

TECHNOSUP

Les FILIÈRES TECHNOLOGIQUES des ENSEIGNEMENTS SUPÉRIEURS

MÉTHODES NUMÉRIQUES

Algorithmes numériques

Fondements théoriques et analyse pratique

Cours, exercices et applications avec MATLAB®

Marie-Hélène MEURISSE



TECHNOSUP

Les FILIÈRES TECHNOLOGIQUES des ENSEIGNEMENTS SUPÉRIEURS

MÉTHODES NUMÉRIQUES

Algorithmes numériques

Fondements théoriques et analyse pratique

Cours, exercices et applications avec MATLAB®

Marie-Hélène MEURISSE

Enseignant-chercheur
INSA de Lyon



Dans la même collection :

- Faire des maths avec Mathematica. Initiation, thèmes d'étude (A) 160 p. N. VERDIER
- Méthodes numériques pour l'ingénieur, (B) 312 p. B. RADI, A. EL HAMI
- Mathématiques avec Scilab (B) 216 p. B. RADI, A.EL HAMI
- Calcul scientifique avec Matlab (C) 288 p. J. KOKO
- Approximation numérique avec MATLAB. Cours et exercices corrigés (C) 312 p. J. KOKO
- Méthodes spectrales (C) 360 p. J-M. LABROSSE
- VBA pour Excel, Bibliothèque mathématique avec applications pratiques (C) 600 p. D. ROUX

ISBN 9782340-091986

©Ellipses Édition Marketing S.A., 2018
32, rue Bague 75740 Paris cedex 15



Le Code de la propriété intellectuelle n'autorisant, aux termes de l'article L. 122-5.2° et 3°a), d'une part, que les « copies ou reproductions strictement réservées à l'usage privé du copiste et non destinées à une utilisation collective », et d'autre part, que les analyses et les courtes citations dans un but d'exemple et d'illustration, « toute représentation ou reproduction intégrale ou partielle faite sans le consentement de l'auteur ou de ses ayants droit ou ayants cause est illicite » (art. L. 122-4).

Cette représentation ou reproduction, par quelque procédé que ce soit constituerait une contrefaçon sanctionnée par les articles L. 335-2 et suivants du Code de la propriété intellectuelle.

www.editions-ellipses.fr

Avant-propos

Les différentes facettes des sciences de l'ingénieur, production, fabrication, conception, développement et recherche... amènent les acteurs scientifiques à associer, dans un cadre multidisciplinaire, approches expérimentales et simulations numériques.

Si ces approches numériques et expérimentales conjointes s'interrogent, se valident, s'enrichissent mutuellement, permettant ainsi de démultiplier l'efficacité de chaque type d'approche séparée, c'est cependant toujours le maillon le plus faible qui définit l'efficacité de l'ensemble.

C'est pourquoi, dans un contexte de développement rapide des puissances des ordinateurs, qui a pour conséquence la croissance spectaculaire de la part prise par la simulation numérique, il est essentiel que chaque scientifique puisse s'appuyer sur un socle de connaissances et de compréhension des principales méthodes numériques.

En effet, une telle culture numérique rend capable de juger de la fiabilité du *maillon numérique* des approches scientifiques, et de dialoguer efficacement avec des numériciens spécialisés.

D'autre part, le développement considérable des logiciels numériques commerciaux, généralistes ou plus spécialisés, n'a pas fait diminuer la nécessité pour les ingénieurs et les chercheurs de posséder une solide base de connaissance et de maîtrise des outils de calcul numérique, bien au contraire. L'enjeu est en effet pour chacun d'être un utilisateur averti de tels logiciels, capable d'assimiler les notices, de cerner les limites, de définir les paramètres adéquats, et de valider les résultats.

Enfin, une bonne connaissance des algorithmes numériques fondamentaux ouvre la possibilité de développer ses propres outils de calcul scientifique et d'en juger la pertinence et l'efficacité.

Cet ouvrage, synthèse de supports de cours magistral et de travaux dirigés, reflète ma longue expérience d'enseignante coordonnatrice du cours d'analyse numérique du département Génie Mécanique Développement de l'Institut National des Sciences Appliquées de Lyon.

Il traite des algorithmes numériques fondamentaux, destinés à la résolution des systèmes linéaires et non linéaires, des problèmes aux valeurs propres, à l'interpolation

et à l'approximation, à la quadrature numérique, ainsi qu'à la résolution des problèmes différentiels à valeurs initiales.

Evitant une présentation de type catalogue, l'approche pédagogique de cet ouvrage vise une compréhension profonde des méthodes, de leurs principes et de leurs performances, et s'appuie délibérément sur des fondements théoriques. La présentation s'attache néanmoins à éviter les développements mathématiques trop longs, très abstraits, ou non constructifs.

Un ensemble d'exercices, de problèmes de synthèse, et d'applications en environnement MATLAB®, assortis d'éléments de réponses et d'aides à l'interprétation, complète l'ouvrage.

Je voudrais remercier tous les collègues enseignants-chercheurs avec lesquels j'ai collaboré dans le cadre de l'enseignement d'analyse numérique, pour leur apport à cette synthèse au travers de nos fructueux échanges, partages d'expériences et discussions.

J'adresse également mes remerciements aux étudiants et collègues qui ont manifesté de l'intérêt pour ce projet rédactionnel et l'ont soutenu.

Merci enfin à Gilles pour son aide patiente à la relecture.

Table des Matières

Partie A - Analyse numérique matricielle

CHAPITRE I

INTRODUCTION A LA RESOLUTION NUMERIQUE DES SYSTEMES LINEAIRES 5

1	<i>Notations</i>	5
2	<i>Complexité des algorithmes</i>	6
3	<i>Conditionnement d'une matrice</i>	9

CHAPITRE II

ALGORITHMES DIRECTS 16

1	<i>Systèmes linéaires triangulaires</i>	16
2	<i>Algorithme de triangulation de Gauss</i>	18
3	<i>Algorithme de factorisation LU</i>	30
4	<i>Algorithme de Householder</i>	37
5	<i>Bilan comparatif</i>	41

CHAPITRE III

ALGORITHMES DE SPLITTING 43

1	<i>Principe</i>	43
2	<i>Algorithme de Jacobi</i>	46
3	<i>Algorithme de Gauss-Seidel – Relaxation</i>	49
4	<i>Convergence</i>	56

CHAPITRE IV

ALGORITHMES DE DESCENTE 59

1	<i>Rappels d'algèbre bilinéaire</i>	59
2	<i>Problème de minimisation équivalent</i>	61
3	<i>Principe et convergence des algorithmes de descente</i>	62
4	<i>Algorithme de Gauss-Seidel</i>	67
5	<i>Algorithme du gradient</i>	70
6	<i>Algorithme du gradient conjugué</i>	73

CHAPITRE V

CALCUL DE VALEURS ET VECTEURS PROPRES 78

1	Généralités.....	78
2	Algorithme de Leverrier	79
3	Algorithme LR	82
4	Algorithme QR	88
5	Algorithme de la puissance itérée	91
6	Algorithme de Jacobi	96
7	Calcul de vecteurs propres a posteriori	99
8	Conditionnement d'un problème de valeurs propres	102

Partie B - Equations non linéaires

CHAPITRE VI

RESOLUTION NUMERIQUE D'UNE EQUATION NON LINEAIRE 104

1	Généralités.....	104
2	Algorithme de substitution	106
3	Algorithmes regula falsi, de la sécante, de dichotomie.....	113
4	Algorithme de Newton	115
5	Algorithme de Graeffe	117
6	Algorithme de Bairstow	121

CHAPITRE VII

SYSTEMES D'EQUATIONS NON LINEAIRES 127

1	Introduction	127
2	Algorithmes de substitution	128
3	Algorithme de Newton-Raphson	133
4	Algorithmes de gradient	136

Partie C - Interpolation, approximation, intégration

CHAPITRE VIII

INTERPOLATION 143

1	Introduction	143
2	Interpolation polynomiale de Lagrange	144

3	<i>Interpolation polynomiale d'Hermite</i>	151
4	<i>Non convergence de l'interpolation polynomiale.....</i>	154
5	<i>Interpolation par fonctions splines cubiques.....</i>	157
6	<i>Interpolation par fraction continue limitée</i>	161

CHAPITRE IX

APPROXIMATION PAR MOINDRES CARRES 165

1	<i>Introduction</i>	165
2	<i>Méthode des moindres carrés discrets.....</i>	167
3	<i>Méthode des moindres carrés discrets polynomiale</i>	169
4	<i>Méthode des moindres carrés continus polynomiale.....</i>	174

CHAPITRE X

INTEGRATION NUMERIQUE 177

1	<i>Généralités.....</i>	177
2	<i>Formules de Newton-Cotes simples.....</i>	178
3	<i>Formules de Newton-Cotes composites</i>	182
4	<i>Algorithme de Romberg</i>	185
5	<i>Formules de quadrature de Gauss.....</i>	188

Partie D - Schémas numériques pour problèmes différentiels à valeurs initiales

CHAPITRE XI

PROBLEMES DE CAUCHY ET SCHEMA D'EULER 200

1	<i>Introduction</i>	200
2	<i>Problèmes de Cauchy.....</i>	200
3	<i>Schéma d'Euler</i>	205
4	<i>Consistance, stabilité, convergence d'un schéma</i>	209
5	<i>Méthodes de tir</i>	217

CHAPITRE XII

SCHEMAS DE RUNGE-KUTTA 222

1	<i>Introduction</i>	222
2	<i>Schémas de type Euler amélioré.....</i>	223
3	<i>Schémas de Runge-Kutta.....</i>	224

CHAPITRE XIII

SCHEMAS A PAS LIES 229

1	<i>Introduction</i>	229
2	<i>Principe</i>	230
3	<i>Consistance</i>	231
4	<i>Schémas explicites</i>	232
5	<i>Schémas implicites</i>	237
6	<i>Schémas de prédiction-correction</i>	240
7	<i>Stabilité</i>	241

EXERCICES ET PROBLEMES DE SYNTHESE 248

<i>partie A - Analyse numérique matricielle</i>	248
<i>partie B - Equations non linéaires</i>	257
<i>partie C - Interpolation, approximation, intégration</i>	263
<i>partie D - Schémas numériques</i>	267
<i>Eléments de réponses</i>	273

INDEX 303

chapitre I

Introduction à la résolution numérique des systèmes linéaires

Ce chapitre traite de questions générales concernant la résolution numérique des systèmes d'équations linéaires. On y définit en particulier les notations utilisées dans les chapitres I à V, consacrés à l'analyse numérique matricielle ; les outils mathématiques, qui permettent d'étudier la qualité des solutions approchées obtenues par les différents algorithmes numériques, sont également présentés.

Les principaux algorithmes de résolution de systèmes linéaires sont présentés dans les chapitres II (algorithmes directs), puis III et IV (algorithmes itératifs, respectivement de splitting et de descente).

1 NOTATIONS

Notations

On considère un système linéaire de n équations à n inconnues :

$$\left\{ \begin{array}{l} a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n = b_2 \\ \dots \\ a_{i1}x_1 + a_{i2}x_2 + \dots + a_{in}x_n = b_i \\ \dots \\ a_{n1}x_1 + a_{n2}x_2 + \dots + a_{nn}x_n = b_n \end{array} \right. \quad \text{noté également} \quad \sum_{j=1}^n a_{ij}x_j = b_i \quad \forall i \in \{1, 2, \dots, n\}, \text{ ou bien}$$

$$Ax = b, \text{ où } A = \begin{pmatrix} a_{11} & a_{12} & \cdot & \cdot & a_{1n} \\ a_{21} & a_{22} & & & a_{2n} \\ \cdot & \cdot & & & \cdot \\ \cdot & \cdot & & & \cdot \\ a_{n1} & a_{n2} & \cdot & \cdot & a_{nn} \end{pmatrix} \in \mathfrak{M}_{n,n}(\mathbb{R}) \text{ désigne la } \textit{matrice du système}$$

linéaire, $x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \in \mathfrak{M}_{n,1}(\mathbb{R})$ désigne le *vecteur des inconnues* et $b = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix} \in \mathfrak{M}_{n,1}(\mathbb{R})$ le *vecteur second membre*.

Ce système est supposé être un système dit *de Cramer*, qui possède une solution unique, donc tel que $\det A \neq 0$.

Taille des systèmes linéaires

La taille des systèmes linéaires rencontrés en ingénierie ou en recherche et développement est très variable, et peut atteindre des valeurs considérables. A titre d'exemple, le calcul des intensités électriques dans les différentes branches d'un circuit en dérivation donne typiquement lieu à un système linéaire de petite taille ($n \approx 2, 3, 4 \dots$). Si l'on considère les systèmes linéaires issus des techniques de discrétisation d'équations différentielles ou d'équations aux dérivées partielles, leur taille est très dépendante de la dimension des équations, et de la nécessité de resserrer le maillage au voisinage des singularités. Si pour une discrétisation d'équations 1D (calcul de la flèche d'une poutre par exemple) on aboutit à un système linéaire dont la taille est typiquement de l'ordre de 10 ou 10^2 , elle sera naturellement de l'ordre de 10^2 à 10^4 pour un problème 2D (calcul de la répartition de température dans une plaque mince par exemple), et de l'ordre de 10^4 à 10^6 pour un problème 3D (calcul de la déformée d'un corps solide par exemple).

Ainsi, dans la plupart des situations pratiques, la solution des systèmes linéaires ne peut pas être recherchée *à la main*. Il est nécessaire de mettre en œuvre un algorithme numérique pour obtenir, avec plus ou moins de précision, une solution approchée. Mais pour les très gros systèmes linéaires, les temps de calcul, même avec un calculateur super puissant peuvent être prohibitifs.

Les critères principaux d'évaluation et de performance des algorithmes numériques, en particulier les algorithmes de résolution de système linéaire, portent donc à la fois sur la précision et sur le temps d'obtention des solutions approchées.

2 COMPLEXITE DES ALGORITHMES

La notion de *complexité* d'un algorithme vise à évaluer le nombre d'opérations coûteuses en temps, nécessaires pour la mise en œuvre de cet algorithme. Elle permet de comparer les algorithmes entre eux, sur un critère de temps de calcul, d'une manière qui ne soit pas dépendante du calculateur lui-même, du langage de programmation, ou du compilateur. Les théories de la complexité algorithmique proposent diverses méthodes pour évaluer la complexité d'un algorithme.

On utilise dans cet ouvrage une définition élémentaire de la complexité d'un algorithme de résolution de système linéaire, suffisante pour hiérarchiser les différents algorithmes sur un critère de temps (donc de coût) de calcul. Elle consiste à ne considérer que les opérations algébriques de type multiplications ou divisions contenues dans cet algorithme.

Définition de la complexité

La complexité d'un algorithme est le nombre de multiplications et de divisions nécessaires pour sa mise en œuvre. Elle n'est un critère de sélection d'algorithme que pour les gros systèmes linéaires, et on la présente généralement avec la notation mathématique $O(f(n))$, qui signifie :

$complexité = O(f(n)) \iff \exists a > 0, b > 0 \text{ et } N > 0 / af(n) \leq complexité \leq bf(n) \quad \forall n > N$

Complexité de la méthode de Cramer

En algèbre linéaire, la méthode de Cramer permet d'obtenir une formulation de la solution exacte du système linéaire, en exprimant chaque composante x_i de la solution en fonction du déterminant de la matrice, et d'un déterminant obtenu en remplaçant la $i^{ème}$ colonne de la matrice par le second membre b :

$$x_i = \frac{\begin{vmatrix} a_{11} & b_1 & \dots & a_{1n} \\ a_{21} & b_2 & & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{n1} & b_n & \dots & a_{nn} \end{vmatrix}}{\det A} \quad \forall i \in \{1,2,\dots,n\}$$

i^{ème} colonne de A

Pour obtenir la solution du système linéaire par cette méthode, il est nécessaire de calculer $n+1$ déterminants de taille $n \times n$, et d'effectuer n divisions. Or le nombre de multiplications nécessaire pour calculer un déterminant de cette taille est :

$$n! \left(1 + \frac{1}{2!} + \frac{1}{3!} + \dots + \frac{1}{(n-1)!} \right).$$

La complexité de la méthode de Cramer est donc : $n \left[n! \left(1 + \frac{1}{2!} + \frac{1}{3!} + \dots + \frac{1}{(n-1)!} \right) + 1 \right].$

Son comportement pour n grand est $O(n \times n!)$.

Avec un super calculateur capable d'effectuer 10^{12} opérations par seconde (1 teraFLOPS), il faudrait, pour résoudre un système de relativement petite taille, $n = 50$, plus de 10^{53} secondes, c'est à dire plus de 10^{45} années !

Ce calcul de complexité montre bien que la méthode de Cramer n'est envisageable que pour les systèmes linéaires de très petite taille, pour lesquels on recherche généralement une solution exacte.

Inversion de la matrice

Puisque le système linéaire est supposé de Cramer, sa matrice est inversible, et il est aisé d'écrire formellement $Ax = b \Leftrightarrow x = A^{-1}b$. Mais cette formule n'est utile pour obtenir la solution du système linéaire que si l'on est capable de calculer l'inverse de la matrice A .

Or, le calcul de A^{-1} revient à résoudre n systèmes linéaires. En effet, par définition : $AA^{-1} = I$.

$$\text{Donc, si } A^{-1} = \begin{pmatrix} \alpha_{11} & \alpha_{1j} & \cdot & \alpha_{1n} \\ \alpha_{21} & \alpha_{2j} & & \alpha_{2n} \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ \alpha_{n1} & \alpha_{nj} & \cdot & \alpha_{nn} \end{pmatrix}, \text{ la } j^{\text{ème}} \text{ colonne de } A^{-1} \text{ est solution du}$$

$$\text{système linéaire : } A \begin{pmatrix} \alpha_{1j} \\ \alpha_{2j} \\ \vdots \\ \alpha_{ij} \\ \vdots \\ \alpha_{nj} \end{pmatrix} = e_j \text{ où } e_j = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{pmatrix} \leftarrow \text{ligne n}^{\circ}j$$

Pour calculer A^{-1} , il s'agit donc de résoudre les n systèmes linéaires associés aux n colonnes inconnues de cette matrice. Il est clair que ce procédé, de même que la méthode de Cramer, n'est pas intéressant pour résoudre le système linéaire initial, puisqu'il consiste à remplacer un seul système linéaire par n systèmes linéaires de même matrice. Chercher à inverser la matrice est donc un très mauvais réflexe lorsque le but des calculs est la résolution d'un système linéaire.

Par contre, on peut noter que, réciproquement, un algorithme de résolution de système linéaire appliqué successivement aux n systèmes linéaires définis ci-dessus, qui permettent de calculer les n colonnes de A^{-1} , peut permettre d'inverser la matrice A , si tel est le but recherché. Les algorithmes directs de la famille LU, LDU ... (cf. chapitre II, § 3) sont particulièrement bien adaptés pour cela.

3 CONDITIONNEMENT D'UNE MATRICE

Le cumul des erreurs d'arrondi dues au calculateur lors de la mise en œuvre d'un algorithme de résolution de système linéaire est susceptible de provoquer un très gros écart entre la solution approchée, théoriquement fournie par l'algorithme, et la solution approchée effective obtenue à l'issue des calculs. L'exemple décrit au § 3.1 illustre de telles situations de *catastrophes numériques*.

La notion de conditionnement d'un système linéaire (qui est également le conditionnement de sa matrice, cf. § 3.3) permet de quantifier la sensibilité d'une solution numérique approchée aux perturbations qui peuvent se produire lors l'exécution d'un algorithme de résolution (erreurs d'arrondi par exemple).

3.1 Un exemple de système linéaire mal conditionné

Considérons le système linéaire $Ax = b$, avec $A = \begin{pmatrix} 10 & 7 & 8 & 7 \\ 7 & 5 & 6 & 5 \\ 8 & 6 & 10 & 9 \\ 7 & 5 & 9 & 10 \end{pmatrix}$ et $b = \begin{pmatrix} 32 \\ 23 \\ 33 \\ 31 \end{pmatrix}$, dont

la solution exacte est $s = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$.

Afin de simuler les erreurs d'arrondi induites par le calculateur, on perturbe successivement les coefficients du second membre puis de la matrice, et on étudie l'incidence de ces perturbations sur la solution.

- Si on remplace b par $b' = \begin{pmatrix} 32.1 \\ 22.9 \\ 33.1 \\ 30.9 \end{pmatrix}$, induisant ainsi une perturbation telle que

$$\max_i \frac{|b'_i - b_i|}{|b_i|} = 4.4\%, \text{ la solution du système linéaire devient } s' = \begin{pmatrix} 9.2 \\ -12.6 \\ 4.5 \\ -1.1 \end{pmatrix}, \text{ et on}$$

observe une erreur relative maximale $\max_i \frac{|s'_i - s_i|}{|s_i|} = 1360\%$

- Si on remplace la matrice A par $A' = \begin{pmatrix} 10 & 7 & 8.1 & 7.2 \\ 7.08 & 5.04 & 6 & 5 \\ 8 & 5.98 & 9.89 & 9 \\ 6.99 & 4.99 & 9 & 9.98 \end{pmatrix}$, induisant ainsi

une perturbation telle que $\max_{i,j} \frac{|a'_{ij} - a_{ij}|}{|a_{ij}|} = 2.9\%$, la solution du système $A'x = b$

devient $s'' = \begin{pmatrix} -71 \\ 137 \\ -34 \\ 22 \end{pmatrix}$, et on observe une erreur relative maximale

$$\max_i \frac{|s''_i - s_i|}{|s_i|} = 13600\%.$$

Ainsi, bien qu'étant de très petite taille, ce système linéaire présente une sensibilité particulière aux erreurs d'arrondi. L'analyse de cette sensibilité, qui conduit à la notion de conditionnement au § 3.3, utilise comme outil mathématique de mesure des erreurs, les normes vectorielles et matricielles dont les propriétés sont rappelées au § 3.2.

3.2 Normes vectorielles, normes matricielles

On rappelle ici quelques propriétés des normes, en mettant naturellement l'accent sur les normes de $\mathbb{R}^n$ (ou de $\mathfrak{M}_{n,1}(\mathbb{R})$), ainsi que sur les normes de matrices carrées (de $\mathfrak{M}_{n,n}(\mathbb{R})$).

Définition d'une norme

Soit E un espace vectoriel sur $\mathbb{R}$ (ou $\mathbb{C}$). Soit N une application de E dans $\mathbb{R}^+$.

N est une norme de E si et seulement si :

- $N(u) = 0 \Rightarrow u = 0$
- $N(\lambda u) = |\lambda| N(u) \quad \forall \lambda \in \mathbb{R} \text{ (ou } \mathbb{C}) \quad \forall u \in E$
- $N(u + v) \leq N(u) + N(v) \quad \forall (u, v) \in E^2$

Les 3 normes usuelles de $\mathfrak{M}_{n,1}(\mathbb{R})$, notées $x \in \mathfrak{M}_{n,1}(\mathbb{R}) \rightarrow \|x\|_i \in \mathbb{R}^+$ sont définies par :

- $\|x\|_1 = \sum_{i=1}^n |x_i|$

- $\|x\|_2 = \left(\sum_{i=1}^n |x_i|^2 \right)^{1/2}$ (norme euclidienne usuelle)
- $\|x\|_\infty = \max_{i \in \{1, 2, \dots, n\}} |x_i|$

Norme matricielle induite

Soit $x \rightarrow \|x\|$ une norme quelconque (vectorielle) de $\mathfrak{M}_{n,1}(\mathbb{R})$. L'application $A \in \mathfrak{M}_{n,n}(\mathbb{R}) \rightarrow \sup_{\substack{x \in \mathfrak{M}_{n,1}(\mathbb{R}) \\ \|x\|=1}} \|Ax\|$ est une norme de l'espace vectoriel des matrices carrées $\mathfrak{M}_{n,n}(\mathbb{R})$, que l'on note généralement :

$$\|A\| = \sup_{\substack{x \in \mathfrak{M}_{n,1}(\mathbb{R}) \\ \|x\|=1}} \|Ax\| = \sup_{\substack{x \in \mathfrak{M}_{n,1}(\mathbb{R}) \\ x \neq 0}} \frac{\|Ax\|}{\|x\|}$$

Aux 3 normes usuelles de $\mathbb{R}^n$ sont ainsi associées 3 *normes matricielles induites* usuelles (on dit également *normes matricielles subordonnées*), telles que :

- $\|A\|_1 = \sup_{\substack{x \in \mathfrak{M}_{n,1}(\mathbb{R}) \\ \|x\|_1=1}} \|Ax\|_1 = \max_{j \in \{1, \dots, n\}} \sum_{i=1}^n |a_{ij}|$
- $\|A\|_2 = \sup_{\substack{x \in \mathfrak{M}_{n,1}(\mathbb{R}) \\ \|x\|_2=1}} \|Ax\|_2 = \left(\rho(A^t A) \right)^{1/2}$, ρ désignant le rayon spectral d'une matrice.

Remarque : si A est *normale* (c'est-à-dire si $A^t A = A A^t$), alors $\|A\|_2 = \rho(A)$.

- $\|A\|_\infty = \sup_{\substack{x \in \mathfrak{M}_{n,1}(\mathbb{R}) \\ \|x\|_\infty=1}} \|Ax\|_1 = \max_{i \in \{1, \dots, n\}} \sum_{j=1}^n |a_{ij}|$

Les normes matricielles induites vérifient les 2 propriétés suivantes (les normes sont dites *sous-multiplicatives*) :

- $\|Ax\| \leq \|A\| \|x\| \quad \forall x \in \mathfrak{M}_{n,1}(\mathbb{R}) \quad \forall A \in \mathfrak{M}_{n,n}(\mathbb{R})$ (c'est une conséquence directe de la définition d'une norme matricielle induite).
- $\|AB\| \leq \|A\| \|B\| \quad \forall (A, B) \in (\mathfrak{M}_{n,n}(\mathbb{R}))^2$

Autres normes matricielles

Si l'on assimile une matrice carrée à un tableau de n^2 coefficients, donc à un élément de $\mathbb{R}^{n^2}$, on peut définir d'autres normes (non induites) matricielles, dont les 3 principales :

$$\begin{aligned}\|A\|_1 &= \sum_{i=1}^n \sum_{j=1}^n |a_{ij}| \\ \|A\|_2 &= \left(\sum_{i=1}^n \sum_{j=1}^n |a_{ij}|^2 \right)^{1/2} \\ \|A\|_\infty &= \max_{i,j \in \{1,2,\dots,n\}} |a_{ij}|\end{aligned}$$

Equivalence de normes

Dans les espaces vectoriels de dimension finie (c'est le cas des espaces vectoriels de matrices), toutes les normes sont équivalentes. Les normes de matrices, induites ou non induites, sont donc toutes équivalentes entre elles. Ce qui se traduit par :

$$\left| \begin{array}{l} \exists c, d > 0 \quad / \quad c \|A\|_k \leq \|A\|_l \leq d \|A\|_k \\ \exists e, f > 0 \quad / \quad e \|A\|_k \leq \|A\|_l \leq f \|A\|_k \\ \exists g, h > 0 \quad / \quad g \|A\|_k \leq \|A\|_l \leq h \|A\|_k \end{array} \right. \quad k = 1, 2, \infty, \quad l = 1, 2, \infty$$

3.3 Conditionnement d'une matrice, d'un système linéaire

Afin de simuler les erreurs d'arrondi induites par le calculateur, comme dans l'exemple présenté au § 3.1, on considère successivement des perturbations des coefficients du second membre b puis de la matrice A , d'un système linéaire de Cramer quelconque $Ax = b$, et on quantifie, grâce aux normes vectorielles et matricielles induites, l'incidence de ces perturbations sur la solution du système.

Perturbation du second membre

On remplace b par $b' = b + \Delta b$; la solution s devient $s' = s + \Delta s$. Compte tenu du fait que, par définition, $As = b$ et $A(s + \Delta s) = b + \Delta b$, on obtient par soustraction $A\Delta s = \Delta b$, c'est-à-dire formellement $\Delta s = A^{-1}\Delta b$.

On en déduit que $\|\Delta s\| \leq \|A^{-1}\| \|\Delta b\|$ puisque les normes sont sous-multiplicatives, puis, comme $\|b\| \leq \|A\| \|s\|$, que :

$$\boxed{\frac{\|\Delta s\|}{\|s\|} \leq \|A\| \|A^{-1}\| \frac{\|\Delta b\|}{\|b\|}}$$

Perturbation de la matrice

On remplace maintenant A par $A' = A + \Delta A$; la solution s devient $s'' = s + \Delta s$. Compte tenu du fait que $As = b$ et $(A + \Delta A)(s + \Delta s) = b$, on obtient par soustraction $\Delta A(s + \Delta s) + A\Delta s = 0$, c'est-à-dire formellement $\Delta s = -A^{-1}\Delta A(s + \Delta s)$.

On en déduit que $\|\Delta s\| \leq \|A^{-1}\| \|\Delta A\| \|s + \Delta s\|$ puisque les normes sont sous-multiplicatives, donc que :

$$\frac{\|\Delta s\|}{\|s + \Delta s\|} \leq \|A\| \|A^{-1}\| \frac{\|\Delta A\|}{\|A\|}$$

Définition du conditionnement

On remarque que les perturbations des coefficients de la matrice ou du second membre conduisent à une même estimation du facteur d'amplification de leur perturbation relative, sur la solution du système linéaire. Cette estimation est égale à $\|A\| \|A^{-1}\|$.

Il est à noter que ce facteur dépend uniquement de la matrice, et non du second membre du système linéaire. On le nomme *conditionnement* de la matrice, qui est également le conditionnement du système linéaire :

$$\text{cond}(A) = \|A^{-1}\| \|A\|$$

Le conditionnement d'une matrice estime le risque d'amplification des erreurs d'arrondi provoquées par le calculateur. Il estime également l'incertitude sur la solution d'un système linéaire, liée à d'éventuelles incertitudes sur ses coefficients.

La valeur du conditionnement dépend la norme de $\mathfrak{M}_{n,1}(\mathbb{R})$ considérée. Il existe donc 3 conditionnements usuels, notés cond_1 , cond_2 , cond_∞ , associés aux 3 normes usuelles de $\mathfrak{M}_{n,1}(\mathbb{R})$ (ou de $\mathbb{R}^n$).

Si on considère à nouveau l'exemple donné au § 3.1, et que l'on choisit la norme euclidienne usuelle pour mesurer les écarts, on constate que $\frac{\|\Delta s\|_2}{\|s\|_2} \approx 8.2$ pour

$\frac{\|\Delta b\|_2}{\|b\|_2} \approx 3.3 \cdot 10^{-3}$. La matrice A étant symétrique, donc normale, elle vérifie

$\text{cond}_2(A) = \rho(A)\rho(A^{-1}) \approx 3029$. On vérifie bien que $\frac{\|\Delta s\|_2}{\|s\|_2} \leq \text{cond}_2(A) \frac{\|\Delta b\|_2}{\|b\|_2} \approx 10.1$

Propriétés du conditionnement

- $\text{cond}(A) = \text{cond}(A^{-1})$
- $\text{cond}_1(I) = \text{cond}_2(I) = \text{cond}_\infty(I) = 1$
- $\text{cond}(A) = \frac{\|A^{-1}\|}{\|A\|} \geq \frac{\|I\|}{\|I\|} = \text{cond}(I)$ puisque les normes sont sous-multiplicatives. Ainsi, pour les normes usuelles, le conditionnement idéal est 1. Le processus de résolution d'un système linéaire ne peut donc qu'amplifier la perturbation des coefficients.
- $\text{cond}(\alpha A) = \text{cond}(A)$ si $\alpha \neq 0$. Il est donc inutile de tenter d'améliorer le conditionnement d'un système linéaire en multipliant toutes les lignes du système linéaire par une même constante (c'est-à-dire en considérant que $Ax = b \Leftrightarrow \alpha Ax = \alpha b$).
- Si la matrice A est *normale* (c'est-à-dire si $A^t A = A A^t$), alors en notant $|\lambda|_{\max}$ et $|\lambda|_{\min}$ respectivement les plus grand et plus petit module de valeur propre de A :

$$\text{cond}_2(A) = \rho(A)\rho(A^{-1}) = \frac{|\lambda|_{\max}}{|\lambda|_{\min}}$$

- Si A est *orthogonale* (c'est-à-dire si $A^t = A^{-1}$), alors $\text{cond}_2(A) = 1$. Un système linéaire dont la matrice est orthogonale est idéal au plan du conditionnement. Cela explique l'intérêt particulier porté aux matrices orthogonales en analyse numérique.

Notion de préconditionnement d'une matrice

Lorsque l'on doit résoudre un système linéaire mal conditionné, il peut être intéressant d'appliquer une technique dite de *préconditionnement* qui améliore (donc diminue) le conditionnement de la matrice du système linéaire.

Les techniques de préconditionnement les plus usuelles ont pour principe de multiplier le système linéaire, à gauche, par une matrice non singulière, C , judicieusement choisie :

$$Ax = b \Leftrightarrow C Ax = C b.$$

On peut distinguer 2 cas particuliers de matrice C , extrêmes en termes d'efficacité :

- $C = I$ (matrice identité), choix dont la complexité de mise en œuvre est nulle, mais l'efficacité sur le conditionnement nulle également ;
- $C = A^{-1}$, choix dont la complexité de mise en œuvre est élevée par rapport à la complexité d'un algorithme de résolution de système linéaire (cf. remarque sur l'inversion de matrice au § 2), mais dont l'efficacité en termes d'amélioration du conditionnement est totale.

Ces deux cas particuliers montrent bien que pour choisir la matrice de préconditionnement C , il convient de faire un compromis entre le supplément de complexité qu'induit cette technique, et l'amélioration attendue sur le conditionnement.

Il existe diverses techniques de préconditionnement, qui ne sont pas développées ici. On peut toutefois citer l'une des plus simples, dite *préconditionnement de Jacobi*, qui consiste à diviser chaque ligne du système linéaire par le terme diagonal correspondant de la matrice (supposé non nul) :

$$C = \text{diag}(A)^{-1} = \begin{pmatrix} (a_{11})^{-1} & 0 & . & . & 0 \\ 0 & (a_{22})^{-1} & 0 & . & . \\ . & 0 & . & & \\ . & . & & & 0 \\ 0 & . & & 0 & (a_{nn})^{-1} \end{pmatrix} \quad (\text{si } a_{ii} \neq 0 \quad \forall i)$$

Ce préconditionnement est en particulier efficace pour les matrices à diagonale dominante.

chapitre II

Algorithmes directs

On considère le système linéaire de n équations à n inconnues, $Ax = b$, à solution unique (on suppose que $\det(A) \neq 0$). Les notations concernant les coefficients et inconnues de ce système linéaire sont définies au chapitre I.

On présente dans ce chapitre les principaux algorithmes *directs* permettant la résolution de tels systèmes linéaires, à savoir l'algorithme de Gauss (simple ou avec pivotage), l'algorithme LU et ses variantes, et l'algorithme de Householder.

Définition

Un algorithme *direct* fournit la solution du système en un nombre fini d'opérations arithmétiques (+, -, ×, ÷).

1 SYSTEMES LINEAIRES TRIANGULAIRES

Les algorithmes directs consistent à remplacer le système linéaire initial par un système triangulaire équivalent (ou un ensemble équivalent de systèmes triangulaires), qui est ensuite résolu. C'est pourquoi cette section préliminaire traite des algorithmes de résolution de systèmes triangulaires, qui sont à associer, dans les algorithmes directs, aux différents algorithmes de triangulation.

1.1 Systèmes triangulaires inférieurs

On appelle *système triangulaire inférieur*, un système linéaire dont la matrice A est triangulaire inférieure, c'est-à-dire telle que $a_{ij} = 0 \quad \forall i \leq n, \quad \forall j / i < j \leq n$.

Le système s'écrit alors :

$$\begin{cases} a_{11}x_1 = b_1 \\ a_{21}x_1 + a_{22}x_2 = b_2 \\ \dots \\ a_{i1}x_1 + a_{i2}x_2 + a_{i3}x_3 + \dots + a_{ii}x_i = b_i \\ \dots \\ a_{n1}x_1 + a_{n2}x_2 + a_{n3}x_3 + \dots + a_{nn}x_n = b_n \end{cases}$$

La résolution d'un tel système s'effectue en considérant les équations par ordre croissant : de la première équation on peut déduire x_1 , puis x_2 de la 2^{ème} équation, ..., x_i de la $i^{\text{ème}}$ équation, et ainsi de suite jusqu'à x_n .

Ce processus est possible puisque, le système étant supposé de Cramer, aucun des

termes diagonaux de la matrice n'est nul.

L'algorithme de résolution d'un système triangulaire inférieur de Cramer s'écrit donc :

$$\boxed{\begin{array}{l} x_1 = \frac{b_1}{a_{11}} \\ \text{Pour } i = 1, 2, \dots, n \text{ faire} \\ \quad x_i = (b_i - \sum_{j=1}^{i-1} a_{ij}x_j) / a_{ii} \\ \text{fin faire} \end{array}}$$

La complexité de cet algorithme (définie comme le nombre de multiplications et divisions, cf. chapitre I, § 2) est $\frac{n(n+1)}{2} = O(n^2)$.

1.2 Systèmes triangulaires supérieurs

Définition

On appelle *système triangulaire supérieur*, un système linéaire dont la matrice est triangulaire supérieure, c'est-à-dire telle que $a_{ij} = 0 \quad \forall i \leq n, \quad \forall j / j < i$.

Un tel système linéaire a l'allure suivante :

$$\left\{ \begin{array}{l} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \dots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \dots + a_{2,n-1}x_{n-1} = b_2 \\ \dots \\ a_{i1}x_1 + a_{i2}x_2 + a_{i3}x_3 + \dots + a_{in}x_n = b_i \\ \dots \\ a_{n-1,n-1}x_{n-1} + a_{n-1,n}x_n = b_{n-1} \\ a_{nn}x_n = b_n \end{array} \right.$$

La résolution s'effectue en considérant les équations par ordre décroissant. De la dernière équation on peut déduire x_n , puis on tire x_{n-1} de la $n-1^{\text{ème}}$ équation, ..., x_i de la $i^{\text{ème}}$ équation, ..., et ainsi de suite en remontant jusqu'à x_1 .

D'où l'algorithme de résolution d'un système triangulaire supérieur de Cramer :

$$\begin{array}{|l}
 x_n = \frac{b_n}{a_{nn}} \\
 \text{Pour } i = n-1, n-2, \dots, 1 \text{ faire} \\
 x_i = (b_i - \sum_{j=i+1}^n a_{ij}x_j) / a_{ii} \\
 \text{fin faire}
 \end{array}$$

La complexité de cet algorithme est identique à celle de la résolution d'un système triangulaire inférieur : $O(n^2)$.

2 ALGORITHME DE TRIANGULATION DE GAUSS

2.1 Principe

Principe de l'algorithme de Gauss

L'algorithme de triangulation de Gauss est une généralisation de la méthode dite d'*élimination*, qui permet la résolution *à la main* des petits systèmes linéaires.

La méthode consiste à exprimer la première inconnue x_1 en fonction des autres inconnues à partir de la première équation, puis à reporter cette expression dans les équations suivantes, qui deviennent ainsi indépendantes de x_1 (x_1 a été *éliminé*). Puis on exprime x_2 à partir de la 2^{ème} équation, et on reporte son expression dans les équations suivantes (on élimine x_2), et ainsi de suite jusqu'à obtenir un système triangulaire supérieur équivalent au système initial, qu'il suffit de résoudre en appliquant l'algorithme présenté au § 1.2.

Exemple de procédé d'élimination sur un système 3 x 3

Considérons le système linéaire de 3 équations à 3 inconnues : (S_0)

$$\begin{cases} 2x_1 + x_2 + x_3 = 4 \\ 3x_1 + 2x_2 + x_3 = 5 \\ x_1 + 3x_2 + 3x_3 = 7 \end{cases}$$

De la première équation on peut déduire que $x_1 = 2 - \frac{x_2}{2} - \frac{x_3}{2}$, expression que l'on reporte dans les équations 2 et 3 pour obtenir le système (S_1) équivalent à (S_0) :

$$(S_0) \Leftrightarrow (S_1) \quad \begin{cases} 2x_1 + x_2 + x_3 = 4 \\ \frac{1}{2}x_2 - \frac{1}{2}x_3 = -1 \\ \frac{5x_2}{2} + \frac{5x_3}{2} = 5 \end{cases}$$

De la deuxième équation de (S_1) on peut déduire que $x_2 = -2 + x_3$, expression que l'on reporte dans l'équation 3 pour obtenir le système (S_2) équivalent à (S_1) et (S_0) :

$$(S_0) \Leftrightarrow (S_1) \Leftrightarrow (S_2) \quad \begin{cases} 2x_1 + x_2 + x_3 = 4 \\ \frac{x_2}{2} - \frac{x_3}{2} = -1 \\ 5x_3 = 10 \end{cases}$$

Notons que ce procédé a pu aboutir, après élimination de x_1 et x_2 , au système triangulaire (S_2) , grâce au fait que les termes diagonaux par lesquels on a effectué des divisions (on les nomme les *pivots*), étaient non nuls.

Les matrices et seconds membres des systèmes linéaires (S_0) , (S_1) , (S_2) sont successivement :

$$A^{(0)} = A = \begin{pmatrix} 2 & 1 & 1 \\ 3 & 2 & 1 \\ 1 & 3 & 3 \end{pmatrix} \quad b^{(0)} = b = \begin{pmatrix} 4 \\ 5 \\ 7 \end{pmatrix} \quad A^{(1)} = \begin{pmatrix} 2 & 1 & 1 \\ 0 & 0.5 & -0.5 \\ 0 & 2.5 & 2.5 \end{pmatrix} \quad b^{(1)} = \begin{pmatrix} 4 \\ -1 \\ 5 \end{pmatrix}$$

$$A^{(2)} = \begin{pmatrix} 2 & 1 & 1 \\ 0 & 0.5 & -0.5 \\ 0 & 0 & 5 \end{pmatrix} \quad b^{(2)} = \begin{pmatrix} 4 \\ 1 \\ 10 \end{pmatrix}$$

2.2 Algorithme de Gauss

On considère le principe d'élimination des inconnues, décrit sur un exemple au § 2.1, que l'on applique à un système de taille quelconque :

$$(S_0) \begin{cases} a_{11}^{(0)} x_1 + a_{12}^{(0)} x_2 + \dots + a_{1n}^{(0)} x_n = b_1^{(0)} \\ a_{21}^{(0)} x_1 + a_{22}^{(0)} x_2 + \dots + a_{2n}^{(0)} x_n = b_2^{(0)} \\ \dots \\ a_{i1}^{(0)} x_1 + a_{i2}^{(0)} x_2 + \dots + a_{in}^{(0)} x_n = b_i^{(0)} \\ \dots \\ a_{n1}^{(0)} x_1 + a_{n2}^{(0)} x_2 + \dots + a_{nn}^{(0)} x_n = b_n^{(0)} \end{cases}$$

où $A^{(0)} = A$ et $b^{(0)} = b$ sont respectivement la matrice et le second membre du système linéaire à résoudre. Si le premier *pivot* $a_{11}^{(0)}$ est non nul, on peut éliminer x_1 :

$$x_1 = \left(b_1^{(0)} - \sum_{j=2}^n a_{1j}^{(0)} x_j \right) / a_{11}^{(0)}$$

On reporte cette expression de x_1 dans le système (S_0) pour obtenir le système (S_1) équivalent au système (S_0) , dont la matrice et le second membre sont respectivement $A^{(1)}$ et $b^{(1)}$:

$$(S_1) \begin{cases} a_{11}^{(1)} x_1 + a_{12}^{(1)} x_2 + \dots + a_{1n}^{(1)} x_n = b_1^{(1)} \\ a_{21}^{(1)} x_1 + a_{22}^{(1)} x_2 + \dots + a_{2n}^{(1)} x_n = b_2^{(1)} \\ \dots \\ a_{i1}^{(1)} x_1 + a_{i2}^{(1)} x_2 + \dots + a_{in}^{(1)} x_n = b_i^{(1)} \\ \dots \\ a_{n1}^{(1)} x_1 + a_{n2}^{(1)} x_2 + \dots + a_{nn}^{(1)} x_n = b_n^{(1)} \end{cases} \quad \text{avec} \quad \begin{cases} a_{1j}^{(1)} = a_{1j}^{(0)} \\ a_{j1}^{(1)} = 0 \end{cases} \quad \forall j, \quad b_1^{(1)} = b_1^{(0)}$$

$$a_{ij}^{(1)} = a_{ij}^{(0)} - \frac{a_{i1}^{(0)} a_{1j}^{(0)}}{a_{11}^{(0)}} \quad b_i^{(1)} = b_i^{(0)} - \frac{a_{i1}^{(0)} b_1^{(0)}}{a_{11}^{(0)}} \quad i, j = 2, \dots, n$$

On a ainsi, en supposant le pivot $a_{11}^{(0)}$ non nul, remplacé le système initial $A^{(0)}x = b^{(0)}$ par un système équivalent, $A^{(1)}x = b^{(1)}$, tel que la partie sous-diagonale de la 1^{ère} colonne de $A^{(1)}$ soit nulle. La figure II.1 présente l'algorithme d'élimination de x_1 , qui permet de calculer les coefficients du système linéaire (S_1) .

Dans l'algorithme de Gauss (cf. figure II.2), cette procédure est ensuite appliquée aux $n-1$ dernières lignes du système linéaire (S_1) (si le pivot $a_{22}^{(1)}$ est non nul), créant ainsi le système équivalent $A^{(2)}x = b^{(2)}$, tel que la partie sous-diagonale des deux premières colonnes de la matrice $A^{(2)}$ est nulle, ..., et ainsi de suite jusqu'au système triangulaire supérieur $A^{(n-1)}x = b^{(n-1)}$.

La résolution de ce système triangulaire peut alors être réalisée grâce à l'algorithme présenté au § 1.2.

Remarques

- Une des caractéristiques de l'algorithme de Gauss est que les calculs successifs de coefficients concernent à la fois les matrices et les seconds membres, contrairement aux algorithmes directs de type LU (cf. § 3). Cet élément de comparaison est susceptible, dans certains cas, d'influencer le choix d'un algorithme direct adéquat.
- Les matrices successives obtenues par l'algorithme de triangulation de Gauss ont toutes le même déterminant, égal au déterminant de A . La dernière matrice, $A^{(n-1)}$, étant triangulaire supérieure, on a donc $\det A = \prod_{i=1}^n a_{ii}^{(n-1)}$. L'algorithme de Gauss constitue ainsi l'une des méthodes de calcul d'un déterminant.

```

 $a_{11}^{(1)} = a_{11}^{(0)}$ 
 $b_1^{(1)} = b_1^{(0)}$ 
pour  $j = 2, 3, \dots, n$ 
 $a_{1j}^{(1)} = a_{1j}^{(0)}$ 
 $a_{j1}^{(1)} = 0$ 
fin pour
pour  $i = 2, 3, \dots, n$ 
pour  $j = 2, 3, \dots, n$ 
 $a_{ij}^{(1)} = a_{ij}^{(0)} - \frac{a_{i1}^{(0)} a_{1j}^{(0)}}{a_{11}^{(0)}}$ 
fin pour
 $b_i^{(1)} = b_i^{(0)} - \frac{a_{i1}^{(0)} b_1^{(0)}}{a_{11}^{(0)}}$ 
fin pour

```

figure II.1 – Algorithme d'élimination de x_1

```

pour  $k = 1, 2, \dots, n-1$ 
pour  $i = 1, 2, \dots, k$ 
 $a_{ii}^{(k)} = a_{ii}^{(k-1)}$ 
 $b_i^{(k)} = b_i^{(k-1)}$ 
pour  $j = i+1, i+2, \dots, n$ 
 $a_{ij}^{(k)} = a_{ij}^{(k-1)}$ 
 $a_{ji}^{(k)} = 0$ 
fin pour
fin pour
pour  $i = k+1, k+2, \dots, n$ 
pour  $j = k+1, k+2, \dots, n$ 
 $a_{ij}^{(k)} = a_{ij}^{(k-1)} - \frac{a_{ik}^{(k-1)} a_{kj}^{(k-1)}}{a_{kk}^{(k-1)}}$ 
fin pour
 $b_i^{(k)} = b_i^{(k-1)} - \frac{a_{ik}^{(k-1)} b_k^{(k-1)}}{a_{kk}^{(k-1)}}$ 
fin pour
fin pour

```

figure II.2 – Algorithme de Gauss

Mise en œuvre de l'algorithme de Gauss

On peut remarquer que dans l'algorithme de Gauss (cf. figure II.2), lors du calcul des coefficients du système linéaire $A^{(k)}x = b^{(k)}$, chaque coefficient du système linéaire $A^{(k-1)}x = b^{(k-1)}$, $a_{ij}^{(k-1)}$ ou $b_i^{(k-1)}$ ne sert que lors du calcul du nouveau coefficient

correspondant, respectivement $a_{ij}^{(k)}$ ou $b_i^{(k)}$. Ceci permet de stocker tous les coefficients des systèmes linéaires successifs dans un seul tableau.

Il est par exemple possible d'utiliser un tableau rectangulaire, noté C , à n lignes et $n+1$ colonnes. Les n premières colonnes de C sont initialisées avec les coefficients de la matrice A du système linéaire à résoudre, et la $n+1^{\text{ème}}$ colonne est initialisée avec les coefficients du second membre b . Au cours de la procédure de triangulation, les valeurs de C sont transformées selon l'algorithme de Gauss. Il est classique, pour gagner du temps, de ne pas réaliser effectivement la mise à zéro des coefficients de la sous-diagonale. L'organigramme de mise en œuvre de l'algorithme de Gauss présenté sur la figure II.3 correspond à cette option.

2.3 Exemples

On considère 2 exemples de mise en œuvre de l'algorithme de Gauss, selon le mode de stockage et l'organigramme de la figure II.3, sur des systèmes linéaires de 4 équations à 4 inconnues.

Exemple 1

On considère le système linéaire $Ax = b$, avec $A = \begin{pmatrix} 2 & 1 & 1 & 1 \\ 1 & 2 & 1 & 1 \\ 1 & 1 & 2 & 1 \\ 1 & 1 & 1 & 2 \end{pmatrix}$ et $b = \begin{pmatrix} 5 \\ 5 \\ 5 \\ 5 \end{pmatrix}$, dont la

solution exacte est $s = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$. La mise en œuvre en simple précision (32 bits) de

l'algorithme de Gauss conduit à un système triangulé dont les coefficients peuvent être trouvés dans le tableau C final de la figure II.4.

La résolution du système triangulaire fournit ensuite la solution numérique du système

linéaire : $s' = \begin{pmatrix} 1.000000 \\ 1.000000 \\ 0.9999999 \\ 1.000000 \end{pmatrix}$. L'erreur relative, $\frac{\|s' - s\|_{\infty}}{\|s\|_{\infty}}$, est inférieure à 10^{-7} , du fait de

la petite taille du système linéaire et de son bon conditionnement.

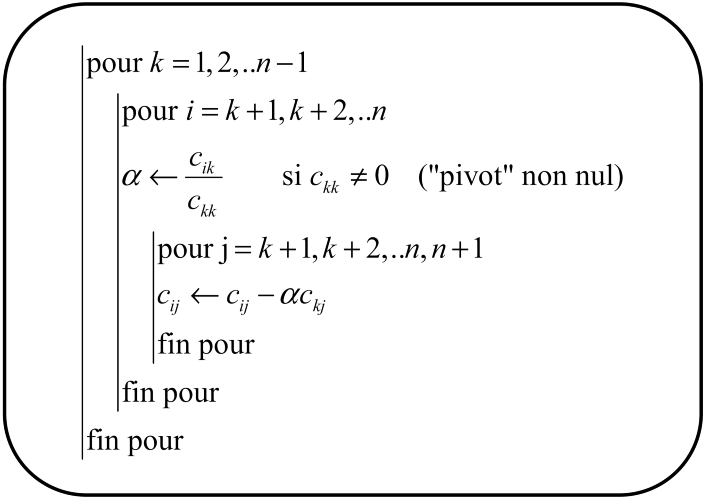


figure II.3 – Organigramme de mise en œuvre de l’algorithme de Gauss ; stockage de tous les coefficients dans un seul tableau rectangulaire C ; les coefficients sous-diagonaux ne sont pas mis à zéro

2.0000000	1.0000000	1.0000000	1.0000000	5.0000000
1.0000000	2.0000000	1.0000000	1.0000000	5.0000000
1.0000000	1.0000000	2.0000000	1.0000000	5.0000000
1.0000000	1.0000000	1.0000000	2.0000000	5.0000000
tableau C initial				
2.0000000	1.0000000	1.0000000	1.0000000	5.0000000
1.0000000	1.5000000	0.5000000	0.5000000	2.5000000
1.0000000	0.5000000	1.3333333	0.3333333	1.6666667
1.0000000	0.5000000	0.3333333	1.2500000	1.2500000
tableau C final				

figure II.4 – Etats initial et final du tableau C associé à l’exemple 1, après mise en œuvre de l’algorithme de Gauss selon l’organigramme de la figure II.3, en simple précision

Exemple 2

On considère le système linéaire $Ax = b$, avec $A = \begin{pmatrix} 10 & 7 & 8 & 7 \\ 7 & 5 & 6 & 5 \\ 8 & 6 & 10 & 9 \\ 7 & 5 & 9 & 10 \end{pmatrix}$ et $b = \begin{pmatrix} 32 \\ 23 \\ 33 \\ 31 \end{pmatrix}$, dont

la solution exacte est $s = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$. La mise en œuvre en simple précision (32 bits) ou en

double précision (64 bits) de l'algorithme de Gauss conduit à un système triangulé dont les coefficients peuvent être trouvés dans le tableau C final de la figure II.5.

En simple précision, la résolution du système triangulaire fournit comme solution numérique du système linéaire :

$$s' = \begin{pmatrix} 0.9999927 \\ 1.000012 \\ 0.9999968 \\ 1.000002 \end{pmatrix}$$

L'erreur relative $\frac{\|s' - s\|_{\infty}}{\|s\|_{\infty}}$, de l'ordre de 10^{-4} , est élevée compte tenu de la petite taille de ce système linéaire, ce qui est dû à son mauvais conditionnement (cf. § 3.1).

En double précision, la solution numérique devient :

$$s'' = \begin{pmatrix} 0.999999999999995 \\ 1.000000000000001 \\ 0.999999999999997 \\ 1.000000000000000 \end{pmatrix}.$$

L'erreur relative $\frac{\|s'' - s\|_{\infty}}{\|s\|_{\infty}}$, de l'ordre de 10^{-15} , montre que la précision des calculs permet de limiter les effets néfastes du mauvais conditionnement.

2.4 Complexité

La complexité de l'algorithme de Gauss est un $O(n^3)$. Ce résultat appelle plusieurs remarques :

- La complexité de la résolution d'un système triangulaire est un $O(n^2)$. Pour les gros systèmes linéaires, le temps de résolution du système triangulé est donc négligeable devant le temps de triangulation.
- Le degré de finesse d'un maillage (éléments finis, différences finies ...) a une énorme influence sur le temps de résolution du système linéaire résultant de la discrétisation des équations. Si par exemple, en affinant un maillage, on multiplie

le nombre d’inconnues par 10, le temps de résolution du système linéaire par l’algorithme de Gauss est alors multiplié par 1000.

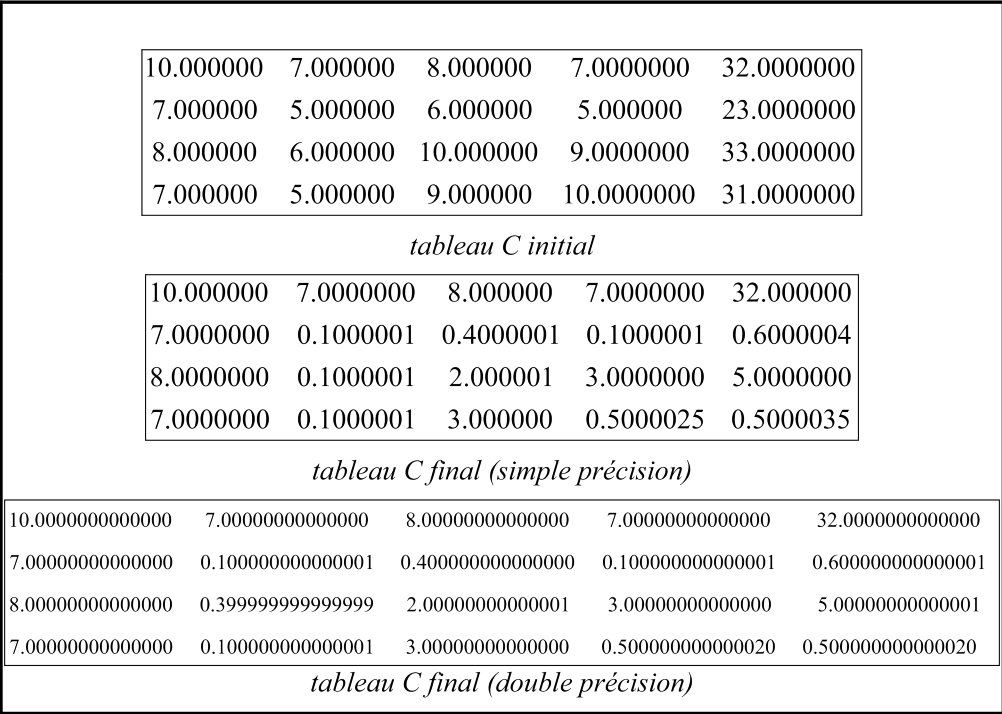


figure II.5 – Etats initial et final du tableau C associé à l'exemple 2, après mise en œuvre de l'algorithme de Gauss selon l'organigramme de la figure II.3, en simple précision, et double précision

2.5 Phénomène d’instabilité

Au cours de l’algorithme de triangulation, on effectue un certain nombre de divisions par des coefficients, nommés *pivots* (les c_{kk} dans l’organigramme de la figure II.3). Il faut donc s’attendre à un blocage lorsque l’on rencontre un pivot nul. Par ailleurs, les valeurs non nulles mais très petites de pivots conduisent à des solutions numériques non satisfaisantes ; ce phénomène est appelé *instabilité de l’algorithme de Gauss*.

Le système linéaire $\begin{pmatrix} \varepsilon & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 0.5 \\ 1 \end{pmatrix}$, dont la solution exacte est $s = \frac{0.5}{1-\varepsilon} \begin{pmatrix} 1 \\ 1-2\varepsilon \end{pmatrix}$ permet d’illustrer simplement ce phénomène d’instabilité. Le procédé de triangulation de Gauss conduit au système équivalent : $\begin{pmatrix} \varepsilon & 1 \\ 0 & 1-\frac{1}{\varepsilon} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 0.5 \\ 1-\frac{1}{2\varepsilon} \end{pmatrix}$.