Laurence Lamoulie
Chrysostome Baskiotis

ANALYSE NUMÉRIQUE

Fascicule 2

Algèbre linéaire

EISTI

Année 2012

INTRODUCTION

Dans la deuxième partie de ce cours nous présentons des méthodes de résolution des systèmes linéaires ainsi que des méthodes d'inversion matricielle.

4

ALGÈBRE LINÉAIRE ET PERTURBATIONS

Nous présentons, au chapitre suivant, des méthodes itératives de résolution d'un système d'équations linéaires et d'inversion d'une matrice. Avant ces méthodes, nous essaierons d'évaluer l'influence sur la solution d'un système linéaire d'une perturbation du second ou du premier membre. Pour établir cette évaluation il est utile de pouvoir évaluer de combien est différente une matrice d'une autre. Par exemple savoir la différence entre $A \cdot A^{-1}$, où A^{-1} est issue d'un calcul numérique, et la matrice identité I. Nous commençons donc par la notion de la norme qui permet de répondre à ce dernier problème.

4.1 Normes vectorielles et matricielles

L'évaluation de la proximité de deux vecteurs se fait en utilisant leur distance qui, à son tour, est calculée en utilisant la notion de la norme d'un vecteur. Pour évaluer donc la proximité

de deux matrices nous devons utiliser une notion analogue à celle de la norme vectorielle, à savoir la norme matricielle. Comme celle-ci est fondée sur celle-là, nous ferons dans la suite un bref rappel des normes vectorielles, avant de présenter les noemes matricielles et leur propriétés. Nous aborderons ensuite l'étude des perturbations de la solution d'un système linéaire.

Remarquons que tous les espaces que nous utiliserons seront de dimension finie.

4.1.1 Normes vectorielles

Plaçons-nous dans $\mathbb{R}^n$ avec n fini. La norme d'un vecteur $x \in \mathbb{R}^n$ est une application $n : \mathbb{R}^n \to \mathbb{R}_+$ qu'on note aussi $\|\cdot\|$, ayant les propriétés suivantes :

(1) $\|x\| \geq 0$ et $\|x\| = 0$ si et seulement si $x = 0$

(2) $\|\alpha x\| = |\alpha| \, \|x\|$, pour $\alpha \in \mathbb{R}$

(3) $\|x + y\| \leq \|x\| + \|y\|$

Une propriété qu'on utilise souvent est la suivante

$$|\, \|x\| - \|y\| \,| \leq \|x - y\|$$

Étant donnés deux vecteurs $x, y \in \mathbb{R}^n$, leur distance est calculée à partir de la norme de leur différence

$$d(x, y) = \|x - y\|$$

La définition la plus générale d'une telle norme est donnée par

$$\|x\|_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{1/p} ; \; p \geq 1 \tag{4.3}$$

Les normes les plus utilisées sont issues de la formule précédente pour $p = 1, 2$ et ∞ :

— Norme 1 : $\|x\|_1 = \sum_{i=1}^{n} |x_i|$.

— Norme 2 ou eudidienne : $\|x\|_2 = \|x\|_2 = \sqrt{x \, x}$

— Norme ∞ ou max : $\|x\|_\infty = \max_{i=1,\ldots n} |x_i|$

De plus, nous avons deux inégalités très utiles pour les calculs

— Inégalité de Hölder

$$\sum_{i=1}^{n} |x_i y_i| \leq \|x\|_p \, \|y\|_q = \left(\sum_{i=1}^{n} |x_i|^p \right)^{1/p} \left(\sum_{i=1}^{n} |y_i|^q \right)^{1/q}$$

— Inégalité de Cauchy (qui est l'inégalité de Hölder pour $p = q = 2$)

$$\sum_{i=1}^{n} |x_i y_i| \leq \sqrt{\sum_{i=1}^{n} |x_i|} \cdot \sqrt{\sum_{i=1}^{n} |y_i|}$$

et les deux théorèmes suivants :

THÉORÈME 4.1.1 Chaque norme $\|\cdot\|$ de $\mathbb{R}^n$ est un application uniformément continue de $\mathbb{R}^n$ muni de la norme max dans $\mathbb{R}$.

DÉMONSTRATION. Du fait que $|\,\|x+\eta\| - \|x\|\,| \leq \|\eta\|$ et si on note $\eta = \sum_{i=1}^{n} \eta_i e_i$, où $\eta = [\eta_1, \ldots, \eta_n]$ et $[e_1, \ldots, e_n]$ la base canonique de $\mathbb{R}^n$, alors nous avons

$$\|\eta\| \leq \sum_{i=1}^{n} |\eta_i| \, \|e_i\| \leq \max_i |\eta_i| \sum_{j=1}^{n} \|e_j\| = M \cdot \|\eta\|_\infty \text{, avec } M = \sum_{j=1}^{n} \|e_j\|$$

Donc pour chaque $\varepsilon > 0$ nous pouvons choisir η tel que $\|\eta\|_\infty = \max_i |\eta_i| \leq \frac{\varepsilon}{M}$, c-à-d. choix indépendant de x, et avoir l'inégalité

$$|\,\|x+\eta\| - \|x\|\,| \leq \varepsilon$$

Par conséquent $\|\cdot\|$ est uniformément continue. □

DÉFINITION 4.1.1 Deux normes $\|\cdot\|_{n_1}$ et $\|\cdot\|_{n_2}$ de $\mathbb{R}^n$ sont équivalentes si

$$\exists c, C \in \mathbb{R}_+ \text{ tels que } \forall x \in \mathbb{R}^n : c\,\|x\|_{n_2} \leq \|x\|_{n_1} \leq C\,\|x\|_{n_2}$$

On note l'équivalence entre les deux normes $\|\cdot\|_{n_1} \sim_n \|\cdot\|_{n_2}$.

Nous avons le théorème suivant :

THÉORÈME 4.1.2 Toutes les normes $\|\cdot\|$ de $\mathbb{R}^n$ sont équivalentes.

DÉMONSTRATION. Considérons l'espace $\mathbb{R}^n$ muni de la norme max $\|\cdot\|_\infty$. On va montrer que toute norme $\|\cdot\|_n$ est équivalente à $\|\cdot\|_\infty$.

Considérons la sphère unité pour la norme max $S = \{x \in \mathbb{R}^n \,/\, \|x\|_\infty = 1\}$. Il s'agit d'un ensemble compact dans $\mathbb{R}^n$. Du fait que $\|x\|_\infty$ est continue d'après le théorème précédent, nous avons que les quantités $c = \min_{x \in S} \|x\|_n > 0$ et $C = \max_{x \in S} \|x\|_n > 0$ existent. Donc pour tout $x \in \mathbb{R}^n$ tel que $x = 0$ on a $x = \frac{x}{\|x\|_\infty} \in S$ et par conséquent $c \leq \|x\|_n \leq C$. Nous avons aussi $\|x\|_n = \left\|\frac{x}{\|x\|_\infty}\right\|_n = \frac{1}{\|x\|_\infty}\|x\|_n$ d'où

$$c\,\|x\|_\infty \leq \|x\|_n \leq C\,\|x\|_\infty$$

□

Le choix de la norme est un élément très important pour un problème d'optimisation de la forme

$$\min_{x \in \mathbb{R}^n} \|Ax - y\|_p$$

Nous pouvons montrer que des valeurs de p qui sont proches de 1 donnent des algorithmes plus stables que des valeurs de p proches de 2.

4.1.2 Normes matricielles

Une norme matricielle est une application $\|\cdot\| : \mathbb{R}^{m\times n} \to \mathbb{R}_+$ qui a les propriétés suivantes :

(1) $\|A\| \geq 0$ et $\|A\| = 0$ si et seulement si $A = 0$.

(2) $\|\alpha A\| = |\alpha| \, \|A\| \quad \forall \alpha \in \mathbb{R}$.

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

Si, de plus, elle vérifie la relation

(4) $\|AB\| \leq \|A\| \, \|B\|$

on dit qu'elle est une norme sous-multiplicative.

Nous pouvons remarquer que toute norme vectorielle peut induire une norme matricielle à l'aide de la relation

$$\|A\| = \sup_{\substack{x\in\mathbb{R}^n \\ \|x\|=1}} \|Ax\| = \sup_{\substack{x\in\mathbb{R}^n \\ x\neq 0}} \frac{\|Ax\|}{\|x\|}$$

Cette norme est appelée norme subordonnée (à la norme vectorielle correspondante) et elle sera notée par

$$\text{lub}(A) = \sup_{\substack{x\in\mathbb{R}^n \\ x\neq 0}} \frac{\|Ax\|}{\|x\|}$$

Pour prouver que lub est une norme, considérons une valeur p telle que $1 \leq p \leq +\infty$. À chaque vecteur $x \in \mathbb{R}^n$ on fait correspondre un réel, qui est la norme vectorielle $\|x\|_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{1/p}$.

Donc $\|A\|_p = \sup_{\substack{x\in\mathbb{R}^n \\ x\neq 0}} \frac{\|Ax\|_p}{\|x\|_p}$ a les propriétés d'une norme matricielle, à savoir

— $\|A\|_p \geq 0$ et, par convention, $\|A\|_p = 0$ si et seulement si $x = 0$.

— $\|\alpha A\|_p = \sup_{\substack{x\in\mathbb{R}^n \\ x\neq 0}} \frac{\|\alpha A x\|_p}{\|x\|_p} = |\alpha| \sup_{\substack{x\in\mathbb{R}^n \\ x\neq 0}} \frac{\|Ax\|_p}{\|x\|_p} = |\alpha| \, \|A\|_p$

— $\|A + B\|_p = \sup_{\substack{x\in\mathbb{R}^n \\ x\neq 0}} \frac{\|(A+B)x\|_p}{\|x\|_p} \leq \sup_{\substack{x\in\mathbb{R}^n \\ x\neq 0}} \frac{\|Ax\|_p}{\|x\|_p} + \frac{\|Bx\|_p}{\|x\|_p} = \|A\|_p + \|B\|_p$

De plus, pour les normes subordonnées, nous avons que $\text{lub}(I) = 1$, où I la matrice identité.

Notons que géométriquement $\text{lub}(A)$ exprime la quantité maximale que la norme du point image Ax peut excéder la norme du point de départ x.

De ce qui précède nous pourrions conclure – abusivement – que pour toute valeur de p entre 1 et $+\infty$ nous pouvons déterminer une norme. En réalité il n'en est rien et nous pouvons déterminer des normes subordonnées pour les valeurs $p = 1, p = 2$ et $p = \infty$ seulement. On a ainsi les trois normes subordonnées :

— Norme 1 – somme des colonnes : $\|A\|_1 = \sup_{\substack{x \in \mathbb{R}^n \\ \|x\|_1 = 1}} \|Ax\|_1 = \max_{j=1,\ldots,n} \sum_{i=1}^{n} |a_{ij}|$

— Norme 2 : $\|A\|_2 = \sup_{\substack{x \in \mathbb{R}^n \\ \|x\|_2 = 1}} \|Ax\|_2 = \sqrt{\rho(A^* A)} = \|A^*\|_2$, où par $\rho(A)$ on note le rayon spectral de A, c-à-d. la plus grande, en module, valeur propre de A.

— Norme infinie – somme des lignes : $\|A\|_\infty = \sup_{\substack{x \in \mathbb{R}^n \\ \|x\|_\infty = 1}} \|Ax\|_\infty = \max_{i=1,\ldots,n} \sum_{j=1}^{n} |a_{ij}|$

Une norme matricielle $\|\cdot\|$ est consistante avec une norme vectorielle $\|\cdot\|$ si $\|Ax\| \leq \|A\| \cdot \|x\|$. Notons que la norme 2 est consistante avec la norme euclidienne.

Les normes consistantes et subordonnées sont sous-multiplicatives, i.e.

$$\|AB\| \leq \|A\| \|B\| \tag{4.4}$$

Une norme qui n'est pas induite par une norme vectorielle est la norme de Frobenius (ou de Schur) donnée par

$$\|A\|_F = \left(\sum_{i=1,j=1}^{m \quad n} |a_{ij}|^2 \right)^{1/2} = \sqrt{\mathrm{tr}(A^* A)}; \; A \in \mathbb{R}^{m \times n}$$

où par tr on a noté la trace de la matrice.

Pour la norme de Frobenius, nous avons $\|I_n\|_F = \sqrt{n}$

On termine cette section par un résultat important :

PROPRIÉTÉ 4.1.1 Soit $\|\cdot\|$ une norme quelconque. Alors le rayon spectral d'une matrice carrée A est au plus égal à la norme de cette matrice

$$\rho(A) \leq \|A\| \tag{4.5}$$

4.1.3 Exercices

EXERCICE 4.1 Calculer les normes 1, 2 et ∞ du vecteur $x = [1, 0, 1, -4]^*$.

EXERCICE 4.2 Calculer les normes 1, 2 et ∞ de la matrice

$$A = \begin{matrix} 2 & 1 & 1 \\ 2 & 3 & 2 \\ 1 & 1 & 2 \end{matrix}$$

EXERCICE 4.3 (1) $\forall x \in \mathbb{R}^n : \|x\|_\infty \leq \|x\|_p \leq n^{\frac{1}{p}} \|x\|_\infty$

(2) En déduire que $\lim_{p \to +\infty} \|x\|_p = \|x\|_\infty$

EXERCICE 4.4 Considérons une norme vectorielle $\|\cdot\|$ et la norme subordonée $\text{lub}(A)$ que l'on notera $\|A\|$. Montrer que si $A, B \in \mathbb{R}^{n\times n}$, alors

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

(2) $\|A\cdot B\| \leq \|A\|\,\|B\|$

(3) $\|A^n\| \leq \|A\|^n$

EXERCICE 4.5 Soit une matrice carrée $A \in \mathbb{R}^{n\times n}$. Montrer que

(1) A est orthogonale, c'est-à-dire $A = A^{-1}$, si et seulement si $\|Ax\|_2 = \|x\|_2, x \in \mathbb{R}^n$.

(2) $\|A\|_2 = 1$ pour toute matrice orthogonale.

EXERCICE 4.6 Montrer que

(1) $\text{lub}(A) \leq \|A\|$

(2) $\text{lub}(AB) \leq \text{lub}(A)\cdot\text{lub}(B)$

(3) Calculer $\text{lub}_\infty(A)$

(4) Calculer $\text{lub}(A)$ pour la norme 2.

4.2 Conditionnement d'une matrice

On sait qu'un problème est bien conditionné si une petite variation des entrées (données) n'entraîne pas une grande variation des sorties (résultats). Afin de caractériser la nature du conditionnement d'un problème, on calcule une valeur caractéristique de ce conditionnement qui s'appelle nombre-condition ou conditionnement du problème (cf. §1.7). Nous allons établir le conditionnement pour les matrices.

On définit le conditionnement d'une matrice A par la quantité

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

Le conditionnement peut aussi être noté par $\kappa(A)$.

La section suivante fournit une justification du choix que nous venons d'effectuer pour le conditionnement d'une matrice.

4.2.1 Exercice

EXERCICE 4.7 Montrer que $\text{cond}(A) \geq 1$.

EXERCICE 4.8 Monter que $\text{cond}(A^2) \leq (\text{cond}(A))^2$ pour toute matrice carrée A.

4.3 Suite de matrices

La convergence d'une suite de matrices de format $(n \times m)$ est équivalente à la convergence de $n \cdot m$ suites scalaires formées par les termes de ces matrices. Nous avons le théorème suivant :

THÉORÈME 4.3.1 Soit $A \in \mathbb{R}^{n \times n}$ une matrice carrée. Alors les propositions suivantes sont équivalentes :

(1) $\lim_{k \to +\infty} A^k = 0$

(2) $\lim_{k \to +\infty} A^k x = 0; \forall x \in \mathbb{R}^n$

(3) $\rho(A) < 1$

(4) $\|A\| < 1$ pour au moins une norme subordonnée.

4.3.1 Exercice

EXERCICE 4.9 (THÉORÈME DE VON NEUMANN) .- Soit $A \in \mathbb{R}^{n \times n}$ matrice carrée avec $\rho(A) < 1$, alors

(1) La matrice $I - A$ est régulière.

(2) $(I - A)^{-1} = \lim_{n \to +\infty} \sum_{k=0}^{n} A^k = \sum_{k=0}^{\infty} A^k$.

Indication.- Si λ_i est la valeur propre de A, alors $1 - \lambda_i$ est la valeur propre correspondante de la matrice $I - A$.

4.4 Bornes de l'erreur de la solution d'un système linéaire

Considérons le système d'équations linéaires

$$A x = b; \quad x \in \mathbb{R}^n, b \in \mathbb{R}^m \tag{4.1}$$

La solution obtenue par des méthodes numériques est entachée d'erreurs d'arrondi. De cette façon, au lieu d'avoir la solution exacte, nous avons une solution approchée $x + \Delta x$, qui peut être vue comme une solution perturbée de la solution exacte. L'objectif ici est d'évaluer la borne supérieure de la perturbation Δx. À cette analyse de l'erreur directe nous pouvons adjoindre une analyse de l'erreur inverse. On s'intéresse ainsi à la perturbation Δx de A et/ ou à la perturbation Δb de b qui conduisent à la solution $x + \Delta x$ et on exprime Δx en fonction de ΔA et Δb.

On commence par un résultat qui montre l'importance de $\frac{1}{\text{cond}(A)}$.

THÉORÈME 4.4.1 Si A est régulière et si

$$\frac{\|\Delta A\|}{\|A\|} < \frac{1}{\text{cond}(A)} \tag{4.2}$$

alors la matrice $A + \Delta A$ est aussi régulière.

On déduit de ce théorème que l'inverse du nombre-condition de la matrice « mesure » la distance qui sépare A d'une matrice singulière. Donc si le nombre-condition d'une matrice A est grand, alors cette matrice est proche de la singularité.

4.4.1 Perturbations de b

Soit le système $A\,x = b$. Si la place du vecteur b nous avons le vecteur perturbé $b + \Delta b$, la solution x est aussi perturbée et elle devient $x + \Delta x$. Nous avons donc le système linéaire :

$$A\,(x + \Delta x) = b + \Delta b,\ A \in \mathbb{R}^{n \times n}$$

On a $A\,\Delta x = \Delta b$ et donc

$$\Delta x = A^{-1}\Delta b$$

d'où

$$\Delta x \leq \|A^{-1}\|\ \|\Delta b\| \tag{4.3}$$

Puisque $A\,x = b$, on a $\|b\| \leq \|A\|\ \|x\|$ et par conséquent

$$\Delta x\ \|b\| \leq \|A\|\ \|A^{-1}\|\ \|x\|\ \|\Delta b\|$$

et si $\|b\| = 0$, on obtient

$$\frac{\Delta x}{x} \leq \kappa(A)\frac{\Delta b}{b} \tag{4.4}$$

Nous constatons ainsi que si $\kappa(A) \gg 1$, une petite perturbation du vecteur b peut provoquer une grande perturbation de la solution.

4.4.2 Perturbations de A

Si la matrice A est perturbée et on a à sa place $A + \Delta A$, alors le système linéaire devient :

$$(A + \Delta A)(x + \Delta x) = b \tag{4.5}$$

d'où on a

$$A\,\Delta x = -\Delta A\,(x + \Delta x) \tag{4.6}$$

et finalement

$$\Delta x = -A^{-1}\Delta A\,(x + \Delta x) \tag{4.7}$$

En utilisant les normes, on a

$$\|\Delta x\| \leq \|A^{-1}\|\ \|\Delta A\|\ (\|x\| + \|\Delta x\|) \Rightarrow \|\Delta x\|\ (1 - \|A^{-1}\|\ \|\Delta A\|) \leq \|A^{-1}\|\ \|\Delta A\|\ \|x\|$$

ce qui donne pour l'erreur sur x :

$$\frac{\Delta x}{x} \leq \frac{\|A^{-1}\|\ \|\Delta A\|}{1 - \|A^{-1}\|\ \|\Delta A\|} = \frac{\|A\|\ \|A^{-1}\|}{1 - \|A^{-1}\|\ \|\Delta A\|} \cdot \frac{\Delta A}{A} \tag{4.8}$$

c'est-à-dire

$$\frac{\Delta x}{x} \leq \kappa(A) \cdot \frac{\Delta A}{A} \tag{4.9}$$

si $A^{-1} \quad \Delta A \quad 1$.

Nous pouvons donc répéter la remarque de la fin du paragraphe précédent.

4.4.3 Perturbations de A et de b

Si A et b subissent des perturbations, le système linéaire devient :

$$(A + \Delta A)(x + \Delta x) = b + \Delta b \tag{4.10}$$

On a

$$\Delta A \cdot x + A \cdot \Delta x + \Delta A \cdot \Delta x = \Delta b$$

En négligeant les termes du 2e ordre, on obtient

$$\Delta x = -A^{-1} \cdot \Delta A \cdot x + A^{-1} \cdot \Delta b$$

En utilisant la norme on a :

$$\Delta x \leq A^{-1} \cdot \Delta A \cdot x + A^{-1} \cdot \Delta b$$

d'où

$$\frac{\Delta x}{x} \leq A^{-1} \cdot \Delta A + A^{-1} \cdot \frac{\Delta b}{x} \leq \frac{A^{-1} \cdot A \cdot \Delta A}{A} + \frac{A^{-1} \cdot A \cdot \Delta b}{A \cdot x}$$

et finalement

$$\frac{\Delta x}{x} \leq \kappa(A) \cdot \frac{\Delta A}{A} + \frac{\Delta b}{b} \tag{4.11}$$

Des trois formules (4.4), (4.9) et (4.11) on conclut que pour avoir une petite modification Δx du vecteur x il n'est pas suffisant que les erreurs relatives $\frac{\Delta A}{A}$ et $\frac{\Delta b}{b}$ soient petites. Il faut, de plus, que la matrice A soit bien conditionnée.

4.4.4 Exercices

EXERCICE 4.10 Soit le système $Ax = b$ et supposons que nous avons une solution approchée $x = x + \Delta x$. Dans ce cas on $Ax = b + \Delta b$.

(1) Montrer que $\Delta x \leq A^{-1} \cdot \Delta b$

(2) Calculer une majoration de la norme de l'erreur relative $\frac{\Delta x}{x}$.

EXERCICE 4.11 Si A matrice carrée avec $A < 1$, alors

(1) $(I + A)^{-1}$ existe

(2) $\|(I + A)^{-1}\| \leq \frac{1}{1 - \|A\|}$

EXERCICE 4.12 Soit A matrice carrée, régulière et $B = A(I + C)$ avec $\|C\| < 1$. Soient aussi x et Δx définis par $Ax = b$ et $B(x + \Delta x) = b$. Supposons que $\text{cond}(A)\frac{\|B - A\|}{\|A\|} < 1$. Montrer que

(1) $\frac{\|\Delta x\|}{\|x\|} \leq \frac{\|C\|}{1 - \|C\|}$

(2) $\frac{\|\Delta x\|}{\|x\|} \leq \frac{\text{cond}(A)}{1 - \text{cond}(A) \cdot \frac{\|B - A\|}{\|A\|}} \cdot \frac{\|B - A\|}{\|A\|}$

EXERCICE 4.13 Soit A matrice carrée regulière et soit B_0 son inverse calculé. Supposons que $\|I - AB_0\| = \rho < 1$.

$\Delta A_0 = I - AB_0$

$B_1 = B_0(I + \Delta A_0)$ et $\Delta A_1 = I - AB_1$

$B_2 = B_1(I + \Delta A_1)$ et $\Delta A_2 = I - AB_2$

.....

(1) Montrer que $B_m = A^{-1}\|I - \Delta A_0^{2^m}\|$

(2) Montrer que $\|B_m - A^{-1}\| \leq \|B_0\| \frac{\rho^{2^m}}{1 - \rho}$

(3) En utilisant Scilab calculer l'inverse de la matrice

$$A = \begin{pmatrix} 1 & 0.42 & 0.54 & 0.66 \\ 0.42 & 1 & 0.32 & 0.44 \\ 0.54 & 0.32 & 1 & 0.22 \\ 0.66 & 0.44 & 0.22 & 1 \end{pmatrix}$$

et améliorer ce calcul en utilisant la procédure décrite ci-dessus.

Peut-on quantifier l'amélioration ainsi obtenue ?

4.5 Analyse active de l'erreur

Nous allons présenter, à l'aide d'un exemple, une méthode de calcul d'erreur d'un algorithme, qui se fait en même temps que le déroulement de l'algorithme.

Rappelons d'abord, la relation (1.4.4) d'un nombre-machine avec le nombre réel qu'il représente :

$$fl(x) = x(1 + \eta), \quad |\eta| \leq eps \tag{4.1}$$

Cette relation peut aussi s'écrire

$$fl(x) = \frac{x}{(1+\eta)}, \quad |\eta| \leq eps \tag{4.2}$$

Supposons que nous voulons faire la somme des nombres x_k; $k = 1, \ldots, n$. Notons la i-ième somme partielle $s_k = x_i + \cdots + x_k$. Le calcul de cette somme par l'ordinateur se fait selon l'itération

$$s_k = s_{k-1} + x_k \tag{4.3}$$

et donne le résultat

$$fl(s_k) = s_k + e_k, \text{ avec } |e_k| \leq eps \tag{4.4}$$

Nous avons aussi d'après (4.1)

$$fl(x_k) = x_k + \eta_k x_k, \text{ avec } |\eta_k| \leq eps \tag{4.5}$$

et d'après (4.2)

$$fl(s_k) = \frac{s_k}{1+\varepsilon_k}, \text{ avec } |\varepsilon_k| \leq eps$$

ce qui compte tenu de (4.4) s'écrit

$$(1+\varepsilon_k) fl(s_k) = fl(s_{k-1}) + fl(x_k) \tag{4.6}$$

En développant et en utilisant de nouveau (4.4) et (4.5), on obtient

$$\begin{aligned} s_k + e_k + \varepsilon_k fl(s_k) &= fl(s_{k-1}) + fl(x_k) \\ &= fl(s_{k-1}) + x_k + \eta_k x_k \\ &= s_{k-1} + e_{k-1} + x_k + \eta_k x_k \end{aligned}$$

En tenant compte de (4.3) on a que

$$e_k = e_{k-1} + \eta_k x_k - \varepsilon_k fl(s_k) \tag{4.7}$$

ce qui donne

$$|e_k| \leq |e_{k-1}| + eps \cdot |x_k| + eps \cdot |fl(s_k)| \tag{4.8}$$

On pose

$$\delta_k = \delta_{k-1} + |x_k| + |fl(s_k)|, \quad \delta_0 = 0 \ (\text{car } e_0 = 0) \tag{4.9}$$

et on a finalement :

$$|e_k| \leq eps \cdot \delta_k \tag{4.10}$$

Cette méthode a été proposée par Wilkinson qui l'a nommée running error analysis et que nous traduisons par analyse active de l'erreur. L'idée fondamentale est que, quand on exécute une opération arithmétique (addition, soustraction, multiplication, division et extraction de la racine caréé) notée $\oplus$, l'erreur peut s'écrire

$$|x \oplus y - fl(x \oplus y)| \leq eps \cdot |fl(x \oplus y)| \tag{4.11}$$

et dans la mesure où on connaît $fl(x \oplus y)$, on peut calculer la borne supérieure de cette erreur.

4.6 Produits vectoriels

Soient deux vecteurs $x, y \in \mathbb{R}^n$ et formons leur produit intérieur $s_n = x\ y$. Notons $s_k = x_1y_1 + \cdots + x_ky_k$ la k-ième somme partielle. En utilisant le modèle (4.1), on obtient

$$\begin{aligned} fl(s_1) &= fl(x_1y_1) = x_1y_1(1+\eta_1) \\ fl(s_2) &= fl(fl(s_1) + x_2y_2) = (fl(s_1) + x_2y_2(1+\eta_2))(1+\eta_3) \\ &= (x_1y_1(1+\eta_1) + x_2y_2(1+\eta_2))(1+\eta_3) \\ &= x_1y_1(1+\eta_1)(1+\eta_3) + x_2y_2(1+\eta_2)(1+\eta_3) \end{aligned}$$

avec $|\eta_k| \leq eps$. Si on suppose que approximativement $1+\eta_k \quad 1 \pm \eta, \quad \forall k$, alors on a ::

$$fl(s_3) = x_1y_1(1 \pm \eta)^3 + x_2y_2(1 \pm \eta)^3 + x_3y_3(1 \pm \eta)^2$$

d'où, en généralisant :

$$fl(s_n) = x_1y_1(1 \pm \eta)^n + x_2y_2(1 \pm \eta)^n + x_3y_3(1 \pm \eta)^{n-1} + \cdots + x_ny_n(1 \pm \eta)^2 \qquad (4.1)$$

Pour simplifier cette expression on utilise le lemme suivant :

LEMME 4.6.1 Si $|\eta_k| \leq eps$ et $\rho_k = \pm 1$ pour $k = 1, \ldots, n$ et si $n \cdot eps < 1$, alors

$$\prod_{k=1}^{n} (1+\eta_k)^{\rho_k} = 1 + \theta_n$$

où

$$|\theta_n| \leq \gamma_n, \quad \text{avec } \gamma_n = \frac{n \cdot eps}{1 - n \cdot eps}$$

En appliquant ce lemme, la relation (4.1) s'écrit :

$$fl(s_n) = x_1y_1(1+\theta_n) + x_2y_2\ 1+\theta_n\ + x_3y_3(1+\theta_{n-1}) + \cdots + x_ny_n(1+\theta_2) \qquad (4.2)$$

Cette relation fournit l'erreur en retard du produit intérieur de deux vecteurs et elle peut être interprétée comme étant la somme exacte des données $x_1, \ldots, x_n$ et de données perturbées $y_1(1+\theta_n), \ldots, y_n(1+\theta_2)$ (ou, alternativement, on peut perturber x_i). Remarquons que chaque perturbation est bornée par γ_k, donc elle est négligable.

Ainsi nous avons pour le produit intérieur la relation

$$fl\ x\ y\ = (x + \Delta x)\ y = x\ (y + \Delta y), \quad \text{avec } \Delta x \leq \gamma_n\ x\ , \ \Delta y \leq \gamma_n\ y \qquad (4.3)$$

qui montre que cette opération est stable si $n \cdot eps < 1$.

Nous pouvons aussi calculer une borne pour l'erreur en avance, en utilisant la dernière relation :

$$x\ y - fl\ x\ y\ \leq \gamma_n \sum_{k=1}^{n} |x_ky_k|$$

Pour le produit extérieur $A = xy^T$, avec $A = (a_{ij})$, on a $fl(a_{ij}) = x_i y_j (1+\eta_{ij})$, où $|\eta_{ij}| \leq eps$. Donc

$$fl(A) = xy^T + \Delta, \quad \text{avec } |\Delta| \leq eps \cdot |xy^T| \qquad (4.4)$$

4.6.1 Exercice

EXERCICE 4.14 Soient $x, y \in \mathbb{R}^n$ avec $n = 2m$, nombre pair. Montrer que l'algorithme suivant pour le calcul de $x \quad y$

— $s_1 \leftarrow x(1:m) \quad y(1:m)$

— $s_2 \leftarrow x(m+1:n) \quad y(m+1:n)$

— $s \leftarrow s_1 + s_2$

réduit la borne supérieure de l'erreur en avance.

Généraliser ce résultat en décomposant en k sous-vecteurs les vecteurs x et y.

Pour quelle valeur de n cette décomposition est réalisable ?

4.7 Multiplication matricielle

Soient deux matrices $A \in \mathbb{R}^{m \times n}, B \in \mathbb{R}^{n \times p}$ et soit $C = A \cdot B$ leur produit. Exprimons $A = \begin{matrix} a_1 \\ \vdots \\ a_m \end{matrix}$ où a_i est un vecteur-ligne. De même on a $B = \begin{matrix} b_1 & \cdots & b_p \end{matrix}$, où b_1 est un vecteur colonne. Nous avons $c_{ij} = a_i b_j$. Donc $fl(c_{ij}) = (a_i + \Delta a_i) b_j$ avec $\Delta a_i \leq \gamma_n \ a_i$. Si on note $C = \begin{matrix} c_1 & \cdots & c_p \end{matrix}$ les colonnes de la matrice C, nous avons

$$fl(c_j) = (A + \Delta A) b_j, \text{avec } \Delta A \leq \gamma_n \ A$$

qui donne l'erreur en retard pour une colonne de la matrice C.

Pour l'erreur en avance, nous avons la borne supérieure suivante :

$$C - fl(C) \leq \gamma_n \ A \quad B \tag{4.1}$$

4.7.1 Exercice

EXERCICE 4.15 Soient $A, B \in \mathbb{R}^{n \times n}$ deux matrices régulières. Montrer que

$$fl(AB) = (A + \Delta A)B$$

et évaluer la borne supérieure de ΔA.

Même chose si on suppose que la matrice B est perturbée d'une matrice ΔB.

4.8 Complexité

En dehors de la précision des calculs d'un algorithme, un autre paramètre important pour le choix d'un algorithme est sa complexité, c'est-à-dire le nombre d'opérations arithmétiques que son exécution nécessite. Cette mesure doit être indépendante de l'ordinateur sur lequel s'exécutera

l'algorithme. On introduit donc, une opération arithmétique abstraite appelée opération flottante standarisée et notée flop.

On convient que les opérations d'addition, soustraction et multiplication repésentent 1 flop. La division et l'extraction de la racine carrée entre 10 et 30 flops. Le calcul des fonctions exponentielles et trigonométriques 50 flops.

Dans plusieurs cas d'évaluation de la complexité d'un algorithme, on cherche à avoir une idée qualitative du nombre d'opérations et non pas le nombre exact de ces opérations. On utilise dans ce cas la notion de l'ordre de la complexité, dont la définition est la suivante :

DÉFINITION 4.8.1 Soit un algorithme dont le nombre d'opérations dépend d'un paramètre n. On dit que la complexité $C(n)$ de l'algorithme est de l'ordre $f(n)$ s'il existe deux constantes a et b telles que

$$C(n) \leq bf(n), \quad \forall n \geq a \tag{4.1}$$

Cette complexité est exprimée en utilisant la O-notation de Landau[1]

$$C(n) = O(f(n)) \tag{4.2}$$

Les algorithmes, en fonction de leur complexité, sont partagées en différentes classes, dont nous donnons ci-après les plus importantes pour l'analyse numérique.

Ordre	Classe	Exemple de $O(f(n))$
$O(1)$	constante	$a \in \mathbb{R}_+$
$O(\log n)$	logarithmique	$a \log n$
$O(n)$	linéaire	$a_1 n + a_0$
$O\ n^2$	quadratique	$a_2 n^2 + a_1 n + a_0$
$O\ n^3$	cubique	$a_3 n^3 + a_2 n^2 + a_1 n + a_0$
$O(n^m)$	polynomiale	$a_m n^m + \cdots + a_0$
$O(c^n)$	exponentielle	$c^{bn} + \ldots$
$O(n!)$	factorielle	$qn!$

4.8.1 Exercices

EXERCICE 4.16 Supposons que nous avons un algorithme pour multiplier deux matrices $n \times n$ en $O(n^\omega)$ opérations avec $2 < \omega < 3$. Montrer que si $A \in \mathbb{R}^{m \times n}, B \in \mathbb{R}^{n \times p}$, alors AB nécessite $O\ n_1^{\omega-2} n_2 n_3$ opérations, où $n_1 = \min\{m, n, p\}$ et n_2, n_3 sont les deux autres dimensions.

N.B. On suppose qu'il existe deux réels k et l tels que $n = km$ et $p = lm$.

EXERCICE 4.17 Soit la matrice triangulaire supérieure par blocs :

$$C = \begin{matrix} I & A & 0 \\ 0 & I & B \\ 0 & 0 & I \end{matrix}$$

Calculer son inverse.

En déduire que nous pouvons calculer la multiplication des matrices A, B en utilisant l'inverse de la matrice C.

[1]La définition de O− notation de Landau est la suivante : Soit $f : \mathbb{R} \to \mathbb{R}$. S'il existe une constante C et $\varepsilon > 0$ tels que

4.9 Multiplication rapide des matrices

Soient deux matrices $A, B \in \mathbb{R}^{n \times n}$ et soit $C = A \cdot B$. L'algorithme de multiplication

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$$

nécessite n^3 multiplications et $n^2(n-1)$ additions. Donc la complexité de la multiplication matricielle des matrices carrées est de l'ordre de $O(n^3)$.

La question posée concerne la possibilité d'avoir une complexité $O(n^\omega)$ avec $\omega < 3$, c'est-à-dire la possibilité d'avoir des algorithmes rapides pour la multiplication. Il y a plusieurs réponses à cette question. Nous présentons les deux premières historiquement.

Winograd en 1967 a utilisé, pour une dimension n paire, la formule

$$x \cdot y = \sum_{k=1}^{n/2} (x_{2k-1} + y_{2k})(x_{2k} + y_{2k-1}) - \sum_{k=1}^{n/2} x_{2k-1}x_{2k} - \sum_{k=1}^{n/2} y_{2k-1}y_{2k} \tag{4.1}$$

Si cette relation est appliquée au produit matriciel, alors les deuxième et troisième termes sont calculés une seule fois pour chaque ligne et chaque colonne. La complexité reste la même mais au lieu de faire n^3 multiplications, on en fera moitié moins.

En 1969 Strassen a présenté une nouvelle technique de calcul du produit matriciel de deux matrices fondée sur la stratégie de diviser et régner. Cet algorithme se decompose en deux phases :

- Phase 1 : (diviser) Le problème est décomposé en deux ou plus sous-problèmes de dimension identique et qui peuvent être résolus indépendamment les uns des autres.
- Phase 2 : (régner) Les solutions des sous-problèmes sont arrangées pour former une solution pour le problème initial.

Nous allons présenter cet algorithme dans le cas simple de deux matrices $A = \begin{matrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{matrix}$ et $B = \begin{matrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{matrix}$. Si $A \cdot B = C = \begin{matrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{matrix}$, on a

$$\begin{aligned} c_{11} &= p_1 + p_4 - p_5 + p_7 \\ c_{12} &= p_3 + p_5 \\ c_{21} &= p_2 + p_4 \\ c_{22} &= p_1 + p_3 - p_2 + p_6 \end{aligned} \tag{4.2}$$

où on a posé

$$\begin{aligned} p_1 &= (a_{11} + a_{22})(b_{11} + b_{22}) \\ p_2 &= (a_{21} + a_{22})b_{11} \\ p_3 &= a_{11}(b_{12} - b_{22}) \\ p_4 &= a_{22}(b_{21} - b_{11}) \\ p_5 &= (a_{11} + a_{12})b_{22} \\ p_6 &= (a_{21} - a_{11})(b_{11} + b_{12}) \\ p_7 &= (a_{12} - a_{22})(b_{21} + b_{22}) \end{aligned} \tag{4.3}$$

$|f(x)| \leq C|x|^p; \forall |x| < \varepsilon$, alors on dit que f est de classe $O(x^p)$ qu'on devait écrire $f(x) \in O(x^p)$ et que, par abus de

Cet algorithme requiert 25 flops (7 multiplications et 18 additions/ soutrctions) tandis l'algorithme classique nécessite 12 flops (8 multiplications et 4 additions). Donc l'algorithme de Strassen devient intéressant quand la format de la matrice $n \times n$ est grand, car les formules (4.2) et (4.3) restent valables si à la place des scalaires on a des matrices. Dans ce cas on partage les matrices initiales en quatre sous-matrices de dimension identique. Chaque sous-matrice ainsi obtenue peut, à son tour, être décomposée en quatre sous-matrices et ainsi de suite, de façon recursive. Si on considère que $n = 2^k$ et que $n_0 \times n_0$ est le format des matrices à l'arrêt de la recursivité, avec $n_0 = 2^r$, on a que l'algorithme de Strassen effectue $M(k,r) = 7^{k-r}8^r$ multiplications et $A(k,r) = 4^r(2^r + 5)7^{k-r} - 6 \cdot 4^k$ additions. La somme $M(k,r) + A(k,r)$ est minimale pour $r = 3$. Dans ce cas on $M(k,r) + A(k,r) = 512 \cdot 7^{k-2} - 6 \cdot 4^k < 4 \cdot 7^k = 2 \cdot 2^{\log_2 7^k} = 4\ 2^{k\ \log_2 7} = 4n^{\log_2 7}$ $4n^{2.807}$.

4.9.1 Exercice

EXERCICE 4.18 On note par $S_n(n_0)$ par le nombre d'opérations de la méthode de Strassen appliquée à une matrice $n \times n$ quand la procédure recursive s'arrête aux matrices $n_0 \times n_0$.

Supposons que n et n_0 sont des puissances de 2.

Pour n grand estimer $S_n(8)/S_n(n)$ et $S_n(1)/S_n(8)$ et expliquer la signification de ces quantités.

4.10 Préconditionnement d'une matrice

Considérons un système linéaire $Ax = b$ Si la matrice A est mal conditionnée, il est conseillé, avant de procéder à la résolution du système, d'effectuer un préconditionnement de la matrice A afin que la matrice résultante après cette opération soit mieux conditionnée. Nous présentons dans ce paragraphe une telle méthode de préconditionnement.

Considérons un système linéaire

$$A_0 x = b,\ A \in \mathbb{R}^{n \times n} \tag{4.1}$$

avec A_0 matrice mal conditionnée. Supposons que nous avons calculer l'inverse B_0 de cette matrice et nous avons que la quantité $A_0 \cdot B_0 - I$ est supérieure à un seuil donné, c'est-à-dire que l'inversion s'est mal deroulée. On construit le système préconditionné

$$B_0 A_0 x = B_0 b \tag{4.2}$$

qui est équivalent au système initial (4.1). On pose $A_1 = B_0 A_0$ et (4.2) devient

$$A_1 x = B_0 b \tag{4.3}$$

Si on suppose que B_0 est régulière, on peut calculer l'inverse de A_1 et soit B_1 cette matrice inverse. Ainsi le produit $B_1 B_0$ est une nouvelle approximation de l'inverse de A_0. Si cette approximation n'est pas bonne, nous pouvons continuer en calculant une nouvelle matrice B_2 qui est l'inverse de la matrice $A_2 = B_1 B_0 A_0$ et ainsi de suite. Nous obtenons ainsi la suite des matrices régulières B_i telles que :

$$B_k B_{k-1} \cdots B_1 B_0 \quad A_0^{-1} \tag{4.4}$$

notation, on écrit $f(x) = O(x^p)$.

Si la matrice inverse obtenue est « assez proche » de A^{-1}, on arrête les itérations.

Nous pouvons aussi utiliser l'expansion polynômiale de von Neumann pour l'inverse d'une matrice. Supposons que la matrice A peut se décomposer en deux matrices selon la formule :

$$A_0 = P - Q \tag{4.5}$$

avec P matrice régulière. Posons

$$G = P^{-1}Q \tag{4.6}$$

et faisons l'hypothèse que le rayon spectral de G est, en module, inférieur à 1 : $|\rho(G)| < 1$. Alors l'expansion polynômiale de von Neumann est donnée par la relation :

$$A_0^{-1} = I + G + G^2 + \cdots + G^{m-1} P^{-1} \tag{4.7}$$

Pour appliquer cette expansion dans notre cas, on constate qu'à chaque étape du calcul nous avons

$$A_i = B_i^{-1} - E_i ; i = 0, 1, \ldots, k \tag{4.8}$$

où B_i est l'inverse approchée de A_i et E_i est la matrice d'erreur de l'approximation. Afin d'exploiter une expansion polynômiale de von Neumann, on doit décompose A_0 en deux matrices. Si on compare avec (4.8) on en conclut que pour la première étape on doit avoir

$$A_0 = B_0^{-1} - E_0 \tag{4.9}$$

Ainsi la formule (4.7) s'applique pour $G = G_0 = B_0E_0$, $P^{-1} = B_0$ et à condition que $|\rho(B_0E_0)| < 1$.

L'expansion de Neumman (4.7) peut aussi se mettre sous la forme

$$A_0^{-1} = I + G^{2^s} \quad I + G^{2^{s-1}} \quad \cdots \quad I + G^2 \quad (I + G)P^{-1} \tag{4.10}$$

qui, dans notre cas, donne

$$A_0^{-1} = I + G_0^{2^s} \quad I + G_0^{2^{s-1}} \quad \cdots \quad I + G_0^2 \quad (I + G_0)B_0 \tag{4.11}$$

à condition que $|\rho(G_0)| < 1$.

Comme

$$B_1 = (B_0A_0)^{-1} = A_0^{-1}B_0^{-1} \tag{4.12}$$

on obtient, à cause de la relation (4.11)

$$B_1 = I + G_0^{2^s} \quad I + G_0^{2^{s-1}} \quad \cdots \quad I + G_0^2 \quad (I + G_0) \quad I + G_0 \tag{4.13}$$

ce qui permet d'amorcer la prochaine étape du calcul de l'expansion.

4.10.1 Exercices

EXERCICE 4.19 Établir un algorithme pour la méthode de préconditionnement exposée ci-dessus.

EXERCICE 4.20 Soit le système linéaire

$$\begin{pmatrix} 5 - \frac{1}{68} & 7 & 6 & 5 \\ 7 & 10 & 8 & 7 \\ 6 & 8 & 10 & 9 \\ 5 & 7 & 9 & 10 \end{pmatrix} x = \begin{pmatrix} 23 \\ 32 \\ 33 \\ 31 \end{pmatrix}$$

avec réponse exacte $x = 1.00063979\ 1\ 1\ 1$.

Vérifier, en utilisant Scilab, la qualité de la résolution directe du système et améliorer cette qualité en utilisant l'algorithme de l'exercice précédent et des fonctions de Scilab.

4.11 Inversion par perturbation des matrices singulières

Considérons un système linéaire

$$A_0 x = b,\ A_0 \in \mathbb{R}^{n \times n}$$

et supposons que la matrice A_0 est mal conditionnée, c'est-à-dire le conditionnement $\kappa(A_0)$ a une très grande valeur, voire elle est singulière, c'est-à-dire $\kappa(A_0)$ est infini. Nous envisageons de faire subir à la matrice A_0 des petites perturbations afin de la rendre bien conditionnée. Il va de soi que ces perturbations ne doivent pas modifier de façon importante la solution du système. Nous obtenons ainsi la matrice perturbée

$$A(\varepsilon) = A_0 + \varepsilon A_1 + \varepsilon^2 A_2 + \cdots, \quad \varepsilon \in \mathbb{R},\ |\varepsilon| < \varepsilon_{max} \tag{4.1}$$

On suppose que $A(\varepsilon)$ est inversible dans le disque $0 < |\varepsilon| < \varepsilon_{max}$. Ainsi la matrice inverse peut être calculée en utilisant un développement en séries de Laurent :

$$A^{-1}(\varepsilon) = \frac{1}{\varepsilon^s} X_0 + \varepsilon X_1 + \varepsilon^2 X_2 + \cdots \tag{4.2}$$

En utilisant le fait que $A(\varepsilon)A^{-1}(\varepsilon) = I$, et en groupant les coefficients de même puissance pour ε, on obtient

$$\begin{aligned} A_0 X_0 &= 0 \\ A_0 X_1 + A_1 X_0 &= 0 \\ &\vdots \\ A_0 X_s + \cdots + A_s X_0 &= I \\ A_0 X_{s+1} + \cdots + A_{s+1} X_0 &= 0 \\ &\vdots \end{aligned} \tag{4.3}$$

Ce système infini d'équations linéaires détermine de façon unique les coefficients de la série de Laurent. Pour la démonstration de ce théorème voir K. E. Avrachenkov, pp. 17-18.

Pour calculer la solution, il faut d'abord déterminer la valeur de s. Pour ce faire on construit de façon itérative les matrices

$$B^{(k)} = \begin{pmatrix} A_0 & 0 & 0 & \cdots & 0 \\ A_1 & A_0 & 0 & \cdots & 0 \\ A_2 & A_1 & A_0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ A_k & A_{k-1} & A_{k-2} & \cdots & A_0 \end{pmatrix} \tag{4.4}$$

Selon le test de Sain et Massey la valeur de s est la valeur minimale de k pour laquelle on a

$$\text{rang}B^{(k)} = \text{rang}B^{(k-1)} + n \tag{4.5}$$

Pour le calcul des matrices X_i on procède par s étapes de réduction. À l'étape on forme le système d'équations

$$\begin{aligned} A_0^{(\ )}X_0^{(\ )} &= R_0^{(\ )} \\ A_0^{(\ )}X_1^{(\ )} + A_1^{(\ )}X_0^{(\ )} &= R_1^{(\ )} \\ &\vdots \quad \vdots \quad \vdots \\ A_0^{(\ )}X_{s-}^{(\ )} + \cdots + A_{s-}^{(\ )}X_0^{(\ )} &= R_{s-}^{(\ )} \end{aligned} \tag{4.6}$$

On remarque que si = 0, nous avons le système initial (4.3) et par conséquent on a

$$\begin{aligned} R_i^{(0)} &= 0, \text{ pour } i = 0,\ldots,s-1 \\ R_s^{(0)} &= I \\ A_i^{(0)} &= A_i, \text{ pour } i = 0,\ldots,s \end{aligned} \tag{4.7}$$

Pour les matrices $A_i^{(\ )}$ et $R_i^{(\ )}$, et = 1,...,s on a les formules suivantes :

$$\begin{aligned} A_i^{(\ )} &= M^{(\ )}U_i^{(\ )}Q^{(\ )},\ i = 0,\ldots,s-1 \\ R_i^{(\ )} &= M^{(\ )} - \sum_{j=0}^{i} U_{i-j}^{(\ )}A_0^{(-1)+}R_j^{(-1)} + R_{i+1}^{(-1)},\ i = 0,\ldots,s-1 \end{aligned} \quad , \quad = 1,\ldots,s \tag{4.8}$$

avec

$$\begin{aligned} &Q^{(\ )} \in \mathbb{R}^{n\times m}, && \text{matrice dont les colonnes forment une base} \\ &&& \text{pour le noyau droit de } A_0^{(-1)} \\ &&& \text{et où } m = n - \text{rang } A_0^{(-1)} \\ &M^{(\ )} \in \mathbb{R}^{m\times n}, && \text{matrice dont les lignes forment une base} \\ &&& \text{pour le noyau gauche de } A_0^{(-1)} \\ &U_0^{(\ )} && = A_1^{(-1)},\quad U_i^{(\ )} = A_{i+1}^{(-1)} - \sum_{j=1}^{i} A_j^{(-1)}A_0^{(-1)}U_{i-j}^{(\ )}, \\ &&& i = 0,\ldots,s-1 \\ &A^{+} && \text{la pseudoinverse de la matrice } A \text{ (cf. inf ra)} \end{aligned} \tag{4.9}$$

Après s étapes on obtient le système final d'équations réduites :

$$A_0^{(s)}X_0^{(s)} = R_0^{(s)} \tag{4.10}$$

On peut démontrer (voir K. E. Avrachenkov, pp. 21-23) que ce système et le système initial donné par (4.3) sont équivalents, donc la matrice $A_0^{(s)}$ est régulière et, par conséquent

$$X_0^{(s)} = A_0^{(s)\ -1}R_0^{(s)} \tag{4.11}$$

Pour obtenir donc la solution $X_0 = X_0^{(0)}$, il faut procéder à une recursion en arrière selon le schéma :

$$X_0^{(-1)} = A_0^{(-1)+}R_0^{(-1)} + Q^{(\ )}X_0^{(\ )}, \quad = s,\ldots,1 \tag{4.12}$$

En posant

$$R_i^{(0)} = \begin{cases} -\sum_{j=1}^{k} A_{i+j} X_{k-j}, & \text{si } i = 0,\ldots,s-1 \\ I - \sum_{j=1}^{k} A_{i+j} X_{k-j}, & \text{si } i = s \end{cases} ; \; k = 1,2,\ldots \tag{4.13}$$

nous pouvons calculer les coefficients de la série de Laurent :

$$X_k = \sum_{i=0}^{s} G_{0i}^{(s)} \cdot R_i^{(0)}; \; k = 1,2,\ldots \tag{4.14}$$

avec $G_{0i}^{(s)} \in \mathbb{R}^{n \times n}$ le bloc $(0,j)$ de la matrice $B^{(k)+}$.

Remarquons que la formule (4.14) est une généralisation de l'expansion de von Neumann. En effet dans cette dernière expansion on suppose que la matrice A_0 est régulière. Donc $s = 0$ et $G^{(0)} = A_0^{-1}$ et la relation (4.14) donne :

$$X_0 = -A_0^{-1}, \quad X_k = -A_0^{-1} \sum_{j=1}^{k} A_j X_{k-j}; \; k = 1,2,\ldots \tag{4.15}$$

qui est l'expansion de von Neumann.

4.11.1 Exercices

EXERCICE 4.21 Établir un algorithme qui permet d'inverser une matrice singulière, en lui faisant subir des petites perturbations.

EXERCICE 4.22 Utiliser l'algorithme de l'exercice précédent et des fonctions de Scilab pour calculer l'inverse de la matrice

$$A_0 = \begin{pmatrix} 1 & 2 & 1 \\ -1 & 1 & 0 \\ 0 & 3 & 1 \end{pmatrix}$$

avec matrice de perturbation

$$A_1 = \begin{pmatrix} 1 & -1 & 0 \\ 0 & 1 & -1 \\ -1 & 0 & 1 \end{pmatrix}$$

4.12 Références

Les livres et articles qui sont pris en compte pour la redaction de ces notes sur l'Algèbre Linéaire et les Perturbations sont les suivants :

J. H. WILKINSON : Rounding errors in algebraic processes, Dover, 1994
J. H. WILKINSON : The algebraic eigenvalue problem, Clarendon Pr., 1965

F. R. GANTMACHER : Théorie des matrice, tome 1, Théorie générale, Dunod, 1966
V. N. FADDEEVA : Computational methods of linear algebra, Dover, 1959
N. J. HIGHAM : Accurancy and stability of numerical algorithms, Siam, 1996
J. STOER, R. BULIRSCH : Introduction to numerical analysis, Second ed., Springer-Verlag, 1992
K. E. AVRACHENKOV : Analytic perturbation theory and its applications, Un. South Australia, 1999

4.A APPENDICE.- BREF RAPPEL DE L'ALGÈBRE LINÉAIRE

Considérons un quelconque problème de l'algèbre linéaire dont la solution est un vecteur $x = [x_1 \cdots x_n] \in U \subset \mathbb{R}^n$. Le calcul de cette solution peut se faire en analyse numérique par un algorithme itératif φ selon le schéma suivant :

$$\begin{array}{l} x(0) \text{ valeur de départ choisie au hasard.} \\ x(k+1) = \varphi(x(k)) \; ; \; k = 0, 1, \ldots \end{array} \tag{4.1}$$

L'algorithme φ est donc une fonction de U dans lui-même et k est le numéro d'itération.

L'application du schéma (4.1) permet d'obtenir une approximation de la vraie solution. Nous dirons que l'algorithme itératif converge vers la solution si $\|x(k+1) - x(k)\| < \text{seuil}$, où $\|\bullet\|$ norme établie selon la métrique usuelle.

Le schéma (4.1) peut être réalisé selon deux méthodes :

— parallèle : Toutes les coordonnées du vecteur $x(k+1)$ sont calculées simultanément:

$$\forall i = 1, \ldots, n \; : \; x_i(k+1) = \varphi_i(x_1(k), \ldots x_n(k))$$

— séquentielle: Pour le calcul d'une coordonnée du vecteur $x(k+1)$ on tient compte des valeurs des coordonnées déjà calculées :

$$\forall i = 1, \ldots, n \; : \; x_i(k+1) = \varphi_i(x_1(k+1), \ldots, x_{i-1}(k+1), x_i(k), \ldots, x_n(k))$$

Notons que l'ordre des indices pour le calcul peut être quelconque et même aléatoire.

En algèbre linéaire il y a aussi des algorithme à résolution directe. On peut considérer que dans ce cas il s'agit des algorithmes séquentiels et dont le nombre d'itérations est égal à l'ordre du problème.

Essayons maintenant de formaliser la notion de la convergence d'un algorithme. À l'algorithme φ on peut associer l'ensemble de ses points fixes $\varphi^{\infty}(U)$ défini par

$$\varphi^{\infty}(U) = \{u \in U \quad | \quad \varphi(u) = u\}$$

Un point fixe u° est dit attractif s'il admet un voisinage $V(u^{\circ}) \subset U$ tel que

$$\forall u \in V(u^\circ), \quad \lim_{k\to\infty} \varphi^k(u) = u^\circ.$$

De la même façon, on dira qu'un sous-ensemble U° est attractif si

$$\forall u^\circ \in U^\circ \quad \exists V(u^\circ) \quad : \forall u \in V(u^\circ), \quad \lim_{k\to\infty} \varphi^k(u) \in U^\circ.$$

Tout l'art de l'analyse numérique pour la résolution d'un problème dans un espace U et dont l'ensemble des solutions forme un ensemble qui sera noté U^*, est de trouver un algorithme φ tel que $\varphi^\infty(U)$ soit « proche » et attractif. Notons que la notion de proximité peut être comprise comme une inclusion dans l'un sens ou dans l'autre - ce qui n'est bien sûr pas la même chose sur le plan du résultat. On pourrait même demander à avoir l'égalité entre $\varphi^\infty(U)$ et U^*, mais ce serait très exigeant et peu efficace. On définit aussi le bassin d'attraction associé à un point fixe u°. C'est l'ensemble de toutes les solutions initiales $u(0)$ qui conduisent à une suite $\varphi^k(u(0))$ convergente vers u°. Un tel sous-ensemble est généralement très difficile à décrire, sauf lorsque que tout U est bassin d'attraction d'un unique point fixe.

On dira qu'un algorithme converge si la suite de valeurs qu'il engendre tend vers une limite dans l'espace considéré. Bien sûr pour que cette convergence soit « intéressante » la limite doit être la solution du problème. En d'autres termes, l'algorithme converge si ses points fixes attractifs sont candidats pour être solution du problème posé et leurs bassins d'attraction recouvrent la totalité de la région de U où nous avons défini notre problème(2).

La vitesse de convergence d'un algorithme mesure le taux de la décroissance vers zéro de la distance entre les valeurs engendrées et leur limite. Par exemple soit, dans $\mathbb{R}^n$, la limite u^* vers laquelle converge la suite $(u(k))_{k\in\mathbb{N}}$ (engendrée par un algorithme φ donné) et $\bullet$ la norme euclidienne :

DÉFINITION 4.A.1 (Vitesse de Convergence).

— Si $\lim_{k\to\infty} \sup \frac{u(k+1)-u^*}{u(k)-u^*} \leq \alpha < 1$, alors on dit que la convergence est linéaire et α est le taux de convergence associé. (On remarque qu'il s'agit ici de la convergence des suites géométriques.)

— Si $\lim_{k\to\infty} \sup \frac{u(k+1)-u^*}{u(k)-u^*} = 0$, alors on dit que la convergence est super-linéaire.

— Si $\exists \gamma > 1$ tel que $\lim_{k\to\infty} \sup \frac{u(k+1)-u^*}{u(k)-u^{*\gamma}} \leq M < +\infty$, alors la convergence est super-linéaire d'ordre γ et, en particulier, si $\gamma = 2$, on parle de vitesse de convergence quadratique.

Ces définitions ont intrinsèquement un caractère asymptotique et on ne peut, en général, démontrer que la convergence est super-linéaire ou quadratique, que dans un voisinage de la limite recherchée u^* (loin de u^*, il se peut très bien que l'algorithme concerné converge lentement – ou pas du tout –, bien que la vitesse de convergence asymptotique soit en théorie quadratique).

[2] Les points fixes non attractifs ne présentent aucun intérêt car leur rencontre, au cours d'itérations, ne peut être que

D'autre part, deux algorithmes ayant la même vitesse de convergence asymptotique peuvent très bien nécessiter des temps de calculs très différents si le nombre d'opérations exécutées à chaque itération est très différent.

Cette définition est néanmoins très utile pour évaluer la qualité de la limite obtenue par un algorithme. Par exemple, supposons, pour fixer les idées, que la vitesse de convergence asymptotique d'un algorithme soit quadratique dans un voisinage de u^*, avec $M = 100$. Il faut donc avoir pour une itération k l'inégalité

$$\frac{u(k+1+i)-u^*}{u(k+i)-u^*} \leq 100, \qquad \forall i = 0, 1, \ldots$$

Si, pour l'itération k, on a $u(k)-u^* \leq 10^{-3}$ c'est-à-dire on est dans un voisinage de la solution, alors on vérifie aisément que $u(k+1)-u^* \leq 10^{-4}$, $u(k+2)-u^* \leq 10^{-6}$, $u(k+3)-u^* \leq 10^{-10}$, etc. L'erreur d'approximation passe donc, en trois itérations, de 10^{-3} à 10^{-10}.

Nous présentons dans les chapitres suivants différentes méthodes de résolution des problèmes d'algèbre linéaire numérique. Lors de la présentation de ces méthodes la terminologie et les résultats fondamentaux de l'algèbre linéaire seront considérés comme connus. Nous nous bornerons donc de lister les principaux résultats qui seront utilisés par la suite.

Définitions et fondements théoriques des applications linéaires

Dans tout le cours de l'analyse numérique, les espaces vectoriels seront considérés sur le corps des réels $\mathbb{R}$.

(1) On appelle application linéaire (ou opérateur linéaire) de l'espace vectoriel U dans l'espace vectoriel V une application f de U dans V vérifiant la propriété suivante :

$$\forall u, u \in E, \forall \alpha, \beta \in \mathbb{R} : f(\alpha u + \beta u) = \alpha f(u) + \beta f(u)$$

(2) Une application linéaire de U dans V est appelée forme linéaire.

(3) Une application linéaire de U dans lui-même est appelée endomorphisme.

(4) Une application linéaire bijective de U dans V est appelée isomorphisme.

(5) Une application linéaire bijective de E dans lui-même est appelée automorphisme.

(6) Soit f une application linéaire bijective (ou isomorphisme) de U sur V. Alors f est inversible et l'application inverse f^{-1} est un isomorphisme de V sur U.

(7) Soit f une application linéaire de U dans V.

- Si f est injective, l'image d'une famille libre de U est une famille libre de V.
- Si f est surjective, l'image d'une famille génératrice de E est une famille génératrice de V.
- Si f est bijective, l'image d'une base de U est une base de F.

(8) Soit f un opérateur linéaire de U dans V. On définit :

- L'image de f : $R(f) = \{y \in V / \exists x \in U \text{ avec } y = f(x)\}$, c'est-à-dire $R(f)$ est composé de tous les vecteurs de V qui peuvent se représenter sous la forme $f(x)$ avec $x \in U$. On a $R(f) = f(E)$ et $R(f)$ est un sous-espace vectoriel de V.

---

fortuite. Par conséquent nous ne pouvons pas fonder sur ces points des shémas itératifs qui convergent.

— Le noyau de f : $N(f) = \{x \in U / f(x) = 0\}$, c'est-à-dire $N(f)$ est composé de tous les vecteurs $x \in U$ qui s'annulent par f. $N(f)$ est un sous-espace vectoriel de U.

(9) Pour l'application linéaire f de U dans V nous avons $\dim(N(f)) + \dim(R(f)) = \dim(U)$.

(10) Si f est injective, alors $N(f) = \{0\}$.

Définitions et fondements théoriques des matrices

Considérons deux espaces vectoriels $U \subseteq \mathbb{R}^m$ avec base $(u_i)_{i=1,\dots,m}$ et $V \subseteq \mathbb{R}^n$ avec base $(v_j)_{j=1,\dots,n}$. Soit $f : U \to V$ une application linéaire. L'image de f est engendrée par les images des vecteurs de la base de U, $f(u_i); i = 1, \dots, m$. En effet, soit $x \in U$. Alors l'image de x par f s'écrit $f(x) = \sum_{i=1}^{m} f(u_i)x$. Avec les composants des vecteurs $f(u_i)$ on peut former le tableau

$$F = \begin{matrix} f_{11} & \cdots & f_{1,m} \\ \vdots & \ddots & \vdots \\ f_{n1} & \cdots & f_{nm} \end{matrix}, \text{ où } f_{ji} \text{ est la } j\text{-ième composante du vecteur } f(u_i)$$

Le tableau F s'appelle la matrice de l'application linéaire f et elle décrit complètement f, à savoir si $x \in U$, alors $f(x) = Fx$.

(1) Une matrice est dite régulière si son inverse existe. Sinon elle est singulière.

(2) Le rang d'un matrice est la dimension de la plus grande sous–matrice carrée qui est régulière. Si $f : U \to V$ application linéaire, alors $\text{rang}(F) = \dim(V)$, où F est la matrice de f.

(3) Une matrice de permutation P est une matrice dont chaque ligne et chaque colonne a un élément égal à 1 et tous les autres sont nuls. Elle s'appelle matrice de permutation car si on la multiplie avec une autre matrice provoque la permutation de deux lignes ou de deux colonnes de cette dernière. De plus nous avons $\det P = (-1)^{k_P}$, où k_P est le nombre de lignes de P qui sont différentes des lignes correspondantes de la matrice identité I.

(4) Une matrice carrée A est orthogonale si $A^{\top} A = I$, d'où $A^{-1} = A^{\top}$.

(5) Une matrice carrée A est diagonalement dominante si $|a_{ii}| \geq \sum_{\substack{k=1 \\ k \neq i}}^{n} |a_{ik}|; i = 1, \dots, n$ et il existe au moins un indice i_0 tel que $|a_{i_0 i_0}| > \sum_{\substack{k=1 \\ k \neq i_0}}^{n} |a_{i_0 k}|$.

La définition est valable si à la place des lignes on utilise les colonnes à la formule précédente.

La matrice est strictement diagonalement dominante si dans la formule précédente nous avons des inégalités strictes. Une telle matrice est régulière.

(6) Une matrice A est symétrique si $A = A^{\top}$.

Une matrice symétrique A est définie positive si

$$x^{\top} A x > 0; \ \forall x \in \mathbb{R}^n$$

Elle est semi-définie positive si

$$x^{\top} A x \geq 0; \ \forall x \in \mathbb{R}^n$$

(7) Factorisation LU.- Toute matrice régulière A peut se factoriser de manière unique selon le produit

$$A = LU$$

avec

$$L = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ l_{21} & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ l_{n1} & l_{n2} & \cdots & 1 \end{pmatrix} ; U = \begin{pmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ 0 & u_{22} & \cdots & u_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & u_{nn} \end{pmatrix}$$

(8) Pour toute matrice A régulière, il existe une matrice de permutation P, telle que

$$PA = LR$$

Dans ce cas nous avons aussi

$$\det A = (-1)^{k_P} \det R = (-1)^{k_P} r_{11} r_{22} \cdots r_{nn}$$

(9) Une matrice symétrique, strictement diagonalement dominante A peut se factoriser de manière unique selon le produit

$$A = LDL$$

avec L matrice triangulaire inférieure et D une matrice diagonale.

(10) Une matrice symétrique, définie positive A peut se factoriser de manière unique selon le produit

$$A = LL$$

avec L matrice triangulaire inférieure avec $l_{ii} > 0$ pour tout i.

(11) Soit une matrice $A \in \mathbb{R}^{n \times n}$ regulière. Un réel $\lambda \in \mathbb{R}$ est une valeur propre de A s'il existe un vecteur $x \in \mathbb{R}^n$, avec $x = 0$, tel que $Ax = \lambda x$. Le vecteur x est appelé vecteur propre de A associé à la valeur propre λ.

(12) Rayon spectral d'une matrice A : $\rho(A) = \max\{|\lambda| / \lambda \in \mathbb{R}, \lambda \text{ valeur propre de } A\}$.

(13) Théorème de Gershgorin.- Toutes les valeurs propres de la matrice A se trouvent dans l'union des cercles de Gershgorin

$$C_i = \{ z \in \mathbb{R} \quad | z - a_{ii} | \leq \sum_{\substack{k=1 \\ k \neq i}}^{n} | a_{ik} | \} \quad ; i = 1, \ldots, n$$

(14) Quotient de Rayleigh.- Si A est une matrice carrée, le quotient de Rayleigh est

$$R[x] = \frac{x^T A x}{x^T x} ; x \in \mathbb{R}^n , x = 0$$

Si A est une matrice carrée symétrique, alors

$$|R[x]| \leq |\lambda_{max}| \quad \forall x \in \mathbb{R}^n$$

où λ_{max} est la plus grande, en module, valeur propre de A .

(15) À tout vecteur $x = [x_1, x_2, \cdots, x_n] \in \mathbb{R}^n$ on associe la matrice–colonne [x] de format (n, 1) et dont la i-ième ligne est la i-ième coordonnée du vecteur x dans la base canonique de $\mathbb{R}^n$. Si on note par M (n, 1) l'ensemble de matrices-colonne de format (n × 1), nous pouvons avoir une application linéaire entre $\mathbb{R}^n$ et M (n, 1) qui est, en plus, bijective. $\mathbb{R}^n$ et M (n, 1) sont donc isomorphes.

(16) Soit A une matrice de format (m, n), le noyau de A est l'ensemble $N(A) = \{x \in \mathbb{R}^n / Ax = 0\}$. N (A) est un sous–espace vectoriel de $\mathbb{R}^n$.

(17) Soit A une matrice de format (m, n), l'image de A est l'ensemble $R(A) = \{y \in \mathbb{R}^m / x \in \mathbb{R}^n$ et $Ax = y\}$. R(A) est un sous–espace vectoriel de $\mathbb{R}^m$ engendrée par les colonnes de A .

(18) On appelle rang d'une matrice A , la dimension de l'image de A : rang(A) = dim R(A).

(19) Soit A une matrice carree d'ordre n, alors rang(A) = n $\Leftrightarrow$ A est inversible.

(20) Théorème noyau–image ou théorème du rang.- Soit A une matrice de format (m, n). Alors : n = dim R(A) + dim N (A) = rang(A) + dim N (A).

(21) Une matrice de rang–colonne plein est une matrice A de format (m, n) dont les colonnes forment une famille libre de $\mathbb{R}^m$. Donc comme $n \leq m$ et que les colonnes de A constituent une base de R(A), la dimension de R(A) est égale au nombre de colonnes de A . Par conséquent dim N (A) = 0, c'est-à-dire que N (A) = {0}.

(22) Soit A une matrice de format (m, n). L'application f_A de $\mathbb{R}^n$ dans $\mathbb{R}^m$ définie par :

$$f_A : x \in \mathbb{R}^n \rightarrow y \in \mathbb{R}^m \text{ telle que } [y] = A\,[x]$$

est une application linéaire de $\mathbb{R}^n$ dans $\mathbb{R}^m$ appelée application linéaire canonique associée à la matrice A .

(23) Soient A une matrice de format m, n et f_A l'application linéaire associée à la matrice A , alors

- $N(f_A) = N(A)$;
- $R(f_A) = R(A)$;
- le rang de l'application linéaire f_A est égal au rang de la matrice A .

5

MÉTHODES DE RÉSOLUTION DE SYSTÈMES LINÉAIRES

5.1 Introduction

L'étude des méthodes de résolution des systèmes linéaires est une étape obligatoire dans un cours d'analyse numérique. En effet, presque tous les calculs passent par la résolution d'un système. On en rencontre dans la discrétisation de problèmes aux limites, dans les problèmes d'approximation par exemple.

La résolution de systèmes n'est plus une opération destinée à être menée à la main. Du fait de l'apparition des ordinateurs, et du développement de méthodes qui leur sont adaptées, on peut envisager de résoudre de grands systèmes. Si la matrice est pleine, c'est à dire contient peu de zéros, on pourra "seulement" résoudre des systèmes à quelques centaines de milliers d'inconnues. Si la matrice est creuse, c'est à dire contient beaucoup de zéros, la résolution de systèmes à plusieurs millions d'inconnues est possible.

La résolution d'un système ne doit pas être comprise comme l'obtention de la solution exacte en un nombre fini d'étapes. Cette vision ne concerne que les méthodes dites "directes". On

est souvent amené à accepter un compromis : obtenir une solution approchée en un nombre fini d'étapes, sachant que la solution exacte serait obtenue en un nombre infini d'étapes, chose impossible à mettre en oeuvre. Ce compromis est le principe même des méthodes itératives. C'est devenu la solution la plus répandue pour résoudre les grands systèmes.

En plus de la taille de la matrice, ses propriétés sont aussi un élément déterminant : certaines assurent la convergence a priori de méthodes itératives. Dans le cas où elles ne sont pas vérifiées par une matrice, ou bien quand on ne pas peut démontrer qu'elles le sont, la méthode itérative est choisie aux risques et périls de l'utilisateur. C'est un risque à éviter, surtout dans des applications industrielles, parfois sensibles. Des navettes spatiales ont explosé pour moins que ...

La résolution des systèmes linéaires est donc à la fois une question de théorie et une question de pratique : il faut choisir le bon algorithme, s'assurer que les conditions sont réunies pour l'utiliser, qu'il sera numériquement stable et assez rapide pour les besoins de la cause.

Il faut repenser la résolution de systèmes dans le cadre d'une mise en oeuvre informatique : nous allons voir sur deux exemples simples que les méthodes utilisables à la main ne sont pas, en général, adaptées à la résolution de systèmes en machine, notamment pour des questions de temps de calcul. Bien sur, de nombreux logiciels proposent des bibliothèques de résolution de systèmes linéaires, qui s'appuient sur des bibliothèques publiques d'algèbre linéaire de base (BLAS). Ces dernières ont été précieuses dans les progrès du calcul scientifique matriciel ; elles permettent d'effectuer de faoptimale les opérations algébriques de base. Les principaux logiciels sont soit du domaine public (LAPACK= Linear Algebra PACKage par exemple), soit des logiciels commerciaux (NAG = Numerical Algorithms Group, IMSL = International Mathematical and Statistical Library, MATLAB,...). Mais en tout état de cause, la possession d'une voiture de course ne permet pas d'avancer quand on ne sait pas la conduire...

5.1.1 Exemple 1 : Économie : analyse d'entrées-sorties

On veut déterminer l'équilibre entre la demande et l'offre de certains biens. Dans le modèle de production considéré, $m \geq n$ usines produisent n produits différents. Elles doivent faire face à une demande interne (l'entrée) nécessaire au fonctionnement propre des usines, ainsi qu'à une demande externe (la sortie) provenant des consommateurs.

La principale hypothèse du modèle de Leontieff (1930) [1] est que le modèle de production est linéaire, c'est à dire que la sortie est proportionnelle à l'entrée utilisée. Sous cette hypothèse, l'activité des usines est entièrement décrite par deux matrices : la matrice d'entrée $C = (c_{ij}) \in R^{n \times m}$ et la matrice de sortie $P = (p_{ij}) \in R^{n \times m}$. Le coefficient c_{ij} (resp. p_{ij}) représente la quantité du $i^{ème}$ bien absorbé (resp. produit) pour la $j^{ème}$ usine sur une période fixée. La matrice $A = P - C$ est appelée matrice d'entrée-sortie : un a_{ij} positif (resp. négatif) désigne la quantité du $i^{ème}$ bien produit (resp. absorbé) par la $j^{ème}$ usine. Enfin, on peut raisonnablement supposer que le système de production satisfait à la demande du marché, qu'on peut représenter par un vecteur $b = (b_i) \in R^n$ (vecteur de la demande finale). La composante b_i représente la quantité du $i^{ème}$ bien absorbé sur le marché. L'équilibre est atteint lorsque le vecteur $x = (x_i) \in R^m$

[1] Wassily Leontieff a reen 1973 le prix Nobel d'économie pour ses travaux

représentant la production totale est égal à la demande totale, c'est à dire

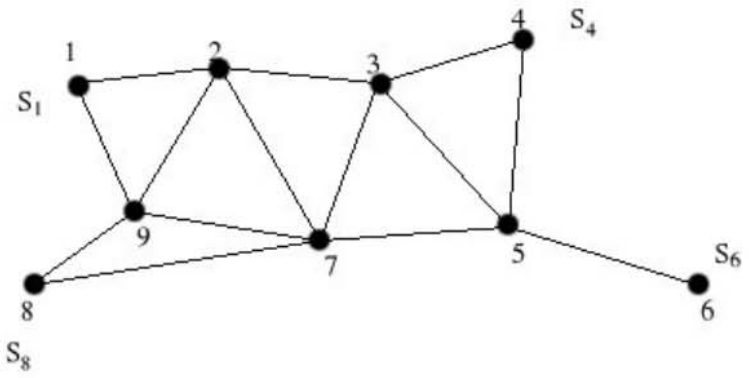
$$A\,x = b, \text{ où } A = P - C$$

5.1.2 Exemple 2 : Résolution d'un problème de réseaux

Un réseau est un ensemble de nœuds P_i et d'arêtes $E_{i,j}$ reliant certains de ces nœuds :

— lignes électriques,

— canalisations d'eaux, égouts,...



Dans chaque arête circule un fluide; à chaque nœud est associé un potentiel. L'intensité (ou le débit) du fluide est proportionnelle à la différence de potentiel entre les deux extrémités de l'arête où il circule ; c'est la loi d'Ohm pour les circuits électriques :

$$q_{i,j} = k_{i,j}\,(u_i - u_j)$$

Une loi physique de conservation (de Kirchoff dans le cas électrique) impose un équilibre : la somme algébrique des intensités en chaque nœud est égale à la valeur de la source (ou du puits) qu'il figure.

Au nœud P_i, on a dans le cas du circuit électrique :

$$S_i = \sum_j q_{i,j} = \sum_j k_{i,j}\,(u_i - u_j)$$

Cette somme peut être étendue aux nœuds adjacents de P_i, les équations d'équilibre s'écrivent :

$$\begin{aligned} S_1 &= k_{1,2}(u_1 - u_2) + k_{1,9}(u_1 - u_9) \\ 0 &= k_{2,1}(u_2 - u_1) + k_{2,9}(u_2 - u_9) + k_{2,7}(u_2 - u_7) + k_{2,3}(u_2 - u_3) \\ &\cdots = \cdots \\ 0 &= k_{9,1}(u_9 - u_1) + k_{9,2}(u_9 - u_2) + k_{9,7}(u_9 - u_7) + k_{9,8}(u_9 - u_8) \end{aligned}$$

de sorte que l'équilibre du système est connu en résolvant le système linéaire

$$A\,u = S$$

avec une matrice A dont les coefficients non nuls sont représentés ci-dessous par une étoile :

$$A = \begin{pmatrix} * & * & & & & & & * \\ * & * & * & & & * & & * \\ & * & * & * & & * & & \\ & & * & * & * & & & \\ & & & * & * & * & * & \\ & & & & * & * & & \\ & * & * & & * & & * & * \\ & & & & & & * & * \\ * & * & & & & & * & * \end{pmatrix}$$

Le second membre est défini par $S^T = (S_1, 0, 0, S_4, 0, S_6, 0, S_8, 0)$.

5.1.3 Comment résoudre ces systèmes

La première idée qui vient est de résoudre ce système en utilisant les formules de Cramer, dites aussi "méthode des déterminants". Ces formules fournissent une solution exacte du système linéaire $A\,x = b$ dans R^n avec $x = (x_1, x_2, \ldots x_n)^T$ donnée par :

$$x_i = \frac{\det A^{(i)}}{\det A}$$

où $A^{(i)}$ désigne la matrice obtenue en substituant dans A la colonne i par le second membre du système.

Il y a donc N + 1 déterminants à calculer, donnés dans le cas général pour une matrice A quelconque par :

$$\det(A) = \sum_{\sigma \in P_n} (\sigma) a_{1,\sigma(1)} \times a_{2,\sigma(2)} \cdots \times a_{n,\sigma(n)}$$

où P_n désigne l'ensemble des permutations de $\{1, \ldots n\}$ qui compte n! éléments et (σ) est + 1 ou − 1, la signature de la permutation. Le coût du calcul d'un tel déterminant est donc tel que :

$$c(n) \sim n \times n! \text{ opérations élémentaires}$$

et le coût global de la méthode est donc

$$C(n) \sim n \times (n-1)! \text{ opérations}$$

Le coût de la résolution d'un système 100 × 100 peut alors être évalué par la formule de Stirling ($n! \sim n^{n+1/2} e^{-n} \sqrt{2\pi}$). Cela conduit à l'évaluation :

$$C(n) \sim 9.4 \times 10^{161}$$

opérations élémentaires. Sur un ordinateur qui réalise 10^9 opérations flottantes par seconde (1 gigaflop), on devra attendre la solution pendant environ 3×10^{145} années !

5.2 Les systèmes faciles à résoudre

Afin de construire des méthodes qui simplifient la résolution d'un système sans passer par des calculs inappropriés pour une machine, voyons deux cas de systèmes dont la résolution est simple à programmer.

5.2.1 Systèmes diagonaux

Si A est une matrice diagonale, c'est à dire si

$$A = (a_{i,j})_{1 \le i,j \le n} \text{ avec } a_{i,j} = 0 \text{ si } i = j$$

le système $Ax = b$ est immédiatement résolu du fait que

$$x_i = \frac{1}{a_{ii}} b_i$$

L'algorithme correspondant est donné par :

Algorithme : Matrice diagonale

```
        {On suppose que A_kk = 0}
    Pour i ← 1 à n faire
      x_i ← b_i / a_ii
    FinPour
```

Le coût est $c(n) = n$ opérations élémentaires

5.2.2 Systèmes triangulaires

La matrice A d'un système triangulaire supérieur est telle que

$$A = (a_{i,j})_{1 \le i,j \le n} \text{ avec } a_{i,j} = 0 \text{ si } i > j$$

Comme A est inversible,

$$a_{i,i} = 0 \text{ si } 1 \le i \le n$$

Le système s'écrit

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n &= b_1 \\ a_{22}x_2 + \ldots + a_{2n}x_n &= b_2 \\ &\vdots \\ a_{nn}x_n &= b_n \end{aligned}$$

On le résout en remontant :

$$\begin{aligned} x_n &= \frac{b_n}{a_{n,n}} \\ x_{n-1} &= (b_{n-1} - a_{n-1,n} x_n) / a_{n-1,n-1} \\ &\vdots \\ x_1 &= (b_1 - a_{12} x_2 - \ldots - a_{1,n} x_n) / a_{1,1} \end{aligned}$$

On obtient alors la solution par un algorithme de substitution rétrograde, dit aussi simplement algorithme de remontée :

Algorithme : Algorithme de remontée

$x_n \leftarrow b_n / a_{n,n}$
Pour i ← n-1 à 1 par pas de -1 faire
  $x_i \leftarrow \left(b_i - \sum_{j=i+1}^{n} a_{i,j} x_j \right) / a_{i,i}$
FinPour

Le coût du calcul d'un $x_k = \left(b_k - a_{k,k+1} x_{k+1} - \ldots - a_{k,n} x_n \right) / a_{k,k}$ se décompose en :

- $(n - k)$ additions,
- $(n - k)$ multiplications,
- 1 division

Le coût total de remontée est donc de

$$\sum_{k=1}^{n} (n - k) = n^2 - \frac{n(n-1)}{2} \sim \frac{n^2}{2} \text{ additions + multiplications}$$

soit n^2 opérations élémentaires.

EXERCICE 5.1 Établir l'algorithme de résolution d'un système triangulaire inférieur

5.3 Méthodes directes

L'idée des méthodes directes est de remplacer la résolution d'un système du type $Ax = b$ dans lequel la matrice A est pleine, par un système ou plusieurs systèmes plus facile(s) à résoudre car creux (i.e. de matrice triangulaire ou diagonale). Le système diagonal serait idéal puisque c'est à la fois le plus rapide et le moins coûteux à résoudre. Mais diagonaliser A, c'est rechercher ses éléments propres et déterminer P et D vérifiant $A = PDP^{-1}$. L'idée est séduisante mais malheureusement inapplicable car la recherche d'éléments propres est beaucoup plus difficile numériquement que la résolution d'un système. L'alternative est donc de remplacer A par le produit de deux matrices triangulaires, notées en général L et U, respectivement triangulaire

inférieure et triangulaire supérieure. En effet on résout alors successivement deux systèmes triangulaires :

$$Ly = b \text{ puis } Ux = y$$

dont la solution x vérifie évidemment $Ax = b$. On va voir dans ce qui suit comment on s'y prend numériquement.

5.3.1 Elimination de Gauss sans recherche de pivot

5.3.1.1 Factorisation LU

Soit à résoudre le système : Trouver $x \in \mathbb{R}^n$ tel que

$$Ax = b$$

pour $A \in \mathbb{R}^{n \times n}$, et $b \in \mathbb{R}^n$. On suppose que A est inversible.

Le but est de se ramener à un système triangulaire. On procède par étapes :

Étape 1 : Elimination de l'inconnue x_1 des lignes 2 à n

Sous l'hypothèse que $a_{11} = 0$ on peut l'utiliser pour éliminer l'inconnue x_1 des lignes 2 à n. Le terme a_{11} est appelé pivot et on note pour la suite :

$$\pi_1 = a_{11}$$

La ligne i devient alors:

$$0x_1 + (a_{i2} - \frac{a_{i1}}{\pi_1}a_{12})x_2 + \ldots + (a_{in} - \frac{a_{i1}}{\pi_1}a_{1n})x_n = b_i - \frac{a_{i1}}{\pi_1}b_1$$

ce que l'on écrit encore

$$a_{i2}^{(2)}x_2 + \ldots + a_{in}^{(2)}x_n = b_i^{(2)}, \forall i > 1$$

en posant

$$\begin{aligned} a_{i2}^{(2)} &= a_{i2} - \frac{a_{i1}}{\pi_1}a_{12} \\ &\vdots \\ a_{in}^{(2)} &= a_{in} - \frac{a_{i1}}{\pi_1}a_{1n} \\ b_i^{(2)} &= b_i - \frac{a_{i1}}{\pi_1}b_1 \end{aligned}$$

Si on pose aussi, pour $1 \leq j \leq n$

$$a_{1j}^{(2)} = a_{1j}, \forall 1 \leq j \leq n \text{ et } b_1^{(2)} = b_1$$

on a obtenu le système équivalent :

$$A^{(2)}x = b^{(2)}$$

avec

$$A^{(2)} = \begin{pmatrix} a_{11}^{(2)} & a_{12}^{(2)} & \dots & a_{1n}^{(2)} \\ 0 & a_{22}^{(2)} & \dots & a_{2n}^{(2)} \\ \vdots & \vdots & & \vdots \\ 0 & a_{n2}^{(2)} & \dots & a_{nn}^{(2)} \end{pmatrix}$$

On voit que

$$A^{(2)} = M^{(1)}A \text{ où } M^{(1)} = \begin{pmatrix} 1 & 0 & \dots & \dots & 0 \\ -\frac{a_{21}}{a_{11}} & 1 & \ddots & & \vdots \\ \vdots & 0 & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ -\frac{a_{n1}}{a_{11}} & 0 & \dots & 0 & 1 \end{pmatrix}$$

Le système s'écrit alors

$$A^{(2)} = M^{(1)}A, \text{ et } b^{(2)} = M^{(1)}b$$

Tout se passe comme si on avait prémultiplié à gauche le système initial par $M^{(1)}$.

Étape k : Elimination de l'inconnue x_k des lignes $k+1$ à n

On suppose que l'on a pu itérer le procédé ci-dessus $k-1$ fois, c'est que l'on n'a jamais rencontré de pivot nul :

$$\pi_i = a_{ii}^{(i)} = 0 \text{ pour } 1 \le i \le k-1$$

On obtient alors le système équivalent :

$$A^{(k)}x = b^{(k)}$$

avec $A^{(k)}$ de la forme

$$A^{(k)} = \begin{pmatrix} a_{11}^{(k)} & a_{12}^{(k)} & \dots & \dots & \dots & \dots & \dots & a_{1n}^{(k)} \\ 0 & a_{22}^{(k)} & \ddots & & & & & a_{2n}^{(k)} \\ \vdots & \ddots & a_{33}^{(k)} & \ddots & & & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & & & \vdots \\ 0 & \dots & \dots & 0 & a_{k-1,k-1}^{(k)} & \dots & \dots & a_{k-1,n}^{(k)} \\ \vdots & & & \vdots & 0 & & & \\ \vdots & & & \vdots & \vdots & & A^{(k)} & \\ 0 & \dots & \dots & 0 & 0 & & & \end{pmatrix} \tag{5.1}$$

où $A^{(k)}$ est une matrice carrée d'ordre $n-k+1$:

$$A^{(k)} = \begin{pmatrix} a_{kk}^{(k)} & \dots & a_{kn}^{(k)} \\ \vdots & & \vdots \\ a_{nk}^{(k)} & \dots & a_{nn}^{(k)} \end{pmatrix} \tag{5.2}$$

Comme précédemment, on doit effectuer une hypothèse sur la valeur du pivot :

$$\pi_k = a_{kk}^{(k)} = 0$$

On peut alors introduire la matrice

$$M^{(k)} = \begin{pmatrix} 1 & 0 & \dots & & \dots & \dots & \dots & 0 \\ 0 & 1 & \ddots & & & & & \vdots \\ \vdots & \ddots & \ddots & \ddots & & & & \vdots \\ \vdots & \vdots & 0 & 1 & \ddots & & & \vdots \\ \vdots & & & -\frac{a_{k+1,k}^{(k)}}{\pi_k} & \ddots & \ddots & & \vdots \\ \vdots & & & \vdots & 0 & \ddots & \ddots & \vdots \\ \vdots & & & \vdots & \vdots & \ddots & \ddots & 0 \\ 0 & 0 & \dots & -\frac{a_{n,k}^{(k)}}{\pi_k} & 0 & \dots & 0 & 1 \end{pmatrix}$$

(5.3)

Remarquons que l'inverse de $M^{(k)}$ est $L^{(k)}$:

$$L^{(k)} = \begin{pmatrix} 1 & 0 & \dots & & \dots & \dots & \dots & 0 \\ 0 & 1 & \ddots & & & & & \vdots \\ \vdots & \ddots & \ddots & \ddots & & & & \vdots \\ \vdots & \vdots & 0 & 1 & \ddots & & & \vdots \\ \vdots & & & \frac{a_{k+1,k}^{(k)}}{\pi_k} & \ddots & \ddots & & \vdots \\ \vdots & & & \vdots & 0 & \ddots & \ddots & \vdots \\ \vdots & & & \vdots & \vdots & \ddots & \ddots & 0 \\ 0 & 0 & \dots & \frac{a_{n,k}^{(k)}}{\pi_k} & 0 & \dots & 0 & 1 \end{pmatrix} \qquad (5.4)$$

Soit

$$A^{(k+1)} = M^{(k)} A^{(k)} \text{ et } b^{(k+1)} = M^{(k)} b^{(k)}$$

alors

$$A^{(k+1)} x = b^{(k+1)}$$

et

$$A^{(k+1)} = \begin{pmatrix} a_{11}^{(k+1)} & a_{12}^{(k+1)} & \cdots & \cdots & \cdots & \cdots & \cdots & a_{1n}^{(k+1)} \\ 0 & a_{22}^{(k+1)} & \ddots & & & & & a_{2n}^{(k+1)} \\ \vdots & \ddots & a_{33}^{(k+1)} & \ddots & & & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & & & \vdots \\ 0 & \cdots & \cdots & 0 & a_{k,k}^{(k+1)} & \cdots & \cdots & a_{k,n}^{(k+1)} \\ \vdots & & & \vdots & 0 & & & \\ \vdots & & & \vdots & \vdots & & A^{(k+1)} & \\ 0 & \cdots & \cdots & 0 & 0 & & & \end{pmatrix}$$

où $A^{(k+1)}$ est une matrice carrée d'ordre $n-k$:

$$A^{(k+1)} = \begin{pmatrix} a_{k+1,k+1}^{(k+1)} & \cdots & a_{k+1n}^{(k+1)} \\ \vdots & & \vdots \\ a_{nk+1}^{(k+1)} & \cdots & a_{nn}^{(k+1)} \end{pmatrix}$$

Après n-1 étapes : Si on itère $n-1$ fois, et si les pivots apparus sont tous non nuls, soit :

$$\pi_i = a_{ii}^{(i)} = 0 \text{ pour } 1 \leq i \leq n-1$$

on arrive au système triangulaire équivalent :

$$A^{(n)}x = b^{(n)}$$

avec

$$A^{(n)} = \begin{pmatrix} \pi_1 & * & * & \cdots & * \\ 0 & \pi_2 & * & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & * \\ 0 & \cdots & \cdots & 0 & \pi_n \end{pmatrix}$$

qui est inversible, si de plus

$$\pi_n = 0$$

Bilan

Si on appelle L la matrice triangulaire inférieure avec des 1 sur la diagonale, et $L^{(k)}$ la matrice définie en (5.4)

$$L = L^{(1)} \cdots L^{(n-1)}$$

et U la matrice triangulaire supérieure

$$U = A^{(n)}$$

on a

$$A = LU$$

On dit qu'on a effectué une factorisation de Gauss ou factorisation LU de A.

REMARQUE 5.3.1 Il est aussi possible avec le même algorithme d'obtenir la factorisation LU d'une matrice de $M_m(C)$, avec $m \geq n$, si les pivots qui apparaissent sont non nuls.

5.3.1.2 Coût de la méthode d'élimination de Gauss

On estime le coût de l'élimination de x_k. Rappelons qu'à l'étape $k-1$, on obtient la matrice A_k donnée en (5.1) comportant le bloc $A^{(k)}$ donné par (5.2) :

$$A^{(k)} = \begin{matrix} a_{kk}^{(k)} & \dots & a_{kn}^{(k)} \\ \vdots & & \vdots \\ a_{nk}^{(k)} & \dots & a_{nn}^{(k)} \end{matrix}$$

On construit tout d'abord $M^{(k)}$ donné par (5.3) donnée par

$$M^{(k)} = \begin{matrix} 1 & 0 & \dots & \dots & \dots & \dots & \dots & 0 \\ 0 & 1 & \ddots & & & & & \vdots \\ \vdots & \ddots & \ddots & \ddots & & & & \vdots \\ \vdots & \vdots & 0 & 1 & \ddots & & & \vdots \\ \vdots & & & -\frac{a_{k+1,k}^{(k)}}{\pi_k} & \ddots & \ddots & & \vdots \\ \vdots & & & \vdots & 0 & \ddots & \ddots & \vdots \\ \vdots & & & \vdots & \vdots & \ddots & \ddots & 0 \\ 0 & 0 & \dots & -\frac{a_{n,k}^{(k)}}{\pi_k} & 0 & \dots & 0 & 1 \end{matrix}$$

il faut pour cela diviser par le pivot sur $n-k$ lignes. On effectue donc $n-k$ divisions.

On construit ensuite le produit $A^{(k+1)} = M^{(k)}A^{(k)}$ (puisque notre objectif est de déterminer à la fin $A^{(n)}$). Pour cela il faut effectuer $(n-k)^2$ additions et multiplications.

Au total :

$$\sum_{k=1}^{n} (n-k)^2 = \frac{1}{3}n(n-1)(n-\frac{1}{2}) \sim \frac{n^3}{3} \text{additions et multiplications.}$$

$$\sum_{k=1}^{n} (n-k) = \frac{1}{2}n(n-1) \text{ divisions}$$

5.3.1.3 Utilisation de la factorisation LU

Calcul de déterminant

Une utilisation de la factorisation LU est le calcul de déterminant de A : en effet, si A admet une factorisation LU, on a

$$\det A = \det(LU) = \det L . \det U$$

Or L est triangulaire à diagonale unité donc de déterminant 1, ce qui donne finalement

$$\det A = \det U = \prod_{i=1}^{n} \pi_i$$

La factorisation LU permet donc de calculer le déterminant de A avec une complexité de l'ordre de $\frac{n^3}{3}$ additions et multiplications, au lieu de n! avec la formule du déterminant !

Résolution de systèmes linéaires

N'oublions pas notre objectif initial : résoudre un système linéaire. Il est clair que la factorisation LU permet de résoudre successivement deux systèmes triangulaires :

$$Ly = b \text{ puis } Ux = y$$

dont la solution x vérifie $Ax = b$.

Au delà de ce point, si on dispose de plusieurs systèmes linéaires de seconds membres différents mais de même matrice A :

$$Ax = b_i \text{ pour } i = 1, \ldots l$$

il suffit de calculer une seule fois les matrices L et U et de les stocker. On peut alors effectuer autant de descentes et de remontées qu'on a de systèmes pour les résoudre tous, sans pour autant refaire la factorisation LU.

Pour l systèmes à résoudre, la complexité est de l'ordre de $\frac{n^3}{3} + l . n^2$ additions et multiplications plutôt que $l \frac{n^3}{3}$.

5.3.1.4 Et le stockage...

Pour une matrice de taille n, on remarque que le nombre de termes à connaître pour disposer des matrices L et U n'est que de n^2, puisque la diagonale unité de L n'est pas à stocker. De ce fait on peut utiliser la place mémoire disponible dans A pour y enregistrer les termes de L et U.

Cette remarque explique que l'algorithme qui suit "écrase" la matrice A par sa décomposition LU.

5.3.1.5 Algorithme

La factorisation LU présentée au paragraphe (5.3.1.1) n'est pas implémentée selon la méthode décrite : il existe une algorithme permettant de calculer directement les termes des matrices L et U. On notera dans l'algorithme ci-dessous que l'ordre de caclul des coefficients n'est pas indifférent.

Voici l'algorithme réalisant la factorisation LU d'une matrice A et stockant cette factorisation dans la place mémoire occupée par A : (la matrice A est perdue, on dit qu'on écrase A).

Algorithme : Factorisation LU

```
Pour j ← 1 à n-1
  Pour i ← j+1 à n
    A(i,j) = A(i,j)/A(j,j) //construction de la j-ième colonne de L
    Pour k ← j+1 à n
      A(i,k) = A(i,k) - A(i,j) × A(j,k) //actualisation des n-j-1 dernières lignes de A
    FinPour
  FinPour
FinPour
```

5.3.1.6 Exercice

EXERCICE 5.2 On considère la matrice de Vandermonde

$$A = (a_{ij}) \text{ avec } a_{ij} = x_i^{j-1}, i,j = 1,...n$$

où les x_i sont n abscisses distinctes.

(1) Programmer la factorisation LU de la matrice A pour des valeurs de n comprises entre 10 et 60 par pas de 10. On pourra utiliser des x_i définis par $x_i = i$, pour $i = 1,...n$.

(2) Calculer et représenter graphiquement le nombre d'opérations effectuées.

(3) Retrouver l'ordre de complexité de la méthode. On pourra utiliser l'option de la commande `plot2d logflag` qui permet d'afficher des échelles logarithmiques, ainsi que la commande `reglin` fournissant les coefficients a et b de l'équation de la droite de régression linéaire appliquée à un nuage de points.

5.3.1.7 CNS d'existence d'une factorisation LU

PROPRIÉTÉ 5.3.1 Soit A une matrice de $M_m(C)$. Pour $1 \leq p \leq n$, on note A_p le bloc

$$A_p = \begin{matrix} a_{11} & \cdots & \cdots & a_{1p} \\ a_{21} & & & a_{2p} \\ \vdots & & & \vdots \\ a_{p1} & \cdots & \cdots & a_{pp} \end{matrix}$$

La matrice A admet une factorisation

$$A = LU$$

où L est triangulaire inférieure avec des 1 sur la diagonale, et U est triangulaire supérieure et inversible si et seulement si tous les blocs $A_p, 1 \leq p \leq n$ sont inversibles. De plus, cette factorisation est unique. De plus, si A est réelle, L et U le sont aussi.

On pourra trouver la démonstration détaillée de ce résultat dans [YA].

5.3.1.8 Exercices

EXERCICE 5.3 Supposons qu'on résolve $A\,x = b$ avec

$$A = \begin{matrix} 1 & 1-\varepsilon & 3 \\ 2 & 2 & 2 \\ 3 & 6 & 4 \end{matrix} \quad \text{et } b = \begin{matrix} 5-\varepsilon \\ 6 \\ 13 \end{matrix}$$

(1) Déterminer pour quelles valeurs de ε la matrice A ne satisfait pas les hypothèse du théorème ci-dessus.

(2) Pour quelles valeurs de ε cette matrice est-elle singulière ?

(3) Est-il possible de calculer la factorisation dans ce cas ?

EXERCICE 5.4 Montrer que la factorisation LU d'une matrice A peut être utilisée pour calculer la matrice inverse A^{-1}. (On remarquera que la j-ième colonne de A^{-1} vérifie le système linéaire $A\,y_j = e_j$, e_j étant le j-ième vecteur de la base canonique.)

EXERCICE 5.5 Considérons la matrice inversible

$$A = \begin{matrix} 1 & 1+0.5.10^{-15} & 3 \\ 2 & 2 & 20 \\ 3 & 6 & 4 \end{matrix}$$

(1) Effectuer la factorisation LU de A à l'aide du programme de l'exercice.

(2) Calculer le résidu $A - LU$, et montrer qu'il vérifie :

$$A - LU = \begin{matrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 4 \end{matrix}$$

(3) Que peut-on en conclure ?

(4) Comparez le résultat que vous obtenez avec celui fourni par scilab avec la routine `lu`. Analysez la différence.

EXERCICE 5.6 Soit la matrice $A \in R^{n\times n}$. On définit la matrice $B = [A, I_n] \in R^{n\times 2n}$, où I_n la matrice identité.

Pour calculer l'inverse de A on établit l'algorithme suivant

Algorithme : `InverMat`

```
Pour j ← 1 à n
  Pour i ← 1 à n
    Si  (i = j)
      Pour k ← 1 à n
        B'(j,k) = B(j,k)/B(j,j)
      FinPour
    Sinon
      Pour k ← 1 à n
        B'(i,k) = B(i,k) - B(i,j) * B'(j,k)
      FinPour
    FinSi
    B ← B'
  FinPour
FinPour
```

(1) Montrer que le résultat est équivalent à multiplier à gauche B par une suite de matrices $C^{(i)}$ que vous calculerez.

(2) Montrer qu'à la fin de l'algorithme les n premières colonnes de la matrice B forment la matrice identité I_n.

(3) En déduire que les n dernières colonnes de B forment A^{-1}.

(4) Application : $A = \begin{matrix} 2 & 4 & 2 \\ 1 & 0 & 3 \\ 3 & 1 & 2 \end{matrix}$ en utilisant les programme `inverMat`

5.4 Méthode de Cholesky

Dans le cas où la matrice A est hermitienne et définie positive, on peut toujours effectuer la factorisation décrite ci-dessus. de plus, on peut trouver une factorisation du type $A = LL^*$ moins gourmande en place mémoire.

5.4.1 Existence de la factorisation

THÉORÈME 5.4.1 Si A est hermitienne et définie positive, alors A admet une unique factorisation LU où L est triangulaire inférieure avec des 1 sur la diagonale et U est triangulaire supérieure et inversible.

DÉMONSTRATION. Si A est hermitienne et définie positive, ses blocs $A_p, 1 \leq p \leq n$ (voir théorème ci-dessus) le sont aussi, et on peut donc appliquer le théorème.

THÉORÈME 5.4.2 Si A est hermitienne et définie positive, alors il existe une unique matrice L triangulaire inférieure et inversible, avec des coefficients positifs sur la diagonale telle que

$$A = LL^*$$

Cette factorisation porte le nom de Cholesky (colonel de l'armée de Napoléon). De plus si A est réelle, symétrique et définie positive, L est réelle.

On pourra trouver la démonstration dans [YA].

REMARQUE 5.4.1 Il est important de noter que les matrices L dans les factorisation LU et de Cholesky sont différentes.

5.4.2 Algorithme

Algorithme : Algorithme de Cholesky

$L(1,1) \leftarrow \sqrt{A(1,1)}$ // Construction de l_{11}
Pour i ← 2 à n
  Pour j ← 1 à i-1
    $L(i,j) \leftarrow \frac{1}{L(j,j)} \left(A(i,j) - \sum_{k=1}^{j-1} L(i,k) \times L(j,k) \right)$
  FinPour
  $L(i,i) \leftarrow \left(a(i,i) - \sum_{k=1}^{i-1} L^2(i,k) \right)^{1/2}$
FinPour

5.4.3 Complexité

On montre facilement que l'on effectue :

— $\frac{n^3}{6}$ additions + multiplications

— $\frac{1}{2}n(n-1)$ divisions

— n évaluations de racines carrées.

REMARQUE 5.4.2 L'intérêt de la méthode de Cholesky réside dans le fait qu'elle demande une place mémoire deux fois inférieure à celle de la factorisation LU, puisqu'on ne stocke que L et que sa complexité est aussi deux fois moindre.

5.5 Elimination de Gauss avec recherche de pivot partiel

La méthode de Gauss peut-être bloquée si on tombe sur un pivot nul. C'est pour cela qu'a été mise au point la méthode dite de Gauss avec pivotage partiel . Elle consiste à échanger les lignes du système lorsqu'on tombe sur un pivot nul, ce qui permet de continuer sans problème.

Elle repose sur l'utilisation de matrices de permutation, dont on rappelle ci-dessous la définition:

DÉFINITION 5.5.1 On appelle permutation de $\{1,...,n\}$ une bijection de $\{1,...,n\}$ sur $\{1,...,n\}$.

DÉFINITION 5.5.2 Soit σ une permutation de $\{1,...,n\}$, on associe à σ une matrice P dite de permutation d'ordre n par

$$P_{ij} = \delta_{\sigma(i)j}$$

c'est à dire

$$P_{ij} = 1 \text{ si } j = \sigma(i)$$
$$P_{ij} = 0 \text{ si } j = \sigma(i)$$

Les matrices de permutation ont des propriétés intéressantes, qui font que leur utilisation pour effectuer les pivotages conduit au résultat suivant :

THÉORÈME 5.5.1 Soit $A \in M_m(R)$, inversible. Alors il existe :

- une matrice de permutation P,
- une matrice triangulaire inférieure L, avec des 1 sur la diagonale,
- une matrice triangulaire supérieure U inversible,

telles que

$$PA = LU$$

REMARQUE 5.5.1 Le pivotage partiel ne sert pas seulement à garantir l'obtention d'une factorisation. Il garantit aussi une meilleure stabilité que celle obtenue par la méthode LU dans le cas de matrices mal conditionnées. On pourra s'en convaincre en examinant le cas du système $Ax = b$, constitué de la matrice A :

$$A = \begin{matrix} 10^{-9} & 1 \\ 1 & 1 \end{matrix}$$

et du second membre b :

$$b = \begin{matrix} 1 \\ 2 \end{matrix}$$

Sa solution est

$$x = \frac{1}{1-10^{-9}}, \frac{1-2.10^{-9}}{1-10^{-9}} \qquad [1,1]$$

Si on le résout sur une machine à 8 chiffres significatifs, la factorisation calculée est

$$U = \begin{matrix} 10^{-9} & 1 \\ 0 & -10^{-9} \end{matrix} \quad \text{et } L = \begin{matrix} 1 & 0 \\ 10^{-9} & 1 \end{matrix}$$

ce qui donne la solution

$$x = [0,1]$$

Si on utilise le pivotage partiel, on obtient :

$$U = \begin{matrix} 1 & 1 \\ 0 & 1 \end{matrix} \quad \text{et } L = \begin{matrix} 1 & 0 \\ 10^{-9} & 1 \end{matrix}$$

ce qui donne la solution

$$x = [1, 1]$$

En exploitant cette remarque, on aboutit à la méthode de Gauss dite de pivot maximal. Elle consiste à prendre, à chaque étape, comme pivot l'élément diagonal qui est le plus grand en valeur absolue afin de limiter les erreurs d'arrondi inévitables dues à un résultat de division trop faible (pertes de chiffres significatifs).

EN SUBSTANCE

- La factorisation LU d'une matrice A consiste à calculer une matrice triangulaire inférieure L et une matrice triangulaire supérieure U telles que $A = LU$.
- La factorisation LU, quand elle existe, n'est pas unique. Cependant, on peut la rendre unique en se donnant des conditions supplémentaires, par exemple en fixant les valeurs des éléments diagonaux de L à 1. Ceci s'appelle factorisation de Gauss.
- La factorisation de Gauss existe et est unique si et seulement si les mineurs principaux de A d'ordre 1 à $n-1$ sont non nuls (autrement, au moins un pivot est nul).
- Quand on trouve un pivot nul, un nouveau pivot peut être obtenu en échangeant des lignes (ou colonnes) convenablement choisies. C'est la stratégie du pivot.
- Le calcul de la factorisation de Gauss nécessite de l'ordre de $2n^3/3$ opérations en général, et seulement de l'ordre de n opérations dans le cas d'un système tridiagonal.
- Pour les amtrices symétriques définies positives, on peut utiliser la factorisation de Cholesky $A = HH^T$, où H est une matrice triangulaire inférieure. Le coût de calcul est de l'ordre de $n^3/3$ opérations.
  La sensibilité du résultat aux perturbations des données dépend du conditionnement de la matrice du système : la solution calculée peut être imprécise quand la matrice est mal conditionnée.

5.6 Bibliographie

Les ouvrages ci-dessous sont disponibles sous forme de fichier téléchargeable sur le site du cours ou sur Arel

[CB] Algèbre matricielle numérique, Claude Brezinski

[YA] Algèbre linéaire et analyse numérique matricielle, Yves Achdou, téléchargeable à l'adresse http:/ / www.ann.jussieu.fr/ ~achdou/ files/ teaching/ linalg/ book.pdf

[AH1]Analyse numérique matricielle, Cours de 3ème année, Alain Huard, téléchargeable à partir du site de l'INSA Toulouse www-gmm.insa-toulouse.fr

[AH2]Analyse numérique des grands problèmes linéaires, Cours de 4ème année, Alain Huard, téléchargeable à partir du site de l'INSA Toulouse www-gmm.insa-toulouse.fr

Les ouvrages ci-dessous sont disponibles en librairie

[QS] Calcul scientifique, Cours, exercices corrigés et illustrations en Matlab et Octave, Alfio Quarteroni, Fausto Saleri, Springer, 2206.

[AF] Analyse numérique pour Ingénieurs, André Fortin, Presses Internationales Polytechnique, 2001.

6

MÉTHODES ITÉRATIVES

Pour la redaction de ce chapitre, nous nous sommes fortement inspirés du livre de A. Quarteroni et al. [QSS] cité en bibliographie.

6.1 Introduction

Une méthode itérative consiste à construire une suite de vecteurs $x^{(0)}, x^{(1)}, ..., x^{(k)}, ...$ qui convergera vers la solution du système linéaire $A x = b$ à résoudre. La notion de convergence de cette suite est traitée grâce aux normes vectorielles qui ont été abordées dans le chapitre précédent. Les conditions de convergence de ces méthodes vont requérir l'emploi de majorations faisant appel aux normes matricielles, également abordées dans le chapitre précédent.

Les méthodes itératives donnent en théorie, la solution x d'un système linéaire après un nombre fini d'itérations. A chaque pas, elles nécessitent le calcul du résidu du système, qui permet de décider quand on estime avoir atteint la solution espérée. Dans le cas d'une matrice pleine, leur coût est donc de l'ordre de n^2 opérations à chaque itération, alors que le coût des méthodes directes est, en tout et pour tout, de l'ordre de $2/3 n^3$. Les méthodes itératives peuvent devenir compétitives si elles convergent en un nombre d'itérations indépendant de n, ou croissant sous-linéairement avec n. Elles sont utilisées soit pour la résolution de systèmes linéaires

de grande taille, soit lorsqu'on dispose d'une solution approchée du système que l'on désire améliorer.

Il est à noter que les méthodes itératives sont sensibles au conditionnement de la matrice du système, et que l'on peut donc être amené à utiliser des techniques de préconditionnement, telles que vues au chapitre précédent.

6.2 Convergence des méthodes itératives

L'idée de base des méthodes itératives est de construire une suite convergente de vecteurs $x^{(k)}$ telle que

$$x = \lim_{k \to +\infty} x^{(k)} \tag{6.1}$$

où x est la solution de

$$Ax = b \tag{6.2}$$

En pratique, le calcul devrait être interrompu à la première itération n pour laquelle $\|x^{(n)} - x\| < \varepsilon$, où ε est une tolérance fixée et $\|\cdot\|$ une norme vectorielle donnée. Mais comme la solution exacte n'est évidemment pas connue, il faudra trouver un critère d'arrêt plus commode (voir section 6.4).

Considérons pour commencer les méthodes itératives de la forme :

$$\begin{aligned} &x^{(0)} \text{ donné} \\ &x^{(k+1)} = Bx^{(k)} + f; \quad k \geq 0 \end{aligned} \tag{6.3}$$

où B désigne une matrice carrée $n \times n$ appelée matrice d'itération et où f est un vecteur dépendant de b (second membre du système à résoudre).

DÉFINITION 6.2.1 Une méthode itérative de la forme (6.3) est dite consistante avec (6.2) si f et B sont tels que $x = Bx + f$, x étant la solution de (6.2), ou de façon équivalente, si f et B satisfont :

$$f = (I - B)A^{-1}b$$

Si on note

$$e^{(k)} = x^{(k)} - x \tag{6.4}$$

l'erreur à l'itération k, la condition (6.1) revient à $\lim_{k \to +\infty} e^{(k)} = 0$ pour toute valeur initiale $x^{(0)}$.

REMARQUE 6.2.1 La seule propriété de consistance ne suffit pas à assurer la convergence d'une méthode itérative, comme le montre l'exemple suivant :

EXEMPLE 6.2.1 On veut résoudre le système linéaire $2Ix = b$ avec la méthode itérative

$$x^{(k+1)} = -x^{(k)} + b$$

qui est clairement consistante. Cette suite n'est pas convergente pour une donnée initiale arbitraire. Si par exemple $x^{(0)} = 0$, la méthode donne $x^{(2k)} = 0$, $x^{(2k+1)} = b$; $k = 0, 1, \ldots$
En revanche si $x^{(0)} = \frac{1}{2}b$ la méthode est convergente.

THÉORÈME 6.2.1 Si la méthode (6.3) est consistante, la suite de vecteurs $x^{(k)}$ de (6.3) converge vers la solution de (6.2) pour toute donnée initiale $x^{(0)}$ si et seulement si $\rho(B) < 1$.

D'après (6.4), et grâce à l'hypothèse de consistance, on a $e^{(k+1)} = Be^{(k)}$, d'où

$$e^{(k+1)} = B^k e^{(0)}; \ \forall k = 0, 1, \ldots \tag{6.5}$$

Il en résulte donc que $\lim_{k \to +\infty} B^k e^{(0)} = 0$ pour tout $e^{(0)}$ si et seulement si $\rho(B) < 1$.

Réciproquement, supposons que $\rho(B) > 1$, alors il existe au moins une valeur propre $\lambda(B)$ de module plus grand que 1. Soit $e^{(0)}$ un vecteur propre associé à λ, alors $Be^{(0)} = \lambda e^{(0)}$. Comme $|\lambda| > 1$, $e^{(k)}$ ne peut pas tendre vers 0 quand $k \to +\infty$

REMARQUE 6.2.2 On a déjà vu que pour une tolérance fixée aussi petite que l'on veut, il existe toujours une norme matricielle telle que la norme d'une matrice A soit arbitrairement proche du rayon spectral de A. De plus on a toujours $\rho(A) \leq \|A\|$. La condition de convergence $\rho(B) < 1$ entraîne donc immédiatement que la condition $\|B\| < 1$, pour une norme matricielle consistante arbitraire, est suffisante pour que la méthode converge.

Il est raisonnable de penser que la convergence est d'autant plus rapide que $\rho(B)$ est petit. Une estimation de $\rho(B)$ peut donc fournir une bonne indication sur la convergence de l'algorithme.

En effet le théorème suivant (cf. [SB]) établit une relation entre l'erreur de la solution obtenue après k itération et le rayon spectral :

THÉORÈME 6.2.2 Pour la méthode (6.3) les erreurs $e^{(k)} = x^{(k)} - x$ satisfont à

$$\sup_{e^{(0)} = 0} \lim_{k \to \infty} \sup \sqrt[k]{\frac{\|e^{(k)}\|}{\|e^{(0)}\|}} = \rho(B)$$

Les itérations définies en (6.3) sont un cas particulier des méthodes itératives de la forme

$$\begin{aligned} x^{(0)} &= f_0(A, b) \\ x^{(n+1)} &= f_{n+1}(x^{(n)}, x^{(n-1)}, x^{(n-m)}, A, b) \ \text{pour } n \geq m \end{aligned}$$

où les f_i sont des fonctions et les $x^{(m)}, \ldots, x^{(1)}$ des vecteurs donnés. Le nombre de pas dont dépend l'itération courante (ici $m + 1$) s'appelle ordre de la méthode. Si les fonctions f_i sont indépendantes de i, la méthode est dite stationnaire. Elle est instationnaire dans le cas contraire. Enfin, si f_i dépend linéairement de $x^{(0)}, \ldots x^{(m)}$, la méthode est dite linéaire, autrement elle est non linéaire.

Au regard de ces définitions, les algorithmes considérés jusqu'à présent sont donc des méthodes itératives linéaires, stationnaires du premier ordre.

6.3 Méthodes itératives linéaires

Une technique générale pour définir une méthode itérative linéaire consistante est basée sur la décomposition de A, aussi appelée splitting, sous la forme $A = P - N$, où P est une matrice inversible.

On se donne $x^{(0)}$, et on calcule $x^{(k)}$ pour $k \geq 1$, en résolvant le système

$$Px^{(k+1)} = Nx^{(k)} + b;\ k \geq 0 \tag{6.1}$$

Selon le formalisme introduit en (6.3), on peut écrire la matrice d'itération $B = P^{-1}N$, et $f = P^{-1}b$ puisqu'on peut écrire de façon équivalente (6.1) comme

$$x^{(k+1)} = P^{-1}Nx^{(k)} + P^{-1}b;\ k \geq 0$$

On peut aussi écrire

$$x^{(k+1)} = x^{(k)} + P^{-1}r^{(k)};\ k \geq 0 \tag{6.2}$$

où

$$\begin{aligned} P^{-1}r^{(k)} &= P^{-1}b + P^{-1}Nx^{(k)} - x^{(k)} \\ &= P^{-1}b + (P^{-1}N - I)x^{(k)} \\ &= P^{-1}\ b + (N - P)x^{(k)} \\ &= P^{-1}\ b - Ax^{(k)} \end{aligned}$$

donc

$$r^{(k)} = b - Ax^{(k)} \tag{6.3}$$

$r^{(k)}$ désigne le résidu à l'itération k. La relation (6.2) montre qu'on doit résoudre un système linéaire de matrice P à chaque itération. En plus d'être inversible, P doit donc être facile à inverser afin de minimiser le coût de calcul. On peut d'ailleurs remarquer que si $P = A$ et $N = 0$, la méthode (6.2) converge en une itération et est équivalente à une méthode directe.

Deux résultats garantissent la convergence de (6.2) sous des hypothèses convenables concernant le splitting de A.

THÉORÈME 6.3.1 Soit $A = P - N$, avec A et P symétriques définies positives. Si la matrice $2P - A$ est définie positive, alors la méthode (6.2) est convergente pour toute donnée initiale $x^{(0)}$ et

$$\rho(B) = \|B\|_A = \|B\|_P < 1$$

De plus, la convergence est monotone pour les normes $\|.\|_P$ et $\|.\|_A$, i.e.

$$\|e^{(k+1)}\|_P \leq \|e^{(k)}\|_P \quad \text{et} \quad \|e^{(k+1)}\|_A \leq \|e^{(k)}\|_A;\ k = 0, 1, \ldots$$

THÉORÈME 6.3.2 Soit $A = P - N$ avec A symétrique définie positive. Si la matrice $P + P^{*} - A$ est définie positive, alors P est inversible, la méthode itérative (6.2) converge de manière monotone pour la norme $\|.\|_A$, et $\rho(B) \leq \|B\|_A < 1$.

6.3.1 Méthodes de Jacobi, Gauss-Seidel et relaxation

On rappelle que l'on veut résoudre le système linéaire $Ax = b$, dans lequel A est une matrice carrée. Pour cela on décompose A sous la forme $A = M - N$. Alors $Ax = b \Leftrightarrow Mx = Nx + b$. Si M est une matrice régulière (i.e. inversible) on définit la méthode itérative par :

$$Mx^{(k+1)} = Nx^{(k)} + b \Leftrightarrow x^{(k+1)} = M^{-1}Nx^{(k)} + M^{-1}b \tag{6.4}$$

REMARQUE 6.3.1 Cette formulation est bien consistante puisque sa solution x^* vérifie (si elle existe) $x^* = M^{-1}Nx^* + M^{-1}b$. Or ceci équivaut à $Mx^* = Nx^* + b \Leftrightarrow (M - N)x^* = b \Leftrightarrow Ax^* = b$ donc x^* est le vecteur solution du système initial. On dit aussi qu'il s'agit d'un problème de point fixe.

6.3.1.1 Méthodes non relaxées

La matrice A peut s'écrire aussi $A = D - E - F = D + L + U$ (choix dépendant des auteurs), où D est la diagonale de A, $-E = L$ sa partie triangulaire inférieure et $-F = U$ sa partie triangulaire supérieure, i.e. : $A = \begin{pmatrix} \ddots & & -F \\ & D & \\ -E & & \ddots \end{pmatrix} = \begin{pmatrix} \ddots & & U \\ & D & \\ L & & \ddots \end{pmatrix}$.

La décomposition sous la forme $A = M - N$ doit être établie à partir de celle sous la forme $A = D - E - F = D + L + U$: suivant les matrices D, E et F (ou D, L et U) que l'on associe dans M et N on obtient les méthodes de Jacobi, Gauss-Seidel et Richardson :

Méthode	Décomposition $A = M - N$	Matrice $M^{-1}N$ de (6.4)	Description d'une itération
Jacobi	$A = \underbrace{D}_{M} - \underbrace{(E+F)}_{N}$	$J = M^{-1}N = D^{-1}(E+F) = I - D^{-1}A$	$Dx^{(k+1)} = (E+F)x^{(k)} + b$
Gauss-Seidel	$A = \underbrace{(D-E)}_{M} - \underbrace{F}_{N}$	$G = M^{-1}N = (D-E)^{-1}F$	$(D-E)x^{(k+1)} = Fx^{(k)} + b$
Richardson	$A = \underbrace{I}_{M} - \underbrace{(I-A)}_{N}$	$R = M^{-1}N = I - A$	$x^{(k+1)} = (I-A)x^{(k)} + b$

Si on utilise la décomposition $A = D + L + U$, on pour les méthodes de Jacobi et de Gauss-Seidel le tableau suivant :

Méthode	Décomposition $A = M - N$	Matrice $M^{-1}N$ de (6.4)	Description d'une itération
Jacobi	$A = \underbrace{D}_{M} - \underbrace{(L+U)}_{N}$	$J = M^{-1}N = -D^{-1}(L+U)$	$Dx^{(k+1)} = -(L+U)x^{(k)} + b$
Gauss-Seidel	$A = \underbrace{(D+L)}_{M} - \underbrace{U}_{N}$	$G = M^{-1}N = (D+L)^{-1}U$	$(D+L)x^{(k+1)} = Ux^{(k)} + b$

La description des itérations est directement liée à la formulation (6.4) et peut être retrouvée facilement.

REMARQUE 6.3.2 La méthode de Richardson est ici donnée pour mémoire car elle n'a que peu d'intérêt numérique. Cela est notamment dû au fait que le rayon spectral de la matrice d'itération, R, n'est pas très bon en général.

6.3.1.2 Méthodes relaxées

Des performances des méthodes non relaxées, parfois peu satisfaisantes, a rapidement découlé l'émergence de méthodes plus efficaces, dites relaxées. L'idée de base est assez simple : Ces méthodes consistent à reprendre les différents choix du paragraphe précédent en pondérant par un facteur ω (ou μ suivant les auteurs) les facteurs de (6.4). On remplace l'itéré calculé selon l'une des méthodes précédentes ($x^{(k+1)}$ dans (6.5)) par $x^{(k+1)}$ qui fait intervenir $x^{(k)}$ via la formule :

$$x^{(k+1)} = \omega x^{(k+1)} + (1-\omega)x^{(k)} \tag{6.5}$$

On utilise en fait une combinaison convexe de l'itéré au rang k et de l'itéré au rang k + 1 pour minimiser les risques de divergence ou d'oscillation.
Chaque méthode vue au paragraphe précédent peut être relaxée, on obtient alors :

Méthode relaxée	Description d'une itération
Jacobi relaxé	$x^{(k+1)} = (1-\omega)I + \omega D^{-1}(E+F)\ x^{(k)} + \omega D^{-1}b$
Gauss-Seidel relaxé	$x^{(k+1)} = (\frac{D}{\omega} - E)^{-1}\ (\frac{1}{\omega} - 1)D + F\ x^{(k)} + (D - \omega E)^{-1}b$
Richardson relaxé	$x^{(k+1)} = (1-\omega)x^{(k)} + \omega(I - A)x^{(k)} + \omega b = (I - \omega A)x^{(k)} + \omega b$

De même pour la décomposition $A = D + L + U$, on a le tableau suivant:

Méthode relaxée	Description d'une itération
Jacobi relaxé	$x^{(k+1)} = (1-\omega)x^{(k)} - \omega D^{-1}(L+U)x^{(k)} + \omega D^{-1}b$
Gauss-Seidel relaxé	$x^{(k+1)} = (1-\omega)(I + \omega D^{-1}L)^{-1}x^{(k)} - \omega(I + \omega D^{-1}L)^{-1}D^{-1}Ux^{(k)} + \omega(D - \omega E)^{-1}D^{-1}b$

Si on considère l'exemple de la méthode de Jacobi, pour lequel la $i-$ème composante de l'itéré $x^{(k+1)}$ est obtenue par :

$$x_i^{(k+1)} = \frac{1}{a_{ii}}\ b_i - \sum_{j=1, j\neq i}^{n} a_{ij}\, x_j^{(k)}\ ;\ i = 1, \ldots, n \tag{6.6}$$

on obtient ainsi facilement la $i-$ème composante de l'itéré $x^{(k+1)}$ de la méthode de Jacobi relaxée, en appliquant la technique de pondération expliquée ci-dessus :

$$x_i^{(k+1)} = \frac{\omega}{a_{ii}}\ b_i - \sum_{j=1, j\neq i}^{n} a_{ij}\, x_j^{(k)}\ + (1-\omega)x_i^{(k)};\ i = 1, \ldots, n \tag{6.7}$$

La littérature abonde en dénominations variées pour les méthodes de relaxation, signifiant parfois toutes la même chose. Ainsi la méthode de Jacobi relaxée est parfois dite méthode de sur-relaxation, ou méthode JOR, pour Jacobi over relaxation.

REMARQUE 6.3.3 Il faut noter que la méthode de Jacobi relaxée est consistante pour $\omega = 0$ et que pour $\omega = 1$, elle coïncide avec la méthode de Jacobi.

Si on considère maintenant le cas de la méthode de Gauss-Seidel, pour lequel la ième composante de l'itéré $x^{(k+1)}$ est obtenue par :

$$x_i^{(k+1)} = \frac{1}{a_{ii}}\ b_i - \sum_{j=1}^{i-1} a_{ij}\, x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij}\, x_j^{(k)}\ ,\ i = 1, \ldots n \tag{6.8}$$

on obtient ainsi facilement la $i$ème composante de l'itéré $x^{(k+1)}$ de la méthode de Gauss-Seidel relaxée, en appliquant la technique de pondération expliquée ci-dessus :

$$x_i^{(k+1)} = \frac{\omega}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)} \right) + (1-\omega) x_i^{(k)}; \; i = 1, \ldots, n \qquad (6.9)$$

Cette méthode est aussi qualifiée de méthode de sur-relaxation successive ou méthode SOR (pour successive over relaxation)

REMARQUE 6.3.4 Il faut noter que la méthode de Gauss-Seidel relaxée est consistante pour $\omega = 0$ et que pour $\omega = 1$, elle coïncide avec la méthode de Gauss-Seidel. Si $\omega \in]0, 1[$, la méthode est dite de sous-relaxation, par contre si $\omega > 1$, elle est qualifiée de sur-relaxation.

6.3.2 Résultats de convergence pour les méthodes de Jacobi et Gauss-Seidel

Il existe des cas où l'on peut établir des résultats de convergence a priori pour les méthodes examinées à la section précédente. Voici deux résultats dans ce sens :

THÉORÈME 6.3.3 Si A est une matrice à diagonale strictement dominante, les méthodes de Jacobi et de Gauss-Seidel sont convergentes.

THÉORÈME 6.3.4 Si A et $2D - A$ sont symétriques définies positives, alors la méthode de Jacobi est convergente et $\rho(J) = \|J\|_A = \|J\|_D$.

Dans le cas de la méthode de Jacobi relaxée, on peut se passer de la condition sur $2D - A$:

THÉORÈME 6.3.5 Quand A est symétrique définie positive, la méthode de Jacobi relaxée est convergente si $0 < \omega < 2/\rho(D^{-1}A)$

En ce qui concerne la méthode de Gauss-Seidel, on a le résultat suivant :

THÉORÈME 6.3.6 Quand A est symétrique définie positive, la méthode de Gauss-Seidel converge de manière monotone pour la norme $\|.\|_A$.

Enfin, si la matrice A est tridiagonale symétrique définie positive, on peut montrer que la méthode de Jacobi est convergente et que

$$\rho(G) = \rho^2(J)$$

où G et J représentent respectivement les matrices d'itérations de la méthode de Gauss-Seidel et de celle de Jacobi.

Dans ce cas, la méthode de Gauss-Seidel converge plus rapidement que celle de Jacobi.

La littérature regorge de résultats de convergence établis sur des familles de matrices correspondant souvent à la résolution d'équations aux dérivées partielles provenant de la physique. Il serait trop long de les citer ou même de les répertorier toutes. Il faut retenir que dans le cas général de matrices quelconques, il n'existe pas de résultat assurant de la convergence des méthodes précitées. Démontrer la convergence pour une matrice ne vérifiant pas des propriétés classiques peut donc devenir très technique, mais appliquer la méthode itérative sans s'être assuré de la convergence peut aussi être assez risqué...

L'exemple ci-dessous montre que l'on ne peut tirer aucune conclusion a priori sur la convergence des méthodes de Jacobi et de Gauss-Seidel.

EXEMPLE 6.3.1 Considérons les sytèmes 3×3 de la forme $A_i x = b_i$. On choisit b_i de manière à ce que la solution du système soit le vecteur unité, et les matrices A_i sont données par :

$$A_1 = \begin{pmatrix} 3 & 0 & 4 \\ 7 & 4 & 2 \\ -1 & 1 & 2 \end{pmatrix} \quad A_2 = \begin{pmatrix} -3 & 3 & -6 \\ -4 & 7 & -8 \\ 5 & 7 & -9 \end{pmatrix}$$

$$A_3 = \begin{pmatrix} 4 & 1 & 1 \\ 2 & -9 & 0 \\ 0 & -8 & -6 \end{pmatrix} \quad A_4 = \begin{pmatrix} 7 & 6 & 9 \\ 4 & 5 & -4 \\ -7 & -3 & 8 \end{pmatrix}$$

On peut vérifier que la méthode de Jacobi ne converge pas pour A_1 ($\rho(J) = 1.33$), contrairement à celle de Gauss-Seidel. C'est exactement le contraire qui se produit pour A_2($\rho(G) = 1.1$). La méthode de Jacobi converge plus lentement que celle de Gauss-Seidel pour la matrice A_3 ($\rho(J) = 0.44$ et $\rho(G) = 0.018$), alors que la la méthode de Jacobi est plus rapide pour A_4 ($\rho(J) = 0.64$ et $\rho(G) = 0.77$).

Concluons par un dernier résultat :

THÉORÈME 6.3.7 Si la méthode de Jacobi converge alors la méthode JOR converge pour $0 < \omega \leq 1$.

6.3.3 Résultats de convergence pour la méthode de relaxation

Lorsque rien n'est précisé par ailleurs, la méthode diet de relaxation est celle de Guass-Seidel relaxée.

Sans hypothèse particulière sur A, on peut déterminer les valeurs de ω pour lesquelles la méthode SOR ne peut pas converger :

THÉORÈME 6.3.8 On a $\rho(G_\omega) \geq |\omega - 1|$, $\forall \omega \in R$. La méthode SOR diverge donc pour tout $\omega \leq 0$ ou $\omega \geq 2$.

Notons que du tableau donné en section (6.3.1.2) on déduit que la matrice d'itération de la méthode relaxée est donnée par :

$$G_\omega = (\frac{D}{\omega} - E)^{-1} \ (\frac{1}{\omega} - 1)D + F \tag{6.10}$$

que l'on peut aussi exprimer sous la forme équivalente :

$$G_\omega = \omega D^{-1}(I - \omega D^{-1}E)^{-1}\frac{D}{\omega}\ (1-\omega)I + \omega D^{-1}F \tag{6.11}$$

soit

$$G_\omega = (I - \omega D^{-1}E)^{-1}\ (1-\omega)I + \omega D^{-1}F \tag{6.12}$$

Si $\{\lambda_i\}$ désigne l'ensemble des valeurs propres de la matrice d'itération de SOR, alors

$$\prod_{i=1}^{n} \lambda_i = \det\ (1-\omega)I + \omega D^{-1}F = |1-\omega|^n$$

Par conséquent, au moins une valeur propre λ_i est telle que $|\lambda_i| > |1-\omega|$. Pour avoir convergence, il est donc nécessaire que $|1-\omega| < 1$, c'est à dire que $0 < \omega < 2$.

Si on suppose A symétrique définie positive, la condition nécessaire $0 < \omega < 2$ devient suffisante pour avoir convergence. On a en effet le résultat suivant :

THÉORÈME 6.3.9 (Propriété d'Ostrowski)
Si A est symétrique définie positive, alors la méthode SOR converge si et seulement si $0 < \omega < 2$. De plus sa convergence est monotone pour la norme $\|.\|_A$.

Enfin :

THÉORÈME 6.3.10 Si A est à diagonale dominante stricte, alors la méthode SOR converge si $0 < \omega \leq 1$.

Les résultats ci-dessus montrent que SOR converge plus ou moins vite selon le choix du paramètre de relaxation ω. On ne peut donner de réponse satisfaisante à la question du choix du paramètre optimal ω pour lequel le taux de convergence est le plus grand, que dans le cas de matrices particulières. On pourra consulter les ouvrages cités en référence pour plus de détails.

6.4 Test d'arrêt

Dans cette section, nous abordons le problème de l'estimation de l'erreur induite par une méthode itérative (au sens défini dans (6.4)). En particulier, on cherche à évaluer le nombre d'itérations k_{min} nécessaire pour que la norme de l'erreur divisée par celle de l'erreur initiale soir inférieur à un ε fixé.

En pratique, une estimation a priori de k_{min} peut être obtenue à partir de (6.3) qui donne la vitesse à laquelle $e^{(k)} \rightarrow 0$ quand k tend vers l'infini. D'après (6.5) on obtient :

$$\frac{e^{(k)}}{e^{(0)}} \leq B^k$$

Ainsi $\|B^k\|$ donne une estimation du facteur de réduction de la norme de l'erreur après k itérations. Typiquement, on poursuit les itérations jusqu'à ce que

$$\|e^{(k)}\| \leq \varepsilon \|e^{(0)}\| \quad \text{avec } \varepsilon < 1 \tag{6.1}$$

Des considérations sur les propriétés des matrices d'itérations et les normes, conduisent à établir que

$$k_{min} \sim -\frac{\log(\varepsilon)}{R(B)} \tag{6.2}$$

où $R(B)$ est le taux de convergence asymptotique défini par

$$R(B) = -\log\rho(B)$$

Cette dernière estimation est plutôt optimiste, mais elle présente en plus le désavantage de nécessiter le calcul de $\rho(B)$, qui peut constituer un problème en lui-même.

De ce fait, plutôt que des estimations a priori du nombre d'itérations nécessaires, on préfère en général utiliser un indicateur facilement évaluable au cours des itérations. On donne ci-après deux exemples.

6.4.1 Un test d'arrêt basé sur l'incrément

D'après la relation de récurrence sur l'erreur $e^{(k+1)} = Be^{(k)}$, on a

$$\|e^{(k+1)}\| \leq \|B\| \|e^{(k)}\| \tag{6.3}$$

En utilisant la relation de récurrence sur l'erreur, on a :

$$\begin{aligned} e^{(k+1)} &= Be^{(k)} \\ &= B\left(e^{(k)} + e^{(k+1)} - e^{(k+1)}\right) \\ &= B\left(e^{(k+1)} + x^{(k)} - x - x^{(k+1)} + x\right) \\ &\quad Be^{(k+1)} - B\left(x^{(k+1)} - x^{(k)}\right) \end{aligned}$$

d'où

$$\|e^{(k+1)}\| \leq \|B\| \left(\|e^{(k+1)}\| + \|x^{(k+1)} - x^{(k)}\|\right)$$

et donc

$$\|x - x^{(k+1)}\| \leq \frac{\|B\|}{1 - \|B\|} \|x^{(k+1)} - x^{(k)}\| \tag{6.4}$$

En particulier, en prenant $k = 0$ dans (6.4) et en appliquant la formule de récurrence (6.3) on obtient aussi l'inégalité :

$$\|x - x^{(k+1)}\| \leq \frac{\|B\|^{k+1}}{1 - \|B\|} \|x^{(1)} - x^{(0)}\|$$

qu'on peut utiliser pour estimer le nombre d'itérations nécessaires à satisfaire la condition $\|e^{(k+1)}\| \leq \varepsilon$, pour une tolérance ε donnée.

En pratique, on peut estimer B comme suit : puisque

$$x^{(k+1)} - x^{(k)} = -\left(x - x^{(k+1)}\right) + \left(x - x^{(k)}\right) = B\left(x^{(k)} - x^{(k-1)}\right)$$

la quantité B est minorée par

$$c = \frac{\delta_{k+1}}{\delta_k}$$

où

$$\delta_{k+1} = \left\| x^{(k+1)} - x^{(k)} \right\|$$

En remplaçant B par c, le membre de droite de (6.4) suggère d'utiliser l'indicateur suivant pour $e^{(k+1)}$

$$\varepsilon^{(k+1)} = \frac{\delta_{k+1}^2}{\delta_k - \delta_{k+1}} \tag{6.5}$$

Il faut prendre garde au fait qu'avec l'approximation utilisée pour B, on ne peut pas voir $\varepsilon^{(k+1)}$ comme un majorant de $e^{(k+1)}$. Néanmoins $\varepsilon^{(k+1)}$ fournit souvent une indication raisonnable du comportement de l'erreur.

6.4.2 Tests d'arrêt fondés sur le résidu

Un autre critère d'arrêt consiste à tester si $\left\| r^{(k)} \right\| \leq \varepsilon$, pour une tolérance ε fixée. Comme

$$\left\| x - x^{(k)} \right\| = \left\| A^{-1}b - x^{(k)} \right\| = \left\| A^{-1}r^{(k)} \right\| \leq \left\| A^{-1} \right\| \left\| r^{(k)} \right\| \leq \left\| A^{-1} \right\| \varepsilon$$

on doit prendre $\varepsilon \leq \frac{\delta}{\left\| A^{-1} \right\|}$ pour que l'erreur soit inférieure à δ.

Il est en général plus judicieux de considérer un résidu normalisé : on interrompt les itérations quand $\left\| r^{(k)} \right\| / \left\| r^{(0)} \right\| \leq \varepsilon$ ou bien quand $\left\| r^{(k)} \right\| / \left\| b \right\| \leq \varepsilon$ (ce qui correspond au choix $x^{(0)} = 0$). Dans ce dernier cas, le test d'arrêt fournit le contrôle suivant de l'erreur relative

$$\frac{\left\| x - x^{(k)} \right\|}{\left\| x \right\|} \leq \frac{\left\| A^{-1} \right\| \left\| r^{(k)} \right\|}{\left\| x \right\|} \leq \kappa(A) \frac{\left\| r^{(k)} \right\|}{\left\| b \right\|} \leq \kappa(A).\varepsilon$$

On retrouve l'influence du conditionnement de la matrice du système, qui même si la tolérance est faible, pour entraîner une erreur importante sur la solution. Avec la technique de préconditionnement par une matrice P, le critère précédent devient

$$\frac{\left\| P^{-1}r^{(k)} \right\|}{\left\| P^{-1}r^{(0)} \right\|} \leq \varepsilon$$

ce qui permet de s'affranchir de l'influence du conditionnement.

En plus, dans [CB] est donné la majoration suivante pour le résidu :

$$x - x^{(k)} \leq \frac{K}{1-K} \; x^{(k)} - x^{(k-1)}$$

avec $K = I - CA$ où $C = \begin{cases} D^{-1}, & \text{si méthode de Jacobi} \\ (D+L)^{-1}, & \text{si méthode de Gauss-Seidel} \end{cases}$

En conclusion et au delà des tests d'arrêt, il y a deux possibilités pour arrêter un algorithme itératif de résolution d'un système d'équations :

(1) Dépassement du'un nombre d'itérations défini d'avance, et

(2) le résidu devient inférieure à un seuil défini d'avance.

6.5 Exercices

EXERCICE 6.1 Soient $A \in R^{2\times 2}$ et $b \in R^2$. La solution du système $A\,x = b$ s'interprète géométriquement comme le point d'intersection de deux droites

$$(D_1) : a_{11}x_1 + a_{12}x_2 = b_1$$
$$(D_2) : a_{21}x_1 + a_{22}x_2 = b_2$$

On suppose que a_{11} et a_{22} sont non nuls.

(1) Calculer les matrices d'itération des méthodes de Jacobi et de Gauss-Seidel associées à ce système.

(2) Calculer les rayons spectraux de ces matrices. Que remarque-t-on ?

(3) Calculer les rayons spectraux des matrices d'itération des méthodes de Jacobi et de Gauss-Seidel associées au système obtenu en permutant les équations ci-dessus.

EXERCICE 6.2 (1) Interpréter géométriquement les méthodes de Jacobi et de Gauss-Seidel appliquées à $A\,x = b$. Pour cette dernière, on notera que le vecteur $x_{k+1} \in R^2$ est solution du système triangulaire

$$a_{11}x_{k+1,1} + a_{12}x_{k,2} = b_1$$
$$a_{21}x_{k+1,1} + a_{22}x_{k+1,2} = b_2$$

EXERCICE 6.3 Soient

$$A = \begin{pmatrix} 1 & 0 & -1/4 & -1/4 \\ 0 & 1 & -1/4 & -1/4 \\ -1/4 & -1/4 & 1 & 0 \\ -1/4 & -1/4 & 0 & 1 \end{pmatrix} = \begin{pmatrix} I_2 & K \\ K & I_2 \end{pmatrix}$$

et

$$b = \begin{pmatrix} 1 \\ \frac{1}{2} \\ 1 \\ 1 \end{pmatrix}$$

(1) Calculer les matrices d'itération des méthodes de Jacobi, Gauss-Seidel et de relaxation associées à A. On les notera J, G et G_ω.

(2) Donner l'expression en fonction de k, b et K des itérés de la résolution du système par la méthode de Jacobi et de Gauss-Seidel quand le point initial est l'origine. Il est recommandé de tirer parti de la structure par blocs de la matrice.

(3) Calculer les rayons spectraux de J et G.

(a) Montrer que si λ est valeur propre de G_ω alors $\lambda = 1 - \omega$ ou bien λ est racine de l'équation

$$\lambda^2 - \left(2(1-\omega) + \frac{\omega^2}{4}\right)\lambda + (1-\omega)^2 = 0$$

(b) Calculer $\rho(G_\omega)$ en distinguant les cas où les racines de l'équation précédente sont réelles ou non.

(c) Trouver la valeur de ω qui rend $\rho(G_\omega)$ minimum.

EXERCICE 6.4 Soit $a \in \mathbb{R}$ et

$$A = \begin{matrix} 1 & a & a \\ a & 1 & a \\ a & a & 1 \end{matrix}$$

Montrer que A est symétrique définie positive si et seulement si $-1/2 < a < 1$ et que la méthode de Jacobi converge si et seulement si $-1/2 < a < 1/2$.

EXERCICE 6.5 Considérons le système d'équations linéaires

$$Ax = b$$

Pour la résolution on applique une méthode itérative

$$x(0) \in \mathbb{R}^n$$
$$Px(k+1) = Nx(k) + b$$

avec $A = P - N$ et $\rho\left(P^{-1}N\right) < 1$.

À cause des erreurs de calcul on a pour la solution itérative

$$(P + \Delta P_{k+1}) \cdot x(k+1) = Nx(k) + b + \Delta b_k$$

où on a noté par x la représentation machine de x, c'est-à-dire le nombre-machine $m(x)$.

(1) Montrer que

$$P \cdot x(k+1) = Nx(k) + b - d_k$$

et évaluer d_k.

(2) Supposons que $x(0) = x(0)$. Montrer que

$$x(k+1) = \left(P^{-1}N\right)^{k+1} x(0) + \sum_{l=0}^{k} \left(P^{-1}N\right)^{l} P^{-1}(b - d_{k-l})$$

(3) Posons $\Delta P_{k+1} \leq c \|P\|, \delta_k = c(\|P\| \|x(k+1)\| + \|N\| \|x(k)\| + \|b\|)$, d'où $d_k \leq \delta_k$. Considérons x^* le point fixe et soit $e(k+1) = x^* - x(k)$ l'erreur de la solution calculée à chaque itération par rapport à la vraie solution. Montrer que

$$e(k+1) \leq \|P^{-1}N\|^{k+1} e(0) + \sum_{l=0}^{k} \|P^{-1}N\|^l \|P^{-1}\| \delta_{k-l}$$

(4) Posons $\theta = \sup_k \frac{x(k)}{x^*}$ et $\varepsilon(A) = \operatorname{argmin}\left\{\varepsilon \cdot \sum_{l=0}^{k} \|P^{-1}N\|^l \|P^{-1}\| - \sum_{l=0}^{k} \|P^{-1}N\|^l \|P^{-1}\| \geq 0\right\}$. Remarquons que nous avons $\sum_{l=0}^{k} \|P^{-1}N\|^l \|P^{-1}\| \leq \|A^{-1}\|$, car $A = P - N = P(I - P^{-1}N)$ d'où $A^{-1} = (I - P^{-1}N)^{-1}P^{-1} = \sum_{l=0}^{\infty} \|P^{-1}N\|^l \|P^{-1}\|$.

Calculer $e(k+1)$.

(5) Calculer $e(k+1)$ pour l'itération de Jacobi.

6.6 Bibliographie

Les ouvrages ci-dessous sont disponibles sous forme de fichier téléchargeable sur le site du cours ou sur Arel

[CB] Algèbre matricielle numérique, Claude Brezinski

[YA] Algèbre linéaire et analyse numérique matricielle, Yves Achdou, téléchargeable à l'adresse http://www.ann.jussieu.fr/~achdou/files/teaching/linalg/book.pdf

[AH1] Analyse numérique matricielle, Cours de 3ème année, Alain Huard, téléchargeable à partir du site de l'INSA Toulouse www-gmm.insa-toulouse.fr

[AH2] Analyse numérique des grands problèmes linéaires, Cours de 4ème année, Alain Huard, téléchargeable à partir du site de l'INSA Toulouse www-gmm.insa-toulouse.fr

Les ouvrages ci-dessous sont disponibles en librairie

[QS] Calcul scientifique, Cours, exercices corrigés et illustrations en Matlab et Octave, Alfio Quarteroni, Fausto Saleri, Springer, 2206.

[QSS] Numerical Mathematics, Alfio Quarteroni, Riccardo Sacco, Fausto Saleri, Springer, 2000.

[AF] Analyse numérique pour Ingénieurs, André Fortin, Presses Internationales Polytechnique, 2001.

[AD] Analyse numérique matricielle, Cours exercices et corrigés, Luca Amodei, Jean-Pierre Dedieu, Dunod, 2008.

[CD] Numerical Analysis, S. D. Conte, Carl de Boor, McGraw-Hill, 1980

[SB] Introduction to numerical ananlysis, Second Edition, J. Stoer, B. Bulirsch, Springer-Verlag, 1993

Pour des logiciels relatifs aux méthodes itératives, on peut se référer à

http://www.netlib.org/ est un dépôt des programmes d'analyse numérique. Il contient aussi TOMS (Transactions on Mathematical Software).

7

MÉTHODES DE DESCENTE

Dans l'esprit des méthodes itératives construites au chapitre précédent, consistant comme on l'a dit en une suite de vecteurs $x^{(0)}, x^{(1)}, ..., x^{(k)}, ...$ qui convergera vers la solution du système linéaire $A\,x = b$ à résoudre, on a cherché à construire des méthodes plus efficaces. En effet, au siècle dernier sont apparus des moyens de calcul permettant d'envisager la résolution de problèmes liés à la physique, notamment par discrétisation d'équations aux dérivées partielles, dans des domaines aussi variés que l'aéronautique, la météorologie ou la médecine. On en verra un exemple dans la première partie de ce chapitre, qui justifie notamment des inconvénients des méthodes vues au chapitre consacré aux méthodes directes, et ayant nécessité des progrès en la matière.

Pour autant, les progrès effectués dans le domaine ne doivent pas faire oublier au lecteur que les méthodes directes restent dans certains cas intéressantes et efficaces.

7.1 Un exemple d'application

Dans l'intention d'expliquer le contexte d'utilisation des méthodes d'optimisation, nous donnons ci-après un exemple tiré des problématiques d'écoulement (thermique, aéronautique ou mécanique des fluides en général).

On se place dans un espace bidimensionnel homéomorphe à R^2 et on considère le problème modèle dit de Dirichlet : Trouver la fonction $u : R^2 \rightarrow R$ satisfaisant l'équation et les conditions aux limites suivantes :

$$-\frac{\partial^2 u}{\partial x^2} - \frac{\partial^2 u}{\partial y^2} = f(x,y), \qquad 0 < x,y < 1 \tag{7.1}$$
$$u(x,y) = 0 \qquad \text{pour } (x,y) \in \partial\Omega$$

où $\partial\Omega$ désigne la frontière du carré unité $\Omega = \{(x,y)/\ 0 < x,y < 1\} \subset R^2$. On suppose que la fonction f est continue sur $\Omega \cup \partial\Omega$. Pour résoudre ce problème, on peut utiliser la méthode des différences finies, qui consiste à remplacer les opérateurs de différenciation par des quotients de type taux d'accroissement pris en des points situés sur une grille recouvrant le domaine $\Omega \cup \partial\Omega$. On introduit des domaines discrétisés Ω_h et $\partial\Omega_h$ définis par :

$$\Omega_h = \{(x_i, y_i)/\ i,j = 1,2,...N\}$$
$$\partial\Omega_h = \{(x_i,0),(x_i,1),(0,y_i),(1,y_i)/\ i,j = 0,1,2,...N+1\}$$

sur lesquels on définit des points

$$x_i = ih \qquad y_i = jh \qquad i,j = 0,1,2,...N+1$$
$$\text{pour } h = \frac{1}{N+1},\ N \geq 1 \text{ et } N \in N$$

En notant u la solution approchée évaluée aux points de la discrétisation, et en utilisant la notation abrégée

$$u(x_i, y_i) = u_{ij} \qquad i,j = 0,1,2,...N+1$$

on montre que la solution approchée vérifie

$$4u_{ij} - u_{i-1,j} - u_{i+1,j} - u_{i,j-1} - u_{i,j+1} = h^2 f_{ij} + h^2 \tau_{ij}, \qquad i,j = 1,2,...N$$
$$u_{0j} = u_{N+1,j} = u_{i0} = u_{i,N+1} = 0, \qquad i = 0,1,...,N+1$$

où f_{ij} désigne $f(x_i, y_i)$ et τ_{ij} désigne l'erreur commise par approximation du premier et du second membre de l'équation (7.1).

Sous l'hypothèse que le paramètre h soit suffisamment petit, on peut montrer que la solution approchée u converge vers la solution exacte u et qu'elle est solution du système d'inconnue z suivant :

$$4z_{ij} - z_{i-1,j} - z_{i+1,j} - z_{i,j-1} - z_{i,j+1} = h^2 f_{ij}, \qquad i,j = 1,2,...N \tag{7.2}$$
$$z_{0j} = z_{N+1,j} = z_{i0} = z_{i,N+1} = 0, \qquad i = 0,1,...,N+1$$

On a donc à résoudre un système à N^2 inconnues (égales aux valeurs de la fonction z aux points intérieurs de la grille), dont le second membre comporte les valeurs $f(x_i, y_i)$. Si on note les inconnues dans un vecteur (c'est à dire que l'on met bout à bout les inconnues correspondant aux points en colonnes dans la grille) et le second membre par :

$$z = ([z_{11}, z_{21}, ..., z_{N1}, z_{12}, z_{22}, ..., z_{N2}, ...z_{NN}]$$
$$b = h^2([f_{11}, f_{21}, ..., f_{N1}, f_{12}, f_{22}, ..., f_{N2}, ...f_{NN}]$$

alors le système (7.2) est équivalent au système matriciel $Az = b$ dont la matrice A est de dimension $N^2 \times N^2$ et définie par :

$$A = \begin{array}{|c|c|c|c|}
\hline
\begin{pmatrix} 4 & -1 & & \\ -1 & 4 & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 4 \end{pmatrix} & \begin{pmatrix} -1 & & & \\ & -1 & & \\ & & \ddots & \\ & & & -1 \end{pmatrix} & & \\
\hline
\begin{pmatrix} -1 & & & \\ & -1 & & \\ & & \ddots & \\ & & & -1 \end{pmatrix} & \begin{pmatrix} 4 & -1 & & \\ -1 & 4 & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 4 \end{pmatrix} & \begin{pmatrix} -1 & & & \\ & -1 & & \\ & & \ddots & \\ & & & -1 \end{pmatrix} & \\
\hline
 & \begin{pmatrix} -1 & & & \\ & -1 & & \\ & & \ddots & \\ & & & -1 \end{pmatrix} & \begin{pmatrix} 4 & -1 & & \\ -1 & 4 & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 4 \end{pmatrix} & \begin{pmatrix} -1 & & & \\ & -1 & & \\ & & \ddots & \\ & & & -1 \end{pmatrix} \\
\hline
 & & \begin{pmatrix} -1 & & & \\ & -1 & & \\ & & \ddots & \\ & & & -1 \end{pmatrix} & \begin{pmatrix} 4 & -1 & & \\ -1 & 4 & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 4 \end{pmatrix} \\
\hline
\end{array}$$

et peut donc aussi s'écrire en tirant parti de sa structure par blocs sous la forme :

$$A = \begin{pmatrix} A_{11} & A_{12} & & 0 \\ A_{21} & A_{22} & \ddots & \\ & \ddots & \ddots & A_{N-1,N} \\ 0 & & A_{N,N-1} & A_{NN} \end{pmatrix}$$

On observe que la matrice A est symétrique et très creuse puisque ses termes non nuls sont concentrés sur 5 vecteurs, ou autrement dit que chaque ligne comportant 5 valeurs non nulles sur N^2 valeurs, et en fait trois valeurs différentes seulement du fait de la symétrie. Sa largeur de bande dépend de h, le paramètre de discrétisation des points sur la grille, ce qui implique que plus on met de points sur la grille, plus la bande est large.

Lors de résolutions numériques, on cherche évidemment à tirer parti de la structure creuse de la matrice. On voit tout de suite que la méthode de Gauss n'est pas très adaptée du point de vue du stockage, car en introduisant des zéros sous la diagonale, on garde la structure de bande mais on accroît le nombre de termes non nuls en remplissant l'espace entre les lignes de valeurs non nulles. Par contre le nombre d'opérations pour effectuer une décomposition de Choleski ($A = LL^*$ puisque A est symétrique), c'est à dire essentiellement le calcul de L, est d'environ $\frac{N^2}{4}$.

Les méthodes de Jacobi et celle de Gauss-Seidel, quant à elles, requièrent à chaque itération $5N^2$ opérations (une opération est une multiplication ou division plus une addition) au lieu de N^4 pour une matrice pleine, à condition de coder une fonction de multiplication matrice-vecteur optimisée tenant compte de la structure bande.

Les méthodes itératives sont donc plus intéressantes du point de vue du stockage mais

moins du point de vue du temps de calcul. De plus on peut montrer que le conditionnement de la matrice A pour la norme 2 vaut $cond_2(A) = \frac{4}{\pi^2 h^2}$ (cf [SB] pour le détail des calculs) et est donc inversement proportionnel à h, ce qui a pour effet de ralentir la vitesse de convergence des méthodes itératives au fur et à mesure que h augmente.

Ces considérations ont donc occasionné l'apparition de différentes méthodes dans l'objectif d'échapper aux inconvénients ci-dessus :

— les méthodes itératives par blocs, tenant compte de la structure de la matrice, et dont nous ne parlerons pas ici,

— des méthodes itératives moins sensibles au conditionnement de la matrice du système, mais conservant les avantages de stockage, auxquelles nous allons consacrer la suite de ce chapitre.

7.2 Résolution d'un système linéaire : un problème d'optimisation

Remarquons que la résolution d'un système d'équations linéaires

$$A x = b \; ; \; A \in R^{n \times n}, x \in R^n \tag{7.1}$$

peut aussi être vu comme un problème de minimisation d'une fonctionnelle dans le cas où la matrice A est symétrique, définie positive.

En effet, considérons la fonctionnelle

$$J(x) = \frac{1}{2} \; x \;\; Ax \;\; - x \;\; b$$

Pour résoudre le système (7.1) on cherche à calculer un vecteur $x \in R^n$ tel que

$$Ax - b = 0$$

Si on prend le gradient de la fonctionnelle on a

$$\nabla J(x) = \frac{1}{2} \; A \;\; + A \;\; x - x \;\; A = Ax - b$$

Par conséquant la solution x du système est aussi solution de $\nabla J(x) = 0$, c'est-à-dire que la solution x minimise la valeur de la fonctionnelle $J(x)$. Il s'agit donc d'un problème d'optimisation sans contraintes.

Dans ce chapitre on commence par présenter les algorithmes qui permettent de trouver une solution au problème de l'optimisation et on termine en appliquant ces algorithmes dans le cas de la résolution d'un système d'équations linéaires.

Formellement pour un problème d'optimisation on considère une fonctionnelle $J : U \subseteq R^n \rightarrow R$ [1] On suppose que J est partout définie sur U.

Le problème de la minimisation de la fonctionnelle J consiste donc au calcul d'un élément $x^* \in R$ qui minimise J, c'est-à-dire tel que

$$J(x^*) \leq J(x) \;\; \forall x \in R$$

[1]Une fonctionnelle est une fonction au sens classique du terme à ceci près que son argument peut être une autre

On peut aussi avoir un problème de maximisation et, en règle générale, on parle des problèmes d'optimisation sans contraintes.

Lors de la résolution d'un problème d'optimisation se posent essentiellement deux questions:

(1) Quelle méthode doit-on utiliser pour calculer x^*?

(2) Comment s'assurer que la valeur calculée est un minimum global et non pas local ?

7.3 Outils mathématiques

Dans la suite du chapitre on utilisera les trois notions suivantes relatives à la fonctionnelle J :

Gradient de J Le gradient de J est

$$\nabla J = \frac{\partial J}{\partial x_1}, \frac{\partial J}{\partial x_2}, \cdots, \frac{\partial J}{\partial x_n}$$

Hessienne de J La matrice hessienne de J est donnée par $H(J) = \nabla \quad J \cdot \nabla J$ à savoir

$$H(J) = \begin{matrix} \frac{\partial^2 J}{\partial^2 x_1} & \frac{\partial^2 J}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 J}{\partial x_1 \partial x_n} \\ \frac{\partial^2 J}{\partial x_2 \partial x_1} & \frac{\partial^2 J}{\partial x_2^2} & \cdots & \frac{\partial^2 J}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 J}{\partial x_{n1} \partial x_1} & \frac{\partial^2 J}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 J}{\partial^2 x_n} \end{matrix}$$

Notons que le déterminant de la hessienne s'appelle le hessien.

Norme de l'énergie C'est le scalaire $x \overset{2}{A} = x \quad Ax$ où l'indice A se refère à la matrice A .

Les deux dernières notions permettent d'introduire un ordre non complet, au sens des formes quadratiques, dans l'ensemble de matrices symétriques de dimension $(n \times n)$. En effet on pose $A \leq B$ si et seulement si $x \overset{2}{A} \leq x \overset{2}{B}$.

Nous avons aussi besoin de la notion de la convexité :

— Un ensemble E est convexe si pour tout $x, y \in E$ on a que le segment fermé $[x, y]$ est dans U.

— Une fonctionnelle J définie sur un ensemble U convexe, est convexe si

$$J(\lambda x + (1-\lambda)y) \leq \lambda J(x) + (1-\lambda) J(y) \; ; \; 0 < \lambda < 1, \forall x, y \in U$$

Notons que la fonctionnelle est concave si la relation de l'inégalité est dans l'autre sens.

— Si J est fortement convexe de rapport a, alors

$$\forall u, v \in U : \quad J(v) \geq J(u) + \nabla J(u) \quad (v-u) + \frac{a}{2} \quad u - v \quad ^2$$

Nous avons deux conditions pour l'optimalité d'un élément de U :

---

fonction, c'est-à-dire une fonctionnelle peut être une fonction de fonction.

— Une condition nécsessaire : Si $x^* \in U$ est un extremum de J et J est dérivable en x^*, alors

$$\nabla J(x^*) = 0$$

Le contraire n'est pas obligatoirement vrai.

— Une condition suffisante : Soit U convexe et deux fois dérivable sur U. Soit $x^* \in U$ un élément tel que $\nabla J(x^*) = 0$. Si $\nabla^2 J(x)$ est symétrique, definie positive pour tout $x \in U$, alors x^* minimise J sur U.
Notons que si $\nabla^2 J(x)$ est symétrique, définie négative, alors x^* maximise J sur U.

Notons aussi que la convexité forte implique immédiatement l'existence et l'unicité d'une solution optimale.

Nous avons la typologie suivante des points de U selon l'optimisation de J :

— $x^* \in U$ est un point critique de J si elle est dérivable sur ce point et $\nabla J(x) = 0$.

— un point critique $x^* \in U$ de J est non dégénéré si $J \in C^2$ sur une boule ouverte de centre x^* dans U et la hessienne $H(x^*)$ est regulière.

Nous terminons cette section par la définition du gradient lipschitzien

— Si J est à gradient lipschitzien de constante L, on a

$$\forall u, v \in U, \quad J(v) - J(u) \leq \nabla J(u)^T (v-u) + \frac{N}{2} \|v-u\|^2$$

7.4 Méthode de descente : formulation générale

Nous commençons par la définition du minimum local.

DÉFINITION 7.4.1 Un minimum local de J sur U est un vecteur x^* tel qu'il existe une boule $B(x^*, r)$ de centre x^* et rayon $r > 0$, avec

$$\forall x \in B(x^*, r) \cap U : J(x) \geq J(u^*)$$

DÉFINITION 7.4.2 Un minimum global de J sur U est un vecteur x^* tel que

$$\forall x \in U : \quad J(x) \geq J(x^*)$$

Cette distinction étant faite, notons que sauf situation très favorable (il n'existe qu'un seul minimum ou bien on est dans le cadre de l'optimisation linéaire, quadratique à matrice positive ou optimisation convexe), nous nous contenterons d'un minimum local de J.

Pour trouver un minimum, les méthodes présentées dans la suite partent du point de vue intuitif suivant : on connaît un point x ainsi que la valeur de $J(x)$. En se déplaçant localement à partir de x, on peut déterminer si J augmente ou diminue. Le déplacement le plus simple étant la ligne droite (dans un espace euclidien), on effectuera donc des petites excursions à partir du x vers différentes directions d, en essayant d'améliorer la valeur du critère J.

DÉFINITION 7.4.3 On appelle direction admissible en x un vecteur d (direction) le long duquel on pourra se déplacer en partant de x tout en restant dans U, c'est-à-dire tel qu'il existe un $h > 0$ de sorte que nous ayons $[x, x + hd] \subset U$. On notera $D(x)$ l'ensemble des directions admissibles en x.

DÉFINITION 7.4.4 Un vecteur $d \in D(x)$ est une direction de descente pour une fonctionnelle J au point $x \in U$ si pour tout réel $\gamma > 0$ il existe un $h \in]0, \gamma[$ tel que $J(x + hd) \leq J(x)$.

Notons que si x est un minimum local de J, il n'existe aucune direction de descente pour J au point x.

L'idée de la méthode de descente est de partir d'un point $x^{(0)} \in U$ tel que $\nabla J\left(x^{(0)}\right) = 0$. Pour calculer le prochain point $x^{(1)} \in U$ on utilise une direction de descente d au point $x^{(0)}$ et on a $x^{(1)} = x^{(0)} + hd$ avec $J\left(x^{(1)}\right) \leq J\left(x^{(0)}\right)$. De cette façon on réduit la valeur de la fonctionnelle J.

La méthode de descente se caractérise par deux choix :

— Le choix de la direction de la descente.
  – Méthode de Cauchy : $d = -\nabla J(x)$. Elle donne naissance aux algorithmes de gradient qui minimisent $\nabla J(x) d = -\|\nabla J(x)\|^2$ qui est la dérivée de la fonctionnelle $J(x + hd)$. Cette méthode est fondée sur le fait qu'en utilisant la formule de Taylor au premier ordre, on peut approcher J au voisinage de $x^{(0)}$ par la fonction

$$J\left(x^{(1)}\right) = J\left(x^{(0)}\right) + \nabla J\left(x^{(0)}\right) \cdot \left(x^{(1)} - x^{(0)}\right) + O\left(\nabla J\left(x^{(0)}\right)\right)$$

  Considérons la droite passant par $x^{(0)}$ le long du gradient de J en $x^{(0)}$:

$$x(h) = x^{(0)} - h \times \nabla J\left(x^{(0)}\right)$$

  Si h est choisi positif, alors, en posant $x^{(1)} = x(h)$

$$J\left(x^{(1)}\right) = J\left(x^{(0)}\right) - h\left\|\nabla J\left(x^{(0)}\right)\right\|^2 + O\left(h\left\|\nabla J\left(x^{(0)}\right)\right\|\right)$$

  et si h est suffisamment petit, on aura

$$J\left(x^{(1)}\right) < J\left(x^{(0)}\right)$$

  En d'autres termes, la direction opposée à celle du gradient est une direction de descente. C'est la meilleure direction localement (c'est-à-dire pour h petit). De ce fait, la méthode de Cauchy fournit à chaque itération une direction de descente. Par contre sa convergence est lente.
  – Méthode de Newton : $d = -H^{-1}(J(x))\nabla J(x)$, qui est une direction de descente si $H(J(x))$ est définie positive. De ce fait la méthode de Newton ne fournit pas obligatoirement une direction de descente. Par contre, si elle converge, sa convergence est plus rapide que celle de la méthode de Cauchy.

— Choix du pas h qui doit être défini de sorte que le nombre d'itérations soit minimal. Il y a deux techniques :

– Pas fixe.- On choisit un pas fixe pour l'ensemble des itérations.

– Pas optimal.- Le pas est choisi à chaque itération de sorte que la fonction J $x^{(0)} + \gamma d$ soit minimale.

7.5 Algorithme du gradient à pas fixe

Cet algorithme utilise la méthode de Cauchy avec un pas h constant et fixé d'avance.

ALGORITHME DU GRADIENT À PAS FIXE (DE PLUS GRANDE PENTE)

— En partant de $x^{(0)}$, on calcule la direction de descente

$$d_0 = -\nabla J \quad x^{(0)}$$

et le nouveau point $x^{(1)}$

$$x^{(1)} = x^{(0)} + h \times d_0$$

— À l'étape k, connaissant $x^{(k)}$, on calcule la direction de la descente

$$d_k = -\nabla J \quad x^{(k)}$$

et le nouveau point $x^{(k+1)}$ par

$$x^{(k+1)} = x^{(k)} + h \times d_k$$

— On décide d'arrêter l'algorithme lorsqu'un test de convergence

$$\nabla J \quad x^{(k)} \quad < \theta = \text{seuil}$$

est vérifié.

Pour la convergence, nous avons le théorème suivant :

THÉORÈME 7.5.1 Supposons que

H_1 la fonctionnelle J est de classe C dans U;

H_2 l'ensemble $U_0 = \quad x \in U \quad J(x) \leq J \quad x^{(0)} \quad \subset U$ est fermé;

H_3 $\forall x \in U_0$ on a $c \cdot I \leq \nabla^2 J(x) \leq C \cdot I$, où I la matrice identité, et

H_4 Le pas h est choisi tel que $h < \frac{2}{C}$.

Alors l'algorithme du gradient à pas fixe converge vers un minimum local $x^* \in U$ non dégénéré.
De plus, nous avons

$$x^* - x^{(0)} \quad \leq \frac{\nabla J \quad x^{(0)}}{c}$$

7.6 Algorithme du gradient à pas variable

Cet algorithme effectue à chaque itération, étant donnée la direction de descente $d = \nabla J(x)$, le calcul d'un pas h qui minimise la fonctionnelle $J(x + hd)$.

ALGORITHME DU GRADIENT À PAS OPTIMAL

— En partant de $x^{(0)}$, on calcule la direction de descente

$$d_0 = -\nabla J\left(x^{(0)}\right)$$

le pas optimal

$$h_0 = \arg\min_{h>0} J\left(x^{(0)} - h \times \nabla J\left(x^{(0)}\right)\right)$$

et le nouveau point $x^{(1)}$

$$x^{(1)} = x^{(0)} + h \times d_0$$

— À l'étape k, connaissant $x^{(k)}$, on calcule la direction de la descente

$$d_k = -\nabla J\left(x^{(k)}\right)$$

le pas optimal

$$h_k = \arg\min_{h>0} J\left(x^{(k)} - h \times \nabla J\left(x^{(k)}\right)\right)$$

et le nouveau point $x^{(k+1)}$ par

$$x^{(k+1)} = x^{(k)} + h \times d_k$$

— On décide d'arrêter l'algorithme lorsqu' un test de convergence

$$\left\|\nabla J\left(x^{(k+1)}\right)\right\| < \theta = \text{seuil}$$

est vérifié.

7.6.1 Convergence de l'algorithme

Habituellement on prend $\theta = 10^{-6} \times \left\|\nabla J\left(x^{(0)}\right)\right\|$. Ce test évite d'accumuler des itérations qui n'apportent plus rien à la qualité de la solution trouvée. Il ne donne en revanche aucune garantie pour l'optimalité de la solution trouvée. En particulier il faut vérifier que le point $x^{(k+1)}$ correspondant est bien un minimum, car il se peut qu'il soit un point selle.

FAIT 7.1 L'algorithme du gradient à pas optimal a les propriétés suivantes :

— Deux directions de recherche successives sont orthogonales. En effet, dans le calcul de h_k, si la minimisation est exacte (c'est à dire si l'on trouve exactement le meilleur h), la condition nécessaire d'optimalité de la recherche linéaire s'écrit

$$\frac{d}{dh} J\left(x^{(k)} - h \times \nabla J\left(x^{(k)}\right)\right)\Big|_{h=h_k} = -\nabla J\left(x^{(k+1)}\right) \cdot \nabla J\left(x^{(k)}\right) = 0.$$

— Dans le cas où J est quadratique, de la forme $J(u) = \frac{1}{2} u\ Au - b\ u$ avec A symétrique définie positive, on peut calculer facilement le paramètre h_k. Il est donné par l'equation $\nabla J\left(x^{(k+1)}\right) \cdot \nabla J\left(x^{(k)}\right) = 0$ ou encore

$$A x^{(k+1)} - b \quad A x^{(k)} - b = A\left(x^{(k)} - h_k\right) A x^{(k)} - b - b \quad A x^{(k)} - b = 0.$$

Nous considérons maintenant la convergence de la méthode du gradient à pas optimal dans le cas quadratique. Nous avons le théorème suivant :

THÉORÈME 7.6.1 Si $J(x) = \frac{1}{2} x\ Ax - b\ x$ avec A symétrique définie positive, la méthode du gradient à pas optimal converge vers l'optimum unique $x^* = A^{-1} b$. De plus

$$x^{(k+1)} - x^* < x^{(k+1)} - x^* \quad {}^2_A \quad \frac{\kappa(A) - 1}{\kappa(A) + 1}^2$$

où $\kappa(A)$ est le conditionnement de A relativement à la norme 2.

Nous avons aussi le lemme et le théorème suivants :

LEMME 7.6.1 (INÉGALITÉ DE KANTOROVICH) Si A est une matrice symétrique définie positive d'ordre n, avec $\kappa(A) = \frac{\lambda_+}{\lambda_-}$, où λ_+ et λ_- la plus grande et la plus petite respectivement valeurs propres, alors :

$$\frac{x^{\ 2}}{x\ {}^2_A \times x\ {}^2_{A^{-1}}} \geq \frac{4\kappa(A)}{(1+\kappa(A))^2}$$

THÉORÈME 7.6.2 Si J est continûment différentiable et fortement convexe, alors l'algorithme de gradient à pas optimal converge vers l'unique optimum.

7.7 Méthode de Newton

Le principe de la méthode de Newton pour l'optimisation est de minimiser successivement les approximations au second ordre de la fonctionnelle J :

ALGORITHME DE NEWTON

— Soit $x^{(0)} \in U$. Par un développement de Taylor au second ordre au voisinage de $x^{(0)}$, on obtient :

$$J^0(x) = J\ x^{(0)} + \nabla J\ x^{(0)}\ x - x^{(0)} + \frac{1}{2}\ x - x^{(0)}\ J\ x^{(0)}\ x - x^{(0)}$$

On minimise la fonctionnelle quadratique $J^0(x)$, ce qui fournit un vecteur $x^{(1)}$ qui est solution du système linéaire

$$H\ J\ x^{(0)}\ x^{(1)} = H\ J\ x^{(0)}\ x^{(0)} - \nabla J\ x^{(0)}$$

et qui s'écrit comme suit : $x^{(1)} = x^{(0)} - H\ J\ x^{(0)}\ ^{-1} \nabla J\ x^{(0)}$

— À l'itération k, on construit J^k, approximation quadratique de J au voisinage de $x^{(k)}$, que l'on minimise pour obtenir $x^{(k+1)}$, défini par

$$H\ J\ x^{(k)}\ \delta_k = -\nabla J\ x^{(k)}\ , \quad x^{(k+1)} = x^{(k)} + \delta_k . \tag{7.1}$$

On notera le parti pris qui consiste à ne pas écrire $x^{(k+1)} = x^{(k)} - H\ J\ x^{(k)}\ ^{-1} \nabla J\ x^{(k)}$ dans l'algorithme. En effet, cette syntaxe sous-entendrait que l'on procède à l'inversion $H\ J\ x^{(k)}$ ce qui n'est absolument pas nécessaire.

7.7.1 Propriétés de l'algorithme de Newton.

L'algorithme de Newton est la généralisation multi-dimensionnelle de la méthode Newton-Raphson, appliquée à la recherche des racines de $G(x) = \nabla J(x)$. On pourrait démontrer la convergence dans tout voisinage d'un minimum local, ainsi qu'une vitesse de convergence quadratique. Cette méthode fonctionne très bien pour des problèmes de petites dimensions (quelques dizaines de variables), lorsque le calcul de la hessienne $H(J)$ est facile. Dans les autres cas, on préferera souvent une méthode de gradient conjugué (cf. infra) ou une méthode de quasi-Newton[2].

Dans le cas quadratique, elle fournit évidemment la solution du problème en une itération : $x^{(1)} = A^{-1}b$. Mais ceci est bien sûr une illusion puisqu'il reste à résoudre le système linéaire $Ax = b$, c'est à dire $Ax^{(1)} = b$ qui constitue le plus gros du travail dans ce cas. Dans le cas général, pour résoudre le système $H\ J\ x^{(k)}\ \delta_k = -\nabla J\ x^{(k)}$, lorsque $H\ J\ x^{(k)}$ est définie positive, la factorisation LU est la mieux adaptée.

La méthode de Newton peut aussi bien fournir des maxima locaux ou des points-selle, puisqu'elle cherche seulement à satisfaire la condition nécessaire d'optimalité $\nabla J(x) = 0$. En d'autres termes, la direction de Newton, δ_k n'est pas forcément une direction de descente ! Si $H\ J\ x^{(k)}$ est définie positive, alors $\nabla J\ x^{(k)}\ \delta_k = -\nabla J\ x^{(k)}\ ^2_{H(J(x^{(k)}))} < 0$ pour $\nabla J\ x^{(k)} = 0$. Si $H\ J\ x^{(k)}$ n'est pas définie positive, il existe des modifications qui permettent d'avoir

[2] Cette méthode dépasse le cadre de ce cours

une convergence globale alliant les avantages de la vitesse de convergence quadratique près de la solution et ceux d'une méthode de descente.

7.8 Méthode de gradient conjugué

Soit J la fonctionnelle quadratique $J(u) = \frac{1}{2} u \; Au - b \; u$. Nous aurons besoin de la définition suivante :

DÉFINITION 7.8.1 A étant une matrice symétrique donnée, deux directions d_0 et d_1 sont dites conjuguées par rapport à A si $(d_0) \; A d_1 = 0$.

Le lemme suivant est aussi utile.

LEMME 7.8.1 Si A est symétrique définie positive, et si les vecteurs non nuls $(d_1, \cdots, d_k)$ sont conjugués deux à deux par rapport à A, alors ils forment une famille libre.

Ce résultat s'applique d'une manière très astucieuse : si l'on note x^* la solution (unique) de $Ax = b$, et si l'on dispose d'une famille de n vecteurs conjugués par rapport à A, alors, la famille étant une base de R^n, on peut exprimer x^* dans cette base :

$$x^* = \sum_{j=0}^{n-1} h_j d_j$$

On calcule aisément les coefficients h_i en multipliant cette équation à gauche par $d_i \; A$, d'où

$$d_i \; Ax^* = h_i \times \; d_i \; Ad_i \; = h_i \times \; d_i \; {}^2_A$$

Comme $Ax^* = b$, on peut donc calculer les h_i uniquement à partir des données du problème, A et b :

$$h_i = \frac{d_i \; b}{d_i \; {}^2_A}.$$

Nous avons ainsi mis en evidence une méthode directe de calcul de x^*. De plus, cette méthode nécessite au plus n étapes (trouver les directions conjugués d_i) pour la fonctionnelle quadratique J. La seule difficulté est donc de construire successivement des directions conjuguées. Comme on l'a vu plus haut, les vecteurs propres de A constituent une solution, mais c'est une solution onéreuse. Essayons de mettre en œuvre une méthode plus simple, qui optimise J en même temps qu'elle construit de nouvelles directions conjuguées :

— Fixons un $x^{(0)}$, et choisissons $d_0 = \nabla J(x^{(0)})$.

— Notons $x^{(1)}$ le vecteur obtenu par application de la méthode du gradient à pas optimal, le long de $x^{(0)}$:

$$x^{(1)} = x^{(0)} - h_0 d_0$$

Remarquons que h_0 a été calculé plus haut. Il est donné par,

$$h_0 = \frac{\nabla J\ x^{(0)}\ ^2}{\nabla J\ x^{(0)}\ ^2_A}$$

ce que l'on écrira

$$h_0 = \frac{d_0\ \nabla J\ x^{(0)}}{d_0\ ^2_A}$$

On dispose maintenant de $\nabla J(x^{(1)})$ (orthogonal à $\nabla J(x^{(0)})$).

— Recherchons une nouvelle direction d_1 comme combinaison, linéaire de d_0 et $\nabla J\ x^{(1)}$, qui soit conjuguée de d_0 par rapport à A [3].
Comme on désire que d_1 et d_0 soient conjugués, nous avons :

$$0 = d_0\, A\, d_1 = d_0\, A\, \nabla J\ x^{(1)} - \beta_0 d_0\, A\, d_0 \Rightarrow \beta_0 = \frac{d_0\, A\, \nabla J\ x^{(1)}}{d_0\, A\, d_0}.$$

ce qui permet d'évaluer une nouvelle quantité β.

— Nous pouvons maintenant optimiser J le long de la direction d_1, ce qui fournit

$$x^{(2)} = x^{(1)} - h_1 d_1, \qquad \text{avec } h_1 = \frac{d_1 \cdot \nabla J\ x^{(1)}}{d_1\ ^2_A}$$

et on recherche une nouvelle direction d_2, de forme

$$d_2 = \nabla J\ x^{(2)} - \beta_1 d_1$$

qui soit conjuguée par rapport à d_0 et d_1, etc.

Le fait que d_2 soit conjuguée avec d_1 s'impose naturellement, comme précédemment, en choisissant β_1 ad hoc :

$$\beta_1 = \frac{d_1\, A\, \nabla J\ x^{(2)}}{d_1\, A\, d_1}$$

Il est plus étonnant que d_2 ainsi définie soit conjuguée avec d_0. Cette propriété provient du fait que J est quadratique.

[3] ces deux vecteurs sont les seuls dont on dispose, pour l'instant, et qui fournissent de l'information sur la fonctionnelle

ALGORITHME DE GRADIENT CONJUGUÉ

— On choisit $x^{(0)}$ et on calcule la direction de descente

$$d_0 = \nabla J\ x^{(0)}$$

— À l'itération $k \geq 1$, on calcule la valeur du pas

$$h_k = \frac{d_{k-1} \cdot \nabla\ J\ x^{(k-1)}}{d_{k-1} H\ J\ x^{(k-1)}\ d_{k-1}}$$

le nouveau point

$$x^{(k)} = x^{(k-1)} - h_k d_{k-1}$$

et la nouvelle direction, conjuguée aux précédentes

$$d_k = \nabla J\ x^{(k)} - \frac{H\ J\ x^{(k)}}{H\ J\ x^{(k-1)}} d_k$$

— On décide d'arrêter l'algorithme lorsqu' un test de convergence

$$\nabla J\ x^{(k)} < \theta = \text{seuil}$$

est vérifié.

Cette méthode est donc à peine plus compliquée à mettre en œuvre que la méthode du gradient à pas optimal, mais elle converge en n itérations lorsque J est quadratique. On peut aussi considérer la méthode du gradient conjugué comme une méthode de relaxation appliquée à J dans le système de coordonnées induit par les vecteurs propres de A . La recherche de directions conjuguées est alors simplement une façon de décomposer J par rapport à ces nouvelles coordonnnées. L'algorithme de gradient conjugué admet plusieurs formulations différentes qui sont équivalentes pour les fonctions quadratiques mais peuvent avoir des caractéristiques différentes si J est plus générale.

7.9 Application à la résolution d'un système linéaire

Nous avons vu au début de ce chapitre que la solution d'un système linéaire $A x = b$ peut être vue comme la minimisation d'une fonctionnelle J associée à A et b. La résolution du système devient alors un problème d'optimisation qu'on peut traiter itérativement en utilisant les algorithmes de gradient. Nous donnons ci-après l'algorithme du gradient conjugué dans le cas d'un système linéaire.

Soit une matrice carrée A de dimension n, symétrique définie positive et b un vecteur fixé. On pose

$$J(x) = \frac{1}{2}\ x\ Ax - b\ x$$

J

ALGORITHME DE DE GRADIENT CONJUGUÉ APPLIQUÉ À LA RÉSOLUTION D'UN SYSTÈME LINÉAIRE

(1) On choisit un vecteur initial $x^{(0)} \in R^n$. On calcule le résidu

$$r_0 = A x^{(0)} - b$$

et la direction

$$d_0 = r_0$$

— À l'itération $k \geq 1$, on calcule la valeur du pas

$$h_k = \frac{d_{k-1} \cdot r_{k-1}}{d_{k-1} A d_{k-1}}$$

le nouveau point

$$x^{(k)} = x^{(k-1)} + h_k d_{k-1}$$

le résidu

$$r_k = r_{k-1} - h_k A d_{k-1}$$

et la nouvelle direction, conjuguée aux précédentes

$$d_k = r_k - \frac{(A d_{k-1}) \cdot r_k}{(A d_{k-1}) \cdot d_{k-1}}$$

— On décide d'arrêter l'algorithme lorsqu' un test de convergence

$$r_k < \theta = \text{seuil}$$

est vérifié.

On peut démontrer que l'algorithme de gradient conjugé converge en n itérations si A est symétrique, définie positive. Néanmoins il faut faire attention aux erreurs dues à la précision de la machine. Plus la valeur de n est grande, plus les erreurs de calcul s'accumulent avec comme conséquence que les directions successives ne soient pas conjuguées. Dans ce cas il est possible que la méthode ne converge pas au bout de n itérations.

Notons, pour finir, que la contrainte d'avoir une matrice A symétrique, définie positive, afin que les méthodes de gradient puissent s'appliquer, n'est pas limitative. En effet, considérons le système (7.1)

$$A x = b \tag{7.1}$$

avec A matrice carrée, régulière, quelconque. On peut appliquer les méthodes de gradient au système

$$A \quad A y = A \quad b \tag{7.2}$$

parce que la matrice A A est symétrique, définie positive. On obtient donc la solution

$$y = \quad A \quad A \quad ^{-1} A \quad b = A^{-1} b = x$$

c'est-à-dire la solution du système (7.2) est la même que celle du système (7.1).

7.10 Exercice

EXERCICE 7.1 Soit la fonctionnelle

$$J(x) = \frac{1}{2} x \quad A x + x \quad b ; A \in R^n$$

avec A matrice symétrique définie positive avec valeurs propres $\lambda_1 \geq \ldots \geq \lambda_n > 0$.

(1) Calculer x^* le point qui minimise $J(x)$, ainsi que la valeur de $J(x^*)$.

(2) Calculer pour l'algorithme à pas fixe, la direction d_k à la k-ième itération.

(3) Calculer $x^{(k+1)}$ et $J \quad x^{(k+1)}$ pour l'algorithme à pas fixe.

(4) Il s'agit maintenant de trouver un pas h pour cette itération qui optimise le critère, c'est-à-dire qui minimise la valeur de $J \quad x^{(k+1)}$.

Calculer la valeur de ce pas h.

(5) En utilisant la nouvelle valeur calculée de h, évaluer le prochain point $x^{(k+1)}$ et la valeur de $J \quad x^{(k+1)}$ en fonction de $x^{(k)}$ et d_k.

(6) Montrer qu'on a pour le taux de convergence

$$\frac{J \quad x^{(k+1)} \quad - J(x^*)}{J \quad x^{(k)} \quad - J(x^*)} = 1 - \frac{1}{\beta}$$

où

$$\beta = \frac{d_k A d_k \quad d_k A^{-1} d_k}{d_k \quad d_k^{\ 2}}$$

(7) Sachant que

$$\beta \leq \frac{(\lambda_1 + \lambda_n)}{4 \lambda_1 \lambda_n}$$

montrer pour le taux de convergence l'inégalité suivante :

$$\frac{J \quad x^{(k+1)} \quad - J(x^*)}{J \quad x^{(k)} \quad - J(x^*)} \leq \quad \frac{\kappa(A) - 1}{\kappa(A) + 1}^{\ 2}$$

où $\kappa(A)$ est le conditionnement de la matrice selon la norme 2.

(8) Application.- Calculer une borne supérieure du taux de convergence pour le système

$$\begin{matrix} 20 & 5 \\ 20 & 2 \end{matrix} \quad x = \begin{matrix} 14 \\ 6 \end{matrix}$$

Remarques.

7.11 Références

Les livres qui sont pris en compte pour la redaction de ce chapitre sont les suivants :

E. ANGELINI : Polycopié sur l'optimisation, ENST,
http:/ / perso.telecom-paristech.fr/ angelini/ master_spsiv/ optimization/
D. P. BERTSEKAS, J. N. TSITSIKLIS : Parallel and distributed computation, Prentice-Hall, 1989
C. BREZINSKI : Projection methods fot systems of equations, Elsevier, 1997
P. G. CIARLET : Introduction à l'analyse numérique matricielle et à l'optimisation, Masson, 1988
P. G. CIARLET, B. MIARA, J.M. THOMAS : Exercices d'analyse numérique matricielle et d'optimisation, 2e édition, Masson, 1987
F. R. GANTMACHER : Théorie des matrice, tome 1, Théorie générale, Dunod, 1966
C. D. MEYER : Matrix analysis and applied linear algebra, SIAM, 2001
R. D. MILNE : Applied functional analysis, Pittman, 1980
A. QUARTERONI, R. SACCO, F. SALERI : Numerical mathematics, Springer, 2000

8

CHOIX DES MÉTHODES DE RÉSOLUTION DE SYSTÈMES LINÉAIRES ET PRÉCONDITIONNEMENT

8.1 Introduction

Le fait de disposer de différentes méthodes de résolution d'un système linéaire est évidemment un avantage, mais cela ne résout pas la question qui se pose au néophyte : quelle méthode choisir dans un cas précis ? Au risque de ne pas rassurer le lecteur, on peut affirmer qu'il n'existe pas de méthode meilleure que les autres dans l'absolu, il n'y a que des cas particuliers dans lesquels le choix devra s'orienter vers telle méthode plutôt que telle autre. L'ingénieur aura à charge de déterminer les caractéristiques particulières de la matrice ou de la solution cherchée, afin de choisir une méthode adéquate. En cela il pourra s'aider des points de repères fournis dans le premier paragraphe de ce chapitre. Dans les cas difficiles, ou simplement lorsqu'une amélioration de la convergence est nécessaire, il pourra avoir recours aux techniques de préconditionnement abordées dans le second paragraphe.

Notons que la caractérisation des propriétés d'une matrice fait appel à des notions d'algèbre linéaire, et que la maîtrise de cette matière sera donc particulièrement utile (encore une fois).

8.2 Ce qui se voit à l'œil nu

Avant d'aller plus loin, il semble bon de rappeler que la condition sine qua non dont on doit s'assurer avant tout, est que la matrice soit carrée et régulière. Dans tous les cas contraires (non carrée et/ ou non régulière) on se reportera à la décomposition en valeurs singulières et aux méthodes de moindres carrés vues dans la suite de ce cours. On considérera donc ici dorénavant que la matrice du système est carrée et régulière.

Dans le choix de la méthode de résolution d'un système linéaire le premier paramètre à prendre en compte est la taille du système, n. Dans le cas des systèmes de petite taille, disons $n < 10$ pour fixer les idées, toutes les méthodes feront l'affaire en général, sauf cas particuliers rares. On peut indifféremment ou presque, choisir une méthode directe ou itérative, sachant que dans la mesure où on peut accéder à une solution "exacte" en un nombre fini d'opérations il ne faut pas s'en priver : on privilégiera donc les méthodes directes en général.

Dans le cas des systèmes de grande taille, il en va tout autrement. Le choix de la méthode doit tenir compte de différents facteurs :

- les propriétés de la matrice : symétrie, définie positivité, structure creuse ou pleine, conditionnement,
- les besoins de l'utilisateur : précision, rapidité, parallélisation,
- les moyens de calcul disponibles : accès mémoire, processeurs rapides,
- le temps de développement : de nombreuses librairies existent et il est souvent plus intéressant de bien choisir une méthode déjà programmée et testée que de développer son propre code.

Nous donnons ci-après quelques éléments de choix relatifs au premier point, qui relève du périmètre de l'analyse numérique. Le second et le troisième seront relatifs au contexte technique, le dernier au contexte du projet et à son budget.

8.2.1 Systèmes linéaires creux

De nombreux problèmes de résolution d'équations aux dérivées partielles conduisent à des matrices creuses. On en a donné un exemple dans le chapitre précédent. Comme on l'a déjà dit, lors de résolutions numériques, on cherche évidemment à tirer parti de la structure creuse de la matrice. On voit tout de suite que la méthode de Gauss n'est pas très adaptée du point de vue du stockage, car en introduisant des zéros sous la diagonale, on garde la structure de bande mais on accroît le nombre de termes non nuls en remplissant l'espace entre les lignes de valeurs non nulles. Par contre le nombre d'opérations pour effectuer une décomposition de Cholesky ($A = LL$ lorsque A est symétrique), c'est à dire essentiellement le calcul de L, est d'environ $\frac{n^2}{4}$, où n l'ordre de la matrice.

Les méthodes de Jacobi et celle de Gauss-Seidel, quant à elles, requièrent à chaque itération $5n^2$ opérations (une opération est une multiplication ou division plus une addition) au lieu de n^4 pour une matrice pleine, à condition de coder une fonction de multiplication matrice-vecteur optimisée tenant compte de la structure bande.

Les méthodes itératives sont donc plus intéressantes du point de vue du stockage mais moins du point de vue du temps de calcul.

8.2.1.1 Cas des largeurs de bande constantes

Matrices symétriques

On se limite dans un premier temps aux matrices symétriques car les problèmes qui mènent à la résolution de matrices bandes, sont souvent également générateurs de matrices symétriques. Si la largeur de bande est constante en fonction de la taille, et faible vis à vis de cette dernière, il est intéressant de rester sur une méthode de Cholesky (A est décomposée sous la forme $A = LDL^t$), car on peut montrer facilement (cf exercice ci-dessous) que le profil de A est conservé (i.e. la largeur de bande de la partie inférieure de A est la même que celle de la partie inférieure de L). Par contre il est à noter que dans le cas de matrices non symétriques, la largeur de bande de L ne se transmet à U.

Du fait de la propriété de conservation de la largeur de bande, on gagnera sur le stockage et sur le nombre d'opérations. On aura intérêt dans ce cas, à utiliser un stockage particulier, dit "stockage profil". Il fait appel à la notion de profil, définie ci-après :

DÉFINITION 8.2.1 Le profil d'une matrice symétrique A est $\{(i,j), 1 \leq i \leq n, j_i \leq j \leq i\}$ où j_i est l'indice de la colonne du premier élément non nul de la ligne i.

L'intérêt de ce rangement réside dans le fait que le profil de L est inclus dans celui de A. Ainsi, si l'on réserve une quantité de place mémoire suffisante pour stocker le profil de A, on pourra ranger au fur et à mesure les coefficients de la matrice L dans cette place. Les éléments diagonaux de L valant 1 n'ayant pas besoin d'être stockés, les éléments de D seront rangés à la place des coefficients diagonaux de A. On constate alors le double avantage de ce rangement : d'une part on fait une grande économie de place mémoire (à condition que le profil de A soit assez petit), d'autre part la gestion des données reste assez simple : il suffit de stocker le profil de A sous forme d'un tableau, la taille et la forme de ce tableau n'évoluant pas au cours de la factorisation.

EXERCICE 8.1 En exprimant le terme général L_{ij} de la matrice L de la décomposition de Cholesky en fonction de celui de A, de termes de D et des termes L_{ik} et L_{jk} pour k inférieur à j, montrer par récurrence que si A est tridiagonale alors la matrice L l'est aussi.

Matrices non symétriques

DÉFINITION 8.2.2 Soit A une matrice de taille n. On appelle largeur de bande inférieure (resp. supérieure) de la matrice A, l'entier q (resp. p) tel que

$$\forall i = 1,...n, \forall j = 1,..,i-q, \quad A_{ij} = 0$$
$$\forall j = 1,..,n, \forall i = 1,...j-p, \quad A_{ij} = 0$$

Si la matrice A est symétrique alors la largeur de bande inférieure est égale à la largeur de bande supérieure

On peut montrer aussi dans le cas de matrices non symétriques, que le profil se conserve :

THÉORÈME 8.2.1 On suppose que A est une matrice de taille n, possédant une factorisation LU. Si A est de largeur de bande supérieure q, et de largeur de bande inférieure p, alors U est de largeur de bande q et L de largeur de bande p.

On pourra trouver la démonstration de ce théorème dans [GV].

Il est important de noter que dans le cas où n p et n q alors on peut facilement établir un algorithme de décomposition LU revenant à 2npq opérations.

EXERCICE 8.2 Modifier l'algorithme de la factorisation LU pour tenir compte du cas d'une matrice A de largeur de bande supérieure q, et de largeur de bande inférieure p.
De même, modifier les algorithmes de descente et de remontée permettant la résolution du système A x = b.

8.2.1.2 Cas des largeurs de bande non constantes

Nous nous limitons au cas de matrices symétriques, possédant de nombreuses valeurs nulles mais ne présentant pas de structure bande. Dans ce cas, il serait évidemment maladroit de ne pas tenir compte des coefficients nuls, mais la notion de bande ne le permet pas. On utilise alors par exemple une représentation des valeurs non nulles sous forme de graphe, puis une renumérotation des sommets du graphe permet de transformer la matrice en une matrice de profil minimal.

À la matrice A de taille n, associons un graphe G = g(A) défini par :

— Les sommets de G sont numérotés de 1 à n.
— Les sommets i et j de G sont reliés si et seulement si A_{ij} est non nul.

Inversement, nous pouvons associer à tout graphe G une matrice A telle que G = g(A).

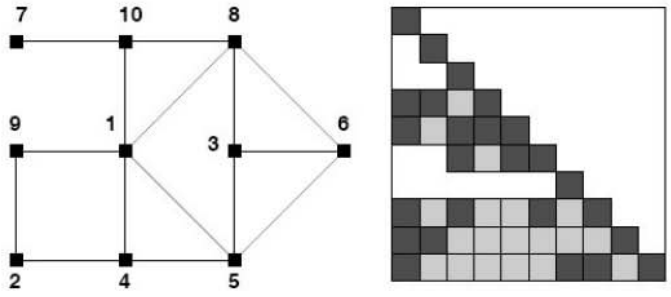


Figure 8.1: Numérotation initiale

Le graphe g(A) traduit en fait les relations entre les inconnues du système : les inconnues x_i et x_j sont en effet reliés si $A_{ij} = 0$, c'est-à-dire si les sommets i et j sont reliés. Si nous renumérotons les sommets sans changer la structure du graphe, nous ne changeons pas le système, mais seulement l'ordre dans lequel apparaissent les inconnues. Il faut donc trouver une numérotation optimale du graphe, c'est-à-dire une numérotation dont la matrice associée a un profil de dimension minimale.

L'algorithme suivant est dû à Cuthill et Mc Kee, il permet de trouver de manière heuristique une bonne numérotation du graphe.

Algorithme (Cuthill-Mc Kee direct)

(1) On choisit un premier sommet.

(2) On numérote ses voisins.

(3) On numérote les voisins non encore numérotés du numéro 2, puis du numéro 3, . . .

À chaque étape, lorsqu'il y a plusieurs sommets à numéroter, on numérote en premier les sommets qui ont le moins de voisins non encore numérotés.

Cet algorithme consiste en fait en un parcours en largeur du graphe.

Chaque sommet du graphe est visité une et une seule fois. (Voir Fig. 2)

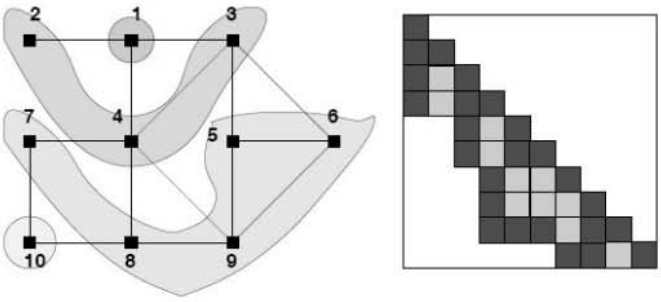


Figure 8.2: 11 zéros dans le profil

8.2.2 Systèmes avec matrices pleines

Dans le cas de matrices pleines, de grande taille puisqu'on a déjà traité le cas des matrices de taille réduite, le nombre d'opérations à effectuer rend l'utilisation des méthodes directes très délicate :

— D'une part, le nombre d'opérations va occasionner des erreurs de calculs qui pénalisent le résultat

— D'autre part, il est nécessaire de faire tous les calculs jusqu'au dernier pour pouvoir obtenir la solution : il est impossible d'obtenir une solution approchée, mais si cette dernière suffirait aux besoins de l'utilisateur.

De ce fait, les méthodes itératives sont en général privilégiées dans ce cas. Ce n'est pas pour autant si simple. De nombreux problèmes peuvent se poser :

(1) La matrice peut ne pas être symétrique, ni définie positive, on verra au paragraphe (8.2.2.1) une piste de solution dans ce cas

(2) La matrice peut avoir un mauvais conditionnement, on verra dans le paragraphe (8.3) quelques techniques destinées à palier ce problème

Pour ces différentes raisons, une importante littérature est développée sur ces sujets. Face au problème de la résolution d'un système linéaire, et confronté à des exigences de performances ou de qualité de résultat, l'ingénieur devra envisager plusieurs solutions tirées de l'état de l'art, et faire preuve d'une bonne dose d'esprit critique.

8.2.2.1 Méthode de l'équation normale

Dans l'objectif de résoudre le système linéaire $A\,x = b$ quand la matrice A n'est pas symétrique, on peut résoudre le système équivalent :

$$A\ \ A\,x = A\ \ b \tag{8.1}$$

qui présente une matrice à la fois symétrique et définie positive.Le système est connu sous le nom de systèmes d'équations normales associé au système $A\,x = b$. Il est associé au problème de minimisation au sens des moindres carrés :

$$\text{Trouver le minimum de } \ \ b - A\,x\ \ _2$$

On peut remarquer que l'équation (8.1) est utilisée pour résoudre les problème aux moindres carrés pour des systèmes surdéterminés, c'est à dire pour des matrices rectangulaires de taille $n x m$, $m < n$.

Une alternative à cette méthode est de poser $x = A\ \ u$ et de résoudre le problème en u :

$$A\,A\ \ u = b \tag{8.2}$$

Une fois que la solution u est obtenue, il suffit de la prémultiplier par A^T pour obtenir x.

On remarque qu'à partir du moment où la matrice du système est symétrique définie positive, on peut utiliser par exemple un algorithme de gradient conjugué pour résoudre le système. Néanmoins, il serait trompeur de considérer qu'on a trouvé dans cette astuce une excellente idée. En effet, dans le cas où la matrice A est dotée d'un mauvais conditionnement, celui de $A^T A$ est encore pire. Remarquons que le conditionnement en norme 2 de la matrice $A^T A$ est donné par

$$\kappa_2(A\ \ A) = \ \ A\ \ A\ \ _2\ \ \ \ A\ \ A\ \ ^{-1}\ \ _2$$

Or $\ \ A\ \ A\ \ _2 = \rho(A\ \ A)$ car $A\ \ A$ est symétrique. Mais par ailleurs $\ \ A\ \ _2 = \ \ \overline{\rho(A\ \ A)}$ dans le cas général, donc en utilisant le même raisonnement pour le second membre :

$$\kappa_2(A\ \ A) = \ \ A\ \ _2^2\ \ A^{-1}\ \ _2^2 = \kappa_2^2(A)$$

On a donc montré que le conditionnement de $A\ \ A$ est le carré de celui de A.

Dans le cas où la matrice A est bien conditionnée, la méthode de l'équation normale peut donc s'avérer une bonne solution. Par contre dans le cas de conditionnements moyens ou mauvais, les calculs des itérés peuvent conduire à la divergence ou à une propagation des erreurs telle

qu'on n'obtient aucune solution satisfaisante. On est donc encore conduit à chercher à améliorer le conditionnement de la matrice d'un système.

8.2.2.2 Alternative à la méthode de l'équation normale

Il existe différentes alternatives à la méthode ci-dessus. Citons par exemple, le système équivalent à $A\,x = b$:

$$\begin{matrix} I & A \\ A & 0 \end{matrix} \begin{matrix} r \\ x \end{matrix} = \begin{matrix} b \\ 0 \end{matrix}$$

qui, avec $r = b - A\,x$ et $A^T r = 0$ conduit bien sûr à résoudre le système initial. La méthode est équivalente à la résolution d'un problème de minimisation sous contrainte :

$$\text{Trouver le minimum de } \quad r - b \ \frac{2}{2} \text{ sous la contrainte } A \quad r = 0$$

Un autre système symétrique est obtenu par :

$$\begin{matrix} 0 & A \\ A & 0 \end{matrix} \begin{matrix} A\,x \\ x \end{matrix} = \begin{matrix} b \\ A\ b \end{matrix}$$

On remarque néanmoins que le système n'est en général pas plus facile à résoudre que le système initial, et que des méthodes comme le gradient conjugué présentent les mêmes inconvénients vis à vis du conditionnement.

8.3 Préconditionnement

L'idée du préconditionnement est simple : il s'agit de trouver une matrice C telle que CA soit mieux conditionnée que A, et telle que le produit CA soit peu coûteux à calculer. L'idéal serait de choisir $C = A^{-1}$ mais c'est évidemment trop coûteux, on choisit donc d'approcher l'inverse de A. Dans le cas où on veut utiliser le gradient conjugué ou ses dérivés, il est essentiel de conserver les propriétés de symétrie et de définie positivité de A pour le produit CA, afin de conserver la convergence.

8.3.1 Décomposition de A

Les préconditionnements les plus simples sont basés sur les méthodes linéaires, donc sur une décomposition de A.

Le préconditionnement diagonal, dit aussi de Jacobi, consiste à choisir $C = D^{-1}$ où D est la matrice dont la diagonale est égale à celle de A. Malgré sa simplicité, c'est un préconditionneur efficace, et très peu coûteux. On ne peut donc pas s'en passer dans tous les cas où il peut être d'une quelconque utilité.

Le précontionnement SSOR consiste à choisir $C = (D + \omega L)D^{-1}(D + \omega U)$ avec $A = D + L + U$, où D est diagonale, L triangulaire inférieure, U triangulaire supérieure. En général on choisit $\omega = 1$.

Ces deux préconditionneurs sont symétriques définis positifs, dès que A l'est.

8.3.2 Préconditionneur polynômial

Nous allons reprendre la méthode du préconditionnement présenté à la section 2.10 du chapitre 2. On suppose que la résolution de $A x = b$ n'est possible que de façon approchée. En conséquence on recherche une solution x qui diffère de x d'une erreur e :

$$x = x + e$$

De $A x = b$ on tire

$$-Ae = b - Ax = r$$

Si on recherche z solution de

$$Az = r$$

on dispose d'une connaissance de l'erreur e, mais ceci revient à résoudre le système initial. Pour contourner cette difficulté, on approche A par M, et on résout à la place

$$Mz = r \tag{8.1}$$

L'une des solutions est d'appliquer p étapes d'une méthode stationnaire sous la forme

$$M_1 z^{(k+1)} = N_1 z^{(k)} + r,$$
$$z^{(0)} = 0$$

Si on pose $G = M_1^{-1} N_1$, alors :

$$z = z^{(p)} = (I + G + G^2 + \ldots + G^{p-1}) M_1^{-1} r$$

On peut alors poser, par analogie avec l'équation (8.1) :

$$M^{-1} = (I + G + G^2 + \ldots + G^{p-1}) M_1^{-1}$$

Bien entendu, la fait que M soit symétrique, définie positive contraint le choix de M_1, N_1 et p. Le fait que le préconditionneur M soit un polynôme de la matrice G, donne son nom à la méthode.

Nous pouvons rapprocher cette méthode avec les méthodes itératives de résolution de systèmes linéaires. En effet nous avons considéré, au chapitre 3, la matrice A comme étant la somme de trois matrices $M = L + D + U$, où L matrice triangulaire inférieure, D matrice diagonale et U matrice triangulaire supérieure. Dans ce cas, nous pouvons obtenir la décomposition

$$A = M - N$$

et, selon la méthode utilisée, nous avons

- soit $M = D$ et $N = -(L + U)$, pour la méthode Jacobi. Le preconditionneur $M^{-1} = D^{-1}$ il est facile à calculer, mais il n'est pas très puissant.
- soit $M = (D + L)$ et $N = -U$, pour la méthode de Gauss-Seidel.

En posant $G = M^{-1} N$, l'expansion polynômiale donne

$$A_0^{-1} = I + G + G^2 + \cdots M^{-1} \tag{8.2}$$

Pour pouvoir faire un calcul fini, il faut que $|\rho(G)| < 1$. Dans ce cas, on a

$$A_0^{-1} \approx I + G + G^2 + \cdots + G^p \; M^{-1}$$

Ce type de préconditionneur est intéressant du point de vue de la vectorisation ou parallélisation du code. Il a donc rencontré un vif succès dès l'apparition des machines correspondantes.

8.3.3 Factorisation incomplète

Les méthodes itératives sont souvent appliquées à des matrices creuses, dont une grande partie des coefficients sont nuls. Comme on l'a vu quand la matrices n'ont pas une forme bande, l'inconvénient des méthodes directes (et donc des préconditionneurs basées dessus) est de remplir les places occupées par des zéros, et donc d'augmenter le coût de stockage induit par une factorisation. L'idée des factorisations incomplètes, est de limiter le remplissage, et de ne pas nécessiter trop d'opérations, donc de ne pas engendrer trop d'erreurs numériques. On obtient alors

$$A = LU + R$$

et on choisit pour préconditionneur

$$C = U^{-1}L^{-1}$$

On parle dans ce cas de méthode ILU(k) : ILU pour Incomplete LU factorization, et le paramètre k indique combien de zéros de la matrice initiale ont été remplacés par des coefficients non nuls dans le préconditionneur.

Dans le cas général, on utilise la notion de profil P tel que :

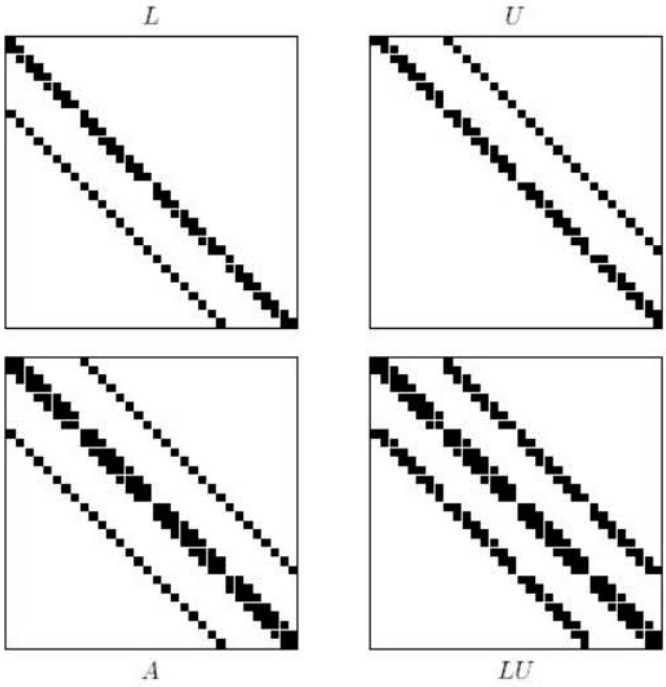
$$P \subset \{(i,j)/\, i = j, 1 \leq i,j \leq n\}$$

qui représente l'ensemble des couples (i,j) tels que pour une matrice quelconque M : $M_{ij} = 0$. On adapte alors l'algorithme de la décomposition LU pour ne calculer que les éléments hors du profil :

```
Pour k = 1,...n − 1 Faire
    Pour i = k + 1,...n et si (i,k) ∉ P Faire
        a_ik = a_ik / a_kk
        Pour j = k + 1,...n et si (i,j) ∉ P Faire
            a_ij = a_ij − a_ik * a_kj
        FinPour
    FinPour
FinPour
```

En pratique, il serait assez coûteux et maladroit de tester pour chaque couple (i,j) ou (i,k) s'il appartient au profil. On procède donc différemment en pratique.

L'une des solutions pour choisir le profil P de la factorisation incomplète, est de prendre précisément celui de la matrice A. Pour une matrice pentadiagonale, on aura donc la configuration suivante :



C'est dans des situations de ce type que la factorisation incomplète de Cholesky a été établie pour la première fois.

8.3.4 Inverse approché

On recherche ici une matrice C qui minimise $I - CA$ ou $I - AC$ pou rune norme à préciser. Ce préconditionnement peut s'avérer très efficace, mais couteux à calculer. En outre les conditions d'existence de C sont mal définie dans le cas où A n'est pas symétrique.

8.3.5 Multigrilles et multiniveaux

Les méthodes multigrilles et multiniveaux sont des méthodes itératives qui peuvent être utilisées en soi. Mais comme pou les méthodes linéaires de type Gauss-Seidel, il est possible de définir un préconditionnement à partir de ces méthodes. Leur intérêt est de réduire notablement le conditionnement de A lorsqu'elle provient de la résolution par discrétisation d'un problème d'équations aux dérivées partielles.

8.4 Gradient conjugué préconditionné

On a vu que dans de nombreux cas, la méthode itérative sera préférée à une méthode directe, et dans ce cas, c'est très souvent le Gradient Conjugué qui sera choisi. Dans les méthodes de préconditionnement vues au paragraphe (8.3) on ne présuppose pas quelle méthode est choisie, on

peut donc appliquer n'importe quelle méthode de préconditionnement associée à une méthode de résolution. Dans les faits, et toujours dans le but d'améliorer l'algorithme utilisé, on a intégré le préconditionnement à l'algorithme de gradient conjugué. Il en découle de nombreuses variantes, que nous ne détaillerons pas toutes ici.

Considérons un système linéaire $A x = b$, de matrice symétrique définie positive. L'idée du gradient conjugué préconditionné est d'appliquer le gradient conjugué à un système transformé :

$$Ax = b$$

où C est une matrice symétrique définie positive telle que

$$\begin{aligned} A &= C^{-1} A C^{-1} \\ x &= Cx \\ b &= C^{-1} b \end{aligned}$$

On doit bien sûr choisir C de telle façon que A ait un meilleur conditionnement que A. Pour des raisons qui s'éclairciront dans la suite, on doit également prévoir que C^2 soit simple à évaluer.

On peut mettre l'algorithme du gradient conjugué sous la forme suivante :

```
Soit x(0) donné dans Rn
k = 0
Calculer r(0) = b – A x(0)
Tant que r(k) = 0 Faire
    k = k + 1
    Si k = 1 alors
        p(1) = r(0)
    sinon
        βk = – p(k–1) A r(k–1) / p(k–1) A p(k–1)
        p(k) = r(k–1) + βk p(k–1)
    FinSi
    αk = p(k) r(k–1) / p(k) A p(k)
    x(k) = x(k–1) + αk p(k)
    r(k) = b – A x(k)
FinTantQue
x = x(k)
```

En remarquant que l'on peut calculer les résidus récursivement, on peut remplacé $r^{(k)} = b - A x^{(k)}$ par $r^{(k+1)} = r^{(k)} - \alpha_k A p^{(k)}$. L'expression de β_k se transforme alors en

$\beta_k =$ $r^{(k)-1}$ $r^{(k-1)}$ $r^{(k-2)}$ $r^{(k-2)}$ et α_k peut être remplacé par

$\alpha_k =$ $r^{(k-1)}$ r^{k-1} $p^{(k)}$ $A p^{(k)}$. Si on applique l'algorithme ainsi transformé au système $Ax = b$ on obtient :

Soit $x^{(0)}$ donné dans R^n
$k = 0$
Calculer $r^{(0)} = b - Ax^{(0)}$
Tant que $r^{(k)} = 0$ Faire
  $k = k + 1$
  Si $k = 1$ alors
    $p^{(1)} = r^{(0)}$
  sinon
    $\beta_k = - \quad r^{(k-1)} \quad r^{(k-1)} \qquad r^{(k-2)} \quad r^{(k-2)}$
    $p^{(k)} = r^{(k-1)} + \beta_k^{(k-1)} p^{(k-1)}$
  FinSi
  $\alpha_k = \quad r^{(k-1)} \quad ^T \; r^{(k-1)} \qquad p^{(k)} \quad C^{-1} A C^{-1} p^{(k)}$
  $x^{(k)} = x^{(k-1)} + \alpha_k p^{(k)}$
  $r^{(k)} = r^{(k-1)} - \alpha_k C^{-1} A C^{-1} p^{(k)}$
FinTantQue
$x = x^{(k)}$

On récupère en sortie un $x^{(k)}$ qui est une approximation de x, lequel permet de calculer x par $x = C^{-1}x$. Néanmoins pour éviter de calculer explicitement C^{-1}, on peut définir des intermédiaires de calcul incluant C^{-1}:

$$p^{(k)} = Cp^{(k)}$$
$$x^{(k)} = Cx^{(k)}$$
$$r^{(k)} = Cr^{(k)}$$

En choisissant alors comme préconditionneur $M = C^2$ (qui reste définie positive) et en notant $z^{(k)}$ la solution de $Mz^{(k)} = r^{(k)}$ alors on aboutit à l'algorithme du gradient conjugué préconditionné :

```
Soit x^(0) donné dans R^n
k = 0
Calculer r^(0) = b − A x^(0)
Tant que r^(k) = 0 Faire
    Résoudre M z^(k) = r^(k)
    k = k + 1
    Si k = 1 alors
        p^(1) = z^(0)
    sinon
        β_k = − r^(k−1) z^(k−1) r^(k−2) z^(k−2)
        p^(k) = z^(k−1) + β_k p^(k−1)
    FinSi
    α_k = r^(k−1) z^(k−1) p^(k)^T A p^(k)
    x^(k) = x^(k−1) + α_k p^(k)
    r^(k) = r^(k−1) − α_k A p^(k)
FinTantQue
x = x^(k)
```

Le choix de la matrice de préconditionnement peut être fait selon les principes énoncés au paragraphe (8.3)

8.5 Raffinement itératif

Dans le cas où les solutions obtenues par résolution du système, préconditionné ou pas, sont entachées d'erreurs numériques, il est possible de recourir à un post-traitement du vecteur solution. On peut supposer, suivant le principe de l'analyse régressive, que la perturbation due aux erreurs d'arrondi, ne porte que sur le second membre on a par exemple réalisé en machine une factorisation :

$$L_c U_c = A + E$$

et on a résolu exactement le système

$$(A + E)x_c = b$$

On vérifie généralement dans ce cas que le résidu calculé $r_c = b - A x_c$ n'est pas nul.

Posons $x^{(0)} = x_c$ et $r^{(0)} = r_c$. Ces valeurs initialisent un processus récursif. On calcul alors pour tout $k \geq 0$:

$$\begin{aligned} (A + E)e^{(k+1)} &= r^{(k)} \\ x^{(k+1)} &= x^{(k)} + e^{(k+1)}_c \\ r^{(k+1)} &= b - A x^{(k+1)}_c \end{aligned}$$

En supposant que l'erreur d'arrondi porte essentiellement sur la résolution des systèmes

linéaires, on pose :

$$x^{(k+1)} - x^{(k)} = (A + E)^{-1} r^{(k)}$$

$$e^{(k+1)} = (A + E)^{-1} A \quad x - x^{(k)}$$

d'où on tire

$$x - x^{(k+1)} = \quad I - (A + E)^{-1} A \quad x - x^{(k)}$$

Posons $G = I - (A + E)^{-1} A = I - \ I + A^{-1} E \ ^{-1}$, on a donc obtenu pour tout $k \geq 1$:

$$x - x^{(k+1)} = G \quad x - x^{(k)} \ = G^{k+1} \quad x - x^{(0)}$$

Supposons alors que pour une norme induite, on ait $A^{-1} E \ = \alpha < 1$ alors on aura

$$G = \sum_{k=1}^{\infty} (-A^{-1} E)^k$$

de sorte que $G \ \leq \frac{\alpha}{1-\alpha}$. On obtient alors

$$x - x^{(k)} \ \leq \quad \frac{\alpha}{1-\alpha} \quad ^{k} \quad x - x^{(0)}$$

L'expérience prouve qu'en deux itérations ce processus peut être utile, mais pas plus.

EXERCICE 8.3 Déterminant et conditionnement

(1) Calculez en fonction de n le déterminant et le conditionnement de la matrice carrée A d'ordre n définie par

$$A = \begin{pmatrix} 1 & & & \\ & 10 & & \\ & & \ddots & \\ & & & 10 \end{pmatrix}$$

(2) De même, calculez en fonction de n le déterminant et le conditionnement de la matrice carrée B d'ordre n définie par :

$$B_{ii} = 1, 1 \leq i \leq n$$
$$B_{i,i+1} = 2, 1 \leq i \leq n-1$$
$$B_{ij} = 0 \text{ sinon}$$

(3) Concluez quant au lien qui peut exister entre conditionnement et déterminant. Expliquez votre point de vue dans le cas général.

EXERCICE 8.4 Notion de préconditionnement

Soient

$$A = \begin{matrix} 1 & 0 \\ 0 & 10^{-6} \end{matrix} \quad \text{et} \quad \Delta A = \begin{matrix} 10^{-8} & 0 \\ 0 & 10^{-14} \end{matrix}$$

(1) Calculez le conditionnement de A en norme 2.

(2) Que vaut

$$\frac{\|\Delta A\|_2}{\|A\|_2}$$

Déduisez-en une estimation de $\frac{\|\Delta x\|_2}{\|x\|_2}$, où x et $x + \Delta x$ sont solutions des systèmes $Ax = b$ et $(A + \Delta A)(x + \Delta x) = b$ avec b un vecteur quelconque.

(3) Que vaut le conditionnement de $2A$, $-4A$, puis de αA avec α un réel quelconque ?

(4) Soit

$$D = \begin{matrix} 1 & 0 \\ 0 & 10^{6} \end{matrix}$$

Exprimez $D \cdot A$ et $D \cdot \Delta A$.

(5) Que vaut le conditionnement de $D \cdot A$ en norme 2 ? Que vaut

$$\frac{\|D \cdot \Delta A\|_2}{\|D \cdot A\|_2} ?$$

(6) Déduisez-en une nouvelle estimation de $\frac{\|\Delta x\|_2}{\|x\|_2}$

(7) Quelle conclusion en tirez-vous ?

8.6 Préconditionnement et erreur de calcul

Lors de l'application d'un algorithme itératif à la solution du système $Ax = b$, la seule information disponible à chaque itération est le résidu $r^{(k)} = b - Ax^{(k)}$. Par contre l'erreur $e^{(k)} = x - x^{(k)}$ est inconnue. Du fait que $r^{(k)} = Ae^{(k)}$, on a la relation

$$\frac{1}{\kappa(A)} \cdot \frac{r^{(k)}}{r^{(0)}} \leq \frac{e^{(k)}}{e^{(0)}} \leq \kappa(A) \frac{r^{(k)}}{r^{(0)}} \leq \kappa^2(A) \frac{e^{(k)}}{e^{(0)}} \tag{8.1}$$

De cette relation on en conclut que si $\kappa(A)$ est proche de 1, le quotient des normes des résidus est relié au quotient des normes des erreurs et donc si le résidu décroît, l'erreur aussi décroît. Par contre si le coonditionnement est grand, le comportement de deux quotients peut être différent, l'un augmentant et l'autre diminuant. Ces remarques montrent l'importance d'avoir un conditionnement proche de 1 et, par voie de conséquence, l'importance du préconditionnement.

Nous pouvons aussi utiliser les relations (8.1) pour élaborer un critère d'arrêt d'une méthode itérative. Soit $x + \Delta x$ la solution calculée. Elle est solution du système perturbé $(A + \Delta A)(x + \Delta x) = b$

(on suppose que le second membre est sans erreur). On calcule le résidu $r = b - A(x + \Delta x)$. Nous avons la relation

$$\frac{r}{A \cdot x + \Delta x} \leq \frac{A + \Delta A}{A} \tag{8.2}$$

Par conséquent si la norme du résidu relatif $\frac{r}{A \cdot x + \Delta x}$ est petite, le résultat est acceptable. Sinon, la solution est relative au système initial qui a subi une grande perturbation. Remarquons que si $\frac{r}{A \cdot x + \Delta x}$ est proche de la précision de l'ordinateur, alors la solution a une bonne précision.

8.6.1 Exercices

EXERCICE 8.5 Démontrer la relation (8.1).

EXERCICE 8.6 Démontrer la relation (8.2)

8.7 Bibliographie

Les ouvrages ci-dessous sont disponibles sous forme de fichier téléchargeable sur le site du cours ou sur Arel

[CB] CLAUDE BREZINSKI : Algèbre matricielle numérique,

[YA] YVES ACHDOU : Algèbre linéaire et analyse numérique matricielle„ téléchargeable à l'adresse http:/ / www.ann.jussieu.fr/ ~achdou/ files/ teaching/ linalg/ book.pdf

[AH1] ALAIN HUARD : Analyse numérique matricielle, Cours de 3ème année, téléchargeable à partir du site de l'INSA Toulouse www-gmm.insa-toulouse.fr

[AH2] ALAIN HUARD : Analyse numérique des grands problèmes linéaires, Cours de 4ème année, téléchargeable à partir du site de l'INSA Toulouse www-gmm.insa-toulouse.fr

[YS] YOUSEF SAAD : Iterative Methods for Sparse Linear Systems, téléchargeable à l'adresse www.stanford.edu/ class/ cme324/ saad.pdf

Les ouvrages ci-dessous sont disponibles en librairie

[QS] ALFIO QUARTERONI, FAUSTO SALERI : Calcul scientifique, Cours, exercices corrigés et illustrations en Matlab et Octave, Springer, 2206.

[AF] ANDRÉ FORTIN : Analyse numérique pour Ingénieurs, Presses Internationales Polytechnique, 2001.

[AD] LUCA AMODEI, JEAN-PIERRE DEDIEU : Analyse numérique matricielle, Cours exercices et corrigés, Dunod, 2008.

[SB] J. STOER, R. BULIRSCH : Introduction to numerical Analysis, Third Edition, Springer, 2000.

[GV] G. GOLUB, C. VAN LOAN : Matrix Computations, Third Edition, The Johns Hopkins University Press, 1996.

[DW] DAVID S. WATKINS : Fundamentals of Matrix computation, Second Edition, Wiley, 2002

[CBr] CLAUDE BREZINSKI : Projection methods for systems of equations, Elsevier, 1997

TABLE DES MATIÉRES