, \mathbf{x}^{(0)}, B$

sortie : $\mathbf{x}^{(k+1)}$

répéter pour $k = 0, 1, 2, \dots$, jusqu'à l'arrêt

déterminer $\mathbf{e}^{(k)}$ tel que $B\mathbf{e}^{(k)} = \mathbf{b} - A\mathbf{x}^{(k)}$ % résolution

$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \mathbf{e}^{(k)}$ % correction

Notons qu'une méthode stationnaire ne peut être compétitive que si la résolution du système approché (4.6) est sensiblement moins coûteuse que la résolution du système de départ (4.4). Autrement dit, la matrice B doit être plus facile à «inverser» que la matrice A .

4.2.2 Convergence

On dit qu'une méthode itérative *converge* si la séquence $\mathbf{x}^{(0)}, \mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots$ de solutions approchées produite par la méthode tend vers la solution exacte \mathbf{x} . La convergence est une propriété souhaitable, mais elle n'est pas garantie a priori. C'est pourquoi une attention particulière est prêtée aux conditions qui assurent la convergence d'une méthode stationnaire. Ici on présente le résultat de base qui relie les propriétés de convergence aux valeurs propres d'une matrice particulière, appelée matrice d'itération.

Notre analyse exploite la relation suivante, qui relie la correction à l'itération $k + 1$ à celle à l'itération k :

$$\begin{aligned}
 \mathbf{x} - \mathbf{x}^{(k+1)} &= (\mathbf{x} - \mathbf{x}^{(k)}) - \underbrace{(\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)})}_{\mathbf{e}^{(k)}} && \text{(voir (4.7))} \\
 &= (\mathbf{x} - \mathbf{x}^{(k)}) - B^{-1}(\mathbf{b} - A\mathbf{x}^{(k)}) && \text{(par (4.6))} \\
 &= (\mathbf{x} - \mathbf{x}^{(k)}) - B^{-1}A(\mathbf{x} - \mathbf{x}^{(k)}) && \text{(car } A\mathbf{x} = \mathbf{b}) \\
 &= (I - B^{-1}A)(\mathbf{x} - \mathbf{x}^{(k)}) .
 \end{aligned}$$

Sous une forme condensée cette relation s'écrit comme

$$\mathbf{x} - \mathbf{x}^{(k+1)} = E(\mathbf{x} - \mathbf{x}^{(k)}) , \quad (4.8)$$

où $E = I - B^{-1}A$ est la *matrice d'itération*.

Le théorème suivant donne alors une condition suffisante de convergence d'une méthode itérative stationnaire. Plus particulièrement, il montre que si les valeurs propres de la matrice d'itération E sont toutes inférieures à 1 en valeur absolue, alors la suite de vecteurs $\mathbf{w}^{(k)} = \mathbf{x} - \mathbf{x}^{(k)}$ converge vers $\mathbf{0}$ et la suite de vecteurs $\mathbf{x}^{(k)}$ converge donc vers \mathbf{x} .

THÉORÈME 1. *Soit E une matrice $n \times n$. Si toutes les valeurs propres λ_i de E satisfont $|\lambda_i| < 1$, $i = 1, \dots, n$, alors la suite définie par*

$$\mathbf{w}^{(k+1)} = E \mathbf{w}^{(k)}$$

converge vers $\mathbf{0}$ pour tout choix de $\mathbf{w}^{(0)}$.

DÉMONSTRATION. Nous nous limitons au cas où la matrice E est diagonalisable. Dans ce cas, il existe une matrice P telle que

$$P^{-1}EP = \text{diag}(\lambda_i) =: \Lambda.$$

On a alors (et ce pour toute norme vectorielle $\|\cdot\|$ présenté dans l'Annexe 2.A.1 et la norme matricielle correspondante)

$$\|\mathbf{w}^{(k)}\| = \|E^k \mathbf{w}^{(0)}\| = \|P \Lambda^k P^{-1} \mathbf{w}^{(0)}\| \leq \underbrace{\|P\| \|P^{-1}\|}_{\text{const}} \underbrace{\|\mathbf{w}^{(0)}\| \left(\max_i |\lambda_i| \right)^k}_{\rightarrow 0}. \quad \blacksquare$$

4.2.3 Méthodes stationnaires classiques

Nous présentons ici deux choix de la matrice B qui correspondent à deux méthodes classiques : celle de Jacobi et celle de Gauss-Seidel. Pour motiver ces choix, notons que la correction obtenue avec (4.6) ne peut être proche de la correction idéale obtenue avec (4.4) que si la matrice auxiliaire B est proche de la matrice A du système, et donc, construite sur base de cette dernière. Au moins deux options qui mènent à une matrice B facile à «inverser» sont envisageables : B correspond à la partie diagonale de A (cas Jacobi) ou à la partie triangulaire inférieure² de A (cas Gauss-Seidel). Les deux choix sont représentés schématiquement sur la Figure 4.3.

Méthode de Jacobi

La méthode de Jacobi est une méthode stationnaire qui utilise comme approximation de la matrice A du système sa partie diagonale. Autrement dit, il s'agit d'une méthode stationnaire avec la matrice $B = B_J$ définie par

$$(B_J)_{ij} = \begin{cases} a_{ii} & \text{si } i = j, \\ 0 & \text{sinon.} \end{cases} \quad (4.9)$$

Bien entendu, la méthode n'est applicable que si la matrice B_J est régulière ; comme elle est diagonale, elle est régulière ssi $a_{ii} \neq 0$, $i = 1, \dots, n$.

2. Le cas de la partie triangulaire supérieure est similaire et mène à une méthode qu'on appelle aussi méthode de Gauss-Seidel ; pour distinguer ces deux variantes on parle de la méthode de Gauss-Seidel progressive si B est triangulaire inférieure et de la méthode de Gauss-Seidel rétrograde si B est triangulaire supérieure.

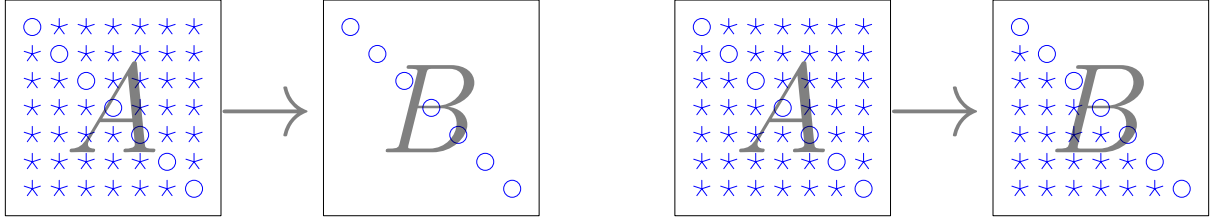


FIGURE 4.3 – Représentation schématique de la construction de la matrice B dans le cas des méthodes de Jacobi (gauche) et de Gauss-Seidel (droite).

Une autre formulation possible présente une itération de la méthode de Jacobi sous une forme *élément par élément*. Pour introduire cette formulation nous regroupons les étapes (4.6) et (4.7), ce qui donne

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + B_J^{-1}(\mathbf{b} - A\mathbf{x}^{(k)}).$$

Sous la forme élément par élément cela donne

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

et donc

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1, j \neq i}^n a_{ij} x_j^{(k)} \right), \quad i = 1, \dots, n. \quad (4.10)$$

La forme élément par élément possède aussi une interprétation plus intuitive qu'on présente dans l'exemple suivant.

EXEMPLE 27. Soit le système régulier (système de l'Exemple 1, Chapitre M)

$$\begin{cases} 2x_1 & -x_2 & & = 1 \\ -x_1 & +2x_2 & -x_3 & = 1 \\ & -x_2 & +x_3 & = 1 \end{cases}.$$

Si on isole dans chacune des trois équations la variable qui porte le numéro de l'équation et l'exprime par rapport aux autres variables, on obtient

$$\begin{cases} x_1 & = \frac{1}{2}(1 + x_2) \\ x_2 & = \frac{1}{2}(1 + x_1 + x_3) \\ x_3 & = 1 + x_2 \end{cases}. \quad (4.11)$$

La méthode de Jacobi est obtenue à partir de cet ensemble d'équations en remplaçant x_i

par $x_i^{(k+1)}$ dans les membres de gauche et par $x_i^{(k)}$ dans les membres de droite, soit

$$\begin{cases} x_1^{(k+1)} &= \frac{1}{2}(1 + x_2^{(k)}) \\ x_2^{(k+1)} &= \frac{1}{2}(1 + x_1^{(k)} + x_3^{(k)}) \\ x_3^{(k+1)} &= 1 + x_2^{(k)} \end{cases} .$$

Vous pouvez vérifier que cette approche donne exactement les mêmes équations que celles obtenues dans le cas élément par élément (4.10); il s'agit en fait d'une autre manière d'obtenir ces équations.

Pour finir, les quelques solutions approchées générées par la méthode de Jacobi avec $\mathbf{x}_0 = \mathbf{0}$ comme solution initiale sont

$$\underbrace{\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}}_{\mathbf{x}^{(0)}} \Rightarrow \underbrace{\begin{pmatrix} 0.5 \\ 0.5 \\ 1 \end{pmatrix}}_{\mathbf{x}^{(1)}} \Rightarrow \underbrace{\begin{pmatrix} 0.75 \\ 1.25 \\ 1.5 \end{pmatrix}}_{\mathbf{x}^{(2)}} \Rightarrow \dots \Rightarrow \underbrace{\begin{pmatrix} 2.95\dots \\ 4.93\dots \\ 5.91\dots \end{pmatrix}}_{\mathbf{x}^{(30)}}$$

alors que la solution exacte est $\mathbf{x} = \begin{pmatrix} 3 \\ 5 \\ 6 \end{pmatrix}$.

Finalement, notons que la méthode de Jacobi converge si, comme démontré dans le Théorème 1,

$$|\lambda_i(I - B_J^{-1}A)| < 1$$

pour toute valeur propre $\lambda_i(I - B_J^{-1}A)$ de la matrice d'itération $I - B_J^{-1}A$.

Méthode de Gauss-Seidel

La méthode de Gauss-Seidel utilise comme approximation de la matrice A du système sa partie triangulaire inférieure. Plus précisément, il s'agit d'une méthode stationnaire avec la matrice $B = B_{GS}$ définie par

$$(B_{GS})_{ij} = \begin{cases} a_{ij} & \text{si } j \leq i, \\ 0 & \text{sinon.} \end{cases} \quad (4.12)$$

Pour utiliser la méthode il faut s'assurer au préalable que la matrice B_{GS} est inversible; comme B_{GS} est une matrice triangulaire, elle est inversible ssi $a_{ii} \neq 0$, $i = 1, \dots, n$.

La méthode de Gauss-Seidel peut aussi être mise sous une forme élément par élément. Pour ce faire on regroupe d'abord les deux étapes (4.6) et (4.7) en une seule

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + B_{GS}^{-1}(\mathbf{b} - A\mathbf{x}^{(k)}) .$$

Ensuite, on multiplie à gauche par B_{GS} les deux membres de cette égalité pour obtenir

$$B_{GS}\mathbf{x}^{(k+1)} = \mathbf{b} - (A - B_{GS})\mathbf{x}^{(k)} .$$

En utilisant le fait que $A - B_{GS}$ est la partie triangulaire strictement supérieure de A , on a sous la forme élément par élément

$$\sum_{j \leq i} a_{ij} x_j^{(k+1)} = b_i - \sum_{j > i} a_{ij} x_j^{(k)}, \quad i = 1, \dots, n.$$

En subdivisant la somme dans le membre de gauche en deux parties : la somme sur $j < i$ et un terme correspondant au cas $j = i$, en mettant la première des deux parties dans le membre de droite, et en divisant par a_{ii} on aboutit à

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j < i} a_{ij} x_j^{(k+1)} - \sum_{j > i} a_{ij} x_j^{(k)} \right), \quad i = 1, \dots, n. \quad (4.13)$$

Comme dans le cas de Jacobi, la formulation élément par élément possède aussi une interprétation intuitive qu'on détaille dans l'exemple suivant.

EXEMPLE 27. (FIN) Pour obtenir la méthode de Gauss-Seidel d'une manière plus intuitive on utilise comme point de départ le même ensemble d'équations (4.11) que dans le cas de Jacobi ; de plus, on fixe l'ordre d'évaluation à $x_1 \rightarrow x_2 \rightarrow x_3$. Comme précédemment, on remplace dans les membres de gauche tous les x_i par $x_i^{(k+1)}$. Pour ce qui est des membres de droite, on remplace tous les x_i par $x_i^{(k+1)}$ si ces derniers ont déjà été calculés au moment où on considère une équation donnée (compte tenu de l'ordre d'évaluation, cela arrive si i est inférieur au numéro de l'équation) et par $x_i^{(k)}$ sinon. On a ainsi

$$\begin{cases} x_1^{(k+1)} &= \frac{1}{2}(1 + x_2^{(k)}) \\ x_2^{(k+1)} &= \frac{1}{2}(1 + x_1^{(k+1)} + x_3^{(k)}) \\ x_3^{(k+1)} &= 1 + x_2^{(k+1)} \end{cases}.$$

Ici aussi l'approche présentée n'est qu'une autre manière d'obtenir les équations (4.13) de la formulation élément par élément.

Pour finir, les quelques solutions approchées générées par la méthode de Gauss-Seidel avec $\mathbf{x}_0 = \mathbf{0}$ comme solution initiale sont

$$\underbrace{\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}}_{\mathbf{x}^{(0)}} \Rightarrow \underbrace{\begin{pmatrix} 0.5 \\ 0.75 \\ 1.75 \end{pmatrix}}_{\mathbf{x}^{(1)}} \Rightarrow \underbrace{\begin{pmatrix} 0.875 \\ 1.81... \\ 2.81... \end{pmatrix}}_{\mathbf{x}^{(2)}} \Rightarrow \dots \Rightarrow \underbrace{\begin{pmatrix} 2.94 \\ 4.92 \\ 5.92 \end{pmatrix}}_{\mathbf{x}^{(15)}};$$

pour rappel, la solution exacte est $\mathbf{x} = \begin{pmatrix} 3 \\ 5 \\ 6 \end{pmatrix}$.

Pour ce qui est de la convergence de la méthode de Gauss-Seidel, on notera (sans démonstration) que la suite $\mathbf{x}^{(k)}$ converge vers la solution \mathbf{x} du système si A est une

matrice symétrique définie positive (on spécifie dans la section suivante ce que signifie exactement définie positive, un peu de patience!).

4.3 Méthodes basées sur la minimisation d'énergie

Les méthodes itératives les plus populaires appartiennent en grande majorité à la famille des méthodes non stationnaires. Pour ne donner qu'un léger aperçu de cette vaste famille on se limite ici aux systèmes dont la matrice A est symétrique définie positive. On peut alors utiliser une grandeur auxiliaire – l'énergie – pour quantifier la proximité entre une solution approchée et la solution exacte. Dans la suite de cette section on définit une approche générique basée sur la minimisation d'énergie, avant d'aborder deux méthodes plus spécifiques qui en découlent : la méthode du gradient et la méthode du gradient conjugué. Cette dernière est la méthode itérative de référence pour les systèmes symétriques définis positifs, surtout en combinaison avec ce qu'on appelle un préconditionneur ; c'est la raison pour laquelle on donne un aperçu du préconditionnement vers la fin de la section.

4.3.1 Energie du système

Une matrice A est *définie positive* si pour tout vecteur $\mathbf{v} \neq \mathbf{0}$ de dimension compatible on a

$$\mathbf{v}^T A \mathbf{v} > 0.$$

Elle est *symétrique définie positive* si elle est symétrique et définie positive.

On associe à une matrice symétrique définie positive A la *norme énergie* correspondante, définie par

$$\|\mathbf{v}\|_A = \sqrt{\mathbf{v}^T A \mathbf{v}}.$$

Il s'agit bien d'une norme (elle respecte en particulier l'inégalité triangulaire) même si on n'utilise ici que son caractère positif :

$$\|\mathbf{v}\|_A > 0 \quad \forall \mathbf{v} \neq \mathbf{0}.$$

Pour une approximation donnée $\mathbf{x}^{(k)}$ de la solution, l'«énergie» de la différence avec la solution exacte \mathbf{x} correspond alors à

$$f(\mathbf{x}^{(k)}) := \|\mathbf{x} - \mathbf{x}^{(k)}\|_A^2 = \mathbf{x}^{(k)T} A \mathbf{x}^{(k)} - 2 \mathbf{x}^{(k)T} \underbrace{A \mathbf{x}}_{\mathbf{b}} + \underbrace{\mathbf{x}^T A \mathbf{x}}_{\text{const}}. \quad (4.14)$$

Notez en particulier que

$$f(\mathbf{x}^{(k)}) \geq 0 \quad \text{et} \quad f(\mathbf{x}^{(k)}) = 0 \Leftrightarrow \mathbf{x}^{(k)} = \mathbf{x}.$$

Par ailleurs, la fonction $f(\mathbf{x}^{(k)})$ peut être évaluée (à une constante additive près) sans connaître la solution exacte du système (4.1). Dès lors, une manière de s'assurer que la suite $\mathbf{x}^{(k)}$, $k = 1, \dots$, se rapproche de la solution exacte est de choisir $\mathbf{x}^{(k+1)}$ tel que $f(\mathbf{x}^{(k+1)}) < f(\mathbf{x}^{(k)})$.

4.3.2 Minimisation d'énergie

Les méthodes de minimisation d'énergie forment la nouvelle solution approchée en rajoutant à la solution approchée courante un vecteur dont la direction est donnée et dont l'amplitude est choisie pour minimiser l'énergie de l'erreur. Les méthodes particulières diffèrent alors entre elles par le choix de la direction. Deux choix seront abordés dans les sous-sections suivantes, menant à la méthode du gradient et la méthode du gradient conjugué.

Concrètement, on suppose avoir choisi une direction $\mathbf{p} \neq \mathbf{0}$ à l'itération $k + 1$. La solution approchée est alors donnée par

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha \mathbf{p}, \quad \alpha \in \mathbb{R}. \quad (4.15)$$

La valeur de l'amplitude α qui minimise l'énergie est obtenue en étudiant la dérivée de

$$f(\mathbf{x}^{(k+1)}) = (\mathbf{x}^{(k)} + \alpha \mathbf{p})^T A (\mathbf{x}^{(k)} + \alpha \mathbf{p}) - 2(\mathbf{x}^{(k)} + \alpha \mathbf{p})^T \mathbf{b} + \text{const}$$

par rapport à α . Plus précisément, comme la matrice A est symétrique, on a

$$\frac{1}{2} \frac{df}{d\alpha}(\mathbf{x}^{(k+1)}) = \mathbf{p}^T A (\mathbf{x}^{(k)} + \alpha \mathbf{p}) - \mathbf{p}^T \mathbf{b} = \alpha \mathbf{p}^T A \mathbf{p} - \mathbf{p}^T \underbrace{(\mathbf{b} - A \mathbf{x}^{(k)})}_{\mathbf{r}^{(k)}},$$

qui ne s'annule que pour

$$\alpha = \frac{\mathbf{p}^T \mathbf{r}^{(k)}}{\mathbf{p}^T A \mathbf{p}}.$$

Cette valeur de α est toujours bien définie car le dénominateur ne s'annule pas (pourquoi?). De plus, la dérivée seconde de $f(\mathbf{x}^{(k+1)})$, qui vaut $\mathbf{p}^T A \mathbf{p}$, est toujours strictement positive, ce qui implique qu'il s'agit effectivement d'un minimum local (qui est de plus global).

Notez par ailleurs que le choix $\alpha = 0$ donne $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)}$, et donc dans ce cas la solution approchée ne change pas lors de l'itération $k + 1$. Dès lors, si la valeur de α dans l'expression de $\mathbf{x}^{(k+1)}$ est celle qui minimise l'énergie, alors

$$f(\mathbf{x}^{(k+1)}) \leq f(\mathbf{x}^{(k)}).$$

Cette dernière inégalité indique que l'erreur $\mathbf{x} - \mathbf{x}^{(k)}$ sur la solution est toujours décroissante avec k (en norme énergie) quel que soit le choix de \mathbf{p} . En pratique cela est souvent suffisant pour obtenir une convergence de la suite des solutions approchées vers la solution exacte, même si formellement la décroissance de l'énergie ne garantit pas à elle seule la convergence de la méthode (pourquoi?).

L'algorithme générique qui décrit l'ensemble des méthodes basées sur la minimisation d'énergie est alors comme suit. L'utilisation de l'indice k pour la direction \mathbf{p} et l'amplitude α est faite pour rappeler qu'elles changent d'une itération à l'autre; néanmoins, en pratique le même espace mémoire peut être utilisé d'une itération à l'autre pour chacune de ces deux quantités, aussi bien que pour les vecteurs $\mathbf{x}^{(k)}$ et $\mathbf{r}^{(k)}$.

ALGORITHME (MINIMISATION D'ÉNERGIE)entrée : $A, \mathbf{b}, \mathbf{x}^{(0)}$ sortie : $\mathbf{x}^{(k+1)}$ $\mathbf{r}^{(0)} = \mathbf{b} - A\mathbf{x}^{(0)}$

répéter jusqu'à l'arrêt

 choisir $\mathbf{p}^{(k)} \neq \mathbf{0}$ calculer $\alpha_k = \frac{\mathbf{p}^{(k)T} \mathbf{r}^{(k)}}{\mathbf{p}^{(k)T} A \mathbf{p}^{(k)}}$ $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{p}^{(k)}$ $\mathbf{r}^{(k+1)} = \mathbf{r}^{(k)} - \alpha_k A \mathbf{p}^{(k)}$

Notez aussi que le calcul du résidu lors de la dernière étape de l'algorithme n'est pas effectué sur base de la définition (4.5). La récurrence

$$\mathbf{r}^{(k+1)} = \mathbf{r}^{(k)} - \alpha A \mathbf{p}$$

utilisée ici provient de la combinaison de la définition (4.5) (appliquée séparément pour $k+1$ et pour k) avec la récurrence (4.15). La raison derrière ce calcul inhabituel du résidu est que le produit $A\mathbf{p}$ qui intervient dans le calcul du dénominateur de α peut alors être réutilisé. Or, comme le produit matrice-vecteur est potentiellement plus coûteux ($\approx n^2 \text{ flops}$ avec symétrie ; moins pour matrices creuses) que les autres opérations, qui n'impliquent que des vecteurs (et dont le coût est $\mathcal{O}(n)$), cela permet de réduire le coût par itération de l'algorithme. Ce coût est d'ailleurs mesuré en comptant le nombre de produits de A avec un vecteur, ou *MatVec* ; l'algorithme dans sa formulation actuelle coûte donc 1 *MatVec* par itération !

4.3.3 Méthode du gradient

La méthode du gradient est une variante de la méthode de minimisation d'énergie qui utilise comme vecteur de direction \mathbf{p} la direction dans laquelle la variation de l'énergie est localement la plus forte ; c'est-à-dire la direction de la plus grande pente de f en $\mathbf{x}^{(k)}$.

En pratique, cela revient à choisir comme vecteur de direction le vecteur résidu. En effet, la direction de la plus grande pente de f est aussi celle du gradient de cette fonction, et donc \mathbf{p} vaut $\nabla f(\mathbf{x}^{(k)})$. Ainsi, en utilisant les règles de dérivation introduites au Chapitre 3 (voir Propriété 3) pour une matrice A symétrique, on a

$$\mathbf{p} = \nabla f(\mathbf{x}^{(k)}) = 2(A\mathbf{x}^{(k)} - \mathbf{b}) = -2\mathbf{r}^{(k)}.$$

Le facteur -2 dans \mathbf{p} importe peu car l'amplitude de la correction est fixée par après avec le choix du paramètre α .

L'algorithme de la méthode du gradient est alors donné ci-dessous. Comme annoncé, il ne diffère de l'algorithme générique pour les méthodes de minimisation d'énergie que par un choix spécifique du vecteur de direction \mathbf{p} . En particulier, le produit $A\mathbf{p}$ s'effectue

ici aussi une fois par itération, et donc le coût de l'algorithme est aussi de 1 *MatVec* par itération.

ALGORITHME (DU GRADIENT)

entrée : $A, \mathbf{b}, \mathbf{x}^{(0)}$

sortie : $\mathbf{x}^{(k+1)}$

$\mathbf{r}^{(0)} = \mathbf{b} - A\mathbf{x}^{(0)}$

répéter jusqu'à l'arrêt

$\mathbf{p}^{(k)} = \mathbf{r}^{(k)}$

calculer $\alpha_k = \frac{\mathbf{p}^{(k)T} \mathbf{r}^{(k)}}{\mathbf{p}^{(k)T} A \mathbf{p}^{(k)}}$

$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{p}^{(k)}$

$\mathbf{r}^{(k+1)} = \mathbf{r}^{(k)} - \alpha_k A \mathbf{p}^{(k)}$

Notez que la direction $\mathbf{p} = \mathbf{r}^{(k)}$ choisie par la méthode du gradient est toujours non nulle. En effet, dans le cas contraire cela impliquerait un résidu nul, et que donc la solution approchée $\mathbf{x}^{(k)}$ est aussi la solution exacte ; or, dans ce cas la méthode devrait logiquement s'arrêter avant de calculer la direction en question. Par conséquent, la direction \mathbf{p} est toujours non nulle et le dénominateur de l'expression qui définit α est également non nul.

4.3.4 Méthode du gradient conjugué

La méthode du gradient conjugué est une variante de la méthode de minimisation d'énergie qui choisit comme vecteur de direction une combinaison linéaire du résidu et du vecteur de direction à l'itération précédente. La direction à l'itération $k + 1$ vaut donc

$$\mathbf{p}^{(k+1)} = \mathbf{r}^{(k+1)} - \beta \mathbf{p}^{(k)}. \quad (4.16)$$

On choisit le paramètre β de sorte à minimiser l'énergie de la nouvelle solution approchée ; cela revient à satisfaire (voir Annexe 4.A) la condition de conjugaison suivante

$$\mathbf{p}^{(k)T} A \mathbf{p}^{(k+1)} = 0, \quad (4.17)$$

qui implique à son tour, en combinant (4.17) et (4.16),

$$\beta = \frac{\mathbf{p}^{(k)T} A \mathbf{r}^{(k+1)}}{\mathbf{p}^{(k)T} A \mathbf{p}^{(k)}}.$$

L'algorithme de la méthode du gradient conjugué est alors comme suit. Comme avant, la multiplication $A\mathbf{p}$ ne se fait qu'une fois par itération, et donc le coût de l'algorithme reste 1 *MatVec*.

ALGORITHME (DU GRADIENT CONJUGUÉ)entrée : $A, \mathbf{b}, \mathbf{x}^{(0)}$ sortie : $\mathbf{x}^{(k+1)}$ $\mathbf{r}^{(0)} = \mathbf{b} - A\mathbf{x}^{(0)} ; \quad \mathbf{p}^{(0)} = \mathbf{r}^{(0)}$

répéter jusqu'à l'arrêt

$$\text{calculer } \alpha_k = \frac{\mathbf{p}^{(k)T} \mathbf{r}^{(k)}}{\mathbf{p}^{(k)T} A \mathbf{p}^{(k)}}$$

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{p}^{(k)}$$

$$\mathbf{r}^{(k+1)} = \mathbf{r}^{(k)} - \alpha_k A \mathbf{p}^{(k)}$$

$$\text{calculer } \beta_k = \frac{(A \mathbf{p}^{(k)})^T \mathbf{r}^{(k+1)}}{\mathbf{p}^{(k)T} A \mathbf{p}^{(k)}}$$

$$\mathbf{p}^{(k+1)} = \mathbf{r}^{(k+1)} - \beta_k \mathbf{p}^{(k)}$$

Notons (sans faire de démonstration) que, comme dans le cas précédent, la direction $\mathbf{p}^{(k)}$ ne s'annule que quand la solution approchée $\mathbf{x}^{(k)}$ est aussi la solution exacte ; or, dans ce cas la méthode devrait logiquement s'arrêter avant de calculer la direction en question. Par conséquent, le dénominateur des expressions définissant α_k et β_k est également non nul.

Par ailleurs, notez que la méthode du gradient conjugué est toujours meilleure que la méthode du gradient, du moins en norme énergie. En effet, en plus de choisir l'amplitude du vecteur de direction de sorte à minimiser l'énergie, elle choisit, via le paramètre β , la combinaison linéaire du résidu et de la direction précédente qui minimise l'énergie, alors que la méthode du gradient équivaut à toujours utiliser $\beta = 0$.

4.3.5 Préconditionnement

En pratique les méthodes précédentes sont surtout efficaces si elles sont utilisées avec un préconditionneur de bonne qualité. Dans le contexte des méthodes de minimisation d'énergie un *préconditionneur* est une matrice *symétrique définie positive* $B \approx A$ avec B plus facile à «inverser» que A . Le fait d'utiliser un préconditionneur équivaut à résoudre, au lieu du système initial (4.1), le système modifié

$$B^{-1}A\mathbf{x} = B^{-1}\mathbf{b}$$

s'il s'agit d'un *préconditionnement à gauche* ou

$$AB^{-1}\mathbf{y} = \mathbf{b}$$

avec $\mathbf{x} = B^{-1}\mathbf{y}$ s'il s'agit d'un *préconditionnement à droite*. Le but de la démarche est de diminuer drastiquement le nombre d'itérations en résolvant le système preconditionné au lieu du système initial (4.1).

Un bon préconditionneur peut être choisi d'une manière similaire à celle de la matrice B dans les méthodes stationnaires, raison pour laquelle on utilise la même notation.

La seule exigence supplémentaire est d'avoir une matrice B qui est symétrique définie positive. Les candidats les plus simples sont la méthode de Jacobi ($B = B_J$, avec B_J définie par (4.9)), ou encore la méthode de Gauss-Seidel symétrique ($B = B_{GS}^T B_J^{-1} B_{GS}$, avec B_J et B_{GS} définies, respectivement, par (4.9) et (4.12)) ; la méthode de Gauss-Seidel ne convient par contre pas car la matrice B_{GS} n'est en général pas symétrique.

4.4 Contrôle de convergence et critères d'arrêt

On arrête une méthode itérative quand la qualité de la solution approchée est jugée satisfaisante. Cette qualité est souvent mesurée en évaluant (avec la norme $\|\cdot\|$ de votre choix) l'amplitude (disons absolue) de l'erreur commise sur la solution exacte, à savoir

$$\|\mathbf{x}^{(k)} - \mathbf{x}\|.$$

Or, comme la solution exacte \mathbf{x} n'est pas connue, on procède à l'évaluation d'une borne basée sur la norme du résidu. Pour relier cette dernière à l'erreur sur la solution, on utilise l'égalité suivante

$$\mathbf{x} - \mathbf{x}^{(k)} = A^{-1} \mathbf{r}^{(k)},$$

qui est une conséquence de (4.3)–(4.5). De plus, quelle que soit la norme vectorielle $\|\cdot\|$ et la norme matricielle induite par celle-ci, on a

$$\|A^{-1} \mathbf{r}^{(k)}\| \leq \|A^{-1}\| \|\mathbf{r}^{(k)}\| \quad \text{et} \quad \|\mathbf{r}^{(k)}\| = \|AA^{-1} \mathbf{r}^{(k)}\| \leq \|A\| \|A^{-1} \mathbf{r}^{(k)}\|,$$

et finalement

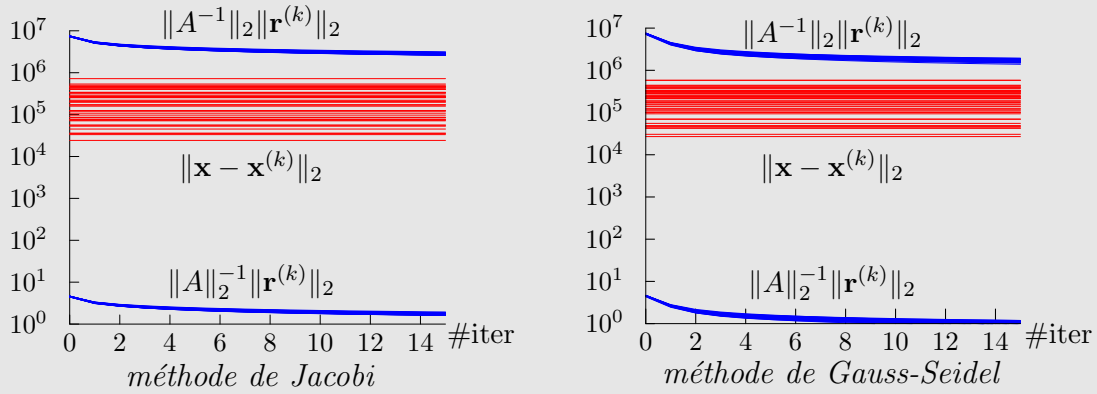
$$\|A\|^{-1} \|\mathbf{r}^{(k)}\| \leq \|\mathbf{x}^{(k)} - \mathbf{x}\| \leq \|A^{-1}\| \|\mathbf{r}^{(k)}\|. \quad (4.18)$$

Cette dernière relation justifie non seulement l'utilisation d'un critère d'arrêt basé sur la norme du résidu, mais explique aussi l'emploi de la norme du résidu pour le contrôle de la convergence d'une méthode itérative.

Pour ce qui est de l'erreur relative, une relation similaire, généralement plus pessimiste, est comme suit (elle est donnée ici sans démonstration)

$$\kappa(A)^{-1} \frac{\|\mathbf{r}^{(k)}\|}{\|\mathbf{b}\|} \leq \frac{\|\mathbf{x}^{(k)} - \mathbf{x}\|}{\|\mathbf{x}\|} \leq \kappa(A) \frac{\|\mathbf{r}^{(k)}\|}{\|\mathbf{b}\|}.$$

EXEMPLE 28. Soit la matrice tridiagonale $A = (a_{ij})$ de dimensions $10^3 \times 10^3$ telle que considérée dans l'Exemple 4 du Chapitre M. On effectue la résolution des systèmes dont la matrice est A par les méthodes de Jacobi et Gauss-Seidel pour 50 membres de droite \mathbf{b} générés de manière aléatoire (via $\mathbf{b} = \text{ones}(1000,1) - 2 * \text{rand}(1000,1)$). Les figures suivantes montrent l'erreur absolue sur la solution et les deux bornes de (4.18) (calculées avec la norme euclidienne) en fonction de l'indice d'itération.



On constate que la borne supérieure, qui fournit une estimation conservative de l'erreur commise, est relativement proche de l'erreur réelle.

Pour ce qui est plus spécifiquement des critères d'arrêt, deux cas de figure sont possibles. Tout d'abord, on peut arrêter une méthode lorsqu'elle a atteint son objectif – le critère correspondant est marqué dans la liste qui suit avec un \oplus . D'autres critères permettent d'arrêter une méthode même si celle-ci n'a pas accompli son travail – on utilise alors un \ominus . Les quelques critères communément utilisés pour les méthodes itératives sont alors les suivants.

- \oplus *La norme de résidu.* L'arrêt intervient lorsque la norme du résidu (4.5) en valeur absolue satisfait

$$\|\mathbf{r}^{(k)}\| \leq \varepsilon_a;$$

alternativement, on peut utiliser la réduction de cette norme *relativement à celle du second membre* \mathbf{b} :

$$\|\mathbf{r}^{(k)}\| \leq \varepsilon_r \|\mathbf{b}\|.$$

Les paramètres ε_a , ε_r sont typiquement fournis par l'utilisateur. Il faut par contre être attentif en utilisant un critère de type $\|\mathbf{r}^{(k)}\| \leq \varepsilon_r \|\mathbf{r}^{(0)}\|$ car la norme $\|\mathbf{r}^{(0)}\|$ du résidu initial dépend du choix de l'approximation initiale $\mathbf{x}^{(0)}$.

- \ominus L'arrêt intervient si le *nombre maximal d'itérations* est dépassé.
- \ominus Pour les méthodes stationnaires on peut montrer que la vitesse de convergence ne peut que se détériorer. On peut dès lors arrêter une méthode stationnaire si la *décroissance du résidu est lente ou inexistante*.

Annexe 4.A Méthode du gradient conjugué : direction optimale

On montre ici que la direction $\mathbf{p}^{(k+1)}$ qui correspond au paramètre β_k optimal doit satisfaire (4.17). Pour cela, on considère la minimisation de

$$f(\mathbf{x}^{(k+2)}) = \mathbf{x}^{(k+2)T} A \mathbf{x}^{(k+2)} - 2 \mathbf{x}^{(k+2)T} \mathbf{b} + \text{const}$$

avec

$$\mathbf{x}^{(k+2)} = \mathbf{x}^{(k+1)} + \alpha_{k+1} \mathbf{p}^{(k+1)} \quad (4.A.1)$$

et

$$\mathbf{p}^{(k+1)} = \mathbf{r}^{(k+1)} - \beta_k \mathbf{p}^{(k)}, \quad (4.A.2)$$

sachant que

$$\mathbf{r}^{(k+2)} = \mathbf{r}^{(k+1)} - \alpha_{k+1} A \mathbf{p}^{(k+1)} \quad (4.A.3)$$

et $\mathbf{p}^{(0)} = \mathbf{r}^{(0)}$.

Le raisonnement général sur la minimisation d'énergie de la sous-section 4.3.2 reste valable et donc

$$\alpha_{k+1} = \frac{\mathbf{p}^{(k+1)T} \mathbf{r}^{(k+1)}}{\mathbf{p}^{(k+1)T} A \mathbf{p}^{(k+1)}}.$$

En multipliant à gauche les deux membres de (4.A.3) par $\mathbf{p}^{(k+1)T}$, on montre que ce choix de α_{k+1} implique

$$\mathbf{p}^{(k+1)T} \mathbf{r}^{(k+2)} = \mathbf{p}^{(k+1)T} \mathbf{r}^{(k+1)} - \alpha_{k+1} \mathbf{p}^{(k+1)T} A \mathbf{p}^{(k+1)} = 0. \quad (4.A.4)$$

Les vecteurs $\mathbf{p}^{(k+1)}$ et $\mathbf{r}^{(k+2)}$ sont donc orthogonaux. Par ailleurs, en combinant (4.A.1) et (4.A.2), on a

$$\mathbf{x}^{(k+2)} = \mathbf{x}^{(k+1)} + \alpha_{k+1} \mathbf{r}^{(k+1)} - \alpha_{k+1} \beta_k \mathbf{p}^{(k)}$$

et, après différentiation par rapport à β_k ,

$$\frac{d\mathbf{x}^{(k+2)}}{d\beta_k} = -\alpha_{k+1} \mathbf{p}^{(k)}.$$

On en déduit

$$\begin{aligned} \frac{df}{d\beta_k}(\mathbf{x}^{(k+2)}) &= \frac{d\mathbf{x}^{(k+2)T}}{d\beta_k} A \mathbf{x}^{(k+2)} + \mathbf{x}^{(k+2)T} A \frac{d\mathbf{x}^{(k+2)}}{d\beta_k} - 2 \frac{d\mathbf{x}^{(k+2)T}}{d\beta_k} \mathbf{b} \\ &= -2 \frac{d\mathbf{x}^{(k+2)T}}{d\beta_k} A (\mathbf{b} - A \mathbf{x}^{(k+2)}) \\ &= 2\alpha_{k+1} \mathbf{p}^{(k)T} \mathbf{r}^{(k+2)} \\ &= 2\alpha_{k+1} \left(\mathbf{p}^{(k)T} \mathbf{r}^{(k+1)} - \alpha_{k+1} \mathbf{p}^{(k)T} A \mathbf{p}^{(k+1)} \right). \end{aligned}$$

Le premier terme s'annule en appliquant l'égalité (4.A.4) obtenue pour l'itération précédente. L'annulation de la dérivée par rapport à β_k implique que soit $\alpha_{k+1} = 0$ (on peut montrer que cela n'arrive que si $\mathbf{r}^{(k+1)} = \mathbf{0}$, et donc, si $\mathbf{x}^{(k+1)}$ est la solution exacte!), soit

$$\mathbf{p}^{(k)T} A \mathbf{p}^{(k+1)} = 0,$$

ce qui est bien la condition recherchée.

Equations et systèmes d'équations non linéaires

Le présent chapitre porte sur la résolution numérique d'équations et systèmes d'équations non linéaires. On commence par introduire les méthodes pour la résolution d'une équation non linéaire à une inconnue. Bien que ce cadre ne soit pas général, il permet d'illustrer une partie des phénomènes qu'on rencontre généralement dans la résolution des systèmes d'équations non linéaires. De plus, certaines des méthodes introduites pour une équation à une inconnue se généralisent au cas des systèmes d'équations non linéaires ; ce cas plus général est abordé dans la deuxième partie du chapitre. Finalement, comme les méthodes considérées sont itératives, la dernière partie est consacrée aux critères d'arrêt.

5.1 Problème

On entend par *équation non linéaire* d'une variable réelle toute équation de la forme

$$f(x) = 0, \quad (5.1)$$

où $f(x) : I \subset \mathbb{R} \mapsto \mathbb{R}$ est une fonction. Dans ce qui suit nous nous limitons aux fonctions continues avec un intervalle I comme domaine de définition ; le fait que I soit un intervalle garantit que si $a, b \in I$, alors tout point entre a et b appartient à I .

Plus généralement, on entend par *système d'équations non linéaires* de n équations à n inconnues (ou plus simplement *système non linéaire*) un système de la forme

$$\begin{cases} f_1(x_1, \dots, x_n) = 0 \\ f_2(x_1, \dots, x_n) = 0 \\ \dots \\ f_n(x_1, \dots, x_n) = 0 \end{cases}$$

avec $f_i : I \subset \mathbb{R}^n \mapsto \mathbb{R}$, $i = 1, \dots, n$, des fonctions qu'on suppose continues. En utilisant les notations vectorielles cela donne

$$\mathbf{f}(\mathbf{x}) = \mathbf{0}, \quad (5.2)$$

où la fonction vectorielle

$$\mathbf{f} : I \subset \mathbb{R}^n \mapsto \mathbb{R}^n : \underbrace{(x_1, \dots, x_n)}_{\mathbf{x}} \mapsto \begin{pmatrix} f_1(x_1, \dots, x_n) \\ f_2(x_1, \dots, x_n) \\ \dots \\ f_n(x_1, \dots, x_n) \end{pmatrix}$$

est donc aussi continue.

On s'intéresse ici à la recherche d'une ou de plusieurs *solutions* (*zéros*, *racines*) de l'équation non linéaire (5.1), c'est-à-dire à la recherche d'un ou plusieurs réels x qui satisfont $f(x) = 0$; dans le cas d'un système (5.2), on recherche un ou plusieurs vecteurs \mathbf{x} qui satisfont $\mathbf{f}(\mathbf{x}) = \mathbf{0}$. Comme en pratique on ne détermine qu'une approximation d'une des solutions de l'équation, on réserve les notations x et \mathbf{x} à la solution exacte vers laquelle on converge (appelée plus bas *la* solution), et on utilise des notations indexées (par exemple, x_k ou \mathbf{x}_k) pour une solution approchée ; les notations x et \mathbf{x} sont aussi utilisées comme des variables génériques. Notons en particulier que la confusion entre l'approximation x_k de la solution scalaire et le k -ème composante du vecteur \mathbf{x} n'a pas lieu d'être car les composantes individuelles d'un vecteur ne sont pas utilisées dans la suite du chapitre.

Notons que le terme *non linéaire* n'est pas utilisé ici dans un sens exclusif ; les méthodes qui suivent s'appliquent également dans le cas particulier d'équations et systèmes d'équations linéaires (c'est-à-dire pour $\mathbf{f}(\mathbf{x}) = A\mathbf{x} - \mathbf{b}$), sans pour autant exploiter explicitement leur linéarité.

Contrairement aux systèmes linéaires réguliers, on n'a a priori aucune garantie que l'équation (5.1) ou le système d'équations non linéaires (5.2) possède une solution, mais il peut en posséder une, plusieurs ou même une infinité. Dans certains cas on peut néanmoins estimer le nombre de solutions d'une équation donnée ; par exemple, dans le cas des *équations algébriques* à coefficients réels $\sum_{k=0}^n a_k x^k = 0$ de degré n (c'est-à-dire telle que $a_n \neq 0$) l'équation possède au plus n racines réelles ; elle possède au moins une racine réelle si n est impair.

5.2 Méthodes d'encadrement

5.2.1 Généralités

Les deux méthodes suivantes sont des *méthodes d'encadrement*. Elles s'appliquent à une fonction f qui est continue sur un intervalle $[a, b]$ (avec $a < b$) et qui satisfait la condition suivante

$$f(a)f(b) < 0, \quad (5.3)$$

appelée *condition d'encadrement*. Le théorème de la valeur intermédiaire garantit alors qu'il existe (au moins) une solution à l'intérieur de cet intervalle. Les méthodes d'encadrement subdivisent à l'itération k un intervalle $[a, b]$ qui satisfait la condition d'encadrement en deux intervalles plus petits $[a, x_k]$ et $[x_k, b]$; si x_k n'est pas une solution de (5.2), au moins un de ces intervalles satisfait la condition d'encadrement (5.3) et devient le nouvel

intervalle $[a, b]$. La procédure est répétée jusqu'à ce que la solution soit localisée avec suffisamment de précision. Les différentes méthodes d'encadrement diffèrent essentiellement par leur choix du point de subdivision x_k . L'algorithme générique pour ces méthodes est comme suit :

ALGORITHME (MÉTHODE D'ENCADREMENT) :
entrées : f, a, b % qui satisfait (5.3)
sortie : x_k
répéter pour $k = 1, 2, \dots$, jusqu'à l'arrêt
 choisir $x_k \in]a, b[$
 si $f(a)f(x_k) < 0$ alors $b := x_k$
 si $f(b)f(x_k) < 0$ alors $a := x_k$
 si $f(x_k) = 0$ alors x_k est une solution

5.2.2 Méthode de dichotomie

La méthode de *dichotomie* est une méthode d'encadrement qui subdivise l'intervalle $[a, b]$ en deux sous-intervalles $[a, x_k]$ et $[x_k, b]$ de longueur égale, avec par conséquent

$$x_k = (a + b)/2.$$

Ainsi, la longueur de l'intervalle qui encadre (au moins) une solution diminue de moitié à chaque itération ; l'algorithme peut s'arrêter quand la longueur $|a - b|$ devient suffisamment petite, sans risque de stagnation.

EXEMPLE 29.

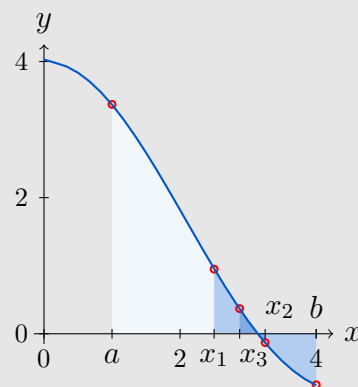
On cherche une solution de l'équation non linéaire $f(x) = 0$ avec

$$f(x) = 4 \sin(x)/x$$

sur l'intervalle $[1, 4]$. Comme f est continue et

$$f(1)f(4) < 0, \quad (5.4)$$

on sait qu'au moins une solution se trouve sur l'intervalle ; de plus, (5.4) indique que la méthode de dichotomie peut être appliquée avec $a = 1$ et $b = 4$. La première itération consiste à choisir $x_1 = (a + b)/2 = 2.5$; comme $f(x_1)f(4) < 0$, au moins une solution se trouve sur $[x_1, 4]$, et la démarche peut être appliquée de nouveau à cet intervalle. Cette première itération ainsi que les deux itérations suivantes sont schématisées à la figure de droite.



5.2.3 Méthode de la fausse position

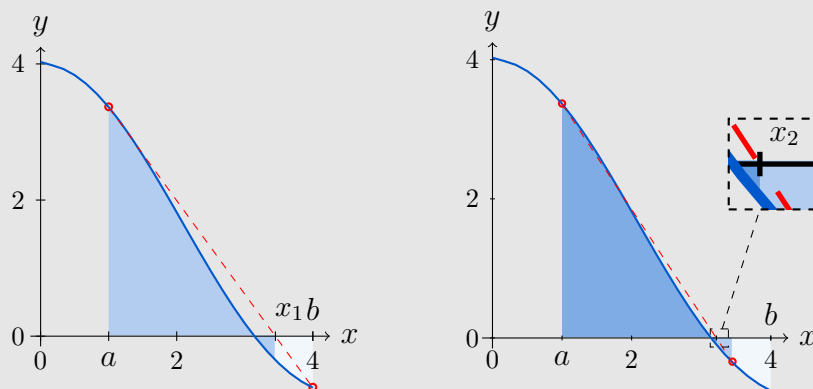
La méthode de la *fausse position* est une méthode d'encadrement qui utilise comme point de subdivision x_k le point où la droite qui relie $(a, f(a))$ et $(b, f(b))$ intersecte l'axe des abscisses ; en d'autres termes :

$$x_k = a - f(a) \frac{b - a}{f(b) - f(a)}.$$

Si la condition d'encadrement (5.3) est satisfaite, l'approximation x_k ainsi obtenue se trouve toujours entre a et b .

Notons que si on interprète x_k comme une approximation de la solution, l'approximation fournie par la méthode de la fausse position est généralement meilleure que le milieu de l'intervalle $[a, b]$. Par contre, il ne permet pas nécessairement de diviser la longueur de $[a, b]$ par un facteur donné ; le critère d'arrêt ne peut donc pas être basé sur la longueur de cet intervalle.

EXEMPLE 30. Les deux premières itérations de la méthode de la fausse position sont illustrées dans les figures suivantes. Comme dans l'exemple précédent, on considère l'équation $f(x) = 0$ avec $f(x) = 4\sin(x)/x$ sur l'intervalle $[a, b] = [1, 4]$. On notera que l'approximation x_2 obtenue durant la deuxième itération est déjà tellement proche de la solution (qui ici vaut π) qu'un agrandissement est nécessaire pour les distinguer. Par contre, la longueur de l'intervalle d'encadrement évolue peu au cours des deux premières itérations ; elle changera peu également durant les itérations suivantes (pourquoi ?).



5.3 Méthodes de Newton

5.3.1 Méthode de Newton-Raphson

La méthode de *Newton-Raphson* (aussi appelée *méthode de Newton*) consiste à approcher la fonction f dans l'équation non linéaire (5.1) par son développement de Taylor d'ordre 1 ; la solution de l'équation approchée donne ainsi une approximation pour une

des solutions de l'équation de départ. Plus précisément, si x_k est une approximation à l'itération k , l'approximation x_{k+1} à l'itération $k+1$ est la solution de

$$f(x_k) + f'(x_k)(x - x_k) = 0,$$

et donc, si $f'(x_k)$ est non nul,

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}. \quad (5.5)$$

Notons que la méthode requiert la dérivée f' de la fonction f , qui doit donc être dérivable ; elle requiert aussi une approximation initiale x_0 de la solution recherchée.

ALGORITHME (MÉTHODE DE NEWTON-RAPHSON) :

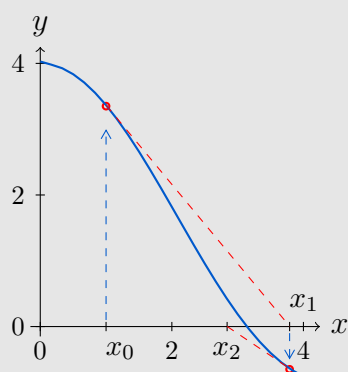
entrées : f, f', x_0

sortie : x_{k+1}

répéter pour $k = 0, 1, 2, \dots$, jusqu'à l'arrêt

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} \quad \% \text{ si } f(x_k), f'(x_k) \text{ existent et } f'(x_k) \neq 0$$

EXEMPLE 31. Les deux premières itérations de la méthode de Newton sont illustrées à la figure suivante. La table à droite de cette figure contient quant à elle les 6 premières approximations de la solution (qui vaut ici π) ; en particulier, la deuxième colonne donne l'approximation à l'itération k pour laquelle le dernier chiffre significatif affiché (en rouge ou gris) est le premier après le point à ne pas être correct.



k	x_k	$ x_k - x $
0	1.0	2.1
1	3.7	$6.5 \cdot 10^{-1}$
2	2.8	$3.0 \cdot 10^{-1}$
3	3.12	$2.2 \cdot 10^{-2}$
4	3.1414	$1.4 \cdot 10^{-4}$
5	3.14159264	$6.6 \cdot 10^{-9}$

On notera que la convergence s'améliore au fur et à mesure qu'on s'approche de la solution recherchée. En substance, le nombre de chiffres significatifs corrects est doublé d'une itération à la suivante : c'est le phénomène de convergence quadratique.

Avant d'expliquer l'observation de l'exemple précédent, clarifions le vocabulaire utilisé. La convergence d'une suite x_k , $k = 0, 1, \dots$, vers x est dite d'ordre p s'il existe une

constante C telle que

$$|x - x_{k+1}| \leq C|x - x_k|^p.$$

Plus particulièrement, si $p = 1$ et $C < 1$ on parle de *convergence linéaire* ; si $p = 2$ on parle de *convergence quadratique*. On parle de *convergence superlinéaire* (appelée encore *convergence surlinéaire*) si

$$\lim_{k \rightarrow \infty} \frac{|x - x_{k+1}|}{|x - x_k|} = 0.$$

Toute convergence d'ordre $p > 1$ est donc superlinéaire.

Le théorème suivant explique pourquoi et dans quel cas la méthode de Newton possède une convergence quadratique vers le réel x qui est une des solutions de $f(x) = 0$. Il spécifie également les conditions requises pour cette convergence :

- une fonction f suffisamment régulière dont la dérivée ne s'annule pas en x ;
- une proximité de la solution x : la convergence quadratique n'a lieu que dans un voisinage de celle-ci.

THÉORÈME 2. *Soit x une solution de $f(x) = 0$ pour une fonction $f \in C^2$ telle que $f'(x) \neq 0$. Alors il existe un intervalle $V(x)$ fermé et centré en x tel que $x_k \in V(x)$ implique*

$$|x - x_{k+1}| \leq C|x - x_k|^2$$

pour x_k et x_{k+1} qui satisfont (5.5), c'est-à-dire pour deux approximations successives obtenues avec la méthode de Newton-Raphson.

DÉMONSTRATION. Le développement de Taylor de f d'ordre 1 par rapport à x_k avec le reste de Lagrange donne

$$\underbrace{f(x)}_{=0 \text{ (car solution)}} = \underbrace{f(x_k) + f'(x_k)(x - x_k)}_{f'(x_k)(x - x_{k+1}) \text{ (par (5.5))}} + \frac{1}{2}f''(c)(x - x_k)^2$$

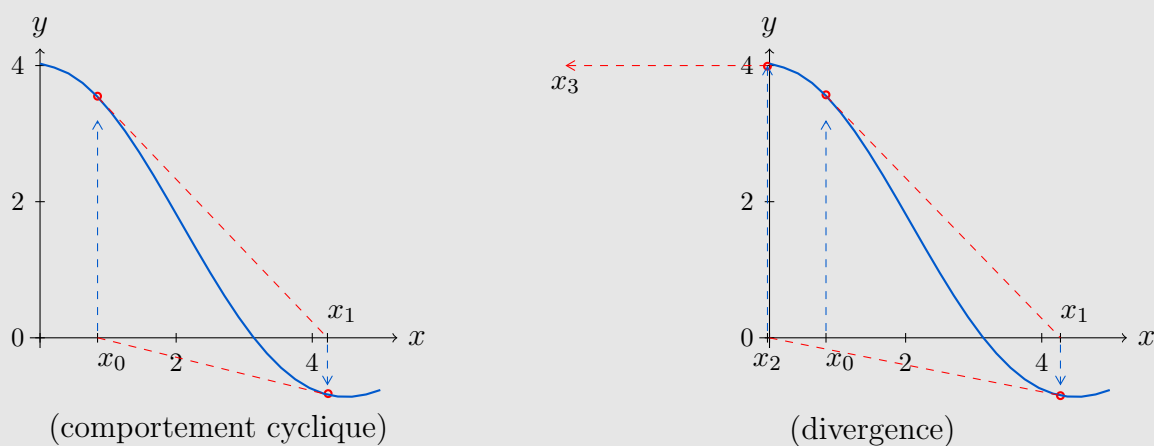
pour un c entre x et x_k . Soit $V(x)$ un intervalle fermé et centré en x tel que f' ne s'annule pas sur cet intervalle (ce qui est toujours possible car f' est continue et $f'(x) \neq 0$). Si $x_k \in V(x)$, alors

$$|x - x_{k+1}| = \left| \frac{f''(c)}{2f'(x_k)}(x - x_k)^2 \right| = \underbrace{\left| \frac{f''(c)}{2f'(x_k)} \right|}_{\leq C} |(x - x_k)^2|.$$

Le premier facteur dans le membre le plus à droite est bien borné ; en effet, il correspond à une fonction continue de deux variables c et x_k (car quotient de fonctions continues dont le dénominateur ne s'annule pas sur $V(x)$) définies sur un ensemble compact $V(x) \times V(x)$ (car $x_k, c \in V(x)$), son image est donc un ensemble compact, et par conséquent aussi borné. ■

Le théorème précédent ne garantit pas pour autant une convergence globale. L'exemple suivant illustre le fait que la méthode de Newton peut ne pas converger, voire même diverger.

EXEMPLE 31. (FIN) On revient ici vers l'Exemple 31. On éloigne progressivement la solution approchée x_0 de la solution exacte $x = \pi$. En prenant $x_0 \approx 0.845$ (au lieu de $x_0 = 1$ précédemment) on obtient la situation illustrée à la figure suivante (partie gauche). On observe que la méthode de Newton manifeste un comportement cyclique ; elle ne converge donc vers aucune solution. Si on s'éloigne davantage de x en choisissant $x_0 \approx 0.83$, on obtient la partie droite de la figure suivante ; la méthode diverge dans ce cas, dans la mesure où la solution approchée x_3 (si elle existe) est assez loin des solutions approchées précédentes.



5.3.2 Newton avec recherche linéaire

Plusieurs approches permettent de combiner la convergence superlinéaire locale (comme celle observée avec la méthode de Newton) avec des propriétés de convergence globale. La méthode de *Newton avec recherche linéaire* en est un exemple. Elle s'obtient en multipliant l'incrément dans la méthode de Newton par un facteur d'amortissement α_k ($\alpha_k > 0$) :

$$x_{k+1} = x_k - \alpha_k \frac{f(x_k)}{f'(x_k)}. \quad (5.6)$$

Le facteur α_k est choisi de manière à satisfaire (parfois avec une marge)

$$|f(x_{k+1})| < |f(x_k)| \quad (5.7)$$

et à éviter ainsi une divergence ou un comportement cyclique. Notons que l'inégalité stricte dans l'expression précédente n'est possible que si $f(x_k) \neq 0$; si $f(x_k) = 0$, alors x_k est une racine, et l'algorithme s'arrête à l'itération k .

Un α_k qui permet de satisfaire (5.7) existe-t-il toujours ? Si $f \in C^1$ et $f'(x_k) \neq 0$ alors la réponse est affirmative, comme le montre le théorème suivant.

THÉORÈME 3. Si $f \in C^1$, et x_k, x_{k+1} deux réels satisfaisant $f(x_k) \neq 0, f'(x_k) \neq 0$ et (5.6) (c'est-à-dire $x_{k+1} = x_k - \alpha_k f(x_k)/f'(x_k)$), alors l'inégalité

$$|f(x_{k+1})| < |f(x_k)|$$

est d'application pour un $\alpha_k > 0$ suffisamment petit.

DÉMONSTRATION. Le développement de Taylor de f d'ordre 0 par rapport à x_k avec le reste de Lagrange donne

$$f(x_{k+1}) = f(x_k) + f'(c) \underbrace{(x_{k+1} - x_k)}_{-\alpha_k f(x_k)/f'(x_k)} = f(x_k) \left(1 - \alpha_k \frac{f'(c)}{f'(x_k)} \right)$$

pour un c entre x_k et x_{k+1} . Par conséquent, comme $f(x_k) \neq 0$, l'inégalité $|f(x_{k+1})| < |f(x_k)|$ est satisfaite si

$$0 < \alpha_k \frac{f'(c)}{f'(x_k)} < 2.$$

D'une part, si α_k est suffisamment petit, (5.6) implique que x_k et x_{k+1} sont suffisamment proches. Comme f' est continue et $f'(x_k) \neq 0$, cela signifie que pour α_k suffisamment petit on a

$$0 < f'(c)/f'(x_k) < C \quad \text{pour tout } c \in [x_k, x_{k+1}].$$

d'où l'inégalité de gauche puisque α_k est positif. D'autre part, l'inégalité de droite s'obtient aussi en choisissant un α_k suffisamment petit, à savoir $\alpha_k < 2C^{-1}$. ■

L'algorithme de *Newton avec recherche linéaire* est alors comme suit. Notons qu'il détermine un α_k suffisamment petit en partant de $\alpha_k = 1$ (ce qui correspond à une itération de Newton-Raphson) et en divisant ce dernier par 2 tant que l'inégalité $|f(x_{k+1})| < |f(x_k)|$ n'est pas satisfaite.

ALGORITHME (NEWTON AVEC RECHERCHE LINÉAIRE) :

entrées : f, f', x_0

sortie : x_{k+1}

répéter pour $k = 0, 1, 2, \dots$, jusqu'à l'arrêt

$$p := \frac{f(x_k)}{f'(x_k)} \quad \% \text{ si } f(x_k), f'(x_k) \text{ existent et } f'(x_k) \neq 0$$

$$\alpha_k = 1$$

(a) répéter jusqu'à l'arrêt % recherche bon α_k

$$x_{k+1} = x_k - \alpha_k p$$

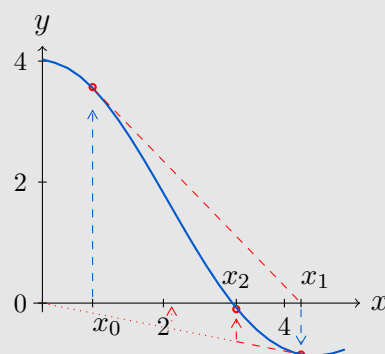
$$\text{si } |f(x_{k+1})| < |f(x_k)|$$

arrêter de répéter (a)

$$\alpha_k := \alpha_k / 2$$

EXEMPLE 32.

Les deux premières itérations de la méthode de Newton avec recherche linéaire sont illustrées à la figure de droite. On reprend ici l'Exemple 31 (fin) avec l'approximation initiale pour laquelle la méthode de Newton diverge. La première itération est celle de la méthode de Newton-Raphson (avec $\alpha_k = 1$ car $|f(x_1)| < |f(x_0)|$), alors que lors de la deuxième itération on divise $\alpha_k = 1$ par 4 pour obtenir une diminution de la norme de f .



5.4 Méthodes pour systèmes non linéaires

Notons que toutes les méthodes vues jusqu'à présent ne se généralisent pas au cas des systèmes d'équations non linéaires. En particulier, il n'existe pas de version vectorielle du théorème de la valeur intermédiaire, et donc les méthodes d'encadrement ne peuvent pas être étendues à des systèmes. En revanche, la généralisation des méthodes de Newton est directe.

5.4.1 Newton-Raphson pour systèmes non linéaires

La dérivation de la méthode de Newton-Raphson suit la même logique que dans la section précédente. Elle revient à approcher la fonction vectorielle \mathbf{f} dans (5.2) par son développement de Taylor d'ordre 1 et d'utiliser comme approximation de la solution la solution exacte de l'équation approchée. En d'autres termes, si \mathbf{x}_k est une approximation de la solution à l'itération k , l'approximation \mathbf{x}_{k+1} à l'itération $k + 1$ est la solution de l'équation suivante

$$\mathbf{f}(\mathbf{x}_k) + \mathbf{f}'(\mathbf{x}_k)(\mathbf{x} - \mathbf{x}_k) = 0,$$

où $\mathbf{f}'(\mathbf{x}) = \left(\frac{\partial f_i}{\partial x_j}(\mathbf{x}) \right)_{i,j=1,\dots,n}$ est la matrice jacobienne; et donc, si $\mathbf{f}'(\mathbf{x}_k)$ est inversible,

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \mathbf{f}'(\mathbf{x}_k)^{-1} \mathbf{f}(\mathbf{x}_k).$$

Contrairement au cas scalaire, ici l'ordre des facteurs dans un produit est important car il s'agit d'un produit matrice-vecteur.

L'algorithme suivant reprend la méthode de Newton pour les systèmes tout en spécifiant que l'opération $\mathbf{f}'(\mathbf{x}_k)^{-1} \mathbf{f}(\mathbf{x}_k)$ correspond à la résolution du système $\mathbf{f}'(\mathbf{x}_k) \mathbf{p} = \mathbf{f}(\mathbf{x}_k)$, et donc peut être effectuée en utilisant la factorisation LU avec pivotage partiel de la matrice jacobienne.

ALGORITHME (NEWTON-RAPHSON POUR SYSTÈMES) :

entrées : \mathbf{f} , \mathbf{f}' , \mathbf{x}_0

sortie : \mathbf{x}_{k+1}

répéter pour $k = 0, 1, 2, \dots$, jusqu'à l'arrêt

$A = \mathbf{f}'(\mathbf{x}_k)$ % évaluer la matrice $\mathbf{f}'(\mathbf{x}_k)$ si elle existe

$\mathbf{b} = \mathbf{f}(\mathbf{x}_k)$ % évaluer le vecteur $\mathbf{f}(\mathbf{x}_k)$ s'il existe

déterminer L , U et P tels que $LU = PA$ % si A est régulière, factoriser A

résoudre $LU\mathbf{p} = P\mathbf{b}$ % résoudre $A\mathbf{p} = \mathbf{b}$

$\mathbf{x}_{k+1} = \mathbf{x}_k - \mathbf{p}$

5.4.2 Méthode de Newton-corde

Le coût de la factorisation LU dans l'algorithme de Newton-Raphson pour systèmes peut devenir assez conséquent si le système non linéaire considéré est de grande taille. Ce coût est d'autant plus important qu'une nouvelle factorisation doit être obtenue à chaque itération. Dans certains cas on peut diminuer ce coût en résolvant le système $A\mathbf{p} = \mathbf{b}$ par une méthode itérative. Alternativement, on peut approcher $\mathbf{f}'(\mathbf{x}_k)$ par $\mathbf{f}'(\mathbf{x}_0)$, cette dernière matrice étant factorisée une seule fois ; cela revient à utiliser l'itération

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \mathbf{f}'(\mathbf{x}_0)^{-1} \mathbf{f}(\mathbf{x}_k)$$

qui correspond à la *méthode de la corde* (ou *Newton-corde*). Cette approche peut être plus efficace même si la convergence est alors généralement plus lente que celle de Newton-Raphson ; en particulier, elle n'est plus quadratique.

ALGORITHME (NEWTON-CORDE POUR SYSTÈMES) :

entrées : \mathbf{f} , \mathbf{f}' , \mathbf{x}_0

sortie : \mathbf{x}_{k+1}

$A = \mathbf{f}'(\mathbf{x}_0)$ % évaluer $\mathbf{f}'(\mathbf{x}_0)$ si elle existe

déterminer L , U et P tels que $LU = PA$ % factoriser A

répéter pour $k = 0, 1, 2, \dots$, jusqu'à l'arrêt

$\mathbf{b} = \mathbf{f}(\mathbf{x}_k)$ % évaluer $\mathbf{f}(\mathbf{x}_k)$ s'il existe

résoudre $LU\mathbf{p} = P\mathbf{b}$ % résoudre $A\mathbf{p} = \mathbf{b}$

$\mathbf{x}_{k+1} = \mathbf{x}_k - \mathbf{p}$

} factorisation
une seule fois
au début !

5.5 Critères d'arrêt

Les méthodes considérées dans ce chapitre sont des méthodes itératives. Il faut dès lors spécifier la manière dont elles sont arrêtées. Les quelques critères suivants sont repartis en deux catégories : \oplus indique que l'arrêt intervient quand l'approximation de la solution

est jugée satisfaisante, alors que \ominus spécifie les conditions dans lesquelles l'arrêt intervient même si la méthode n'est pas parvenue à obtenir une solution acceptable.

\oplus *Critère basé sur la norme de \mathbf{f} .* L'arrêt intervient si la norme de $\mathbf{f}(\mathbf{x}_k)$ satisfait

$$\|\mathbf{f}(\mathbf{x}_k)\| \leq \varepsilon_a.$$

On peut aussi considérer un critère relatif

$$\|\mathbf{f}(\mathbf{x}_k)\| \leq \varepsilon_r \|\mathbf{f}(\mathbf{x}_0)\|,$$

même si un tel critère doit s'utiliser avec prudence car le facteur de normalisation dépend alors du choix de \mathbf{x}_0 . Les valeurs de ε_a , ε_r sont typiquement fournies par l'utilisateur.

- \ominus L'arrêt intervient si le *nombre maximal d'itérations* est dépassé.
- \ominus Pour les méthodes de type Newton, le fait que la fonction \mathbf{f} ou \mathbf{f}' ne soit pas définie en \mathbf{x}_k , ou encore le fait que la dérivée/matrice jacobienne $\mathbf{f}'(\mathbf{x}_k)$ ou son approximation soit non inversible, peut mener à l'arrêt de la méthode.

Interpolation et approximation

Ce chapitre porte sur les problèmes d'interpolation et d'approximation (au sens des moindres carrés) d'un ensemble donné de points dans un plan par une fonction d'une variable réelle. On commence avec l'interpolation et l'approximation polynomiales, en abordant les deux problèmes dans un formalisme commun basé sur les systèmes linéaires. On considère ensuite uniquement le problème de l'interpolation polynomiale, et on constate que pour des points dont les abscisses sont équidistantes ce problème est mal conditionné pour des polynômes d'interpolation de degré élevé. Cette difficulté est contournée ici en interpolant localement les points donnés par des polynômes de degré modéré. L'interpolation par des splines, et en particulier des splines cubiques, est alors une des solutions envisagées.

6.1 Problèmes d'interpolation et d'approximation

Le problème d'*interpolation* consiste à déterminer une fonction appartenant à une classe donnée qui passe par un ensemble donné de points. Ici on considère l'interpolation par des fonctions d'une variable réelle. Pour un ensemble donné de points (x_i, y_i) , $i = 1, \dots, m$, dont les abscisses sont distinctes, le problème d'interpolation revient alors à déterminer une *fonction d'interpolation* f qui appartient à une classe donnée de fonctions et telle que

$$f(x_i) = y_i, \quad i = 1, \dots, m.$$

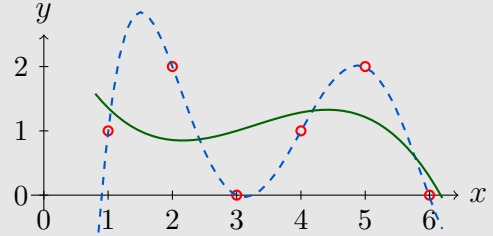
Il n'est pas exclu que dans certains cas la fonction d'interpolation n'existe pas. Il s'avère alors souvent utile de déterminer une fonction f qui appartient à une classe donnée de fonctions et qui passe aussi près que possible de l'ensemble des points considérés ; c'est le problème d'*approximation*. Ici nous considérons plus spécifiquement le problème d'approximation *au sens des moindres carrés* qui consiste à déterminer une fonction f qui minimise

$$\sum_{i=1}^m (f(x_i) - y_i)^2. \quad (6.1)$$

Comme déjà dit, le choix de f est typiquement restreint à une classe donnée de fonctions. En particulier, dans ce qui suit nous considérons l'interpolation et l'approximation par des polynômes dont le degré ne dépasse pas un entier donné.

EXEMPLE 33.

Considérons les 6 points $(1, 1)$, $(2, 2)$, $(3, 0)$, $(4, 1)$, $(5, 2)$ et $(6, 0)$. Comme il existe un et un seul polynôme de degré (au plus) 5 qui passe par tous ces points, le problème d'interpolation de ces points par un polynôme de degré (au plus) 5 possède une et une seule solution ; cette solution est représentée en trait interrompu sur la figure à droite.



Par contre, il n'existe pas de polynôme de degré (au plus) 3 qui passe par l'ensemble de ces points. Le polynôme de degré (au plus) 3 le plus proche de ces points au sens des moindres carrés est représenté sur la même figure en trait continu. On explique dans la section suivante comment ces polynômes peuvent être déterminés en pratique.

6.2 Interpolation et approximation polynomiales

6.2.1 Résolution à l'aide de systèmes linéaires

Dans cette section on explique comment les problèmes d'interpolation et d'approximation polynomiales se ramènent à la résolution de systèmes linéaires.

Commençons par le problème d'interpolation. Il revient à déterminer un polynôme

$$p(x) = \sum_{i=0}^{n-1} c_i x^i = c_0 + c_1 x + \cdots + c_{n-1} x^{n-1}$$

de degré $n-1$ (au plus) qui passe par un ensemble donné de points (x_i, y_i) , $i = 1, \dots, m$. Autrement dit, les coefficients c_i , $i = 0, \dots, n-1$, de ce polynôme doivent satisfaire

$$\begin{cases} c_0 + c_1 x_1 + \cdots + c_{n-1} x_1^{n-1} = y_1 \\ c_0 + c_1 x_2 + \cdots + c_{n-1} x_2^{n-1} = y_2 \\ \vdots \\ c_0 + c_1 x_m + \cdots + c_{n-1} x_m^{n-1} = y_m \end{cases}, \quad (6.2)$$

où la i -ème équation correspond à $p(x_i) = y_i$. Sous forme matricielle le système ainsi obtenu est noté

$$V \mathbf{c} = \mathbf{y},$$

avec

$$V = \begin{pmatrix} 1 & x_1 & \cdots & x_1^{n-1} \\ 1 & x_2 & \cdots & x_2^{n-1} \\ \vdots & \vdots & & \vdots \\ 1 & x_m & \cdots & x_m^{n-1} \end{pmatrix}, \quad \mathbf{c} = \begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_{n-1} \end{pmatrix}, \quad \mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix}.$$

La matrice V qui en résulte est appelée *matrice de Vandermonde*. Elle a comme particularité que sa k -ème colonne est donnée par un vecteur colonne (x_i^{k-1}) , avec $x_i, i = 1, \dots, m$, étant des réels donnés, correspondant ici aux abscisses des points considérés. Autrement dit, sa k -ème colonne peut être obtenue à partir de sa seconde colonne en élevant les éléments de cette dernière à la puissance $k - 1$.

Ainsi, le problème d'interpolation revient à déterminer les coefficients $c_i, i = 0, \dots, n - 1$, en résolvant le système (6.2). Si $m = n$, la matrice du système est régulière, et donc le système possède une et une seule solution. Cela équivaut à l'affirmation qu'un et un seul polynôme de degré au plus $m - 1$ passe par m points dont les abscisses sont toutes distinctes. Bien entendu, le polynôme ainsi déterminé convient également pour tout problème d'interpolation avec un degré polynomial maximal plus élevé (cas $m < n$).

Si $m > n$, le système (6.2) est surdéterminé. Il n'existe donc pas nécessairement de polynôme de degré $n - 1$ qui passe par les m points, et le problème d'interpolation polynomiale n'a généralement pas de solution. Par contre, on peut déterminer la solution du système (6.2) au sens des moindres carrés. Cela est d'autant plus utile que les coefficients $c_i, i = 0, \dots, n - 1$, ainsi obtenus définissent un polynôme qui résout le problème d'approximation au sens des moindres carrés; c'est-à-dire un polynôme qui minimise (6.1). En effet, comme

$$V\mathbf{c} - \mathbf{y} = \begin{pmatrix} c_0 + c_1x_1 + \dots + c_{n-1}x_1^{n-1} - y_1 \\ c_0 + c_1x_2 + \dots + c_{n-1}x_2^{n-1} - y_2 \\ \dots \\ c_0 + c_1x_m + \dots + c_{n-1}x_m^{n-1} - y_m \end{pmatrix}$$

on a aussi

$$\|V\mathbf{c} - \mathbf{y}\|_2^2 = \sum_{i=1}^m (p(x_i) - y_i)^2,$$

et donc les coefficients $\mathbf{c} = (c_i)$ qui minimisent le membre de gauche définissent un polynôme qui minimise le membre de droite.

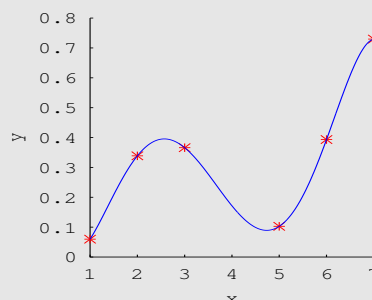
EXEMPLE 34. Ici on illustre la résolution des problèmes d'interpolation et d'approximation polynomiaux basée sur le système (6.2). On considère les points dont les abscisses sont données par le vecteur $\mathbf{x}=[1;2;3;5;6;7]$ et dont les ordonnées sont définies de manière aléatoire par l'instruction $\mathbf{y}=\text{rand}(6,1)$.

La détermination (\mathbf{y} compris l'affichage graphique) du polynôme d'interpolation de degré 5 est alors fait dans le code suivant, avec le résultat affiché à droite du code. Le code consiste en la construction de la matrice de Vandermonde, la résolution du système (6.2) correspondant et l'affichage du polynôme d'interpolation résultant.

```

% matrice de Vandermonde
A = [x.^0 x.^1 x.^2 ...
      x.^3 x.^4 x.^5];
pol = A\y; % système régulier
% afficher le polynôme
xf = 1:0.01:7;
plot(xf, polyval(pol(end:-1:1),xf))

```

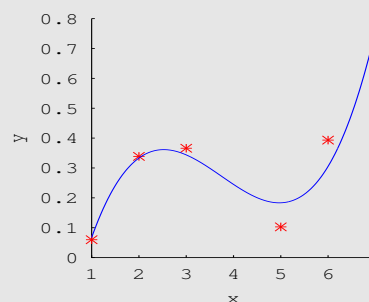


La détermination (y compris l’affichage graphique) du polynôme d’approximations (au sens des moindres carrés) de degré 3 est similaire ; elle est faite dans le code suivant. Notons que la matrice de Vandermonde a cette fois moins de colonnes, et est donc rectangulaire. Le `\` d’Octave effectue alors une résolution du système surdéterminé au sens des moindres carrés.

```

% matrice de Vandermonde
A = [x.^0 x.^1 x.^2 x.^3];
pol = A\y; % système surdéterminé!
% afficher le polynôme
xf = 1:0.01:7;
plot(xf, polyval(pol(end:-1:1),xf))

```



Notons que dans ce qui suit on ne considère que le problème d’interpolation, et donc, pour que ce problème possède une solution, on suppose aussi que $n = m$.

6.2.2 Interpolation avec polynômes de Lagrange

Le polynôme d’interpolation peut aussi être écrit sous une forme explicite à l’aide des polynômes de Lagrange. Le k -ème *polynôme de Lagrange* pour les abscisses x_i , $i = 1, \dots, m$, est défini par

$$\ell_k(x) = \prod_{\substack{i=1 \\ i \neq k}}^m \frac{x - x_i}{x_k - x_i}.$$

Ce polynôme vaut 1 si $x = x_k$ et 0 si $x = x_i$, $i \neq k$; en d’autres termes

$$\ell_k(x_i) = \delta_{ki}.$$

Les polynômes de Lagrange pour 4 et 5 abscisses équidistantes sont représentés sur la Figure 6.1.

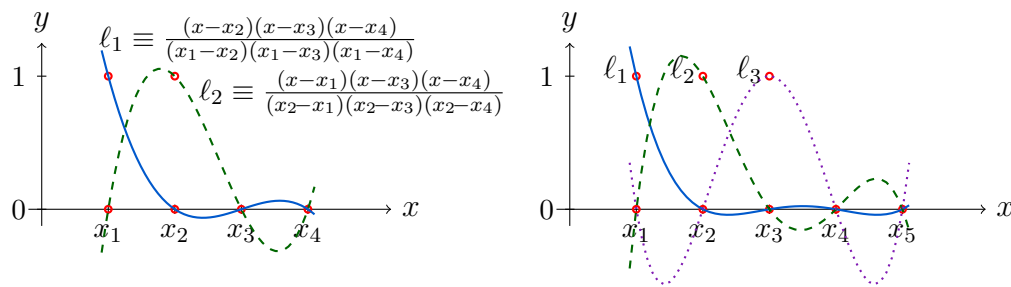


FIGURE 6.1 – Graphes des polynômes de Lagrange $\ell_1(x)$ et $\ell_2(x)$ pour 4 abscisses équidistantes (gauche), et des polynômes de Lagrange $\ell_1(x)$, $\ell_2(x)$ et $\ell_3(x)$ pour 5 abscisses équidistantes (droite) ; les polynômes non représentés s'en déduisent par symétrie.

Le polynôme d'interpolation pour des points (x_i, y_i) , $i = 1, \dots, m$, peut alors s'écrire comme suit

$$p(x) = y_1 \ell_1(x) + y_2 \ell_2(x) + \dots + y_m \ell_m(x).$$

En effet, pour $x = x_i$ tous les termes sauf le i -ème s'annulent dans le membre de droite, et le i -ème terme vaut bien y_i . De plus, comme chacun des polynômes $\ell_k(x)$ est de degré $m - 1$, le polynôme d'interpolation $p(x)$ est de degré au plus $m - 1$.

6.2.3 Conditionnement

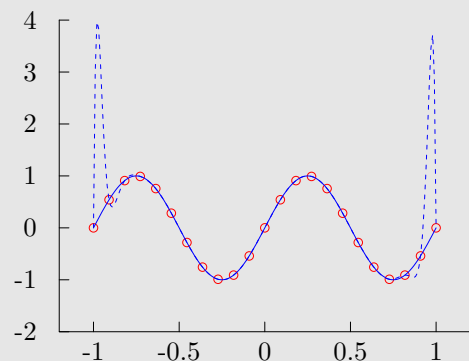
Rappelons que, de manière générale, le conditionnement caractérise la sensibilité d'un problème aux erreurs dans les données. Dès lors, si de petites erreurs dans les abscisses et/ou ordonnées mènent à un polynôme d'interpolation qui varie sensiblement entre les différentes abscisses sur l'intervalle $[x_1, x_m]$, alors le problème d'interpolation est jugé mal conditionné. C'est notamment le cas lorsqu'on considère l'interpolation pour un nombre important d'abscisses équidistantes, comme illustré avec l'exemple suivant.

EXEMPLE 35. (QUARTERONI AND AL.)

On considère le problème d'interpolation par un polynôme de degré (au plus) 22 pour les deux ensembles de points suivants :

- (a) $(x_i, \sin(2\pi x_i))$
- (b) $(x_i, \sin(2\pi x_i) + \delta_i)$ avec $\delta_i = \pm 10^{-4}$

avec x_i , $i = 1, \dots, 23$, des abscisses réparties uniformément entre -1 et 1 . Les polynômes d'interpolation correspondants sont représentés sur la figure à droite : en trait continu pour le cas (a), et en trait interrompu pour le cas (b). On constate que des écarts de l'ordre de 10^{-4} entre les ordonnées des points mènent à des écarts de l'ordre de 1 entre les deux polynômes d'interpolation.



L'exemple précédent montre que le problème d'interpolation polynomiale aux abscisses équidistantes est déjà assez mal conditionné pour un degré polynomial aux alentours de 20. En fait, le conditionnement se dégrade d'avantage avec l'augmentation du degré polynomial. Deux options sont envisageables pour remédier à ce problème : utiliser des abscisses x_i avec un *espacement adapté* (par opposition aux x_i équidistantes), ou bien *limiter le degré* des polynômes utilisés. Dans ce qui suit nous n'explorerons que la deuxième option.

6.3 Interpolation polynomiale par morceaux

La manière la plus simple de limiter le degré polynomial sans pour autant limiter le nombre d'abscisses est d'utiliser l'interpolation par morceaux. Pour $x_1 < x_2 < \dots < x_m$, cela revient à subdiviser l'intervalle $[x_1, x_m]$ en sous-intervalles contigus et disjoints, et d'appliquer ensuite la procédure d'interpolation sur chaque sous-intervalle *séparément*. Bien entendu, le nombre d'abscisses x_i par sous-intervalle doit être faible pour des raisons de stabilité mentionnées auparavant.

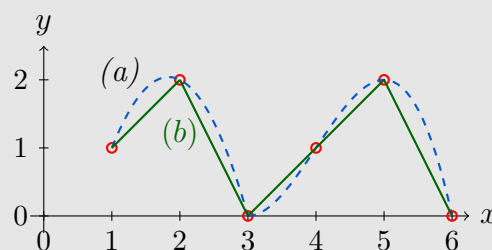
La fonction d'interpolation n'est alors plus polynomiale, mais bien polynomiale par morceaux (c'est-à-dire polynomiale sur chaque sous-intervalle). Elle n'est en général pas continue ; néanmoins, on peut la rendre continue en choisissant des sous-intervalles dont les extrémités coïncident avec les abscisses x_i . Dans ce dernier cas la dérivée de la fonction d'interpolation est généralement toujours discontinue aux extrémités des sous-intervalles. Ce manque de régularité est d'ailleurs le défaut principal de l'approche.

EXEMPLE 33. (SUITE)

Pour l'exemple de la page 120 plusieurs choix de sous-intervalles sont envisageables, dont on ne considère que deux en particulier :

- (a) $[1, 3]$ et $[3, 6]$
- (b) $[1, 2]$, $[2, 3]$, $[3, 4]$, $[4, 5]$ et $[5, 6]$

Les fonctions d'interpolation correspondantes sont représentées sur la figure à droite : en trait interrompu pour les intervalles de (a), et en trait continu pour ceux de (b). On notera que ces fonctions sont continues, mais aussi que leur dérivées sont effectivement discontinues aux extrémités des sous-intervalles.



6.4 Interpolation par splines

L'interpolation polynomiale par morceaux mène à des fonctions d'interpolation qui ne sont pas à dérivée continue ; pour avoir des fonctions suffisamment régulières on utilise

des splines. Plus précisément, pour $x_1 < x_2 < \dots < x_m$, on définit une *spline de degré k* comme une fonction qui :

- est un polynôme de degré k sur $[x_i, x_{i+1}]$, $i = 1, \dots, m-1$ (et donc polynomiale par morceaux) ;
- les $k-1$ premières dérivées sont continues.

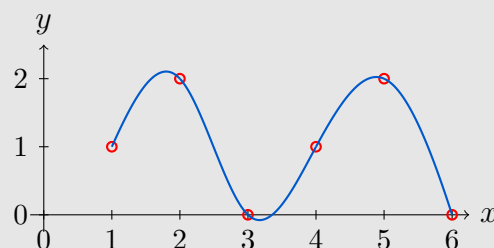
On ne détaille ici que l'interpolation par *splines cubiques*, qui sont donc :

- des *polynômes cubiques par morceaux* ;
- *continues* et dont les *deux premières dérivées* sont aussi *continues*.

Avant d'expliquer comment on détermine la fonction d'interpolation par *splines cubiques* on présente un exemple d'une telle fonction.

EXEMPLE 33. (FIN)

Pour l'exemple de la page 120 la fonction d'interpolation par splines cubiques est représentée sur la figure à droite. On note que la dérivée de la fonction est continue en tout point, et en particulier aux extrémités des sous-intervalles.



La construction de la fonction d'interpolation f par splines cubiques repose sur la vérification des quatre conditions suivantes :

1. f est un polynôme cubique sur chaque intervalle $[x_i, x_{i+1}]$, $i = 1, \dots, m-1$;
2. f est une fonction d'interpolation, et donc $f(x_i) = y_i$, $i = 1, \dots, m-1$; et par conséquent f est continue (y compris aux abscisses x_i , $i = 2, \dots, m-1$) ;
3. la dérivée première f' est continue (y compris aux abscisses x_i , $i = 2, \dots, m-1$) ;
4. la dérivée seconde f'' est continue (y compris aux abscisses x_i , $i = 2, \dots, m-1$).

Plus précisément, la fonction d'interpolation f par splines cubiques est obtenue en résolvant un système linéaire dont les inconnues α_i représentent les valeurs de $f''(x_i)$, $i = 1, \dots, m$. La conditions 1 et 4 sont satisfaites si la dérivée seconde f'' sur le sous-intervalle $]x_i, x_{i+1}[$ est choisie de la forme

$$f''(x) = \alpha_i \frac{x_{i+1} - x}{x_{i+1} - x_i} + \alpha_{i+1} \frac{x - x_i}{x_{i+1} - x_i} \quad \text{pour } x \in]x_i, x_{i+1}[,$$

où α_i , $i = 1, \dots, m$, sont les inconnues à déterminer. En particulier, on a par construction $f''(x_i^+) = f''(x_i^-) = \alpha_i$, $i = 1, \dots, m$. Par ailleurs, en intégrant cette dernière expression deux fois, on obtient, en notant $h_i = x_{i+1} - x_i$,

$$f(x) = \alpha_i \frac{(x_{i+1} - x)^3}{6h_i} + \alpha_{i+1} \frac{(x - x_i)^3}{6h_i} + \beta_i(x - x_i) + \gamma_i \quad \text{pour } x \in [x_i, x_{i+1}] . \quad (6.3)$$

Les valeurs de β_i et γ_i s'obtiennent en imposant la conditions d'interpolation (condition 2) aux abscisses x_i et x_{i+1} , à savoir

$$\begin{aligned} f(x_i) &= y_i & \Rightarrow & \quad \gamma_i = y_i - \alpha_i \frac{h_i^2}{6}, \\ f(x_{i+1}) &= y_{i+1} & \Rightarrow & \quad \beta_i = \frac{y_{i+1} - y_i}{h_i} - (\alpha_{i+1} - \alpha_i) \frac{h_i}{6}. \end{aligned} \quad (6.4)$$

Finalement, en dérivant une fois l'expression (6.3) on a

$$f'(x) = -\alpha_i \frac{(x_{i+1} - x)^2}{2h_i} + \alpha_{i+1} \frac{(x - x_i)^2}{2h_i} + \beta_i \quad \text{pour } x \in]x_i, x_{i+1}[.$$

Dès lors, la condition 3, qui impose la continuité de la dérivée première en x_i , $i = 2, \dots, m-1$, correspond à

$$\underbrace{-\alpha_i \frac{h_i}{2} + \beta_i}_{\text{approximation sur }]x_i, x_{i+1}[} = \underbrace{\alpha_i \frac{h_{i-1}}{2} + \beta_{i-1}}_{\text{approximation sur }]x_{i-1}, x_i[}.$$

En substituant les expressions (6.4) de β_i dans la dernière égalité on a

$$-\alpha_i \frac{h_i}{3} - \alpha_{i+1} \frac{h_i}{6} + \frac{y_{i+1} - y_i}{h_i} = \alpha_i \frac{h_{i-1}}{3} + \alpha_{i-1} \frac{h_{i-1}}{6} + \frac{y_i - y_{i-1}}{h_{i-1}}.$$

Après un regroupement des termes en α et une multiplication par $6/(h_{i-1} + h_i)$ on obtient $m-2$ équations linéaires

$$\frac{h_{i-1}}{h_{i-1} + h_i} \alpha_{i-1} + 2\alpha_i + \frac{h_i}{h_{i-1} + h_i} \alpha_{i+1} = \frac{6}{h_{i-1} + h_i} \left(\frac{y_{i+1} - y_i}{h_i} - \frac{y_i - y_{i-1}}{h_{i-1}} \right),$$

avec $i = 2, \dots, m-1$. Pour avoir un système régulier il reste encore à choisir deux équations supplémentaires. Les *splines naturelles* sont obtenues si les équations $f''(x_1) = \alpha_1 = 0$ et $f''(x_m) = \alpha_m = 0$ sont choisies pour compléter le système.

Intégration numérique

Ce chapitre porte sur l'intégration numérique des fonctions de type $f : A \subset \mathbb{R}^d \mapsto \mathbb{R}$. Il est en grande partie consacré au cas le plus simple : l'intégration de fonctions d'une variable ($d = 1$). On aborde brièvement l'intégration de fonctions de plusieurs variables ($d > 1$) vers la fin du chapitre.

7.1 Fonctions d'une variable

7.1.1 Problème

L'intégration numérique sur un intervalle $[a, b]$ d'une fonction intégrable $f : [a, b] \mapsto \mathbb{R}$ d'une variable réelle consiste à déterminer une approximation numérique de l'intégrale

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

La tâche est typiquement effectuée en deux étapes. On commence par subdiviser l'intervalle d'intégration $[a, b]$ en sous-intervalles $[x_i, x_{i+1}]$, $i = 1, \dots, m-1$, avec

$$a = x_1 < x_2 < \dots < x_m = b.$$

L'intégrale de la fonction f sur l'intervalle $[a, b]$ peut alors être remplacée par une somme d'intégrales de f sur l'ensemble des sous-intervalles ; en d'autres termes

$$\int_a^b f(x)dx = \sum_{i=1}^{m-1} \int_{x_i}^{x_{i+1}} f(x)dx.$$

Ensuite, on évalue séparément les intégrales dans la somme, en remplaçant la fonction f par une approximation «facile à intégrer». Cette dernière approximation résulte souvent d'une interpolation polynomiale présentée au Chapitre 6. La motivation derrière cette démarche en deux étapes est que la fonction f est plus facilement approchée par une fonction «facile à intégrer» sur chacun des sous-intervalles que sur l'intervalle tout entier.

$$\int_{x_i}^{x_{i+1}} f(x) dx \approx \frac{h_i}{2} (f_i + f_{i+1}).$$

Pour une intégrale sur l'entièreté de l'intervalle $[a, b]$, la formule des trapèzes donne, pour des sous-intervalles de même taille $h_i = h, i = 1, \dots, m - 1$,

$$\int_a^b f(x)dx = \sum_{i=1}^{m-1} \int_{x_i}^{x_{i+1}} f(x)dx \approx \frac{h}{2}(f_1 + 2f_2 + \cdots + 2f_{m-1} + f_m).$$

Il est souvent utile de connaître l'erreur commise lorsqu'on approche une intégrale par la formule des trapèzes. Commençons par l'erreur locale. Plus précisément, on parle d'erreur locale $E_{\text{loc}}(h_i)$ pour la formule des trapèzes si un sous-intervalle individuel est considéré ; en d'autres termes, cette erreur vérifie

$$\int_{x_i}^{x_{i+1}} f(x) dx = \frac{h_i}{2}(f_i + f_{i+1}) + E_{\text{loc}}(h_i). \quad (7.1)$$

Le théorème suivant permet de quantifier cette dernière erreur. Ici on ne propose qu'une justification informelle du théorème ; la démonstration rigoureuse est donnée en annexe de ce chapitre.

$$E_{\text{loc}}(h_i) = -\frac{1}{12}h_i^3 f''(c). \quad (7.2)$$
$$f(x) = f(x_i) + f'(x_i)(x - x_i) + \frac{1}{2}f''(x_i)(x - x_i)^2 + \cdots \quad (7.3)$$

En intégrant d'une part l'expression dans (7.3) sur $[x_i, x_{i+1}]$, et en soustrayant de cette dernière $h_i/2$ fois l'expression dans (7.3) évaluée en x_{i+1} on a

$$\begin{aligned} \int_{x_i}^{x_{i+1}} f(x) dx &= f(x_i)h_i + \frac{1}{2}f'(x_i)h_i^2 + \frac{1}{6}f''(x_i)h_i^3 + \dots \\ + \left(f(x_{i+1}) &= f(x_i) + f'(x_i)h_i + \frac{1}{2}f''(x_i)h_i^2 + \dots \right) \times \left(-\frac{1}{2}h_i \right) \\ \hline \int_{x_i}^{x_{i+1}} f(x) dx - \frac{1}{2}f(x_{i+1})h_i &= \frac{1}{2}f(x_i)h_i - \frac{1}{12}f''(x_i)h_i^3 + \dots \end{aligned}$$

Cette dernière égalité n'est autre que (7.1) avec $E_{\text{loc}}(h_i) = -\frac{1}{12}f''(x_i)h_i^3 + \dots$ (comparez avec (7.2)). ■

Pour ce qui est de l'erreur d'intégration globale $E_{\text{glob}}(h)$, avec des sous-intervalles de même taille $h_i = h$, $i = 1, \dots, m-1$, elle est définie par la relation suivante

$$\int_a^b f(x) dx = \frac{h}{2}(f_1 + 2f_2 + \dots + 2f_{m-1} + f_m) + E_{\text{glob}}(h). \quad (7.4)$$

Le théorème suivant permet de quantifier cette dernière erreur.

THÉORÈME 5. *Si $f \in C^2([a, b])$, alors il existe un $c \in]a, b[$ tel que $E_{\text{glob}}(h)$ dans (7.4) est donné par*

$$E_{\text{glob}}(h) = -\frac{1}{12}(b-a)h^2 f''(c). \quad (7.5)$$

DÉMONSTRATION. Notez que comme $[a, b]$ est subdivisé en $m-1$ intervalles de longueur h , on a en particulier $h(m-1) = b-a$. Dès lors, par le Théorème 4 on a

$$E_{\text{glob}}(h) = -\frac{1}{12}(b-a)h^2 \underbrace{\left(\sum_{i=1}^{m-1} \frac{1}{m-1} f''(c_i) \right)}_{=:\mu},$$

avec $c_i \in]x_i, x_{i+1}[$. Comme μ correspond à une moyenne des valeurs de f'' évaluée en certains points de l'intervalle $[c_1, c_{m-1}]$, elle est comprise entre la valeur maximale et minimale de f'' sur $[c_1, c_{m-1}]$. Or, comme f'' est continue sur cet intervalle, par le théorème de la valeur intermédiaire il existe un $c \in [c_1, c_{m-1}] \subset]a, b[$ tel que $\mu = f''(c)$. ■

7.1.3 Formule de Simpson

La *formule de Simpson* est obtenue lorsqu'on intègre sur chaque sous-intervalle $[x_i, x_{i+1}]$ le polynôme d'interpolation quadratique qui passe par les points (x_i, f_i) , $(x_{i+1/2}, f_{i+1/2})$ et (x_{i+1}, f_{i+1}) , où $x_{i+1/2} = (x_i + x_{i+1})/2$ est le milieu du sous-intervalle, et $f_k = f(x_k)$, $k = i, i+1/2, i+1$. En notant $h_i = x_{i+1} - x_i$, la formule de Simpson est donnée par

$$\int_{x_i}^{x_{i+1}} f(x) dx \approx \frac{h_i}{6}(f_i + 4f_{i+1/2} + f_{i+1}).$$

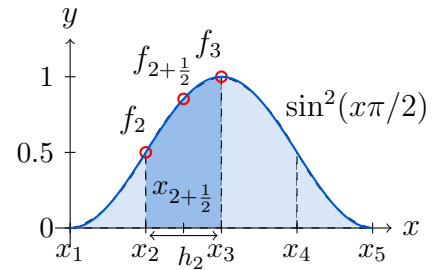


FIGURE 7.2 – Intégration via la formule de Simpson.

Cette formule est exacte pour tout polynôme de degré au plus 3 (et pas 2, comme on pourrait s'attendre avec une interpolation quadratique).

Pour une intégrale sur l'entièreté de l'intervalle $[a, b]$, la formule de Simpson correspond, pour des sous-intervalles de même taille $h_i = h$, $i = 1, \dots, m-1$, à

$$\int_a^b f(x)dx = \sum_{i=1}^{m-1} \int_{x_i}^{x_{i+1}} f(x)dx \approx \frac{h}{6} \left(f_1 + 2 \sum_{i=2}^{m-1} f_i + 4 \sum_{i=1}^{m-1} f_{i+1/2} + f_m \right).$$

Cette dernière expression est parfois appelé *formule composite de Simpson*.

Pour être complet, indiquons sans démonstration le résultat suivant.

THÉORÈME 6. Si $f \in C^4([a, b])$, alors il existe un $c \in]a, b[$ tel que

$$\int_a^b f(x)dx = \frac{h}{6} \left(f_1 + 2 \sum_{i=2}^{m-1} f_i + 4 \sum_{i=1}^{m-1} f_{i+1/2} + f_m \right) - \frac{1}{180}(b-a) \left(\frac{h}{2} \right)^4 f^{(4)}(c).$$

7.1.4 Formules de Newton-Cotes

Les formules des trapèze et de Simpson sont des cas particuliers des formules de Newton-Cotes¹. Ces dernières sont obtenues en intégrant sur chaque sous-intervalle $[x_i, x_{i+1}]$ le polynôme d'interpolation de degré n qui passe par les $n+1$ points $(x_k, f(x_k))$, où x_k sont les abscisses équidistantes et $f_k = f(x_k)$, $k = i, i+1/n, \dots, i+1$. En notant comme avant $h_i = x_{i+1} - x_i$, les formules de Newton-Cotes sont de la forme

$$\int_{x_i}^{x_{i+1}} f(x)dx \approx h_i(w_1 f_i + w_2 f_{i+1/n} + \dots + w_n f_{i+(n-1)/n} + w_{n+1} f_{i+1}).$$

Les valeurs des poids w_i pour les degrés $n \leq 4$ sont données dans la Table 7.1. Le cas $n = 1$ correspond à la formule des trapèzes, alors que le cas $n = 2$ donne la formule de Simpson.

Les formules de Newton-Cotes qui correspondent aux valeurs élevées de n sont rarement utilisées en pratique. La raison en est double. D'une part, pour des valeurs élevées de n ces formules peuvent mener à des problèmes de stabilité numérique dus au mauvais conditionnement des polynômes d'interpolation utilisés dans leur dérivation. D'autre part, la fonction à intégrer peut ne pas être suffisamment régulière pour que l'analyse qui mène aux erreurs d'intégration données dans la Table 7.1 s'applique effectivement. Ce dernier argument est illustré avec l'exemple suivant.

1. Il s'agit plus précisément des variantes fermées des formules de Newton-Cotes.

n	w_1	w_2	w_3	w_4	w_5	deg. exact pol.	$E_{\text{loc}}(h)$	$E_{\text{glob}}(h)$
1	$\frac{1}{2}$	$\frac{1}{2}$				1	$\sim f''h^3$	$\sim (b-a)f''h^2$
2	$\frac{1}{6}$	$\frac{4}{6}$	$\frac{1}{6}$			3	$\sim f^{(4)}h^5$	$\sim (b-a)f^{(4)}h^4$
3	$\frac{1}{8}$	$\frac{3}{8}$	$\frac{3}{8}$	$\frac{1}{8}$		3	$\sim f^{(4)}h^5$	$\sim (b-a)f^{(4)}h^4$
4	$\frac{7}{90}$	$\frac{32}{90}$	$\frac{12}{90}$	$\frac{32}{90}$	$\frac{7}{90}$	5	$\sim f^{(6)}h^7$	$\sim (b-a)f^{(6)}h^6$

TABLE 7.1 – Les valeurs des poids w_i dans les formules de Newton-Cotes de degré $n \leq 4$, ainsi que le degré du polynôme pour lequel la formule est exacte, la forme de l'erreur locale et de l'erreur globale.

EXEMPLE 36. On se propose de mesurer l'erreur globale $E_{\text{glob}}(h)$ commise en approchant les deux intégrales suivantes

$$\int_0^1 \sin(x)dx, \quad \int_0^1 x^{3/2}dx,$$

à l'aide des formules de Newton-Cotes. On procède en subdivisant l'intervalle $[0, 1]$ en $m - 1 = 2^k$, $0 \leq k \leq 9$, sous-intervalles de même largeur et en appliquant les formules de Newton-Cotes sur chaque sous-intervalle. La Figure 7.3 représente l'erreur globale $E_{\text{glob}}(h)$ en fonction du nombre $m - 1$ d'intervalles considérés pour les formules de Newton-Cotes de degrés $n = 1, \dots, 4$; l'erreur pour l'intégrale de $\sin(x)$ est à gauche, celle pour $x^{3/2}$ à droite.

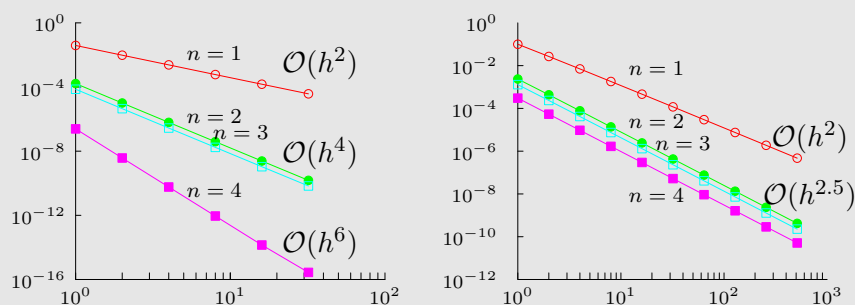


FIGURE 7.3 – Erreur globale des formules de Newton-Cotes pour l'évaluation de l'intégrale de $\sin(x)$ (gauche) et de $x^{3/2}$ (droite) sur l'intervalle $[0, 1]$.

On constate ainsi que le comportement de l'erreur globale dépend de la régularité de la fonction considérée. En effet, comme $\sin(x) \in C^\infty([0, 1])$, on observe pour les différentes valeurs de n la même dépendance en h sur le graphe gauche de la Figure 7.3 que celle donnée dans le Table 7.1. Par contre pour $x^{3/2}$, vu que seule la dérivée première est continue, même l'estimateur (7.5) pour la méthode des trapèzes n'est en toute rigueur pas garanti. En pratique on voit que le comportement théorique est néanmoins observé pour la formule des trapèzes ($n = 1$), mais il ne l'est plus pour les formules de Newton-Cotes de degré $n > 1$.

7.1.5 Méthode de Romberg

La méthode de Romberg représente une autre approche pour obtenir une erreur qui se comporte, pour une fonction f suffisamment régulière et un n ajustable, en h^{2n} . La dérivation de la méthode est basée sur le fait (voir Quarteroni & al., Propriété 8.3) que pour une fonction $f \in C^{2n+2}([a, b])$ la méthode des trapèzes produit une approximation $I_1(h)$ qui satisfait

$$I_1(h) = \int_a^b f(x)dx + C_1h^2 + C_2h^4 + \cdots + C_nh^{2n} + o(h^{2n}).$$

En particulier, si on double le pas d'intégration pour la méthode des trapèzes, la formule donne alors

$$I_1(2h) = \int_a^b f(x)dx + C_14h^2 + C_24^2h^4 + \cdots + C_n4^nh^{2n} + o(h^{2n}).$$

Séparément, les deux valeurs donnent une approximation numérique de l'intégrale $\int_a^b f(x)dx$ avec une erreur de l'ordre de h^2 . Néanmoins, on peut avantageusement combiner ces valeurs ensemble comme suit

$$I_2(h) = \frac{4I_1(h) - I_1(2h)}{3} = \int_a^b f(x)dx - \frac{12}{3}C_2h^4 - \cdots - \frac{4^n - 4}{3}C_nh^{2n} + o(h^{2n})$$

pour faire disparaître le terme en h^2 dans l'expression de l'erreur au profit du terme en h^4 . Si $n > 2$, on peut répéter la démarche avec cette fois $I_2(h)$ combiné avec $I_2(2h)$. Notons que pour calculer ce dernier il faut disposer des valeurs de $I_1(2h)$ (déjà utilisée pour $I_2(h)$) et de $I_1(4h)$, c'est-à-dire des approximations obtenues avec la formule des trapèzes pour les pas d'intégration $2h$ et $4h$. On obtient ainsi la dépendance schématisée pour le cas de $I_n(h)$ quelconque dans la Table 7.2.

$i \backslash j$	1	2	\cdots	$n-1$	n
1	$I_1(h)$	$\longrightarrow I_2(h)$	\cdots	$I_{n-1}(h)$	$\longrightarrow I_n(h)$
2	$I_1(2h)$	$\nearrow I_2(2h)$	\cdots	$I_{n-1}(2h)$	\nearrow
\vdots	\vdots	\vdots			
$n-1$	$I_1(2^{n-1}h)$	$\longrightarrow I_2(2^{n-1}h)$			
n	$I_1(2^n h)$	\nearrow			

TABLE 7.2 – Approximations successives générées par la méthode de Romberg.

Pour ce qui est de l'algorithme de la méthode de Romberg, on présente ici la variante qui s'applique à un intervalle $[a, b]$ non subdivisé en sous-intervalles ; en cas de

subdivision en sous-intervalles la méthode peut être appliquée à chaque sous-intervalle individuellement. La notation $I_{j,i}$ dans l'algorithme qui suit correspond à $I_j(2^{i-1}h)$ avec $h = (b - a)2^{1-n}$; l'algorithme renvoie alors la valeur de $I_{n,1} = I_n(h)$. Notons aussi que pour la méthode des trapèzes le pas d'intégration $2^{i-1}h$ correspond à 2^{n-i} intervalles.

ALGORITHME (MÉTHODE DE ROMBERG) :

entrées : a, b, f, n
 sortie : $I_{n,1}$

% trapezes(a, b, f, k) - méthode des trapèzes pour f sur $[a, b]$ avec k intervalles
 calculer $I_{1,i} = \text{trapezes}(a, b, f, 2^{n-i})$, $i = 1, \dots, n$
 pour $j = 1, \dots, n - 1$
 pour $i = 1, \dots, n - j$
 $I_{j+1,i} = (4^j I_{j,i} - I_{j,i+1}) / (4^j - 1)$

7.2 Fonctions de plusieurs variables

Pour intégrer les fonctions de deux variables et plus, plusieurs options sont envisageables ; elles sont représentées à la Figure 7.4.

Tout d'abord, on peut découper le domaine de la fonction en «tranches» (comme pour le théorème de Fubini), et approcher ensuite l'intégrale sur chaque «tranche» par une intégrale unidimensionnelle multipliée par la largeur de la «tranche» (voir la Figure 7.4(b)). L'intégrale complète est alors la somme des contributions correspondant aux différentes «tranches».

Alternativement, on peut généraliser les idées présentées au cas des fonctions de plusieurs variables. Par exemple, la généralisation de la méthode des trapèzes aux fonctions de deux variables consiste à approcher le domaine d'intégration par une union de triangles T et interpoler la fonction sur chaque triangle par une fonction linéaire de deux variables (voir la Figure 7.4(c)). Pour chaque triangle T cela donne

$$\iint_T f(x, y) \approx \frac{1}{3} \text{aire}(T)(f_1 + f_2 + f_3),$$

où f_1, f_2, f_3 sont les valeurs de f aux sommets du triangle ; l'intégrale complète est alors la somme des contributions correspondant aux différents triangles.

Finalement, on peut aussi utiliser de nouvelles approches. Par exemple, les méthodes Monte-Carlo déterminent une intégrale multiple en évaluant la fonction en des points du domaine générés aléatoirement, et en multipliant ensuite la moyenne de ces valeurs par l'aire du domaine d'intégration (voir la Figure 7.4(d)). Ces méthodes sont typiquement efficaces pour intégrer des fonctions à variables multiples (de l'ordre de 6 et plus).

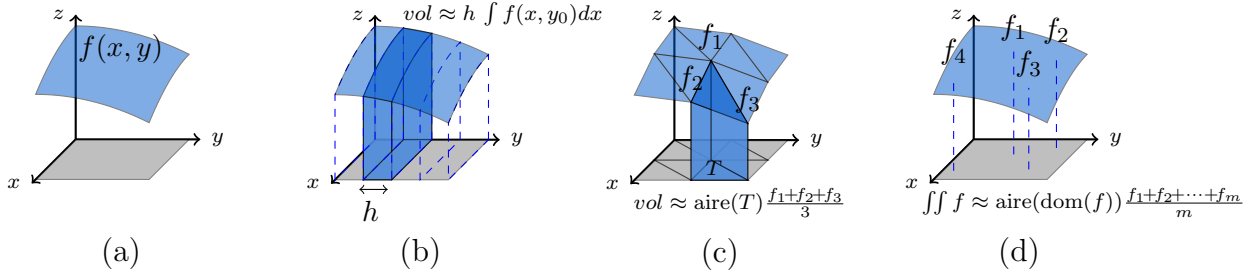


FIGURE 7.4 – Représentation schématique d'une fonction $f(x, y)$ de deux variables (a) et de différentes options pour calculer son intégrale double (b-d).

Annexe 7.A Formule des trapèzes : erreur locale

On commence avec un lemme important, qui est un cas particulier du Théorème 7.2 dans Quarteroni & al. Notez que ce lemme est intéressant en tant que tel, car il donne une erreur d'approximation pour un polynôme d'interpolation de degré 1.

LEMME A.1. Soient $f \in C^2([x_0, x_1])$ et $h = x_1 - x_0$. Pour tout $x \in [x_0, x_1]$ il existe un $c \in]x_0, x_1[$ tel que

$$f(x) - h^{-1}f(x_0)(x_1 - x) - h^{-1}f(x_1)(x - x_0) = \frac{1}{2}f''(c)(x - x_0)(x - x_1). \quad (7.A.1)$$

DÉMONSTRATION. La relation est clairement valable pour $x = x_0$ et $x = x_1$, nous supposons donc que $x \in]x_0, x_1[$. Dans ce cas, la fonction $g(x)$ définie par

$$g(x) := f(x) - h^{-1}f(x_0)(x_1 - x) - h^{-1}f(x_1)(x - x_0)$$

est une combinaison linéaire de fonctions de classe $C^2([x_0, x_1])$, donc une fonction de classe $C^2([x_0, x_1])$. De plus, on a $g(x_0) = g(x_1) = 0$. Par ailleurs, pour tout $t \in [x_0, x_1]$ la fonction

$$h(t) := (x - x_0)(x - x_1)g(t) - (t - x_0)(t - x_1)g(x)$$

est une combinaison linéaire de fonctions de classe $C^2([x_0, x_1])$, donc une fonction de classe $C^2([x_0, x_1])$. Comme $h(t)$ s'annule en x_0 , x_1 et en x , par le théorème de Rolle il existent $c_0 \in]x_0, x[$ et $c_1 \in]x, x_1[$ tels que $h'(c_0) = h'(c_1) = 0$ (notez qu'on dérive par rapport à t , et pas x). En appliquant de nouveau le théorème des accroissements finis à h' , on conclut qu'il existe un $c \in]c_0, c_1[\subset]x_0, x_1[$ tel que $h''(c) = 0$, et donc tel que

$$(x - x_0)(x - x_1)g''(c) = 2g(x).$$

Le lemme est prouvé en constatant que $g''(c) = f''(c)$. ■

Le Théorème 4 correspond alors au résultat suivant avec l'intervalle $[x_0, x_1]$ remplacé par $[x_i, x_{i+1}]$. Notons que le résultat démontré ici est légèrement moins fort, car la constante c dans son énoncé appartient à $[x_0, x_1]$, et non à $]x_0, x_1[$.

THÉORÈME A.2. Soient $f \in C^2([x_0, x_1])$ et $h = x_1 - x_0$. Il existe un $c \in [x_0, x_1]$ tel que

$$\int_{x_0}^{x_1} f(x)dx - \frac{1}{2}h f(x_0) - \frac{1}{2}h f(x_1) = -\frac{1}{12}h^3 f''(c).$$

DÉMONSTRATION. On commence par intégrer l'égalité (7.A.1) sur l'intervalle $[x_0, x_1]$ en notant qu'en toute rigueur la constante c dans cette égalité dépend de x ; cela nous donne

$$\int_{x_0}^{x_1} f(x)dx - \frac{1}{2}h f(x_0) - \frac{1}{2}h f(x_1) = \int_{x_0}^{x_1} \frac{1}{2}f''(c(x))(x - x_0)(x - x_1)dx.$$

Maintenant, comme f'' est continue sur l'intervalle fermé et borné $[x_0, x_1]$, elle possède sur cet intervalle un minimum (noté f''_{\min}) et un maximum (noté f''_{\max}). La double inégalité

$$f''_{\min} \leq f''(c) \leq f''_{\max}, \quad c \in [x_0, x_1],$$

ainsi que le fait que $(x - x_0)(x - x_1) \leq 0$ sur l'intervalle $[x_0, x_1]$ impliquent

$$f''_{\min} \leq \underbrace{\frac{\int_{x_0}^{x_1} f''(c(x))(x - x_0)(x - x_1)dx}{\int_{x_0}^{x_1} (x - x_0)(x - x_1)dx}}_{=: \mu_1} \leq f''_{\max}.$$

Dès lors, par le théorème de la valeur intermédiaire il existe une constante $c \in [x_0, x_1]$ telle que $f''(c) = \mu_1$. Par conséquent

$$\int_{x_0}^{x_1} f(x)dx - \frac{1}{2}h f(x_0) - \frac{1}{2}h f(x_1) = \frac{1}{2}f''(c) \underbrace{\int_{x_0}^{x_1} (x - x_0)(x - x_1)dx}_{= -\frac{1}{6}h^3},$$

ce qui termine la démonstration. ■

Équations différentielles avec conditions initiales

Ce chapitre aborde la résolution d'équations différentielles avec conditions initiales. La présentation est restreinte aux équations et systèmes d'équations du premier ordre (c'est-à-dire aux équations et systèmes d'équations qui contiennent des fonctions inconnues et leur dérivées premières, mais ne contiennent pas leur dérivées d'ordre deux et plus) ; pour ce qui est des équations d'ordre deux et plus, on montre qu'elles se ramènent aux systèmes d'équations du premier ordre, et sont donc implicitement couvertes.

Dans un premier temps, on considère les méthodes numériques d'Euler progressive et d'Euler rétrograde. On explique ensuite ce qu'on entend par méthode explicite, méthode implicite, l'ordre d'une méthode, ainsi que sa stabilité absolue, tout en illustrant ces notions avec les méthodes d'Euler. Dans un deuxième temps, on aborde les méthodes de Crank-Nicolson et de Heun, avant de finir avec un aperçu des méthodes multi-pas.

8.1 Généralités

On commence par introduire le cas d'une équation différentielle du premier ordre avec une condition initiale ; c'est ce qu'on appelle le problème de Cauchy scalaire. On considère ensuite le problème de Cauchy vectoriel, qui correspond à un système d'équations différentielles du premier ordre avec des conditions initiales.

Le problème de Cauchy vectoriel est similaire au problème de Cauchy scalaire. Cette similitude nous permet d'aborder la plupart des notions uniquement dans le cas scalaire, l'extension au problème de Cauchy vectoriel se réduisant à «rajouter des flèches sur les vecteurs». Une discussion spécifique est proposée dans des cas où l'extension est moins triviale.

8.1.1 Équations différentielles du premier ordre

Une équation différentielle relie entre elles une variable t , une fonction inconnue y de cette variable t , et la dérivée y' de cette fonction par rapport à t . Comme cette équation décrit souvent une évolution temporelle, la variable t est appelée *temps*. Dans

ce qui suit, et sauf indication contraire, nous nous limitons à un intervalle temporel fini qui commence en 0 et se termine en T . Dans ce contexte, la condition initiale spécifie la valeur de la fonction inconnue en 0, alors que l'équation différentielle décrit l'évolution de cette fonction entre 0 et T .

Plus formellement, soient un réel $T > 0$ et une fonction donnée $f : [0, T] \times \mathbb{R} \mapsto \mathbb{R}$ de deux variables t et y que l'on suppose continue. Le problème consiste à déterminer une fonction scalaire $y \in C^1([0, T])$ qui satisfait, pour un réel y_0 donné,

$$\begin{cases} \frac{dy}{dt}(t) = f(t, y(t)), & t \in [0, T], & \text{(équation différentielle)} \\ y(0) = y_0. & & \text{(condition initiale)} \end{cases} \quad (8.1)$$

Ce problème est appelé ici *problème de Cauchy scalaire* (ou, simplement, problème de Cauchy).

Notez que si on intègre l'équation différentielle entre 0 et t , et si on utilise

$$\int_0^t \frac{dy}{dt}(\tau) d\tau = y(t) - \underbrace{y(0)}_{=y_0},$$

on obtient

$$y(t) = y_0 + \int_0^t f(\tau, y(\tau)) d\tau. \quad (8.2)$$

C'est la *formulation intégrale* du problème considéré. En particulier, si $f(t, y)$ ne dépend pas de y , le problème consiste à intégrer la fonction f sur l'intervalle $[0, t]$.

8.1.2 Systèmes d'équations différentielles du premier ordre

On présente maintenant le cas de n équations différentielles qui impliquent n fonctions inconnues y_1, \dots, y_n . Soient un réel $T > 0$ et n fonctions données $f_i : [0, T] \times \mathbb{R}^n \mapsto \mathbb{R}$, $i = 1, \dots, n$, supposées continues. Le problème consiste à déterminer n fonctions $y_i \in C^1([0, T])$, $i = 1, \dots, n$, qui satisfont

$$\begin{cases} \frac{dy_1}{dt}(t) = f_1(t, y_1(t), \dots, y_n(t)), \\ \vdots \\ \frac{dy_n}{dt}(t) = f_n(t, y_1(t), \dots, y_n(t)), \\ y_1(0) = y_{01}, \dots, y_n(0) = y_{0n}. \end{cases} \quad \begin{array}{l} \text{(système d'équations différentielles)} \\ \text{(conditions initiales)} \end{array}$$

Pour rendre la similitude avec le cas scalaire plus apparente, nous utilisons les notations vectorielles. Plus précisément, on pose

$$\mathbf{y}(t) = \begin{pmatrix} y_1(t) \\ \vdots \\ y_n(t) \end{pmatrix}, \quad \mathbf{f}(t, \mathbf{y}(t)) = \begin{pmatrix} f_1(t, y_1(t), \dots, y_n(t)) \\ \vdots \\ f_n(t, y_1(t), \dots, y_n(t)) \end{pmatrix}, \quad \mathbf{y}_0 = \begin{pmatrix} y_{01} \\ \vdots \\ y_{0n} \end{pmatrix},$$

avec donc $\mathbf{f}(t, \mathbf{y}) : [0, T] \times \mathbb{R}^n \mapsto \mathbb{R}^n$ une fonction continue de deux variables t et \mathbf{y} . Le problème peut alors s'écrire comme suit

$$\begin{cases} \frac{d\mathbf{y}}{dt}(t) = \mathbf{f}(t, \mathbf{y}(t)), & t \in [0, T], & \text{(équation différentielle vectorielle)} \\ \mathbf{y}(0) = \mathbf{y}_0. & & \text{(condition initiale)} \end{cases} \quad (8.3)$$

Ce problème est appelé ici *problème de Cauchy vectoriel*.

Ici aussi, si on intègre la première équation entre 0 et t , on obtient

$$\mathbf{y}(t) = \mathbf{y}_0 + \int_0^t \mathbf{f}(\tau, \mathbf{y}(\tau)) d\tau.$$

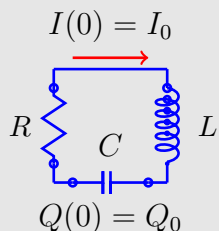
Pour rappel, l'intégrale entre 0 et t d'un vecteur $\mathbf{v}(t) = (v_i(t))$ est un vecteur dont la composante i vaut $\int_0^t v_i(\tau) d\tau$.

8.1.3 Équations d'ordre deux et plus

On considère à présent le cas des équations différentielles d'ordre deux et plus, et on explique comment ces équations peuvent se ramener aux systèmes d'équations du premier ordre. Nous commençons avec un exemple d'équation différentielle d'ordre deux issu de la physique.

EXEMPLE 37.

L'évolution dans le temps du circuit RLC représenté à la figure de gauche est décrite par l'équation différentielle d'ordre 2 suivante dont la fonction inconnue est la charge $Q(t)$ sur le condensateur :



$$\begin{cases} L \frac{d^2 Q}{dt^2} + R \frac{dQ}{dt} + \frac{1}{C} Q = 0, \\ Q(0) = Q_0, \quad \frac{dQ}{dt}(0) = I_0. \end{cases}$$

Alternativement, en introduisant l'inconnue $I(t) = \frac{dQ}{dt}(t)$ du courant, on peut réécrire l'équation précédente comme le système différentiel d'ordre 1 suivant :

$$\begin{cases} \frac{d}{dt} \begin{pmatrix} I \\ Q \end{pmatrix} = \begin{pmatrix} -\frac{R}{L} I - \frac{1}{CL} Q \\ I \end{pmatrix}, \\ Q(0) = Q_0, \quad I(0) = I_0. \end{cases}$$

De manière plus générale, toute équation et tout système d'équations différentielles d'ordre deux et plus peuvent se ramener à un système d'équations du premier ordre. Comme dans l'exemple précédent, il suffit pour ce faire de remplacer les dérivées d'ordre inférieur à l'ordre de l'équation différentielle par des variables indépendantes. Puisque la variable «dérivée $(k+1)$ -ème» est égale à la dérivée de la variable «dérivée k -ème» pour tout k inférieur à l'ordre de l'équation, ces égalités complètent le système différentiel du premier ordre qui en résulte.

En particulier, toute équation différentielle d'ordre deux peut être ramenée à un système de deux équations différentielles du premier ordre via la transformation suivante

$$\begin{cases} \frac{d^2 y}{dt^2} = f(t, y, \frac{dy}{dt}), \\ y(0) = y_0, \quad \frac{dy}{dt}(0) = y'_0, \end{cases} \quad \xleftrightarrow{u = \frac{dy}{dt}} \quad \begin{cases} \frac{d}{dt} \begin{pmatrix} u \\ y \end{pmatrix} = \begin{pmatrix} f(t, y, u) \\ u \end{pmatrix}, \\ \begin{pmatrix} u(0) \\ y(0) \end{pmatrix} = \begin{pmatrix} y'_0 \\ y_0 \end{pmatrix}. \end{cases}$$

8.1.4 Existence, unicité

Les méthodes numériques abordées plus loin dans ce chapitre fournissent toujours une et une seule solution approchée au problème de Cauchy considéré. Néanmoins, cette solution approchée n'a de sens que lorsque la solution exacte du problème existe et est unique. Dès lors, on aborde la question d'existence et d'unicité avant d'examiner les aspects numériques.

Le théorème sur l'existence et l'unicité qui suit fait appel à la notion de fonction lipschitzienne. On dit que la fonction $f(t, y)$ est *lipschitzienne* en y s'il existe un réel ℓ tel que pour tout $t \in [0, T]$, y_1 et y_2 on a

$$|f(t, y_1) - f(t, y_2)| \leq \ell |y_1 - y_2|.$$

En particulier, si la dérivée partielle $\frac{\partial f}{\partial y}(t, y)$ existe et, pour tout $t \in [0, T]$ et tout y , satisfait

$$\left| \frac{\partial f}{\partial y}(t, y) \right| \leq \ell,$$

alors la fonction $f(t, y)$ est lipschitzienne en y . Un exemple d'une fonction lipschitzienne est donné à la Figure 8.1.

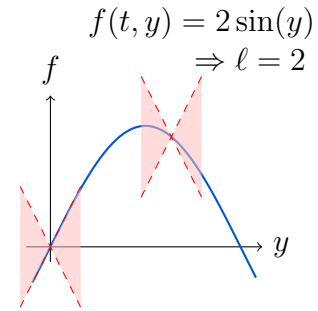


FIGURE 8.1 – Exemple d'une fonction lipschitzienne.

THÉORÈME 7. *Si la fonction $f(t, y)$ est continue sur $[0, T] \times \mathbb{R}$ et lipschitzienne en y , alors le problème de Cauchy*

$$\begin{cases} \frac{dy}{dt}(t) = f(t, y(t)), & t \in [0, T], \\ y(0) = y_0. \end{cases}$$

admet une et une seule solution.

Dans ce qui suit nous supposons que la fonction $f(t, y)$ est continue sur $[0, T] \times \mathbb{R}$ et lipschitzienne en y . Cela nous permet, entre autres, de rechercher la solution du problème de Cauchy sans nous demander si une telle solution existe.

8.1.5 Principe de résolution numérique

Alors que la solution exacte $y(t)$ du problème est une fonction définie sur $[0, T]$, les méthodes numériques ne fournissent qu'une seule solution approchée en un nombre fini de points seulement. Plus précisément, elles approchent la solution exacte $y(t)$ aux abscisses données $0 = t_0 < t_1 < \dots < t_m = T$ par les valeurs y_0, y_1, \dots, y_m de la solution approchée, comme illustré à la Figure 8.2. La distance entre deux abscisses successives

$$h_k = t_{k+1} - t_k$$

est le *pas de discrétisation* (aussi appelé *pas d'intégration*); ici on utilise la notation h sans l'indice k si le pas de discrétisation ne change pas d'une itération à l'autre.

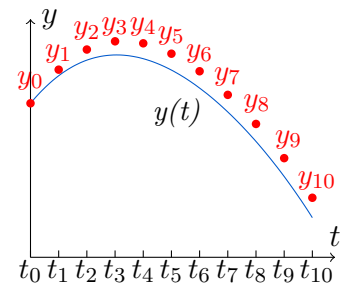


FIGURE 8.2 – Exemple d'une solution numérique.

8.2 Méthodes d'Euler

8.2.1 Méthode d'Euler progressive

On considère à présent les méthodes d'Euler pour la résolution du problème de Cauchy scalaire. La méthode d'*Euler progressive* (encore appelée *Euler explicite*) peut être obtenue de deux manières. Tout d'abord, en partant de la formulation différentielle (8.1), on peut approcher la dérivée au point t_k par

$$f(t_k, y(t_k)) = \frac{dy}{dt}(t_k) \approx \frac{y(t_{k+1}) - y(t_k)}{t_{k+1} - t_k}.$$

En utilisant $h_k = t_{k+1} - t_k$, $y_k \approx y(t_k)$, $y_{k+1} \approx y(t_{k+1})$ et en isolant y_{k+1} on obtient la récurrence

$$y_{k+1} = y_k + h_k f(t_k, y_k) \quad (8.4)$$

qui définit la méthode d'Euler progressive.

Alternativement, on peut obtenir la méthode d'Euler progressive à partir de la formulation intégrale (8.2), en notant que cette dernière implique (par exemple, en prenant la différence entre (8.2) évaluée en $t = t_{k+1}$ et celle évaluée en $t = t_k$)

$$y(t_{k+1}) = y(t_k) + \int_{t_k}^{t_{k+1}} f(\tau, y(\tau)) d\tau. \quad (8.5)$$

En approchant alors l'intégrande $f(\tau, y(\tau))$ par une constante

$$f(\tau, y(\tau)) \approx f(t_k, y_k) = \text{const} \quad \text{sur } [t_k, t_{k+1}],$$

et en utilisant $y_k \approx y(t_k)$, $y_{k+1} \approx y(t_{k+1})$, on retrouve de nouveau la récurrence (8.4).

8.2.2 Méthode d'Euler rétrograde

La méthode d'*Euler rétrograde* (encore appelée *Euler implicite*) est obtenue de manière similaire. Tout d'abord, en partant de la formulation différentielle (8.1), on peut approcher la dérivée au point t_{k+1} (au lieu de t_k pour Euler progressive) par

$$f(t_{k+1}, y(t_{k+1})) = \frac{dy}{dt}(t_{k+1}) \approx \frac{y(t_{k+1}) - y(t_k)}{t_{k+1} - t_k}.$$

En utilisant $h_k = t_{k+1} - t_k$, $y_k \approx y(t_k)$, $y_{k+1} \approx y(t_{k+1})$ on a la récurrence

$$y_{k+1} = y_k + h_k f(t_{k+1}, y_{k+1}) \quad (8.6)$$

qui définit la méthode d'Euler rétrograde. Notez que la seule différence avec la méthode d'Euler progressive réside dans le fait que la fonction $f(t, y)$ est évaluée en t_{k+1} et y_{k+1} au lieu de t_k et y_k pour Euler progressive.

Alternativement, on peut obtenir la méthode d'Euler rétrograde à partir de la formulation intégrale. En approchant alors l'intégrande $f(\tau, y(\tau))$ dans (8.5) par

$$f(\tau, y(\tau)) \approx f(t_{k+1}, y_{k+1}) = \text{const} \quad \text{sur } [t_k, t_{k+1}],$$

et en utilisant $y_k \approx y(t_k)$, $y_{k+1} \approx y(t_{k+1})$, on retrouve aussi (8.6).

8.2.3 Caractère explicite/implicite

Il y a une différence notable entre les deux méthodes d'Euler. La récurrence (8.4) d'Euler progressive permet d'évaluer directement la valeur de y_{k+1} à partir de celle de y_k , les valeurs de temps t_k , t_{k+1} et du pas de discrétisation $h_k = t_{k+1} - t_k$ étant données. Ainsi, on peut déterminer l'ensemble des valeurs y_k comme suit :

$$\begin{aligned} \text{pas 1 : } y_1 &= y_0 + h_0 f(t_0, y_0) \\ \text{pas 2 : } y_2 &= y_1 + h_1 f(t_1, y_1) \\ &\dots \end{aligned}$$

Il s'agit d'une méthode dite *explicite*.

La situation est différente pour la méthode d'Euler rétrograde. Bien que la récurrence (8.6) relie toujours entre elles les valeurs de y_{k+1} et y_k , elle ne permet plus d'exprimer explicitement la première valeur en fonction de la seconde puisque le membre de droite de la récurrence dépend aussi de y_{k+1} . Il s'agit donc d'une méthode *implicite*. Pour ce type de méthodes, la récurrence définit une équation potentiellement non linéaire qu'il faut résoudre avec une des méthodes¹ du Chapitre 5. La démarche de résolution est alors la suivante

$$\begin{aligned} \text{pas 1 : résoudre } y_1 &= y_0 + h_0 f(t_1, y_1) \text{ par rapport à } y_1 \\ \text{pas 2 : résoudre } y_2 &= y_1 + h_1 f(t_2, y_2) \text{ par rapport à } y_2 \\ &\dots \end{aligned}$$

1. Les méthodes de Newton sont particulièrement intéressantes puisque y_k peut alors servir d'approximation initiale pour la recherche de y_{k+1} .

8.2.4 Convergence

On considère à présent les propriétés de convergence. On commence par illustrer ces propriétés pour les méthodes d'Euler à l'aide d'un exemple particulier, et on présente ensuite un résultat général. Pour simplifier la présentation on suppose pour le restant de cette section que le pas de discrétisation h ne change pas d'une itération à l'autre.

EXEMPLE 38. Considérons la résolution numérique du problème de Cauchy suivant

$$\begin{cases} \frac{dy}{dt}(t) = \cos(y) - t, & t \geq 0, \\ y(0) = 0.7, \end{cases}$$

avec les méthodes d'Euler. Les solutions approchées pour la méthode d'Euler progressive avec $h = 0.2, 0.1, 0.05$ sont données à la Figure 8.3 (gauche). A la Figure 8.3 (droite) on trouve l'estimation de l'erreur $|y_k - y(t_k)|$, la solution approchée pour $h = 0.001$ faisant office de solution exacte $y(t)$.

Les résultats montrent que l'estimation de l'erreur $|y_k - y(t_k)|$ diminue lorsque h tend vers 0. De manière plus quantitative, on note que l'estimation de l'erreur est proportionnelle à h : elle est divisée par 2 lorsque h est divisé par 2.

Les résultats pour la méthode d'Euler rétrograde sont donnés à la Figure 8.4 pour les mêmes pas de discrétisation ; ils ont aussi le même comportement.

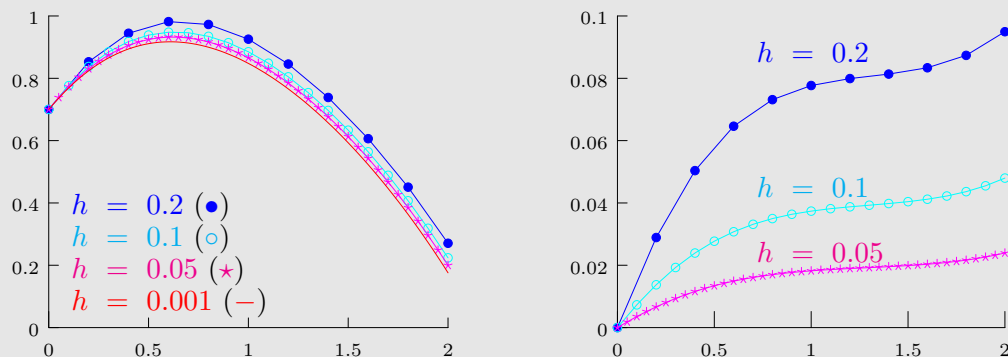


FIGURE 8.3 – Solutions approchées obtenues avec la méthode d'Euler progressive (gauche) et la différence (en valeur absolue) entre ces solutions et la solution approchée pour $h = 0.001$ (droite).

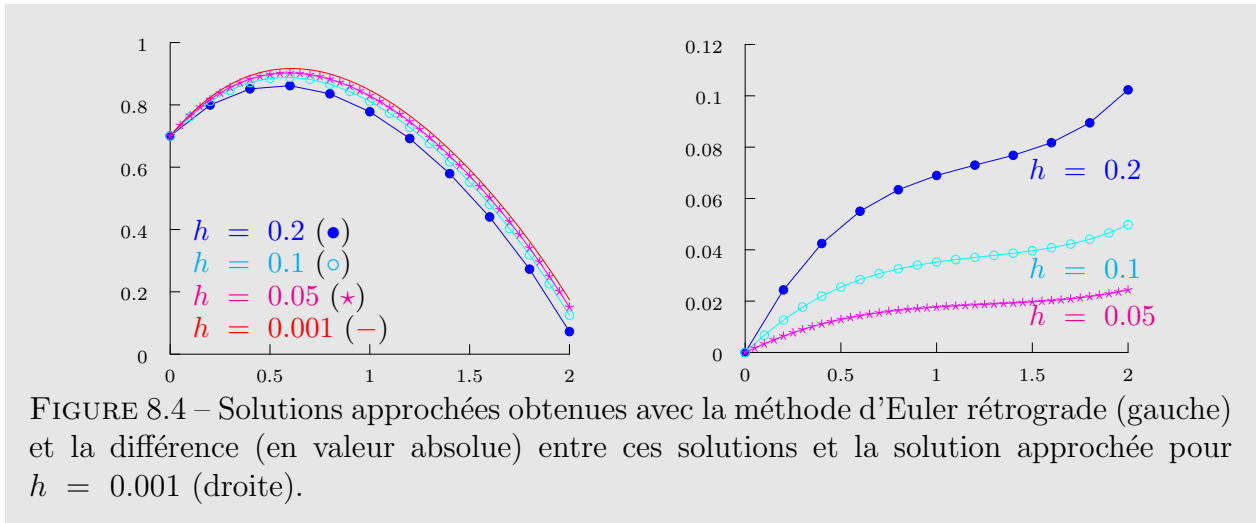


FIGURE 8.4 – Solutions approchées obtenues avec la méthode d’Euler rétrograde (gauche) et la différence (en valeur absolue) entre ces solutions et la solution approchée pour $h = 0.001$ (droite).

Le théorème suivant formalise les observations de l’exemple précédent.

THÉORÈME 8. Soit $f(t, y)$ une fonction continue sur $[0, T] \times \mathbb{R}$ et à dérivées partielles $\frac{\partial f}{\partial t}$ et $\frac{\partial f}{\partial y}$ continues. Supposons de plus que cette fonction est lipschitzienne en y .

Soit $y(t)$ l’unique solution du problème de Cauchy (8.1) associée à f , et soit y_k la solution approchée au point $t_k = hk$, avec $h = T/m$ et $k = 0, \dots, m$, obtenue par la méthode d’Euler progressive ou rétrograde.

Alors il existe une constante $C > 0$ (indépendante de h mais qui peut dépendre de T) telle que

$$\max_k |y_k - y(t_k)| \leq Ch.$$

Autrement dit, pour autant que les hypothèses du théorème soient satisfaites, la solution approchée y_k , $k = 0, \dots, m$, converge vers la solution exacte $y(t)$ lorsque h tend vers 0. De plus, la borne supérieure sur la différence entre la solution exacte et la solution approchée est proportionnelle au pas de discrétisation h . Ce dernier comportement signifie que les méthodes d’Euler sont des méthodes du *premier ordre*. De manière plus générale, une méthode de résolution d’une équation différentielle est d’*ordre* n si

$$\max_k |y_k - y(t_k)| \leq Ch^n.$$

8.2.5 Cas vectoriel

L’expression des méthodes d’Euler dans le cas du problème de Cauchy vectoriel (8.3) est la même que dans le cas scalaire. Plus précisément, pour la méthode d’Euler progressive, la récurrence (8.4) qui définit la méthode devient dans le cas vectoriel

$$\mathbf{y}_{k+1} = \mathbf{y}_k + h_k \mathbf{f}(t_k, \mathbf{y}_k). \quad (8.7)$$

De même, pour la méthode d’Euler rétrograde la récurrence (8.6) devient dans le cas vectoriel

$$\mathbf{y}_{k+1} = \mathbf{y}_k + h_k \mathbf{f}(t_{k+1}, \mathbf{y}_{k+1}). \quad (8.8)$$

8.3 Stabilité

L'étude de la stabilité des méthodes numériques se fait sur base d'une équation différentielle simple. Dans un premier temps, on détaille cette étude pour les méthodes d'Euler. Dans un deuxième temps, on explique pourquoi une équation différentielle simple suffit pour l'étude, et comment interpréter les conclusions de l'étude dans le cas d'équations plus complexes.

8.3.1 Stabilité absolue

La stabilité est étudiée sur base du problème linéaire et homogène suivant

$$\begin{cases} \frac{dy}{dt}(t) = -\beta y(t), & t > 0, \\ y(0) = y_0. \end{cases} \quad (8.9)$$

Contrairement aux sections précédentes, les valeurs de t ne sont plus confinées dans un intervalle. Le problème ci-dessus étant linéaire, homogène, et à coefficient constant, sa solution exacte vaut

$$y(t) = y_0 e^{-\beta t}.$$

Pour l'étude de la stabilité, seules les valeurs $\beta > 0$ nous intéressent, car la solution exacte tend alors vers zéro lorsque $t \rightarrow \infty$.

La stabilité d'une méthode numérique est caractérisée par le comportement de sa solution approchée du problème considéré. Plus précisément, une méthode de résolution est dite *absolument stable* (ou simplement *stable*) pour le problème de Cauchy (8.9) avec $\beta > 0$ si elle produit une séquence y_k , $k = 1, 2, \dots$, d'approximations de $y(t_k)$ telle que

$$y_k \rightarrow 0 \quad \text{lorsque} \quad t_k \rightarrow \infty.$$

En d'autres termes, le comportement asymptotique de la solution approchée doit reproduire celui de la solution exacte.

Pour simplifier la présentation, on suppose dans cette section que le pas de discrétisation h ne change pas d'une itération à l'autre.

8.3.2 Stabilité des méthodes d'Euler

Nous commençons l'étude de stabilité par la méthode d'Euler progressive. La relation de récurrence (8.4) devient pour le problème considéré

$$y_{k+1} = y_k(1 - h\beta), \quad (8.10)$$

ou encore, en appliquant cette relation k fois,

$$y_k = y_0(1 - h\beta)^k.$$

Trois cas de figure sont dès lors possibles selon la valeur de $h\beta$; ils sont illustrés à la Figure 8.5.

$$h\beta < 1$$

Dans ce cas on a

$$y_k = y_0 |1 - h\beta|^k \rightarrow 0$$

et donc la méthode est stable; y_k garde le même signe pour toutes les valeurs de k , et donc n'oscille pas.

$$1 \leq h\beta < 2$$

On a alors

$$y_k = y_0 |1 - h\beta|^k (-1)^k \rightarrow 0;$$

la méthode est toujours stable, mais la solution approchée devient oscillante.

$$h\beta \geq 2$$

Dans ce cas

$$y_k = y_0 |1 - h\beta|^k (-1)^k$$

diverge pour $h\beta > 2$, et $y_k = y_0 (-1)^k$ pour $h\beta = 2$. La méthode est donc instable.

En conclusion, on note que la méthode d'Euler progressive est absolument stable si

$$h\beta < 2.$$

Pour ce qui est de la méthode d'Euler rétrograde, la relation de récurrence (8.6) devient pour le problème considéré

$$y_{k+1} = y_k - h\beta y_{k+1}. \quad (8.11)$$

En isolant y_{k+1} on a

$$y_{k+1}(1 + h\beta) = y_k$$

et donc, en appliquant cette relation k fois,

$$y_k = y_0 \frac{1}{(1 + h\beta)^k}.$$

En particulier, comme h et β sont positifs, on a

$$\frac{1}{1 + h\beta} < 1,$$

et donc $y_k \rightarrow 0$ pour tout $h > 0$. La méthode d'Euler rétrograde est donc *toujours* absolument stable. Cette conclusion est illustrée à la Figure 8.6.

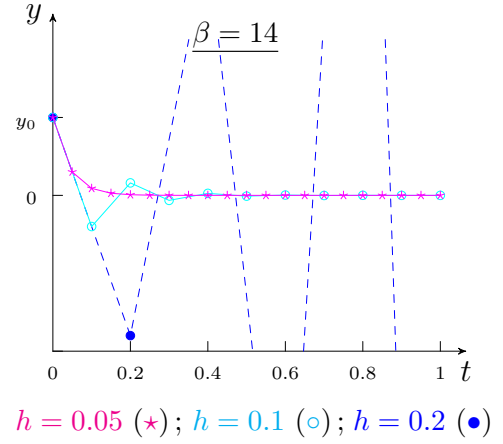


FIGURE 8.5 – Solution du problème (8.9) par la méthode d'Euler progressive pour $h\beta$ valant 0.7, 1.4 et 2.8.

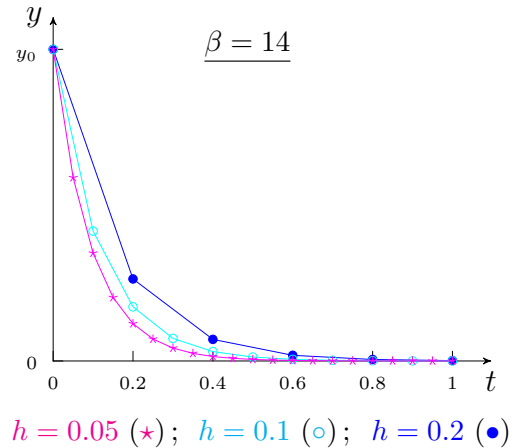


FIGURE 8.6 – Solution du problème (8.9) par la méthode d'Euler rétrograde pour $h\beta$ valant 0.7, 1.4 et 2.8.

8.3.3 Cas plus généraux

Nous expliquons maintenant pourquoi l'étude de stabilité faite sur base d'un problème simple (8.9) reste intéressant pour des cas plus compliqués, mais aussi comment interpréter les résultats de l'étude dans ces cas. L'explication est basée sur trois généralisations du problème de Cauchy (8.9), et les résultats précédents sont réexaminés dans le contexte de ces généralisations. Notons que d'autres généralisations sont possibles, non sans quelques précautions.

GÉNÉRALISATION 1 : PROBLÈME NON HOMOGÈNE

On commence par le problème non homogène

$$\begin{cases} \frac{dy}{dt}(t) = -\beta y(t) + b, & t > 0, \\ y(0) = y_0, \end{cases} \quad (8.12)$$

avec comme avant $\beta > 0$. Ce problème diffère du problème homogène (8.9) de départ par la présence du terme non homogène constant b . On note que le changement de variable $\tilde{y}(t) = y(t) - \frac{b}{\beta}$ permet de retrouver le problème de Cauchy de départ (avec une condition initiale modifiée). Dès lors, comme $\tilde{y}(t) \rightarrow 0$ pour $t \rightarrow \infty$, on a aussi $y(t) \rightarrow \frac{b}{\beta}$ pour $t \rightarrow \infty$.

De même, la récurrence (8.4) qui définit Euler progressive devient

$$y_{k+1} = y_k + h(-\beta y_k + b).$$

En utilisant un changement de variable similaire $\tilde{y}_k = y_k - \frac{b}{\beta}$ on retrouve aussi la récurrence (8.10) du cas homogène

$$\tilde{y}_{k+1} = \tilde{y}_k(1 - h\beta),$$

et par conséquent la même condition de stabilité. Ainsi, toute solution stable \tilde{y}_{k+1} tend vers 0 pour $t \rightarrow \infty$; on a donc aussi $y_{k+1} \rightarrow \frac{b}{\beta}$ pour $t \rightarrow \infty$.

En conclusion, la condition de stabilité permet de toujours s'assurer que la solution approchée tend vers la même valeur asymptotique que la solution exacte. Même si dans le cas non homogène cette valeur asymptotique, donnée par $\frac{b}{\beta}$, dépend du terme non homogène b , la condition de stabilité reste elle inchangé, et donc indépendante de ce dernier paramètre. Ces conclusions sont illustrées à la Figure 8.7.

Pour ce qui est d'Euler rétrograde, on arrive aux mêmes conclusions avec le même changement de variables.

GÉNÉRALISATION 2 : PROBLÈME SCALAIRE GÉNÉRAL

On considère à présent le problème de Cauchy scalaire en toute généralité, soit

$$\begin{cases} \frac{dy}{dt}(t) = f(t, y(t)), & t > 0, \\ y(0) = y_0. \end{cases}$$

Si la fonction f est suffisamment régulière, on peut l'approcher par son développement de Taylor en y au voisinage de (t_k, y_k) , ce qui donne

$$f(t, y) \approx f(t, y_k) + \frac{\partial f}{\partial y}(t, y_k)(y - y_k).$$

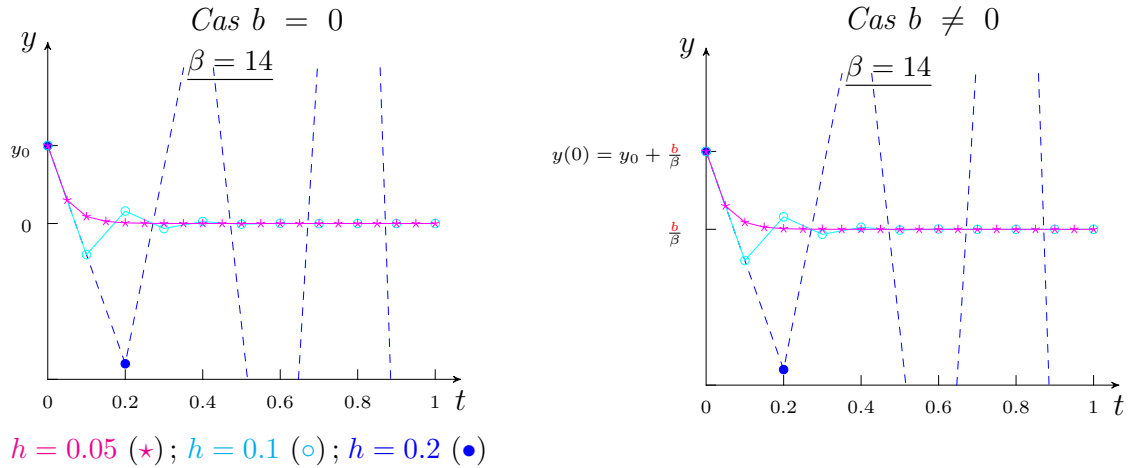


FIGURE 8.7 – Solution des problèmes (8.9) (gauche) et (8.12) (droite) par la méthode d'Euler rétrograde pour $h\beta$ valant 0.7, 1.4 et 2.8 ; pour renforcer la similitude entre les figures la condition initiale du problème (8.12) est modifiée et fixée à $y_0 + \frac{b}{\beta}$.

En notant $\beta(t) = -\frac{\partial f}{\partial y}(t, y_k)$ et $b(t) = f(t, y_k) + \beta(t)y_k$, l'approximation précédente peut être réécrite sous la forme suivante

$$f(t, y) \approx -\beta(t)y + b(t).$$

On en conclut que le problème non homogène approche le cas général sur un intervalle $I(t_k)$ autour de t_k où le développement de Taylor précédent reste une approximation valable et $\beta(t)$, $b(t)$ varient peu.

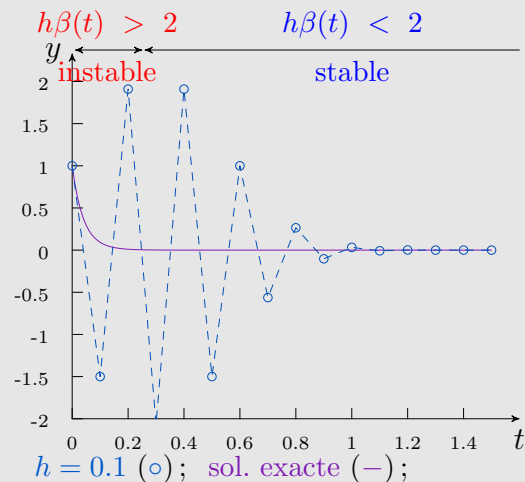
Ainsi, sous l'hypothèse que les conditions qu'on vient de mentionner sont d'application, on peut réutiliser l'analyse de stabilité développée dans la section 8.3.2 à condition d'utiliser $\beta(t)$ tel que défini plus haut. Néanmoins, l'interprétation des résultats ainsi obtenus a cette fois une portée locale, et non plus asymptotique. De plus, elle ne sera utile que si la méthode considérée effectue plusieurs pas sur $I(t_k)$. Comme avant, la perte de stabilité, même locale, peut néanmoins avoir un impact négatif sur la qualité de la solution approchée, comme le montre l'exemple suivant.

EXEMPLE 39.

Soit le problème de Cauchy

$$\begin{cases} \frac{dy}{dt}(t) = -\frac{25}{t+1}y(t), & t > 0, \\ y(0) = y_0. \end{cases}$$

Dans le cas considéré le membre de droite est déjà égal à son développement de Taylor en y ; les paramètres sont $b = 0$ et $\beta(t) = \frac{25}{t+1}$. La solution exacte de ce problème² ainsi que celle obtenue avec la méthode d'Euler progressive de pas $h = 0.1$ sont données sur la figure de droite.



On constate que même si la stabilité n'est perdue que localement, au début du processus de résolution, l'effet dû à cette perte est assez perceptible.

GÉNÉRALISATION 3 : PROBLÈME LINÉAIRE VECTORIEL À COEFFICIENTS CONSTANTS

On considère à présent le problème linéaire homogène vectoriel

$$\begin{cases} \frac{d\mathbf{y}}{dt}(t) = -A\mathbf{y}(t), & t > 0, \\ \mathbf{y}(0) = \mathbf{y}_0, \end{cases} \quad (8.13)$$

avec A une matrice $n \times n$ qui ne dépend pas de la variable t (et donc le problème est à coefficients constants). On suppose que A est diagonalisable, c'est-à-dire qu'il existe une matrice P inversible telle que $A = P^{-1}\text{diag}(\beta_i)P$, avec β_i , $i = 1, \dots, n$, les valeurs propres de A . Il est à noter que même si la matrice A est réelle, ses valeurs propres sont potentiellement complexes, ce qui mène à une présentation des résultats légèrement différente du cas scalaire.

Commençons par le problème considéré à proprement parler. Le changement de variables $\mathbf{z}(t) = P\mathbf{y}(t)$ le transforme en un système de n problèmes de Cauchy scalaires découplés de la forme

$$\begin{cases} \frac{dz_i}{dt}(t) = -\beta_i z_i(t), & t > 0, \\ z_i(0) = (P\mathbf{y}_0)_i, \end{cases}$$

avec $i = 1, \dots, n$. La solution du i -ème problème scalaire est donc $z_i(t) = z_i(0)e^{-\beta_i t}$, et elle tend vers zéro si $\text{Re}(\beta_i) > 0$. Dès lors, la solution du système initial $\mathbf{y}(t) = P^{-1}\mathbf{z}(t)$ tend vers zéro si et seulement si le vecteur $\mathbf{z}(t)$ des solutions des problèmes scalaires découplés tend vers zéro, et donc si

$$\text{Re}(\beta_i) > 0 \text{ pour } i = 1, \dots, n.$$

Par analogie avec le cas scalaire, l'étude de la stabilité sera donc limitée au cas où toutes les valeurs propres ont une partie réelle positive.

Pour ce qui est de la méthode d'*Euler progressive*, la récurrence (8.7) qui définit la méthode devient dans le cas vectoriel

$$\mathbf{y}_{k+1} = \mathbf{y}_k + h \underbrace{\mathbf{f}(t_k, \mathbf{y}_k)}_{-A\mathbf{y}_k} = (I - hA)\mathbf{y}_k.$$

En utilisant le changement de variables $\mathbf{z}_k = P\mathbf{y}_k$, ainsi que la décomposition $A = P^{-1}\text{diag}(\beta_i)P$, on obtient les n relations de récurrence suivantes

$$(\mathbf{z}_{k+1})_i = (1 - h\beta_i)(\mathbf{z}_k)_i, \quad i = 1, \dots, n.$$

2. La solution exacte de ce problème est $y(t) = \frac{y_0}{(t+1)^{25}}$.

Ces relations de récurrence sont identiques à la récurrence (8.10) du cas scalaire, avec les valeurs propres β_i de la matrice A jouant le rôle de la constante β . La condition de stabilité de la méthode d'Euler progressive, donnée dans ce cas par

$$|1 - h\beta_i| < 1 \quad \text{pour toute valeur propre } \beta_i \text{ de } A,$$

est donc aussi identique au cas scalaire, excepté qu'elle couvre maintenant aussi le cas des valeurs β_i complexes.

Les même raisonnement s'applique dans le cas de la méthode d'Euler rétrograde. Plus précisément, en appliquant la récurrence (8.8) d'Euler rétrograde au problème vectoriel considéré, on obtient

$$\mathbf{y}_{k+1} = \mathbf{y}_k + h \underbrace{\mathbf{f}(t_{k+1}, \mathbf{y}_{k+1})}_{-A\mathbf{y}_{k+1}} = \mathbf{y}_k - hA\mathbf{y}_{k+1}$$

et donc, en isolant \mathbf{y}_{k+1} , on a

$$\mathbf{y}_{k+1} = (I + hA)^{-1}\mathbf{y}_k.$$

En utilisant le changement de variables $\mathbf{z}_k = P\mathbf{y}_k$, ainsi que $A = P^{-1}\text{diag}(\beta_i)P$, on obtient n relations de récurrence

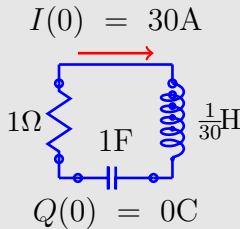
$$(\mathbf{z}_{k+1})_i = (\mathbf{z}_k)_i / (1 + h\beta_i), \quad i = 1, \dots, n.$$

Ce sont les mêmes relations de récurrence que la relation (8.11) dans le cas scalaire, excepté que maintenant les β_i sont potentiellement complexes. Le caractère complexe ne change néanmoins pas les conclusions obtenues dans le cas réel. En effet, pour $h > 0$ et $\text{Re}(\beta_i) > 0$ on a

$$\left| \frac{1}{1 + h\beta_i} \right| < 1,$$

et donc la méthode d'Euler rétrograde est stable pour toute valeur du pas h de discrétisation.

EXEMPLE 40. On reprend l'Exemple 3 de la Motivation (voir aussi l'Exemple 37 du présent chapitre) qui correspond à un système de deux équations différentielles



$$\begin{cases} \underbrace{\frac{d}{dt} \begin{pmatrix} I \\ Q \end{pmatrix}}_{\frac{d\mathbf{y}}{dt}(t)} = \underbrace{\begin{pmatrix} -30 & -30 \\ 1 & \end{pmatrix}}_{-A} \underbrace{\begin{pmatrix} I \\ Q \end{pmatrix}}_{\mathbf{y}(t)} \\ Q(0) = 0, \quad I(0) = 30, \end{cases}$$

lequel décrit le circuit RLC représenté ci-dessus. Les valeurs propres de la matrice A du système sont $\beta_1 \approx 29$ et $\beta_2 \approx 1$.

La solution exacte et les solutions approchées obtenues avec la méthode d'Euler progressive sont représentées à la Figure 8.8. La condition de stabilité est respectée pour

$h = 1/20$ (figure de gauche), mais pas pour $h = 1/14$ (figure de droite), ce qui a clairement un impact sur la qualité de la solution approchée.

Les résultats correspondants à la méthode d'Euler rétrograde pour les mêmes pas de discrétisation sont eux donnés à la Figure 8.9. On constate que, même si la partie rapidement variable de la solution (t entre 0 et 0.2) n'est pas toujours bien reproduite par la solution approchée, la reproduction de la partie qui varie lentement (t à partir de 0.2) est tout à fait satisfaisante. De manière plus générale, c'est un des avantages de la méthode d'Euler rétrograde que de pouvoir correctement reproduire les transitoires lentes sans que le pas de discrétisation h soit limité par la présence des transitoires rapides.

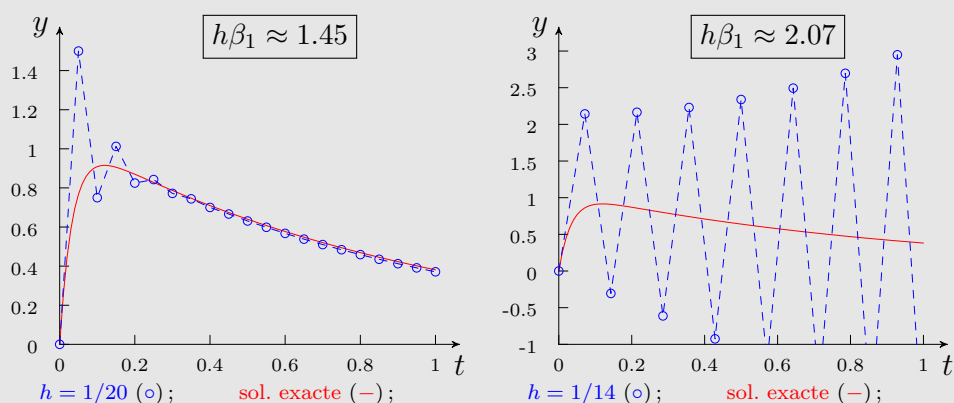


FIGURE 8.8 – Solution exacte (en trait continu), et solutions approchées (en trait interrompu) obtenues avec la méthode d'Euler **progressive** pour les pas $h = 1/20$ (gauche) et $h = 1/14$ (droite).

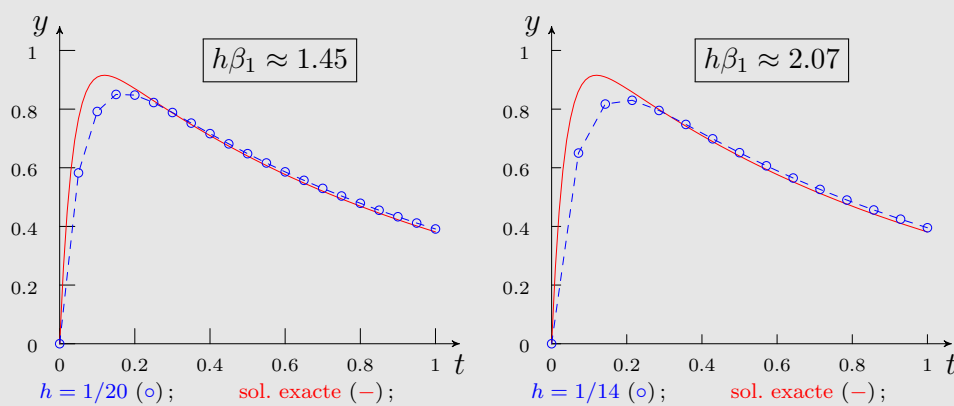


FIGURE 8.9 – Solution exacte (en trait continu), et solutions approchées (en trait interrompu) obtenues avec la méthode d'Euler **rétrograde** pour les pas $h = 1/20$ (gauche) et $h = 1/14$ (droite).

Le théorème qui suit permet de résumer les observations faites dans cette sous-section tout en les étendant au cas des matrices non diagonalisables. Il est donné ici

sans démonstration.

THÉORÈME 9. *Soit un problème de Cauchy vectoriel*

$$\begin{cases} \frac{dy}{dt}(t) = -A y(t), & t > 0, \\ y(0) = y_0, \end{cases}$$

où A est une matrice réelle d'ordre $n \times n$. Soient $\beta_i \in \mathbb{C}$, $i = 1, \dots, n$ les n valeurs propres de A .

Alors la solution exacte de ce problème est de la forme

$$y(t) = \sum_{i=1}^n c_i p_i(t) e^{-\beta_i t}, \quad c_i \in \mathbb{R}^n,$$

où $p_i(t)$ est un polynôme dont le degré est strictement inférieur à la multiplicité algébrique de β_i . De plus,

- a) la solution exacte $y(t)$ tend vers zéro si $\operatorname{Re}(\beta_i) > 0$;
- b) la méthode d'Euler rétrograde est absolument stable pour tout h ;
- c) la méthode d'Euler progressive est absolument stable si $|1 - h\beta_i| < 1$, $i = 1, \dots, n$.

Notons pour finir que l'extension du théorème précédent au problème de Cauchy vectoriel (8.3) non linéaire (en fait, même au problème linéaire où la matrice A dépend de la variable t) n'est pas aussi triviale que ce qui est fait dans la Généralisation 2, comme souligné dans l'Annexe 8.B.

8.4 Méthodes du second ordre

Les méthodes numériques considérés jusqu'à présent dans ce chapitre – celles d'Euler progressive et d'Euler rétrograde – sont des méthodes du premier ordre. Dans cette section on aborde deux méthodes du second ordre : celles de Crank-Nicolson et de Heun. Pour chacune de ces méthodes on obtient d'abord la récurrence qui la définit, et présente ensuite ses principales caractéristiques.

8.4.1 Méthode de Crank-Nicolson

La méthode de Crank-Nicolson (appelée aussi méthode du trapèze) est obtenue en appliquant la formule des trapèzes à l'équation intégrale (8.5). Plus précisément, la formule des trapèzes mène à l'approximation suivante

$$\int_{t_k}^{t_{k+1}} f(\tau, y(\tau)) d\tau \approx \frac{1}{2} h_k (f(t_k, y(t_k)) + f(t_{k+1}, y(t_{k+1}))) .$$

En utilisant cette dernière ainsi que $y_k \approx y(t_k)$, $y_{k+1} \approx y(t_{k+1})$ dans l'équation intégrale (8.5) on retrouve la récurrence pour la méthode de Crank-Nicolson

$$\boxed{y_{k+1} = y_k + \frac{1}{2}h_k (f(t_k, y_k) + f(t_{k+1}, y_{k+1}))}, \quad (8.14)$$

où, comme avant, $h_k = t_{k+1} - t_k$. Notons que la méthode est

- *implicite* (y_{k+1} est présent dans le membre de droite);
- du *second ordre*;
- *absolument stable* quel que soit le pas h_k (tout comme Euler rétrograde).

Les deux dernières caractéristiques sont illustrées avec les deux exemples suivants.

EXEMPLE 38. (SUITE) Reprenons le problème de Cauchy

$$\begin{cases} \frac{dy}{dt}(t) = \cos(y) - t, & t \geq 0, \\ y(0) = 0.7, \end{cases}$$

et résolvons-le par la méthode de Crank-Nicolson. Comme pour les méthodes d'Euler au début du chapitre on représente à la Figure 8.10 les solutions pour les différents pas de discrétisation (gauche) et une estimation de l'erreur sur la solution (droite). Notez que les pas d'intégration considérés sont deux fois plus grands que pour les méthodes d'Euler; malgré cela, la solution approchée reproduit fidèlement la solution exacte même pour le pas le plus grand. Notez aussi le comportement qualitatif de l'erreur : elle est divisée par un facteur 4 lorsque le pas h est divisé par un facteur 2; c'est une caractéristique d'une méthode de second ordre.

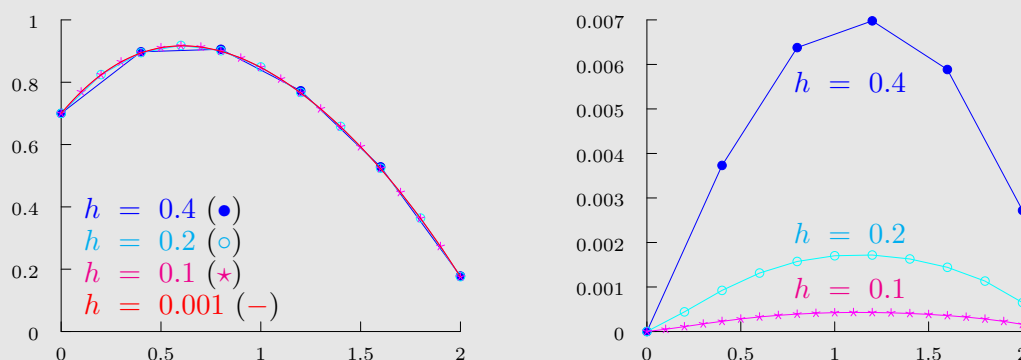


FIGURE 8.10 – Solutions approchées obtenues avec la méthode de Crank-Nicolson (gauche) et la différence (en valeur absolue) entre ces solutions et la solution approchée pour $h = 0.001$ qui fait office de solution exacte $y(t)$ (droite).

EXEMPLE 40. (SUITE) On continue à travailler avec le système d'équations différentielles qui décrit le circuit RLC. La Figure 8.11 représente les solutions approchées obtenues avec la méthode de Crank-Nicolson. La figure de gauche correspond à la solution obtenue en respectant la condition de stabilité d'Euler progressive alors que celle de droite correspond à la solution obtenue en ne respectant pas cette condition. Le comportement de la solution est stable dans les deux cas, en accord avec le fait que la méthode de Crank-Nicolson est stable pour toute valeur du pas h .

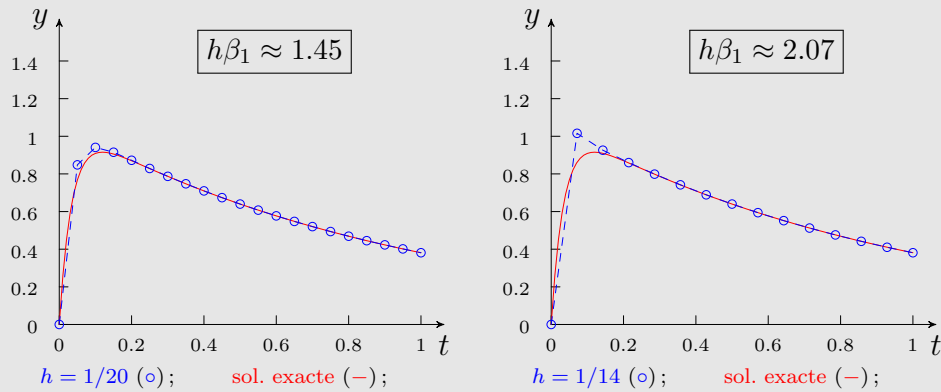


FIGURE 8.11 – Solution exacte (en trait continu), et solutions approchées (en trait interrompu) obtenues avec la méthode de Crank-Nicolson pour les pas $h = 1/20$ (gauche) et $h = 1/14$ (droite).

Le fait que la méthode de Crank-Nicolson soit absolument stable pour toute valeur du pas $h > 0$ n'implique pas pour autant qu'elle produise une solution acceptable pour tout h . En particulier, lorsque le pas h est bien supérieur aux valeurs du pas qui satisfont le critère de stabilité d'Euler progressive, la solution approchée obtenue avec la méthode de Crank-Nicolson peut être assez imprécise. Cette imprécision est caractérisée par des oscillations d'origine numérique à l'échelle du pas h . L'avantage de la méthode de Crank-Nicolson lié à son caractère absolument stable pour tout h est donc à relativiser, d'autant plus que la méthode d'Euler rétrograde n'a pas ce même défaut. L'exemple suivant illustre ce phénomène, alors que son explication est détaillée dans l'Annexe 8.C.

EXEMPLE 40. (SUITE) On reste avec le système d'équations différentielles pour le circuit RLC. La Figure 8.12 illustre les solutions approchées obtenues avec les méthodes de Crank-Nicolson (gauche) et d'Euler rétrograde (droite) pour le pas $h = 1/4$ (c'est-à-dire $h\beta \approx 7.5$, et donc loin du seuil associé au critère de stabilité d'Euler progressive). On note la présence d'oscillations dans la solution approchée obtenue avec la méthodes de Crank-Nicolson, alors que de telles oscillations ne sont pas présentes ni dans la solution exacte, ni dans la solution numérique obtenue avec la méthode d'Euler rétrograde.

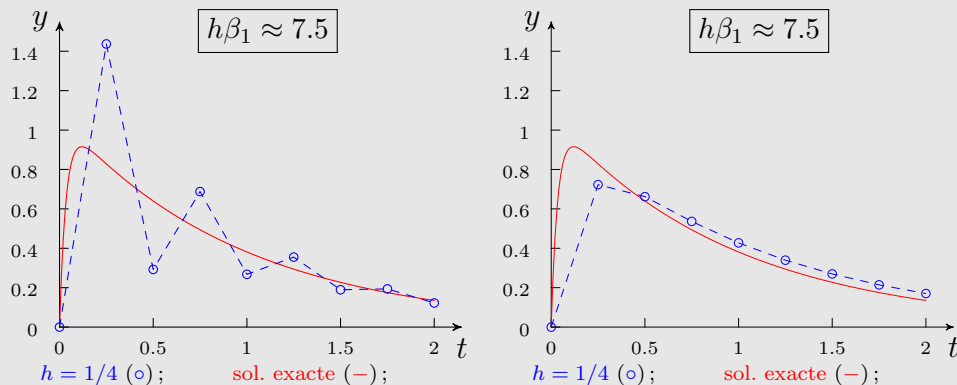


FIGURE 8.12 – Solution exacte (en trait continu), et solutions approchées (en trait interrompu) obtenues avec les méthodes de Crank-Nicolson (gauche) et d'Euler rétrograde (droite) pour le pas $h = 1/4$.

8.4.2 Méthode de Heun

La méthode de *Heun* (ou de *Runge-Kutta* d'ordre 2) est une méthode qu'on obtient lorsqu'on rend la méthode de Crank-Nicolson explicite à l'aide de la méthode d'Euler progressive. Plus précisément, en remplaçant dans le membre de droite de (8.14) le terme y_{k+1} par la formule $y_k + h_k f(t_k, y_k)$ d'Euler progressive on obtient

$$y_{k+1} = y_k + \frac{1}{2} h_k (f(t_k, y_k) + f(t_{k+1}, y_k + h_k f(t_k, y_k))) ;$$

cette récurrence définit la méthode de Heun. Notons que cette méthode est :

- *explicite* (par construction) ;
- du *second ordre* ;
- dans le cas où β est réel³, la méthode est *absolument stable* si $h\beta < 2$ (tout comme la méthode d'Euler progressive).

EXEMPLE 38. (FIN) On reprend ici le même problème de Cauchy que dans les parties précédentes de l'exemple. Les solutions approchées obtenues avec la méthode de Heun sont données à la Figure 8.13 à gauche, alors que l'estimation de l'erreur est à droite. On note que la méthode est un peu moins précise que celle de Crank-Nicolson (comparez les Figures 8.10 et 8.13), mais la dépendance en h de l'erreur est celle d'une méthode du second ordre.

3. Voir section 8.3 pour les notations.

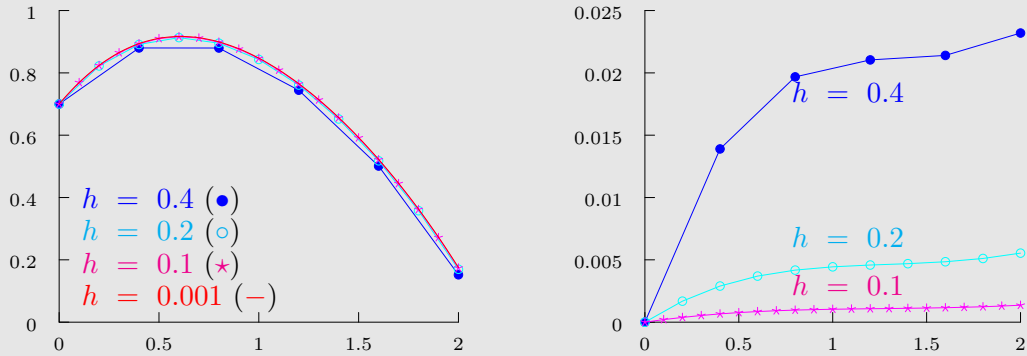


FIGURE 8.13 – Solutions approchées obtenues avec la méthode de Heun (gauche) et la différence (en valeur absolue) entre ces solutions et la solution approchée pour $h = 0.001$ qui fait office de solution exacte $y(t)$ (droite).

EXEMPLE 40. (FIN) On reprend ici le même problème de Cauchy vectoriel – circuit RLC – que dans les parties précédentes de l'exemple. Les solutions approchées du problème obtenues avec la méthode de Heun sont données à la Figure 8.14. On note que le respect de la condition de stabilité d'Euler progressive (figure de gauche) mène à une solution stable, alors que le non respect de cette condition (figure de droite) donne une solution instable.

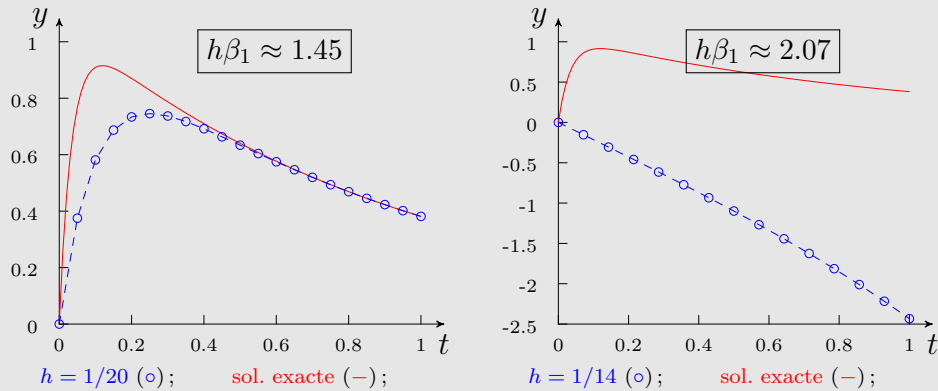


FIGURE 8.14 – Solution exacte (en trait continu), et solutions approchées (en trait interrompu) obtenues avec la méthode de Heun pour les pas $h = 1/20$ (gauche) et $h = 1/14$ (droite).

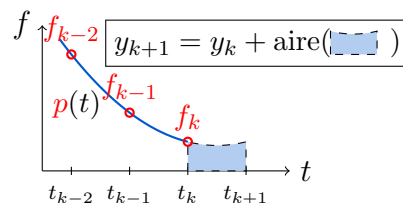
8.5 Méthodes multi-pas

Les méthodes d'Euler, de Crank-Nicolson et de Heun ont en commun le fait qu'elles ne relient entre elles que deux approximations successives de la fonction inconnue, à savoir y_{k+1} et y_k ; il s'agit de méthodes à un pas. Une méthode à $q + 1$ pas (appelée méthode

multi-pas) permet de relier y_{k+1} à $y_k, y_{k-1}, \dots, y_{k-q}$. Les méthodes multi-pas principales sont :

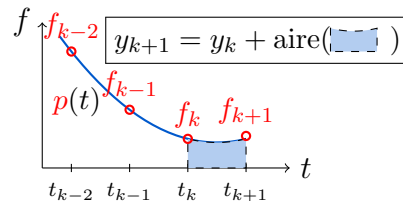
MÉTHODES D'ADAMS-BASHFORTH

Ces méthodes consistent à approcher $f(t, y(t))$ par un polynôme d'interpolation $p(t)$ passant par $f(y_k, t_k), f(y_{k-1}, t_{k-1}), \dots, f(y_{k-p}, t_{k-p})$.



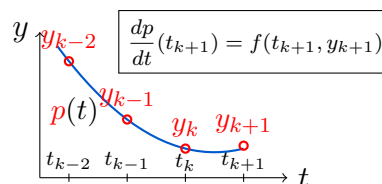
MÉTHODES D'ADAMS-MOULTON

Ces méthodes consistent à approcher $f(t, y(t))$ par un polynôme d'interpolation $p(t)$ passant par $f(y_{k+1}, t_{k+1}), f(y_k, t_k), \dots, f(y_{k-q}, t_{k-q})$.



MÉTHODES BDF

Ces méthodes consistent à approcher $y(t)$ par un polynôme d'interpolation $p(t)$ passant par $y_{k+1}, y_k, \dots, y_{k-q}$.



Annexe 8.A Régions de stabilité

La *région de stabilité* pour une méthode numérique donnée est l'ensemble des points dans le plan complexe qui correspondent aux valeurs de $-h\beta$ pour laquelle la méthode est absolument stable.

Les régions de stabilité des méthodes d'Euler rétrograde et de Crank-Nicolson comprennent le demi-plan gauche du plan complexe. Pour ce qui est des méthodes d'Euler progressive et de Heun, leur région de stabilité est représentée à la Figure 8.15.

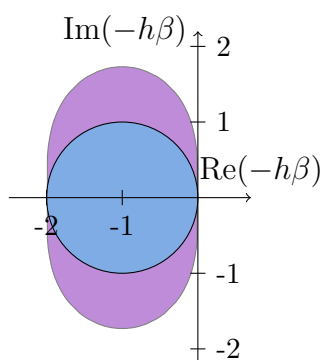


FIGURE 8.15 – Régions de stabilité des méthodes d'Euler progressive (région en forme de disque) et de Heun (région ovale).

Annexe 8.B Compléments sur la stabilité

Le but de cette annexe est d'illustrer les limites de l'analyse effectuée dans la Section 8.3 de ce chapitre, et de généraliser la notion de stabilité au cas des problèmes de Cauchy vectoriels non linéaires.

En effet, l'analyse effectuée dans la Section 8.3 ne se généralise pas directement aux problèmes de Cauchy vectoriels (8.3) non linéaires. En particulier, les Généralisations 1-3 peuvent laisser croire que pour analyser la stabilité d'un problème de Cauchy vectoriel (8.3) il suffit d'approcher localement (comme dans la Généralisation 2) la fonction $\mathbf{f}(t, \mathbf{y})$ au voisinage de \mathbf{y}_k par son développement de Taylor d'ordre 1 par rapport à \mathbf{y} , et d'analyser (à l'instar de ce qui est fait dans la Généralisation 3) les valeurs propres de (l'opposé de) la matrice jacobienne $(\frac{\partial f_i}{\partial x_j})$ qui apparaît dans le terme linéaire du développement. L'exemple suivant montre que cette démarche a ses limites.

EXEMPLE 41. (QUARTERONI & AL.)

Le problème de Cauchy vectoriel (dont la condition initiale est choisie en $t = 1$ pour simplifier la suite)

$$\begin{cases} \frac{d\mathbf{y}}{dt}(t) = -A(t)\mathbf{y}(t), & t \geq 1, \\ \mathbf{y}(1) = \mathbf{y}_1, \end{cases}$$

avec

$$A(t) = \begin{pmatrix} t^{-1}/2 & -2t^{-3} \\ t/2 & t^{-1}/2 \end{pmatrix},$$

est déjà linéaire et homogène, et la fonction $\mathbf{f}(t, \mathbf{y}) = -A(t)\mathbf{y}$ est égale à son propre développement de Taylor par rapport à \mathbf{y} . Néanmoins, ce problème n'est pas à coefficients constants, car la matrice $A(t)$ dépend de la variable t .

Les valeurs propres de la matrice $A(t)$ sont $\{\beta_1, \beta_2\} = \{(1 \pm 2i)t^{-1}/2\}$, et la partie réelle de ces valeurs propres est positive. Cela mène à croire que la solution du problème considéré est décroissante. Or, on peut vérifier que

$$\mathbf{y}(t) = \begin{pmatrix} t^{-3/2} \\ t^{1/2}/2 \end{pmatrix}$$

est la solution du problème pour la condition initiale $\mathbf{y}_1 = (1, 1/2)$, et la deuxième composante de cette solution (ainsi que $\|\mathbf{y}(t)\|_2$ pour $t > 2$) est bien croissante pour un t croissant.

Pour expliquer d'où vient cette surprise (du moins, dans ce cas-ci), il faut noter que la matrice $P(t)$ de changement de base dans la décomposition en valeurs propres $A(t) = P(t)^{-1}\text{diag}(\beta_i)P(t)$ dépend de t , contrairement à la matrice P de changement de base dans la Généralisation 3 (pourquoi cela est important?).

Indiquons maintenant ce qu'on entend dans cette annexe par une équation différen-

tielle (ED)

$$\frac{d\mathbf{y}}{dt}(t) = \mathbf{f}(t, \mathbf{y}(t)) \quad (8.B.1)$$

stable. On dit qu'une ED est *stable* (ou encore *contractive*) si pour deux solutions \mathbf{y} et \mathbf{v} de cette ED correspondant à deux conditions initiales différentes on a

$$\|\mathbf{y}(t) - \mathbf{v}(t)\|_2 \leq \|\mathbf{y}(t_0) - \mathbf{v}(t_0)\|_2$$

pour tout $t > t_0$; pour rappel, $\|\cdot\|_2$ est la norme euclidienne. Dit autrement, l'ED est stable si la différence (en norme euclidienne) entre ses deux solutions ne fait que diminuer avec le temps t . Comme les solutions \mathbf{y} et \mathbf{v} sont supposées de classe $C^1([0, T])$, cela implique en particulier (en dérivant par rapport à t la fonction monotone décroissante $\|\mathbf{y}(t) - \mathbf{v}(t)\|_2^2$ et en utilisant (8.B.1)) que

$$(\mathbf{y}(t) - \mathbf{v}(t))^T (\mathbf{f}(t, \mathbf{y}) - \mathbf{f}(t, \mathbf{v})) \leq 0. \quad (8.B.2)$$

Notons que l'inégalité (8.B.2) peut être vérifiée sans exiger que \mathbf{y} et \mathbf{v} soient des solutions de l'ED (8.B.1), ce qui rend son utilisation plus facile.

Pour ce qui est de la stabilité des méthodes numériques, une méthode est dite *BN-stable* si deux solutions approchées \mathbf{y}_k et \mathbf{v}_k , $k = 0, \dots, m$, obtenues en appliquant cette méthode (avec \mathbf{y}_0 et \mathbf{v}_0 comme conditions initiales) à une ED (8.B.1) qui vérifie (8.B.2) satisfont

$$\|\mathbf{y}_{k+1} - \mathbf{v}_{k+1}\|_2 \leq \|\mathbf{y}_k - \mathbf{v}_k\|_2,$$

pour tout $k = 0, \dots, m-1$.

Le résultat suivant indique alors que la méthode d'Euler rétrograde est BN-stable.

THÉORÈME 10. *La méthode d'Euler rétrograde est BN-stable.*

Démonstration. Les récurrences de la méthode d'Euler rétrograde pour deux conditions initiales \mathbf{y}_0 et \mathbf{v}_0 satisfont

$$\begin{aligned} \mathbf{y}_{k+1} &= \mathbf{y}_k + h_k \mathbf{f}(t_{k+1}, \mathbf{y}_{k+1}), \\ \mathbf{v}_{k+1} &= \mathbf{v}_k + h_k \mathbf{f}(t_{k+1}, \mathbf{v}_{k+1}), \end{aligned}$$

$k = 1, \dots, m-1$, et donc, en mettant les termes avec l'indice k à gauche et ceux avec l'indice $k+1$ à droite, on obtient

$$\begin{aligned} \mathbf{y}_k &= \mathbf{y}_{k+1} - h_k \mathbf{f}(t_{k+1}, \mathbf{y}_{k+1}), \\ \mathbf{v}_k &= \mathbf{v}_{k+1} - h_k \mathbf{f}(t_{k+1}, \mathbf{v}_{k+1}). \end{aligned}$$

En utilisant ces dernières expressions ainsi que (8.B.2) on a

$$\begin{aligned}
\|\mathbf{y}_{k+1} - \mathbf{v}_{k+1}\|_2^2 - \|\mathbf{y}_k - \mathbf{v}_k\|_2^2 &= \underbrace{-h_k^2 \|\mathbf{f}(t_{k+1}, \mathbf{y}_{k+1}) - \mathbf{f}(t_{k+1}, \mathbf{v}_{k+1})\|_2^2}_{\leq 0} \\
&\quad + 2h_k \underbrace{(\mathbf{y}_{k+1} - \mathbf{v}_{k+1})^T (\mathbf{f}(t_{k+1}, \mathbf{y}_{k+1}) - \mathbf{f}(t_{k+1}, \mathbf{v}_{k+1}))}_{\leq 0 \text{ par (8.B.2)}} \\
&\leq 0. \quad \blacksquare
\end{aligned}$$

Notons (sans le démontrer) que la BN-stabilité n'est pas équivalente à la stabilité absolue (sans conditions) telle que définie dans la Section 8.3. En particulier, la méthode de Crank-Nicolson est absolument stable sans conditions mais n'est pas BN-stable. Notons aussi une légère inconsistance dans les deux définitions de stabilité : l'une exige que l'écart entre deux solutions n'augmente pas, alors que l'autre implique (en choisissant $\beta > 0$) que ce même écart (dans le cas du problème de Cauchy (8.9)) tend vers zéro.

Pour ce qui est du cas scalaire, notons que la condition (8.B.2) pour deux scalaires v et y devient

$$(f(t, y) - f(t, v))(y - v) \leq 0,$$

ce qui est équivalent au fait que la fonction $f(t, y)$ est décroissante en y . D'un autre côté, si la fonction $f(t, y)$ est de classe C^1 en y , cette décroissance implique que $\frac{\partial f}{\partial y} \leq 0$. Cette observation est consistante avec la Généralisation 2 de la Section 8.3, où on approche $\beta(t)$ (qui doit être strictement positif pour une ED stable) par $-\frac{\partial f}{\partial y}$.

Annexe 8.C Qualité d'approximation en fonction de $h\beta$

Dans cette annexe, on considère de nouveau le problème de Cauchy scalaire (8.9) et on examine la qualité d'approximation des méthodes numériques en fonction du paramètre $h\beta$. La solution exacte du problème étant une exponentielle, les différentes méthodes vues dans ce chapitre peuvent être considérées comme des approximations de celle-ci.

Plus précisément, la solution exacte du problème (8.9) est donnée par $y(t) = y_0 e^{-\beta t}$ qui, aux instants $t_k = hk$, $k = 0, 1, \dots$, est donnée par

$$y(t_k) = y_0 e^{-\beta hk} = y_0 (e^{-h\beta})^k.$$

Comme dans le cas du problème (8.9), les solutions approchées obtenues avec les méthodes numériques considérées auront toutes la forme $y_0 (a(h\beta))^k$, on ne s'intéresse ici qu'à la qualité de l'approximation de l'exponentielle $e^{-h\beta}$ par un tel facteur $a(h\beta)$.

Dans le cas de la méthode d'Euler rétrograde, la solution approchée satisfait (voir section 8.3.2)

$$y_k = y_0 \left(\frac{1}{1 + h\beta} \right)^k,$$

et donc $a_{\text{ER}}(h\beta) = 1/(1 + h\beta)$. Pour la méthode de Crank-Nicolson un calcul similaire à celui effectué pour les méthodes d'Euler dans la section 8.3.2 permet de montrer que

$$y_k = y_0 \left(\frac{1 - h\beta/2}{1 + h\beta/2} \right)^k,$$

et donc $a_{\text{CN}}(h\beta) = (1 - h\beta/2)/(1 + h\beta/2)$.

L'exponentielle $e^{-h\beta}$ et les deux facteurs sont représentés sur la Figure 8.16 en fonction de $h\beta$. On constate en particulier que le facteur $a_{\text{CN}}(h\beta)$ est une meilleure approximation de $e^{-h\beta}$ pour les petites valeurs de $h\beta$, mais devient moins précis à partir de $h\beta \approx 2.6$. La méthode de Crank-Nicolson peut donc être moins précise que la méthode d'Euler rétrograde pour des grandes valeurs de $h\beta$. On notera par ailleurs que $a_{\text{CN}}(h\beta)$ est proche de -1 pour des grandes valeurs de $h\beta$, ce qui explique la présence des oscillations. Ces dernières peuvent décroître relativement lentement malgré le fait qu'elles soient dues à des transitoires rapides dans la solution. Le fait que les deux méthodes soient absolument stables pour tout h est aussi reflété dans la figure car $a_{\text{ER}}(h\beta)$ et $a_{\text{CN}}(h\beta)$ sont tous les deux compris entre -1 et 1 .

A titre d'exercice, le lecteur peut identifier les facteurs $a(h\beta)$ associés aux méthodes d'Euler progressive et de Heun et les visualiser en fonction de $h\beta$.

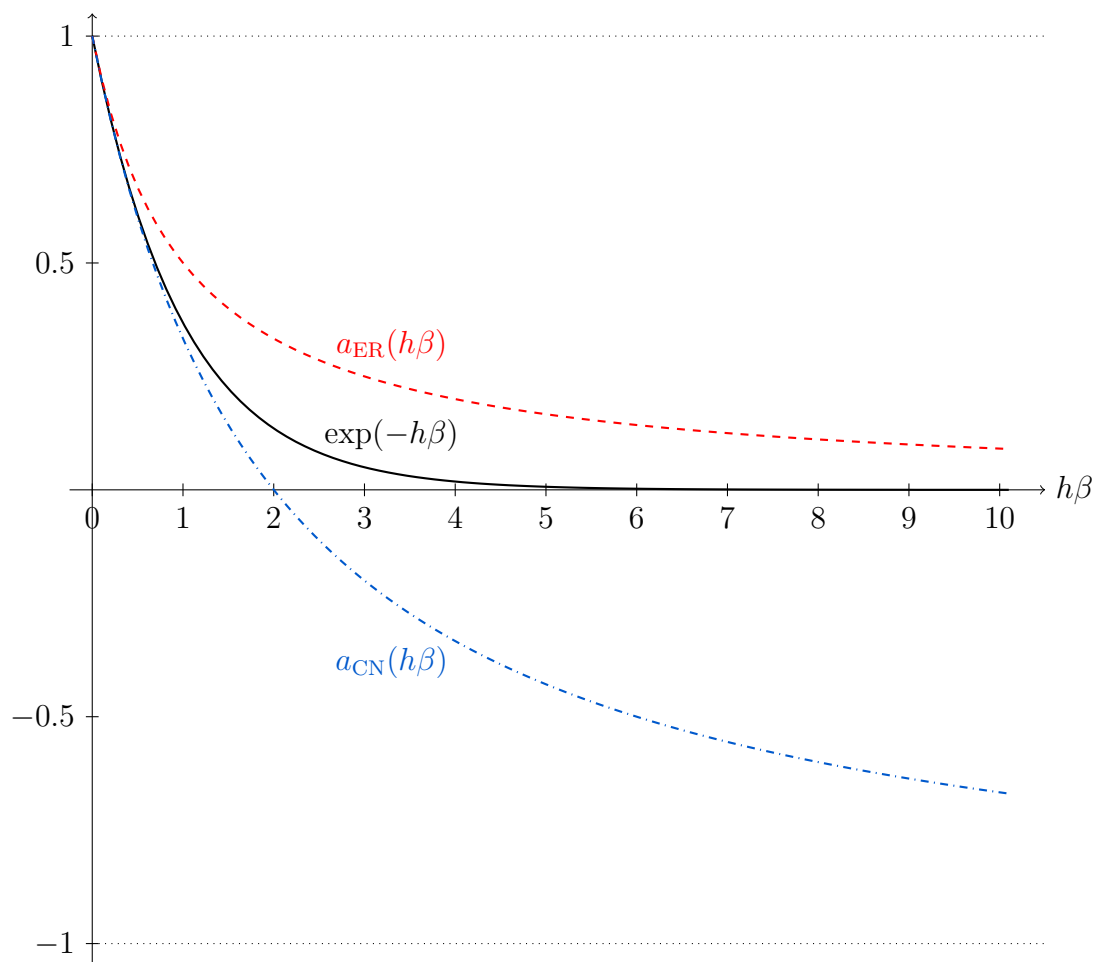


FIGURE 8.16 – L'exponentielle $e^{-h\beta}$ (trait continu) et ses approximations $a_{\text{ER}}(h\beta) = 1/(1 + h\beta)$ (trait interrompu) et $a_{\text{CN}}(h\beta) = (1 - h\beta/2)/(1 + h\beta/2)$ (trait mixte) en fonction de $h\beta$.

Équations différentielles avec conditions aux limites

Ce chapitre porte sur les problèmes aux limites à une dimension, qui sont composés d'une équation différentielle ordinaire avec deux conditions aux limites. Plus précisément, on y considère deux problèmes aux limites particuliers : le problème de Poisson, et celui de convection-diffusion. Pour les deux problèmes, l'équation différentielle est une équation linéaire d'ordre deux. De plus, on se limite aux cas où l'équation différentielle est définie sur un intervalle ; les conditions aux limites sont alors imposées aux extrémités de celui-ci. Après quelques commentaires sur l'existence, l'unicité, et le principe de résolution numérique, on présente plus en détail les méthodes de résolution approchée des deux problèmes.

9.1 Généralités

9.1.1 Problème aux limites

Un problème aux limites est composé d'équations différentielles et de conditions aux limites. Ici on considère une équation différentielle linéaire d'ordre deux définie sur l'intervalle $[0, L]$; elle relie sur cet intervalle une fonction f donnée avec (une combinaison linéaire de) la fonction inconnue y , sa dérivée première $\frac{\partial y}{\partial x}$ et sa dérivée seconde $\frac{\partial^2 y}{\partial x^2}$. Notez en particulier que la variable dont dépendent les fonctions f et y est notée x , et non t comme au chapitre précédent ; ce choix est motivé par l'interprétation typique du problème aux limites considéré ici, qui correspond plutôt à une loi de distribution dans l'espace qu'une évolution dans le temps. Pour ce qui est des conditions aux limites, elles sont imposées aux extrémités 0 et L de l'intervalle considéré (une condition par extrémité), en y spécifiant la valeur de la fonction inconnue y ou de sa dérivée $\frac{\partial y}{\partial x}$.

Plus précisément, le problème aux limites considéré consiste à déterminer une fonction $y(x) \in C^2([0, L])$ qui satisfait

$$\begin{cases} -\frac{d^2 y}{dx^2} + p(x)\frac{dy}{dx} + q(x)y = f(x), & x \in [0, L], & (\text{équation différentielle}) \\ \text{cl}(0) = c_0, \text{cl}(L) = c_L. & & (\text{conditions aux limites}) \end{cases} \quad (9.1)$$

Ici, $f(x), p(x), q(x) : [0, L] \mapsto \mathbb{R}$ sont des fonctions continues données, c_0 et c_L sont deux réels donnés, et $\text{cl}(x)$ correspond ici à une des deux options suivantes pour les conditions aux limites

$$\text{cl}(x) = \begin{cases} y(x) & (\text{condition de Dirichlet}) ; \\ \frac{dy}{dx}(x) & (\text{condition de Neumann}). \end{cases} \quad (9.2)$$

Notez que l'équation différentielle linéaire d'ordre deux dans (9.1) peut aussi être utilisée dans un problème de Cauchy (voir, par exemple, section 8.1.3). La différence vient alors du fait que les conditions sont dans ce cas-ci imposées aux deux extrémités de l'intervalle sur lequel la solution est recherchée (une condition par extrémité), et non à une seule extrémité, comme c'est le cas pour les problèmes de Cauchy.

9.1.2 Existence, unicité et cas particuliers

Le théorème suivant énumère les conditions sous lesquelles le problème aux limites (9.1) possède une et une seule solution. Rappelons qu'il est important de s'assurer de l'existence et de l'unicité de la solution, sans quoi la solution approchée produite par une méthode numérique fournira des informations potentiellement incomplètes (par exemple, en cas de non unicité) ou erronées (par exemple, en cas de non existence) sur l'ensemble des solutions.

THÉORÈME 11. *Soit le problème aux limites (9.1) décrit dans la section 9.1.1. Si les fonctions $f(x), p(x), q(x)$ sont continues, $q(x)$ est plus grande ou égale à zéro sur $[0, L]$, et*

- *soit les conditions aux limites sont de type Dirichlet,*
- *soit $p(x) = 0$ et au plus une des deux conditions est de type Neumann,*

alors le problème aux limites (9.1) admet une et une seule solution.

Dans la suite de ce chapitre on se limite à deux problèmes aux limites particuliers : celui de Poisson, et celui de convection-diffusion. Ces problèmes sont définis plus loin. Notez que le théorème précédent est vérifié dans les deux cas, à condition de considérer des fonctions $p(x)$ et $f(x)$ continues, et de se limiter à au plus une condition de type Neumann.

PROBLÈME DE POISSON :

$$\begin{cases} -\frac{d^2 y}{dx^2}(x) = f(x), & x \in [0, L], \\ \text{cl}(0) = c_0, \text{cl}(L) = c_L. \end{cases} \quad (9.3)$$

PROBLÈME DE CONVECTION-DIFFUSION :

$$\begin{cases} -\frac{d^2 y}{dx^2}(x) + p(x)\frac{dy}{dx}(x) = f(x), & x \in [0, L], \\ y(0) = c_0, y(L) = c_L. \end{cases} \quad (9.4)$$

EXEMPLE 42. Cet exemple illustre pourquoi le problème de Poisson avec deux conditions de type Neumann n'a pas une et une seule solution. Plus précisément, selon les valeurs de c_0 et c_L ce problème soit possède une infinité de solutions, soit n'en a aucune. Pour illustrer ce point, on choisit $f(x) = 0$, ce qui réduit le problème aux limites ainsi considéré à

$$\begin{cases} -\frac{d^2y}{dx^2}(x) = 0, & x \in [0, L], \\ \frac{dy}{dx}(0) = c_0, \quad \frac{dy}{dx}(L) = c_L. \end{cases}$$

La solution de l'équation différentielle est alors de la forme

$$y = ax + b,$$

avec $a, b \in \mathbb{R}$. De plus, les deux conditions aux limites deviennent, respectivement, $a = c_0$ et $a = c_L$. Dès lors, si $c_0 = c_L$, tout fonction de la forme

$$y = c_0x + \text{const}$$

est solution de ce problème. Par contre, si $c_0 \neq c_L$, les deux conditions aux limites ne peuvent pas être satisfaites simultanément, et le problème n'a pas de solution.

9.1.3 Principe de résolution numérique

Les méthodes numériques ne fournissent en général qu'une solution approchée du problème aux limites considéré. En particulier, la méthode des *différences finies* consiste à approcher la solution exacte $y(x)$ sur l'intervalle $[0, L]$ aux $m + 1$ points

$$0 = x_0 < x_1 < \dots < x_m = L,$$

par y_0, y_1, \dots, y_m . On suppose pour simplifier que $x_k = kh$, $k = 0, \dots, m$, avec

$$h = L/m$$

le *pas de discrétisation*.

Typiquement, une solution approchée est obtenue en résolvant un système linéaire qui représente une version discrète du problème aux limites. Avec la méthode des différences finies, la plupart des équations dans un tel système linéaire sont obtenues en approchant les dérivées dans l'équation différentielle par des formules de différences finies.

Pour les dérivées d'ordre un et deux on reprend plus loin les quelques formules de différences finies utilisées dans ce cours. La justification de ces formules est donnée en Annexe 9.A.

Pour ce qui est de la *dérivée première*, les trois formules de différences finies suivantes sont de loin les plus courantes. Les termes d'erreur sont valables si la fonction $y(x)$ est de classe C^2 (pour les deux premières formules) ou C^3 (pour la dernière formule). La Figure 9.2 à droite illustre ces formules.

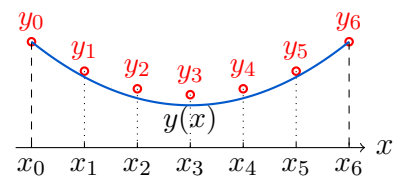


FIGURE 9.1 – Exemple d'une solution numérique.

— Différence progressive :

$$\frac{dy}{dx}(x) = \frac{y(x+h) - y(x)}{h} + \mathcal{O}(h)$$

— Différence rétrograde :

$$\frac{dy}{dx}(x) = \frac{y(x) - y(x-h)}{h} + \mathcal{O}(h)$$

— Différence centrée :

$$\frac{dy}{dx}(x) = \frac{y(x+h) - y(x-h)}{2h} + \mathcal{O}(h^2)$$

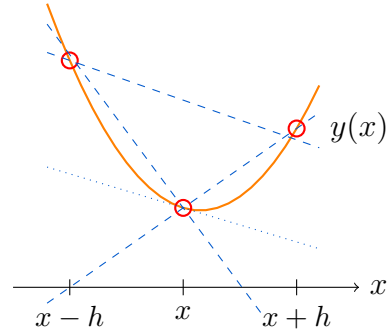


FIGURE 9.2 – Tangente en x (en pointillé) et sécantes correspondantes aux formules de différences finies en x (trait interrompu).

Pour ce qui est de la *dérivée seconde*, la seule formule qui nous est utile est la formule de *différences centrées* suivante. Le terme d'erreur est valable si la fonction $y(x)$ est de classe C^4 .

$$\frac{d^2y}{dx^2}(x) = \frac{y(x+h) + y(x-h) - 2y(x)}{h^2} + \mathcal{O}(h^2) \quad (9.5)$$

Finalement, pour ce qui est des détails de la méthode de différences finies pour les problèmes de Poisson et de convection-diffusion, ils sont présentés dans la suite de ce chapitre.

9.2 Problème de Poisson

Le problème de Poisson (9.3) est repris ici pour rappel :

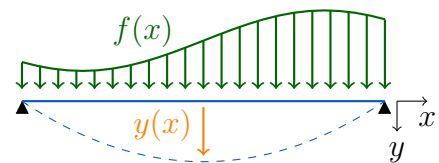
$$\begin{cases} -\frac{d^2y}{dx^2}(x) = f(x), & x \in [0, L], \\ \text{cl}(0) = c_0, \text{cl}(L) = c_L, \end{cases}$$

avec $\text{cl}(x)$ donné par (9.2).

9.2.1 Applications

Les différentes variantes du problème de Poisson, principalement à deux et trois dimensions, forment une famille de problèmes aux limites particulièrement utiles. Ils interviennent dans la modélisation de plusieurs problèmes d'ingénieur, dont deux applications unidimensionnelles sont brièvement présentées ci-dessous.

APPLICATION 1 : Le problème de Poisson décrit le déplacement $y(x)$ d'une corde de longueur L (longueur au repos) soumise à une densité de charge $f(x)$. Si la corde est fixée aux deux extrémités (par exemple, à la même hauteur), cette fixation est décrite par les conditions aux limites de Dirichlet (par exemple, $y(0) = y(L) = 0$).



APPLICATION 2 : Le problème de Poisson (9.3) décrit aussi la distribution stationnaire de température $y(x)$ dans une barre homogène; $f(x)$ est alors proportionnelle à la production de chaleur linéique. La condition de Dirichlet $y = c$ revient à fixer la température c à une des extrémités; la condition de Neumann $\frac{dy}{dx} = c'$ permet d'imposer un flux de chaleur à travers celle-ci.



9.2.2 Approximation aux points intérieurs

Rappelons que la méthode des différences finies consiste à approcher le problème aux limites par un système d'équations linéaires. Plus précisément, on associe à chaque point x_k du domaine, $k = 0, \dots, m$, une équation dans ce système linéaire. Pour un point x_k intérieur (avec donc $k \neq 0, m$), l'équation linéaire est obtenue en approchant la dérivée seconde dans l'équation différentielle (9.3) par la formule (9.5) de différences centrées :

$$f(x_k) = -\frac{d^2y}{dx^2}(x_k) \approx \frac{-y_{k-1} + 2y_k - y_{k+1}}{h^2}. \quad (9.6)$$

Cela permet d'obtenir $m - 1$ équations linéaires reliant les différentes valeurs de y_k , $k = 0, \dots, m$, entre elles.

EXEMPLE 43. Pour les points x_0, \dots, x_6 représentés à la Figure 9.1 les $m - 1 = 5$ équations sont¹

$$\left\{ \begin{array}{llllll} -y_0 & +2y_1 & -y_2 & & & = h^2 f(x_1) \\ & -y_1 & +2y_2 & -y_3 & & = h^2 f(x_2) \\ & & -y_2 & +2y_3 & -y_4 & = h^2 f(x_3) \\ & & & -y_3 & +2y_4 & -y_5 = h^2 f(x_4) \\ & & & & -y_4 & +2y_5 -y_6 = h^2 f(x_5) \end{array} \right.$$

Notez que le nombre d'inconnues (ici, 7) est supérieur au nombre d'équations (ici, 5). Ceci est dû au fait qu'on n'a pas exploré les conditions aux limites. Autrement dit, on n'a pas construit les équations correspondant aux points x_0 et x_6 . Notez aussi que la procédure utilisée avec les points internes n'est pas correcte dans le cas des points x_0 et x_6 (pourquoi?).

9.2.3 Condition de Dirichlet

Les $m - 1$ équations obtenues aux points intérieurs ne sont pas suffisantes pour déterminer les $m + 1$ inconnues du problème; les 2 équations restantes sont obtenues en utilisant les conditions aux limites. Dans le cas de conditions aux limites de Dirichlet, celles-ci correspondent à $y(x) = c_x$ avec $x = 0, L$. Ainsi, la condition spécifie directement la valeur d'une des inconnues du système : $y_0 = c_0$ pour la condition de Dirichlet en 0,

1. Notez la multiplication par le facteur h^2 par rapport à l'expression dans (9.6); cette normalisation nous aide dans la suite de la section 9.2 à garder la matrice du système symétrique.

ou $y_m = c_L$ pour celle en L . Ces égalités peuvent être utilisées soit pour augmenter le nombre d'équations (de une par condition de Dirichlet), soit, puisque'on connaît la valeur de l'inconnue concernée, pour réduire le nombre d'inconnues, en substituant toutes les occurrences des inconnues correspondantes par leur valeur. La deuxième approche, qui consiste à réduire le nombre d'inconnues, est préférable en pratique.

EXEMPLE 43. (SUITE) Exemple 43 avec les deux conditions de Dirichlet $y(0) = y_0 = c_0$ et $y(L) = y_m = c_L$ mène (si on élimine les inconnues correspondantes) au système d'équations suivant

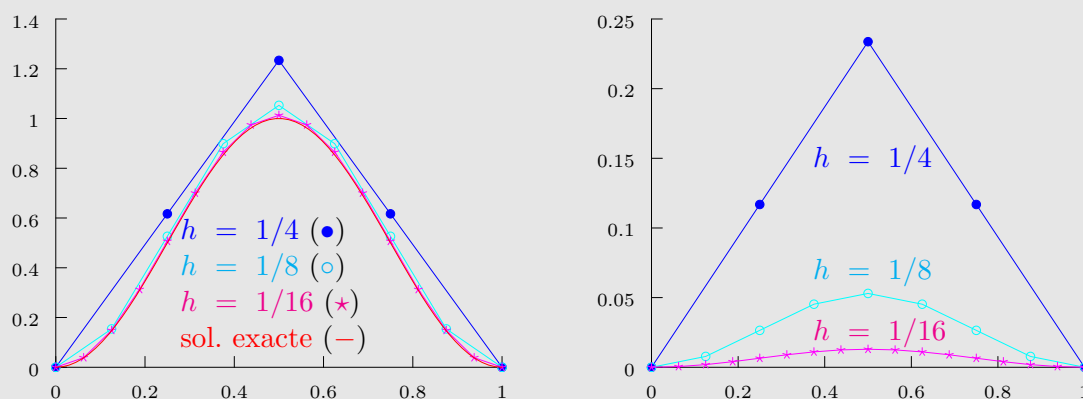
$$\begin{cases} 2y_1 & -y_2 & & & & = h^2 f(x_1) + c_0, \\ -y_1 & +2y_2 & -y_3 & & & = h^2 f(x_2), \\ & -y_2 & +2y_3 & -y_4 & & = h^2 f(x_3), \\ & & -y_3 & +2y_4 & -y_5 & = h^2 f(x_4), \\ & & & -y_4 & +2y_5 & = h^2 f(x_5) + c_L. \end{cases}$$

Notez que cette fois-ci le nombre d'équations est égale au nombre d'inconnues.

EXEMPLE 44. On considère la résolution du problème aux limites suivant

$$\begin{cases} -\frac{d^2 y}{dx^2}(x) = -2\pi^2 \cos(2\pi x), & x \in [0, 1], \\ y(0) = 0, & y(1) = 0, \end{cases}$$

par la méthode des différences finies. La figure de gauche donne les solutions *approchées* pour $h = 1/4, 1/8, 1/16$ et la solution *exacte* $y(x) = \sin^2(\pi x)$; celle de droite présente l'erreur $|y_k - y(x_k)|$.



Notez le respect des conditions de Dirichlet. Notez aussi que l'erreur semble être proportionnelle à h^2 : elle est divisée par un facteur proche de 4 lorsque le pas h est divisé par 2.

9.2.4 Condition de Neumann

Considérons à présent les conditions de Neumann, c'est-à-dire les conditions de la forme

$$\frac{dy}{dx}(x) = c_x \quad \text{avec } x = 0 \text{ ou } x = L.$$

Prenons par exemple la condition $\frac{dy}{dx}(0) = c_0$. Une option est d'approcher directement la dérivée dans cette condition par une formule de différence non centrée (ici, différence progressive), ce qui donne

$$(y(h) - y(0))/h = c_0 + \mathcal{O}(h),$$

où le terme d'erreur $\mathcal{O}(h)$ est valable pour $y \in C^2$. Cela amène à l'équation suivante

$$(y_1 - y_0)/h = c_0. \quad (9.7)$$

Alternativement, on peut utiliser l'équation différentielle (9.3) dans la dérivation de l'équation linéaire recherchée. Plus précisément, en utilisant le fait que la dérivée seconde est une dérivée de la dérivée, et en approchant la dérivée première «externe» par une formule de différence progressive, on peut transformer l'équation différentielle (9.3) en 0 comme suit

$$(h/2) \cdot f(0) = -(h/2) \cdot \frac{d^2y}{dx^2}(0) = - \underbrace{\frac{dy}{dx}(h/2)}_{(y(h)-y(0))/h + \mathcal{O}(h^2)} + \underbrace{\frac{dy}{dx}(0)}_{c_0} + \mathcal{O}(h^2),$$

ce qui donne

$$(y(h) - y(0))/h = c_0 - (h/2) \cdot f(0) + \mathcal{O}(h^2),$$

où le terme d'erreur $\mathcal{O}(h^2)$ est valable pour $y \in C^3$. Ainsi, l'équation à rajouter au système linéaire est

$$(y_1 - y_0)/h = c_0 - (h/2) \cdot f(0). \quad (9.8)$$

Notez que l'erreur commise pour obtenir l'équation (9.7) est de l'ordre de $\mathcal{O}(h)$, alors que celle pour (9.8) est de l'ordre de $\mathcal{O}(h^2)$; cette dernière est donc (asymptotiquement) plus précise, du moins si la solution y est suffisamment régulière.

EXEMPLE 43. (FIN) Exemple 43 avec les conditions aux limites $\frac{dy}{dx}(0) = c_0$ et $y(L) = c_L$ mène au système suivant (ici pour la condition de Neumann on a utilisé la formule plus précise (9.8))

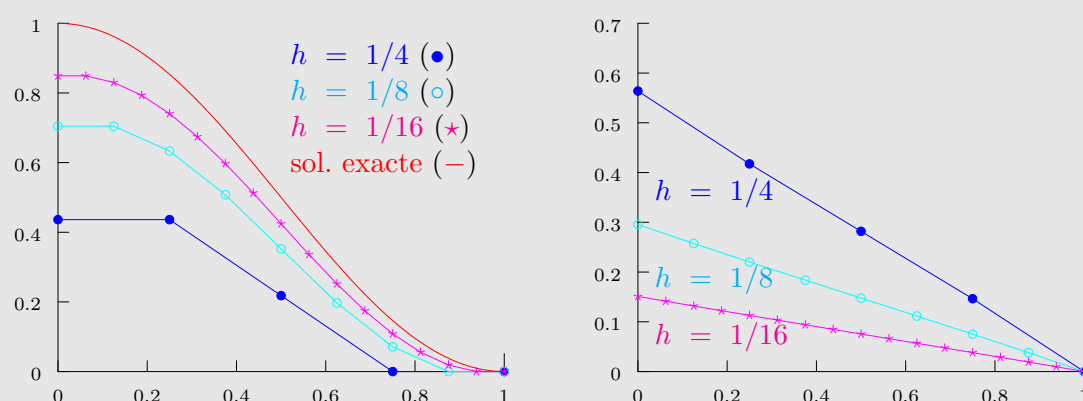
$$\left\{ \begin{array}{lll} y_0 & -y_1 & = -hc_0 + (h^2/2) \cdot f(0) \\ -y_0 & +2y_1 & -y_2 = h^2 f(x_1) \\ & -y_1 & +2y_2 & -y_3 = h^2 f(x_2) \\ & & -y_2 & +2y_3 & -y_4 = h^2 f(x_3) \\ & & & -y_3 & +2y_4 & -y_5 = h^2 f(x_4) \\ & & & & -y_4 & +2y_5 = h^2 f(x_5) + c_L \end{array} \right.$$

Ici aussi le nombre d'équations est égal au nombre d'inconnues, même si ce nombre n'est pas le même que lorsque les deux conditions sont de type Dirichlet.

EXEMPLE 44. (FIN) On considère la résolution du problème aux limites

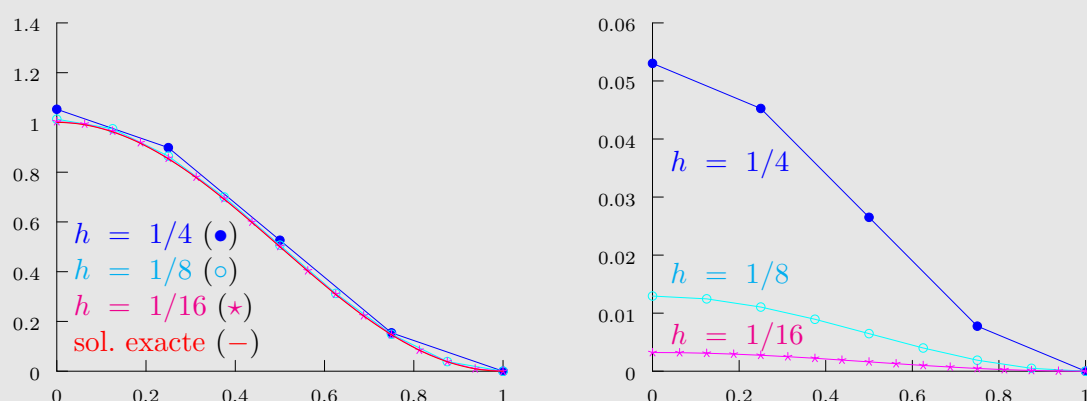
$$\begin{cases} -\frac{d^2 y}{dx^2}(x) = -\pi^2/2 \cdot \cos(\pi(x+1)), & x \in [0, 1], \\ \frac{dy}{dx}(0) = 0, \quad y(1) = 0 \end{cases}$$

par la méthode des différences finies et en utilisant, pour commencer, l'approximation *moins précise* (9.7) pour la condition de Neumann. La figure de gauche donne les solutions *approchées* pour $h = 1/4, 1/8, 1/16$ et la solution *exacte* $y(x) = \sin^2(\pi(x+1)/2)$; celle de droite présente l'erreur $|y_k - y(x_k)|$.



Notez un respect strict de la condition de Neumann. Par ailleurs, l'erreur semble être ici proportionnelle à h .

Maintenant, résolvons le même problème aux limites en utilisant l'approximation *plus précise* (9.8) pour la condition de Neumann. La solution est présentée dans les figures suivantes, dont la description est similaire à celles vues plus haut.



Outre un respect moins strict de la condition de Neumann (qui est néanmoins respectée asymptotiquement lorsque $h \rightarrow 0$!) on note que l'erreur est plus petite que dans le cas précédent, et semble être ici proportionnelle à h^2 .

9.3 Problème de convection-diffusion

Le problème de convection-diffusion (9.4) est repris ici pour rappel :

$$\begin{cases} -\frac{d^2y}{dx^2}(x) + p(x)\frac{dy}{dx}(x) = f(x), & x \in [0, L], \\ y(0) = c_0, \quad y(L) = c_L. \end{cases}$$

Il diffère de celui du Poisson par la présence d'un terme de convection $p(x)\frac{dy}{dx}(x)$. Dans cette section, on envisage deux manières possibles d'approcher ce terme via une formule de différences finies : celle basée sur la formule de différence centrée, et celle basée sur le schéma upwind.

9.3.1 Différence centrée

La discrétisation du problème de convection-diffusion est similaire à celui du problème de Poisson : l'approximation du terme de la dérivée seconde (dit de diffusion) et le traitement des conditions de Dirichlet se fait ici comme dans la section précédente. Dans ce qui suit nous envisagerons les approximations possibles pour le terme de la dérivée première (dit de convection).

Pour un point x_k intérieur ($k \neq 0, m$), il est naturel d'approcher la dérivée première du terme de convection par une formule de différence centrée, ce qui donne

$$\frac{dy}{dx}(x_k) = \frac{y(x_{k+1}) - y(x_{k-1}))}{2h} + \mathcal{O}(h^2),$$

où le terme d'erreur $\mathcal{O}(h^2)$ est valable pour $y \in C^3$. La version discrète de l'équation différentielle devient alors

$$(-y_{k+1} + 2y_k - y_{k-1}) + (hp(x_k)/2)(y_{k+1} - y_{k-1}) = h^2 f(x_k);$$

ici, le premier terme du membre de gauche correspond à h^2 fois l'approximation de la dérivée seconde, alors que le deuxième est h^2 fois l'approximation du terme de convection.

Notons que dans le cas où

$$hp(x) \gg 1,$$

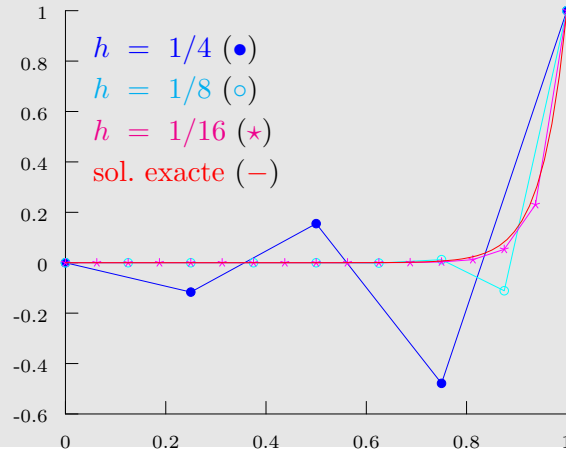
le second terme est dominant, et il ne relie que des inconnues dont les indices ont la même parité. Cette particularité du lien entre les inconnues peut mener à des oscillations dans la solution, comme illustré avec l'exemple suivant.

EXEMPLE 45. On considère la résolution du problème aux limites

$$\begin{cases} -\frac{d^2y}{dx^2}(x) + 20\frac{dy}{dx}(x) = 0, & x \in [0, 1], \\ y(0) = 0, \quad y(1) = 1 \end{cases}$$

par la méthode des différences finies et en approchant la dérivée première dans le terme de convection par la formule de *différence centrée*. La figure suivante donne les solutions

approchées pour $h = 1/4, 1/8, 1/16$ et la solution *exacte* $y(x) = (e^{20x} - 1)/(e^{20} - 1)$. On constate la présence des oscillations dans la solution $y(x)$ pour des valeurs élevées du pas h .



9.3.2 Schéma upwind

Pour éviter ces problèmes d'instabilité on utilise le schéma *upwind*. Concrètement, si $p(x_k) > 0$, le terme de convection est approché par la formule de différence rétrograde

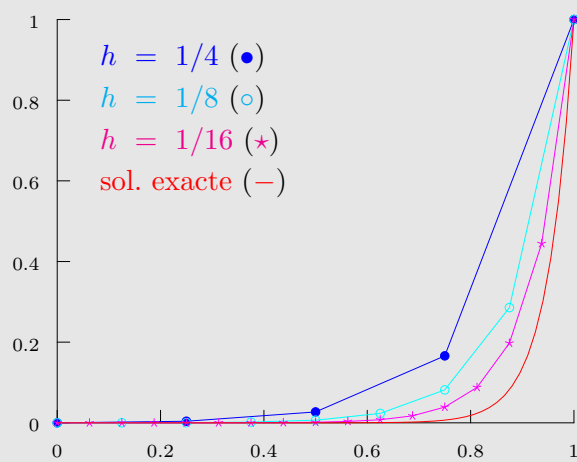
$$\frac{dy}{dx}(x_k) = \frac{y(t_k) - y(t_{k-1})}{h} + \mathcal{O}(h),$$

alors que pour $p(x_k) < 0$, la différence progressive est utilisée :

$$\frac{dy}{dx}(x_k) = \frac{y(t_{k+1}) - y(t_k)}{h} + \mathcal{O}(h);$$

dans les deux cas le terme d'erreur $\mathcal{O}(h)$ est valable pour $y \in C^2$. Notons que ce schéma d'approximation est moins précis, mais il permet d'éviter des oscillations telles qu'observées dans le cas de la différence centrée, et est donc plus stable, comme le montre dans le cadre ci-dessous la suite de l'exemple précédent.

EXEMPLE 45. (FIN) On reprend l'Exemple 45 mais cette fois avec la dérivée première dans le terme de convection étant approchée par un *schéma upwind*. A nouveau, la figure suivante contient les solutions *approchées* pour $h = 1/4, 1/8, 1/16$ et la solution *exacte* $y(x) = (e^{20x} - 1)/(e^{20} - 1)$. On constate l'absence d'oscillations dans la solution approchée, même si cette solution est maintenant moins précise.



Annexe 9.A Formules de différences finies : terme d'erreur

On commence par les formules de différences finies correspondant à la *dérivée première*. Pour justifier le terme d'erreur dans les formules de différence progressive et différence rétrograde, on suppose que la solution y est de classe C^2 . Le terme d'erreur est alors la conséquence du fait que le développement de Taylor satisfait la relation suivante

$$y(x \pm h) = y(x) \pm h \frac{dy}{dx}(x) + \mathcal{O}(h^2),$$

où le terme $\mathcal{O}(h^2)$ est, par exemple, le reste de Lagrange. Par contre, si y est de classe C^3 , on peut alors écrire le développement de Taylor sous la forme suivante

$$y(x \pm h) = y(x) \pm h \frac{dy}{dx}(x) + \frac{h^2}{2} \frac{d^2y}{dx^2}(x) + \mathcal{O}(h^3).$$

L'erreur pour la formule de différence centrée est obtenue en soustrayant la formule avec “+” de celle avec “−” et en divisant par $2h$:

$$\frac{y(x+h) - y(x-h)}{2h} = \frac{dy}{dx}(x) + \mathcal{O}(h^2).$$

Pour ce qui est de la *dérivée seconde*, on suppose que la solution y est de classe C^4 . Le développement de Taylor peut alors être écrit comme

$$y(x_k \pm h) = y(x_k) \pm h \frac{dy}{dx}(x_k) + \frac{h^2}{2} \frac{d^2y}{dx^2}(x_k) \pm \frac{h^3}{3!} \frac{d^3y}{dx^3}(x_k) + \mathcal{O}(h^4).$$

Dès lors, en additionnant les formules avec “+” et “−”, on a

$$y(x_k + h) + y(x_k - h) = 2y(x_k) + h^2 \frac{d^2y}{dx^2}(x_k) + \mathcal{O}(h^4),$$

et la formule pour la dérivée seconde (avec le terme d'erreur correspondant) s'en suit directement :

$$\frac{d^2y}{dx^2}(x_k) = \frac{y(x_k - h) - 2y(x_k) + y(x_k + h)}{h^2} + \mathcal{O}(h^2).$$