Algèbre Linéaire Numérique

Patrick AMESTOY & Philippe BERGER

15 Mars 2005

Table des matières

Chapitre 1

Représentation des nombres réels et erreurs numériques

1.1 Représentation en machine des nombres réels

1.1.1 Représentation normalisée

Les nombres réels sont codés en machine à l'aide d'un nombre fini de chiffres. On s'intéressera essentiellement à la représentation des réels en virgule flottante. La représentation normalisée pour les nombres positifs est la suivante : $x = m.b^e$ où m est appelé la mantisse, b la base et e l'exposant. La mantisse est normalisée, soit $1 \leq m < b$ (ce qui assure une unicité de la représentation des nombres). La plupart des calculateurs utilisent une base égale à 2 mais certains ont utilisé la base 8 (octale) ou 16 (hexadécimale), ou même 10 pour des calculatrices.

L'ensemble F des nombres réels flottants représentés en machine est caractérisé par 4 paramètres :

- la base b
- le nombre de chiffres en base b pour la mantisse, soit p
- l'exposant minimum E_{min} et maximum E_{max}.

On utilise en général un bit de signe pour la mantisse, soit : $x = (-1)^s m.b^e$ si $x \neq 0$, $1 \leq m < b$. Donc :

$$m = a_1 + \frac{a_2}{b} + \frac{a_3}{b^2} + \ldots + \frac{a_p}{b^{p-1}}$$

$$1 \leq a_1 \leq b-1 \qquad 0 \leq a_i \leq b-1 \qquad i = 2, \ldots, p$$

Si $x = 0$ alors $e = m = 0$.

Conséquence : Les éléments de F sont limités en valeur absolue i.e. F^+ est inclus dans un intervalle $[x_{min}, x_{max}]$ où x_{min} et x_{max} sont strictement positifs et un nombre réel quelconque n'est pas toujours représentable exactement. x_{min} et x_{max} peuvent facilement se calculer :

$$x_{\min} = (1 + \frac{0}{b} + \ldots + \frac{0}{b^{p-1}}).b^{E_{\min}} = b^{E_{\min}}$$

$$x_{\max} = (b - 1 + \frac{b-1}{b} + \ldots + \frac{b-1}{b^{p-1}}).b^{E_{\max}} = (b-1)(1 + \frac{1}{b} + \ldots + \frac{1}{b^{p-1}}).b^{E_{\max}}$$

$$x_{\max} = (b-1)\frac{1-\frac{1}{b^p}}{1-\frac{1}{b}} b^{E_{\max}} = b(1-\frac{1}{b^p})b^{E_{\max}}$$

On définit la précision machine *macheps* comme l'intervalle entre 1.0 (représentable exactement en machine) et le nombre réel représentable exactement en machine immédiatement supérieur à 1.0 :

$$1 = (1 + \frac{0}{b} + \ldots + \frac{0}{b^{p-1}}).b^0$$

$$1 + macheps = (1 + \frac{0}{b} + \ldots + \frac{0}{b^{p-2}} + \frac{1}{b^{p-1}}).b^0 \qquad macheps = \frac{1}{b^{p-1}}$$

F est symétrique par rapport à 0. Les nombres réels ne pouvant être représentés exactement sont approchés par un élément de F. Soit $fl(x)$ la représentation en machine de x. Cette approximation est le plus souvent obtenue à partir d'une troncature ou d'un arrondi.

On peut optimiser encore le mode de représentation en introduisant la notion de nombres flottants dénormalisés pour coder des nombres très petits (compris par exemple entre 0 et x_{min}) : l'exposant vaut E_{min} et les premiers chiffres de la mantisse peuvent être nuls. Le plus petit nombre strictement positif représentable devient alors $b^{E_{min}-(p-1)}$.

1.1.2 Arrondis et troncature

Soit la représentation exacte de x :

$$x = (-1)^s(a_1 + \frac{a_2}{b} + \ldots + \frac{a_p}{b^{p-1}} + \frac{a_{p+1}}{b^p} + \ldots).b^e$$

où les coefficients a_i sont les chiffres représentant x en base b.
Soient $m = a_1 + \frac{a_2}{b} + \ldots + \frac{a_p}{b^{p-1}}$ et $h = \frac{a_{p+1}}{b^p} + \ldots$.
On a donc $x = (-1)^s(m+h).b^e$

Troncature $fl(x) = (-1)^s(a_1 + \frac{a_2}{b} + \ldots + \frac{a_p}{b^{p-1}}).b^e$ *i.e.* $fl(x) = (-1)^s m.b^e$

Arrondi au plus près

Soient $x_1 = (-1)^s m.b^e$ et $x_2 = (-1)^s(m + \frac{1}{b^{p-1}}).b^e$
(il s'agit des deux nombres représentables encadrant x). Alors :

$fl(x) = x_1$ si $0 \le h < \frac{macheps}{2}$ ou encore $0 \le a_{p+1} < \frac{b}{2}$

$fl(x) = x_2$ si $\frac{macheps}{2} \le h < macheps$ ou encore $\frac{b}{2} \le a_{p+1} < b$

Arrondi par excés
Si $h = 0$ alors $fl(x) = x_1$ sinon
$fl(x) = x_2$ si $x > 0$ $fl(x) = x_1$ si $x < 0$

Arrondi par défaut
Si $x > 0$ alors $fl(x) = x_1$
Si $x < 0$ et $h = 0$ alors $fl(x) = x_1$
Si $x < 0$ et $h \neq 0$ alors $fl(x) = x_2$

1.1.3 La norme IEEE

C'est le standard de représentation des nombres flottants incluant : le format des nombres, les modes d'arrondi possibles, le traitement des exceptions (overflow, division par zéro,...), les procédures de conversion (en décimal par exemple) et l'arithmétique. Cependant certains gros calculateurs ont conservé leurs formats spécifiques.

Les nombres sont représentés sous la forme : $x = (-1)^s m.b^e$ avec $1 \leq m < b$
La base est 2. Le premier bit de la mantisse étant toujours égal à 1, il n'est pas mémorisé (premier bit implicite).

Un exposant négatif ou nul a un premier bit égal à 0, un exposant positif un premier bit égal à 1. Si q est le nombre de bits dédié à l'exposant, les valeurs 0 et $2^q - 1$ sont réservées, et les exposants sont codés sur l'intervalle $[1, 2^q - 2]$. La représentation en machine d'un exposant e est donc $e - E_{min} + 1$.

Exemple avec $q = 4$: $\quad E_{min} = -6$

valeur décimale	*codage binaire IEEE*
réservé	0000
-6	0001
-5	0010
-4	0011
-3	0100
-2	0101
-1	0110
0	0111
1	1000
2	1001
3	1010
4	1011
5	1100
6	1101
7	1110
réservé	1111

Simple précision IEEE

Codage sur 32 bits. Le bit 31 représente le signe, l'exposant est codé sur 8 bits (bits 30 à 23), la mantisse sur 23 bits (bits 22 à 0) plus 1 bit implicite.

$$E_{min} = \{00000001\} = -126 \qquad E_{max} = \{11111110\} = 127$$

$$x_{min} = 2^{-126} \approx 1.18\ 10^{-38} \qquad x_{max} = (2 - \frac{1}{2^{23}}).2^{127} \approx 3.40\ 10^{38}$$

$$macheps = 2^{-23} \approx 1.19\ 10^{-7}$$

Double précision IEEE

Codage sur 64 bits. Le bit 63 représente le signe, l'exposant est codé sur 11 bits (bits 62 à 52), la mantisse sur 52 bits (bits 51 à 0) plus 1 bit implicite.

$$E_{min} = -1022 \qquad E_{max} = 1023$$

$$x_{min} = 2^{-1022} \approx 2.23\ 10^{-308} \qquad x_{max} = (2 - \frac{1}{2^{52}}).2^{1023} \approx 1.79\ 10^{308}$$

$$macheps = 2^{-52} \approx 2.22\ 10^{-16}$$

Quelques représentations externes en double précision :

Si e' est la représentation binaire en machine de l'exposant,

$e' = 2047$ et $m \neq 0$ NAN (Not A Number)
$e' = 2047$ et $m = 0$ $\pm\infty$
$0 < e' < 2047$ $(-1)^s\ 1.m * (2^{e'-1023})$ nombres normalisés
$e' = m = 0$ 0
$e' = 0$ et $m \neq 0$ $(-1)^s\ 0.m * (2^{-1022})$ nombres dénormalisés

L'arrondi doit satisfaire les propriétés suivantes :

$$\forall x \in F \quad fl(x) = x$$

$$-x_{max} \leq x \leq y \leq x_{max} \ \Rightarrow\ fl(x) \leq fl(y)$$

Les 4 modes d'arrondi, précédemment décrits, conviennent.

Calcul de l'erreur relative entre x et $fl(x)$,selon le mode d'arrondi, en fonction de *macheps*

La troncature est équivalente à l'arrondi par défaut si $x \geq 0$ et à l'arrondi par excés si $x \leq 0$. Pour ces arrondis, il suffit donc de considérer (par exemple) le cas de l'arrondi par défaut :

Soit $x \geq 0 :$ $x = (a_1 + \frac{a_2}{b} + \ldots + \frac{a_p}{b^{p-1}} + \frac{a_{p+1}}{b^p} + \ldots).b^e = (m+h).b^e$

avec $m = a_1 + \frac{a_2}{b} + \ldots + \frac{a_p}{b^{p-1}}$ et $1 \leq m < b, \quad h \leq \frac{1}{b^{p-1}}$

Si $x_2 > x \geq x_1 > 0$ $fl(x) = x_1$ alors $fl(x) = m.b^e$

et $\frac{|x - x_1|}{|x|} = \frac{|h|}{|m+h|} \leq \frac{1}{b^{p-1}}$

d'où $\frac{|x - fl(x)|}{|x|} \leq \frac{1}{b^{p-1}} = macheps$

Dans le cas de l'arrondi au plus près, par analogie avec la démonstration précédente, si $fl(x) = x_1$ alors $h \leq \frac{1}{2\ b^{p-1}}$

donc $\frac{|x - x_1|}{|x|} = \frac{|h|}{|m+h|} \leq \frac{1}{2\ b^{p-1}}$ d'où $\frac{|x - fl(x)|}{|x|} \leq \frac{1}{2\ b^{p-1}} = \frac{macheps}{2}$

Lors de l'exécution des opérations arithmétiques, le respect de la norme IEEE nécessite l'introduction de bits supplémentaires :

- sticky bit : indique si un nombre décimal est nul et aide à déterminer les arrondis par défaut et par excés d'un nombre.
- bit d'arrondi (correspondant à a_{p+1} la première position non représentable en base b)
- bit de parité (permettant d'équilibrer les arrondis au plus près)

1.1.4 Erreurs d'arrondi sur les opérations arithmétiques

Absorption : Lors de l'addition d'un nombre positif très grand x et d'un nombre positif très petit y, y est absorbé par x si $|y| \leq \frac{|x| \; macheps}{2}$, alors $fl(x+y) = fl(x)$

Elimination : Lors de l'addition de deux nombres x et y de signes opposés et très voisins en valeur absolue, il y a élimination si $|x+y| \leq \frac{macheps(|x|+|y|)}{2}$, alors $f(x+y) = 0$

Elimination globale : Soit $S = \sum_{i=1}^{n} x_i$. Il y a élimination globale des x_i si $|S| \leq \frac{macheps}{2} \sum_{i=1}^{n} |x_i|$

1.2 Exemples de calcul d'erreur

On notera toujours $fl(a \; op \; b) = (a \; op \; b)(1+\epsilon)$ avec $|\epsilon| \leq u$ et $u = macheps$ ou $\frac{macheps}{2}$ suivant le mode d'arrondi.

1.2.1 Erreur d'arrondi sur une somme

Soit l'algorithme calculant la somme des coefficients d'un vecteur $x = [x_k]$:

$S = 0$
Pour $k = 1$ *à* n
$\quad S = S + x_k$
Finpour

Soient : $S = \sum_{i=1}^{n} x_i \qquad S_k = fl(\sum_{i=1}^{k} x_i)$

$S_2 = fl(x_1 + x_2) = (x_1 + x_2)(1+\epsilon_2)$ d'où $S_2 - (x_1 + x_2) = \epsilon_2(x_1 + x_2)$
$S_3 = fl(S_2 + x_3) = (S_2 + x_3)(1+\epsilon_3) = ((x_1 + x_2) + \epsilon_2(x_1 + x_2) + x_3)(1+\epsilon_3)$
$S_3 - (x_1 + x_2 + x_3) = (x_1 + x_2 + x_3)\epsilon_3 + (x_1 + x_2)\epsilon_2 + (x_1 + x_2)\epsilon_2\epsilon_3$

d'où, en négligeant $(x_1 + x_2)\epsilon_2\epsilon_3$:
$S_3 - (x_1 + x_2 + x_3) = (x_1 + x_2 + x_3)\epsilon_3 + (x_1 + x_2)\epsilon_2$
Par récurrence : $S_n = fl(S)$

$$S_n - S = (\sum_{i=1}^{n} x_i)\epsilon_n + (\sum_{i=1}^{n-1} x_i)\epsilon_{n-1} + \ldots + (x_1 + x_2)\epsilon_2$$

$$S_n - S = x_1(\sum_{i=2}^{n} \epsilon_i) + x_2(\sum_{i=2}^{n} \epsilon_i) + x_3(\sum_{i=3}^{n} \epsilon_i) + \ldots + x_n\epsilon_n$$

En posant $\epsilon_1 = 0$:

$$S_n - S = x_1(\sum_{i=1}^{n} \epsilon_i) + x_2(\sum_{i=2}^{n} \epsilon_i) + x_3(\sum_{i=3}^{n} \epsilon_i) + \ldots + x_n\epsilon_n$$

1.2.2 Erreur d'arrondi sur un produit scalaire

Soit l'algorithme calculant le produit scalaire entre deux vecteurs $x = [x_k]$ et $y = [y_k]$:

$S = 0$
Pour $k = 1$ *à* n
$S = S + x_k y_k$
Finpour

On note $S = \sum_{i=1}^{n} x_i y_i$ $S_k = fl(\sum_{i=1}^{k} x_i y_i)$

$fl(a+b) = (a+b)(1+\varepsilon)$ $fl(a.b) = (a.b)(1+\delta)$

On suppose une exécution en simple précision IEEE avec un mode d'arrondi au plus près :

$|\varepsilon| \leqslant u = \frac{macheps}{2} \simeq \frac{1.19\ 10^{-7}}{2}$ $|\delta| \leqslant u = \frac{macheps}{2}$

Calcul de S_n sans simplification

$S_1 = fl(x_1.y_1) = x_1.y_1(1+\delta_1)$

$S_2 = fl(S_1 + x_2.y_2) = (S_1 + fl(x_2.y_2))(1+\varepsilon_2)$
$= (x_1.y_1(1+\delta_1) + x_2.y_2(1+\delta_2))(1+\varepsilon_2)$
$= x_1.y_1(1+\delta_1)(1+\varepsilon_2) + x_2.y_2(1+\delta_2)(1+\varepsilon_2)$

$S_3 = fl(S_2 + x_3.y_3) = (S_2 + fl(x_3.y_3))(1+\varepsilon_3)$
$= x_1.y_1(1+\delta_1)(1+\varepsilon_2)(1+\varepsilon_3) + x_2.y_2(1+\delta_2)(1+\varepsilon_2)(1+\varepsilon_3)$
$+x_3.y_3(1+\delta_3)(1+\varepsilon_3)$

Par récurrence :
$S_k = x_1.y_1(1+\delta_1)(1+\varepsilon_2)\dots(1+\varepsilon_k) + x_2.y_2(1+\delta_2)(1+\varepsilon_2)\dots(1+\varepsilon_k)$
$x_3.y_3(1+\delta_3)(1+\varepsilon_3)\dots(1+\varepsilon_k) + \dots\dots + x_k.y_k(1+\delta_k)(1+\varepsilon_k)$

Au final et en posant $\varepsilon_1 = 0$
$S_n = x_1.y_1(1+\delta_1)(1+\varepsilon_1)\dots(1+\varepsilon_n) + x_2.y_2(1+\delta_2)(1+\varepsilon_2)\dots(1+\varepsilon_n)$
$x_3.y_3(1+\delta_3)(1+\varepsilon_3)\dots(1+\varepsilon_n) + \dots\dots + x_n.y_n(1+\delta_n)(1+\varepsilon_n)$
$= \sum_{k=1}^{n} x_k.y_k(1+\delta_k) \prod_{i=k}^{n}(1+\varepsilon_i)$

Utilisation d'un lemme d' "arithmétique" avec contrainte sur la taille des vecteurs

lemme : Si $1+\alpha = \prod_{k=1}^{n}(1+\beta_k)$ avec $|\beta_k| \leq u$ et $nu \leq 0.01$ alors $|\alpha| \leq 1.01\ nu$

Application : soient $k = 1,\dots,n$ $(1+\gamma_k) = (1+\delta_k)\prod_{i=k}^{n}(1+\varepsilon_i)$

$k = 1,\dots,n$ $i = k,\dots,n$ $|\delta_k| \leqslant u$ $|\varepsilon_i| \leqslant u$

Si $n \leqslant 16665$:

$k = 1,\dots,n$ $u(n-k+2) \leqslant u(n+1) \leqslant \frac{macheps}{2}.16666 \simeq 0.0099996 \leqslant 0.01$

donc $k = 1,\dots,n$ $|\gamma_k| \leqslant 1.01\ u(n-k+2) \leqslant 0.0101$

$$S_n = \sum_{k=1}^{n} x_k.y_k(1+\gamma_k) \qquad S_n - S = \sum_{k=1}^{n} x_k.y_k.\gamma_k$$

$$|S_n - S| \leqslant \sum_{k=1}^{n} |x_k|.|y_k|.|\gamma_k| \leqslant 0.0101 \sum_{k=1}^{n} |x_k|.|y_k|$$

$$\frac{|fl({}^t x.y) - {}^t x.y|}{|{}^t x.y|} \leqslant 0.0101 \frac{{}^t|x|.|y|}{|{}^t x.y|}$$

1.3 Exemples de problèmes mal posés

Un problème stable est un problème peu sensible aux perturbations sur les données initiales. Pour un problème mal posé (ou mal conditionné), le résultat devient complètement différent dès que l'on perturbe un tant soit peu les données initiales.

1.3.1 Premier exemple : une limite de suite

Soit la suite $\{x_n\}$ définie par :

$$x_{n+1} = 111 - \frac{1130}{x_n} + \frac{3000}{x_n x_{n-1}} \qquad x_0 = \frac{11}{2} \qquad x_1 = \frac{61}{11}$$

Etude théorique de la limite

a) On peut ramener le problème précédent à une récurrence linéaire d'ordre 3 de la forme : $a_n = \sum_{i=1}^{3} \alpha_i a_{n-i}$

En effet, soit $x_n = \frac{a_{n+1}}{a_n}$ alors $\frac{a_{n+2}}{a_{n+1}} = 111 - 1130\frac{a_n}{a_{n+1}} + 3000\frac{a_{n-1}}{a_{n+1}}$

La suite $\{a_n\}$ vérifie donc :

$$a_{n+2} = 111a_{n+1} - 1130a_n + 3000a_{n-1}$$

$$a_1 = a_0 x_0 = 11\frac{a_0}{2} \qquad a_2 = a_1 x_1 = 61\frac{a_0}{2}$$

(avec a_0 quelconque mais non nul)

b) a_n s'écrit alors $\sum_{i=1}^{3} c_i q_i^n$ où les q_i sont les 3 racines de l'équation :

$$q^3 = \sum_{i=1}^{3} \alpha_i q_i^{3-i}$$

Cette équation s'écrit $q^3 - 111q^2 + 1130q - 3000 = 0$ avec cpmme racines $q_1 = 5$, $q_2 = 6$, $q_3 = 100$

c) On peut alors en déduire la limite de la suite $\{x_n\}$. En effet, la suite $\{a_n\}$ devient :

$$a_n = c_1 5^n + c_2 6^n + c_3 100^n$$

où les coefficients c_1, c_2 et c_3 sont les solutions du système linéaire :

$$a_0 = c_1 + c_2 + c_3$$
$$a_1 = 5c_1 + 6c_2 + 100c_3$$
$$a_2 = 25c_1 + 36c_2 + 10000c_3$$

i.e. $\quad c_1 = c_2 = \dfrac{a_0}{2} \quad$ et $\quad c_3 = 0$

Dans le cas général, en supposant que $c_3 \neq 0$:

$$x_n = \frac{a_{n+1}}{a_n} = \frac{c_1 5^{n+1} + c_2 6^{n+1} + c_3 100^{n+1}}{c_1 5^n + c_2 6^n + c_3 100^n}$$

$$= \frac{c_3 100^n [5\frac{c_1}{c_3}(\frac{5}{100})^n + 6\frac{c_2}{c_3}(\frac{6}{100})^n + 100]}{c_3 100^n [\frac{c_1}{c_3}(\frac{5}{100})^n + \frac{c_2}{c_3}(\frac{6}{100})^n + 1]} = \frac{100 + u_n}{1 + v_n}$$

avec $\quad \lim_{n\to+\infty} u_n = \lim_{n\to+\infty} v_n = 0 \quad$ donc $\quad \lim_{n\to+\infty} x_n = 100$

Par une analyse similaire :

Si $\quad c_3 = 0 \quad$ et $\quad c_2 \neq 0, \quad \lim_{n\to+\infty} x_n = 6$

Si $\quad c_2 = c_3 = 0 \quad$ et $\quad c_1 \neq 0, \quad$ la suite est stationnaire de valeur 5

Si $\quad c_1 = c_2 = c_3 = 0 \quad$ impossible car $a_0 \neq 0$

Etude expérimentale de la limite

On effectue le calcul des n premiers termes de cette suite sur une machine (ici en simple précision IEEE mais le comportement reste le même en double précision).

n	x_n
0	$\frac{11}{2}$
1	$\frac{61}{11}$
2	5.59016
3	5.63344
4	5.67478
5	5.71568
6	5.79015
7	6.49022

n	x_n
8	16.7229
9	71.0688
10	97.6242
11	99.8574
12	99.9916
13	99.9995
14	100.0000
15	100.0000

Or cette suite devrait logiquement converger vers 6 car $c_3 = 0$ et $c_2 \neq 0$!

Le problème vient de la représentation en machine de la valeur $\frac{61}{11} \approx 5.5454545\ldots$. Ce nombre n'est pas représentable exactement en base 2 que ce soit en simple ou double précision. L'inévitable erreur d'arrondi revient à étudier la convergence avec une valeur $c_3 \neq 0$.

1.3.2 Deuxième exemple : un calcul d'intégrale

Exercice 1

On considére l'intégrale : $\quad I_n = \int_0^1 x^n e^{-x} dx$

1) Démontrer la récurrence suivante, qui constitue l'algorithme le plus naturel pour calculer cette intégrale : $\quad I_0 = 1 - \dfrac{1}{e} \qquad I_n = nI_{n-1} - \dfrac{1}{e}$

2) On suppose que tous les calculs sont effectués de manière exacte et que la seule erreur d'arrondi est introduite lors de la première itération i.e. la valeur calculée est $I'_0 = I_0 + \Delta I_0$**. Soit** $I_n' = I_n + \Delta I_n$**, la valeur calculée à l'étape** n**. Exprimer** ΔI_n **en fonction de** ΔI_0**. Quelle conclusion peut-on tirer ?**

3) Supposons maintenant connue la valeur I_p''**, avec** $p > n$**, approximation de** I_p **avec une erreur de** δI_p **:** $I_p'' = I_p + \delta I_p$**. On suppose que tous les calculs sont effectués de manière exacte et que la seule erreur d'arrondi est** δI_p**. Comment calculer** I_n **à partir de** I_p **? Quelle est l'erreur** δI_n **résultant de cet algorithme ? Quelle conclusion peut-on tirer ?**

Illustration : $n = 50$, calculs en double précision IEEE.

1ère méthode : $\Delta I_0 = 2.22\ 10^{-16}$ $(macheps)$ $\Delta I_{50} = 6.75\ 10^{48}$

2ème méthode : $p = 60$ $(arbitrairement\ choisi)$ $I_{60}'' = 1$

(approximation grossière car compte tenu des variations de $f(x) = x^{60}e^{-x}$ dans $[0, 1]$, on sait que son intégrale exacte est inférieure à 1 et donc $|\delta I_{60}| < 1$)

$$\delta I_{50} < 3.66\ 10^{-18}$$

Chapitre 2

Rappels sur les matrices de $\mathcal{M}_n(\mathbb{R})$

$A \in \mathcal{M}_n(\mathbb{R}) \quad A = [a_{ij}] \quad i = 1, \ldots, n \quad j = 1, \ldots, n$
${}^t(A.B) = {}^tB.{}^tA \qquad (AB)^{-1} = B^{-1}.A^{-1} \qquad {}^t(A^{-1}) = ({}^tA)^{-1}$
Soit $(.|.)$ un produit scalaire dans $\mathbb{R}^n$,
$\forall (x, y) \in (\mathbb{R}^n)^2 \qquad (x|A.y) = ({}^tA.x|y) = {}^tx.A.y$

2.1 Définitions

A symétrique $\Leftrightarrow \quad A = {}^tA \qquad (a_{ij} = a_{ji} \quad i, j = 1, \ldots, n)$
A hermitienne $(A \in \mathcal{M}_n(\mathbb{C})) \quad \Leftrightarrow \quad A = {}^\dagger A \qquad (a_{ij} = \bar{a_{ji}} \quad i, j = 1, \ldots, n)$
Sous-matrice principale de A de dimension p $(p \leqslant n)$: sous-matrice formée des p premières lignes et des p premières colonnes de A.

Déterminant de $A = det(A) = |A|$

$$|\alpha A| = \alpha^n |A| \qquad |A| = |{}^tA| \qquad |A.B| = |A|.|B| \qquad |A^{-1}| = \frac{1}{|A|}$$

A inversible $\Leftrightarrow \quad |A| \neq 0 \quad \Leftrightarrow \quad rang(A) = n \quad \Leftrightarrow \quad A$ régulière
A définie $\Leftrightarrow \quad [\ (x|Ax) = 0 \Longrightarrow x = 0\ ]$
A positive $\Leftrightarrow \quad \forall x \in \mathbb{R}^n \quad (x|Ax) \geqslant 0$
A définie positive $\Leftrightarrow \quad \forall x \in \mathbb{R}^n$ et $x \neq 0 \quad {}^tx.A.x > 0$ ou $(x|Ax) > 0$
A orthogonale $\Leftrightarrow \quad A^{-1} = {}^tA$
$trace(A) =$ somme des coefficients diagonaux de A

$\lambda \in \mathbb{C}$ valeur propre de $A \Leftrightarrow \exists x \in \mathbb{C}^n$ non nul, appelé vecteur propre, t.q. $A.x = \lambda x$
Sous-espace propre associé à λ : le sous-espace vectoriel constitué des vecteurs propres associés à λ.

FIG. 2.1 – Matrices diagonales et tridiagonales

FIG. 2.2 – Matrice diagonales et tridiagonales par blocs

FIG. 2.3 – Matrices triangulaires

2.2 Propriétés

2.2.1 Matrice triangulaire

Si A est une matrice triangulaire inférieure (resp. supérieure) et inversible, son inverse est triangulaire inférieur (resp. supérieur). Son déterminant est égal au produit des coefficients diagonaux. Le produit de deux matrices triangulaires inférieures (resp. supérieures) est triangulaire inférieur (resp. supérieur).

2.2.2 Matrice symétrique

Ses valeurs propres sont réelles mais non nécessairement distinctes (i.e. un sous-espace propre peut être de dimension > 1). Les vecteurs propres associés à des valeurs propres distinctes sont orthogonaux (i.e. les sous-espaces propres associés sont orthogonaux). Il existe une base orthogonale (ou orthonormée) formée de vecteurs propres.

2.2.3 Matrice symétrique définie positive

Les sous-matrices principales sont définies positives, les éléments diagonaux sont strictement positifs, les valeurs propres sont strictement positives (non nécessairement distinctes). Pour une matrice définie, les valeurs propres sont non nulles. Pour une matrice symétrique positive, les valeurs propres sont positives ou nulles.

2.3 Normes et conditionnement

2.3.1 Normes matricielles

$\mathcal{M}_n(\mathbb{R})$ est un espace vectoriel de dimension finie donc toutes les normes sont équivalentes. $\|.\|$, norme matricielle, satisfait aux mêmes propriétés que les normes vectorielles :

$$\begin{gathered}
\forall A \in \mathcal{M}_n(\mathbb{R}) \quad \|A\| \geq 0 \\
\|A\| = 0 \Leftrightarrow A = 0 \\
\forall (A, B) \in \mathcal{M}_n(\mathbb{R})^2 \quad \|A + B\| \leq \|A\| + \|B\| \\
\forall A \in \mathcal{M}_n(\mathbb{R}) \quad \alpha \in \mathbb{R} \quad \|\alpha A\| = |\alpha|.\|A\|
\end{gathered}$$

Norme de Froebenius ou F-norme :

$$A = [a_{ij}] \quad i, j = 1, \ldots, n \qquad \|A\|_F = \sqrt{\sum_{i=1}^{n} \sum_{j=1}^{n} |a_{ij}|^2}$$

Norme du Max : $A \in \mathcal{M}_n(\mathbb{R}) \quad \|A\|_\Delta = \max_{i,j=1,\ldots,n} |a_{ij}|$

p-norme :
Soit la norme de $\mathbb{R}^n$ définie par : $\|x\|_p = (|x_1|^p + \ldots + |x_n|^p)^{\frac{1}{p}}$
(La 2-norme correspond à la norme euclidienne)

$$A \in \mathcal{M}_n(\mathbb{R}) \quad \|A\|_p = \sup_{\|x\|_p \neq 0} \frac{\|A.x\|_p}{\|x\|_p} = \sup_{\|x\|_p = 1} \|A.x\|_p$$

de manière plus générale, soit $\|.\|$ une norme vectorielle quelconque, la norme matricielle définie par :

$$A \in \mathcal{M}_n(\mathbb{R}) \quad \|A\| = \sup_{\|x\| \neq 0} \frac{\|A.x\|}{\|x\|}$$

est appelée norme matricielle induite par $\|.\|$.

Propriété d'une norme matricielle induite par une norme vectorielle

Soit $\|.\|$ une norme vectorielle. On considére la norme matricielle définie par :

$A \in \mathcal{M}_n(\mathbb{R}) \quad \|A\| = \sup_{\|x\| \neq 0} \frac{\|A.x\|}{\|x\|}$ alors $\|A.B\| \leq \|A\|.\|B\|$

démonstration :

$\forall x \neq 0 \quad \|A.x\| \leq \|A\|.\|x\|$ d'où :

$\|A.B.x\| = \|A.(B.x)\| \leq \|A\|.\|B.x\| \leq \|A\|.\|B\|.\|x\|$

$$\frac{\|A.B.x\|}{\|x\|} \leq \|A\|.\|B\|$$

et, en particulier, en prenant le Sup du premier membre : $\|A.B\| \leq \|A\|.\|B\|$

Autres propriétés : $\|A\|_2 \leq \|A\|_F \leq \sqrt{n}\|A\|_2 \qquad \|A\|_\Delta \leq \|A\|_2$

Exercice 2

Soit $A \in \mathcal{M}_n(\mathbb{R}) \qquad A = \begin{pmatrix} I_1 & B \\ 0 & I_2 \end{pmatrix}$

où I_1 **matrice identité de** $\mathcal{M}_p(\mathbb{R})$ $(p < n)$, I_2 **matrice identité de** $\mathcal{M}_{n-p}(\mathbb{R})$ **et** $B \in \mathcal{M}_{p,n-p}(\mathbb{R})$

1) Montrer que A **est inversible et calculer son inverse.**

2) On note $||.||_F$ **la norme matricielle de Froebenius. Montrer que** $K(A) = ||A||_F.||A^{-1}||_F = n + ||B||_F^2$

Exercice 3

$\forall x \in \mathbb{R}^n$**, on définit la norme** $||x||_\infty = \max_{i=1,\ldots,n} |x_i|$

où $x_i \quad i = 1, \ldots, n$ **sont les composantes du vecteur** x.

Montrer que, pour la norme matricielle induite :

$$\forall A \in \mathcal{M}_n(\mathbb{R}) \qquad ||A||_\infty = \max_{i=1,\ldots,n} \sum_{j=1}^{n} |a_{ij}|$$

où $a_{ij} \quad i, j = 1, \ldots, n$ **sont les coefficients de la matrice** A.

2.3.2 Conditionnement

Caractériser le conditionnement du système $A.x = b \quad \Leftrightarrow$ Caractériser les variations de la solution x si A et/ou b subissent une perturbation (due aux erreurs numériques et/ou informatiques).

Il y a plusieurs approches possibles pour caractériser le conditionnement. Par exemple, soit la perturbation du système :

$(A + \epsilon F).x(\epsilon) = b + \epsilon f$ avec $\epsilon \in \mathbb{R}^+ \quad F \in \mathcal{M}_n(\mathbb{R}) \quad f \in \mathbb{R}^n$

Supposons la fonction $x(\epsilon)$ dérivable dans un voisinage de 0 :

$$F.x(\epsilon) + (A + \epsilon F).x'(\epsilon) = f \;\Rightarrow\; F.x(0) + A.x'(0) = f$$

Soit $x(0) = x$: $x'(0) = A^{-1}.(f - F.x)$.

D'autre part, développement de Taylor de $x(\epsilon)$ en 0 : $x(\epsilon) = x + \epsilon x'(0) + o(\epsilon^2)$

Soit $\|.\|$ une norme matricielle induite (par exemple la 2-norme) :

$$\|x(\epsilon) - x\| = \|\epsilon A^{-1}.(f - F.x) + o(\epsilon^2)\| \leq \epsilon\|A^{-1}.(f - \epsilon F)\| + o(\epsilon^2)$$

$$\leq \epsilon\|A^{-1}\|.\|f - F.x\| + o(\epsilon^2) \leq \epsilon\|A^{-1}\|.(\|f\| + \|F.x\|) + o(\epsilon^2)$$
$$\leq \epsilon\|A^{-1}\|.(\|f\| + \|F\|.\|x\|) + o(\epsilon^2)$$

$$\frac{\|x(\epsilon) - x\|}{\|x\|} \leq \epsilon\|A^{-1}\|.(\frac{\|f\|}{\|x\|} + \|F\|) + o(\epsilon^2) \leq \epsilon\|A\|.\|A^{-1}\|.(\frac{\|f\|}{\|A\|.\|x\|} + \frac{\|F\|}{\|A\|})$$

or $\|A\|.\|x\| \geq \|A.x\| = \|b\|$ donc

$$\frac{\|x(\epsilon) - x\|}{\|x\|} \leq \|A\|.\|A^{-1}\|.(\frac{\epsilon\|f\|}{\|b\|} + \frac{\epsilon\|F\|}{\|A\|}) + o(\epsilon^2)$$

Soient $\rho_b = \dfrac{\epsilon\|f\|}{\|b\|}$ erreur relative sur b, $\rho_A = \dfrac{\epsilon\|F\|}{\|A\|}$ erreur relative sur A

$$\frac{\|x(\epsilon) - x\|}{\|x\|} \leq \|A\|.\|A^{-1}\|.(\rho_A + \rho_b) + o(\epsilon^2) = K(A)(\rho_A + \rho_b) + o(\epsilon^2)$$

$K(A) = \|A\|.\|A^{-1}\|$ est le nombre de conditionnement de la matrice A.

Plus $K(A)$ est élevé, plus la solution du système peut être perturbée par des erreurs sur A ou b. La résolution d'un système avec une matrice mal conditionnée est un problème mal posé (au sens du Chapitre1).

La valeur minimale que peut prendre $K(A)$ (système très bien conditionné) est 1. En effet :

$$\|A.A^{-1}\| = \|I\| = 1 \leq \|A\|.\|A^{-1}\| = K(A)$$

Selon la norme retenue, $K(A)$ peut, bien sûr, prendre des valeurs différentes.

Propriétés :

- Si A possède n valeurs propres réelles (par exemple, A symétrique) :
  $|\lambda_1| \geq \ldots \geq |\lambda_n|$ alors $K(A) = \|A\|_2.\|A^{-1}\|_2 = \dfrac{|\lambda_1|}{|\lambda_n|}$
- Si A est orthogonale, on montre que $\|A\|_2 = \|^tA\|_2 = 1$ et $K(A) = 1$

Exercice 4

Soient $A \in \mathcal{M}_n(\mathbb{R})$ inversible, $\Delta A \in \mathcal{M}_n(\mathbb{R})$ telle que $(A+\Delta A)$ inversible, $(x, \Delta x, b) \in (\mathbb{R}^n)^3$ tels que $A.x = (A+\Delta A).(x+\Delta x) = b \neq 0$, $\|.\|$ la 2-norme sur $\mathcal{M}_n(\mathbb{R})$, $K(A)$ le nombre de conditionnement de A associé à $\|.\|$. Montrer que : $\dfrac{\|\Delta x\|}{\|x+\Delta x\|} \leq K(A).\dfrac{\|\Delta A\|}{\|A\|}$

Exercice 5

Soit $A \in \mathcal{M}_n(\mathbb{R})$. On note $\|.\|$ la 2-norme de $\mathcal{M}_n(\mathbb{R})$.
Soient $\lambda_i \in \mathbb{C}^n \quad i = 1, \ldots, n$ les valeurs propres de A, on note $\rho(A)$, le rayon spectral de A, défini par $\rho(A) = \max\limits_{i=1,\ldots,n} |\lambda_i|$

1) Montrer que $\|A\| = \sqrt{\rho({}^tA.A)}$

2) Que devient ce résultat pour une matrice symétrique ?

2.4 Systèmes à structure particulière

2.4.1 Systèmes triangulaires

Les algorithmes de résolution d'un système triangulaire inférieur (resp. supérieur) sont appelés algorithmes de descente (resp. remontée). Seul le cas d'un algorithme de descente sera traité.

Soit $A.x = b \qquad b = [b_i] \quad i = 1, \ldots, n$
$A = [a_{ij}] \quad i, j = 1, \ldots, n$ avec $\forall (i,j) \in \{1, \ldots, n\}^2 \;\; a_{ij} = 0$ si $j > i$ et $a_{ii} \neq 0$

Algorithme sans reports

$a_{11}x_1 = b_1 \;\Rightarrow\; x_1 = \dfrac{b_1}{a_{11}}$

$a_{21}x_1 + a_{22}x_2 = b_2 \;\Rightarrow\; x_2 = \dfrac{1}{a_{22}}(b_2 - a_{21}x_1)$

$i = 3, \ldots, n \qquad a_{i1}x_1 + \ldots + a_{ii-1}x_{i-1} + a_{ii}x_i = b_i \;\Rightarrow\; x_i = \dfrac{1}{a_{ii}}(b_i - \sum\limits_{j=1}^{i-1} a_{ij}x_j)$

Algorithme avec reports

$x_1 = \dfrac{b_1}{a_{11}} \qquad i = 2, \ldots, n \qquad b_i^{(1)} = b_i - a_{i1}x_1$

$x_2 = \dfrac{b_2^{(1)}}{a_{22}} \qquad i = 3, \ldots, n \qquad b_i^{(2)} = b_i^{(1)} - a_{i2}x_2$

$j = 3, \ldots, n-1 : \qquad x_j = \dfrac{b_j^{(j-1)}}{a_{jj}} \qquad i = j+1, \ldots, n \qquad b_i^{(j)} = b_i^{(j-1)} - a_{ij}x_j$

$x_n = \dfrac{b_n^{(n-1)}}{a_{nn}}$

2.4.2 Systèmes tridiagonaux

$$A = \begin{pmatrix} a_1 & e_1 & 0 & \dots & 0 \\ f_2 & a_2 & e_2 & \ddots & 0 \\ 0 & \ddots & \ddots & \ddots & 0 \\ 0 & \ddots & \ddots & \ddots & e_{n-1} \\ 0 & \dots & 0 & f_n & a_n \end{pmatrix} = tridiag(f_i, a_i, e_i)$$

On peut, dans ce cas, soit utiliser une méthode de résolution générale en l'adaptant à la structure particulière de A, soit utiliser une méthode spécifique. Parmi ces dernières méthodes, l'algorithme de réduction cyclique (qui appartient à la famille des algorithmes de réduction) est le plus connu.

Cas où $n = 2^p - 1$

La première étape de réduction globale consiste à effectuer une réduction élémentaire pour chacune des équations de numéro pair $i = 1, \dots, 2^{p-1} - 1$:

$$\begin{aligned} Eq.\ (2i-1)\ : &\quad f_{2i-1}x_{2i-2} + a_{2i-1}x_{2i-1} + e_{2i-1}x_{2i} = b_{2i-1} \\ Eq.\ (2i)\ : &\quad f_{2i}x_{2i-1} + a_{2i}x_{2i} + e_{2i}x_{2i+1} = b_{2i} \\ Eq.\ (2i+1)\ : &\quad f_{2i+1}x_{2i} + a_{2i+1}x_{2i+1} + e_{2i+1}x_{2i+1} = b_{2i+1} \end{aligned}$$

Si a_{2i-1} et a_{2i+1} sont non nuls, la combinaison linéaire :

$$\frac{-f_{2i}}{a_{2i-1}} Eq.(2i-1) + Eq.(2i) + \frac{-e_{2i}}{a_{2i+1}} Eq.(2i+1)$$

correspond à l'équation $(2i)$ réduite : $f_{2i}^{(1)}x_{2i-2} + a_{2i}^{(1)}x_{2i} + e_{2i}^{(1)}x_{2i+2} = b_{2i}^{(1)}$ avec :

$$f_{2i}^{(1)} = -\frac{f_{2i-1}f_{2i}}{a_{2i-1}} \qquad a_{2i}^{(1)} = a_{2i} - \frac{f_{2i}e_{2i-1}}{a_{2i-1}} - \frac{e_{2i}f_{2i+1}}{a_{2i+1}}$$

$$e_{2i}^{(1)} = -\frac{e_{2i}e_{2i+1}}{a_{2i+1}} \qquad b_{2i}^{(1)} = b_{2i} - \frac{b_{2i-1}f_{2i}}{a_{2i-1}} - \frac{b_{2i+1}e_{2i}}{a_{2i+1}}$$

A l'issue de cette première étape de réduction et en ne considérant que les équations de numéro pair, un nouveau système est obtenu et ce nouveau système est toujours tridiagonal ($tridiag(f_{2i}^{(1)}, a_{2i}^{(1)}, e_{2i}^{(1)})$) avec un nombre d'inconnues ramené à $2^{p-1} - 1$.

Si la réduction est effectuée $(p-1)$ fois, seule l'équation centrale subsiste sous la forme :

$$a_{2^{p-1}}^{(p-1)}.x_{2^{p-1}} = b_{2^{p-1}}^{(p-1)}$$

Aprés calcul de $x_{2^{p-1}}$, on reprend le système obtenu après $(p-2)$ étapes de réduction : il s'agit d'un système tridiagonal 3×3 dont l'inconnue centrale est connue $\Rightarrow$ on en déduit deux autres inconnues, puis on remonte au système précédent $7 \times 7 \dots\dots$ Ce procédé, qui comporte $(p-1)$ étapes jusqu'à obtention de la solution complète, est appelé restitution.

Cas général : $n \neq 2^p - 1$

Suivant la valeur de n, en appliquant des réductions successives, on obtient soit une seule équation, soit un système de deux équations à deux inconnues. L'algorithme est globalement inchangé.

2.4.3 Systèmes tridiagonaux par blocs

Par rapport à un système tridiagonal simple :
- les réels a_i deviennent des blocs carrés A_i
- les réels e_i (resp. f_i) deviennent des blocs rectangulaires E_i (resp. F_i)

Même remarque que pour un système tridiagonal simple : on peut utiliser un algorithme de résolution général adapté à la structure particulière de la matrice ou un algorithme spécifique. Dans les cas usuels (cf. cours E.D.P. 2ème A), les blocs A_i sont généralement tridiagonaux et de mêmes dimensions, les blocs E_i et F_i sont carrés et diagonaux. On peut alors généraliser l'algorithme de réduction cyclique :
- Lors d'une réduction élémentaire, les divisions par $a_i^{(r)}$ (devant être nécessairement non nuls dans le cas d'un sytème tridiagonal simple) sont remplacées par une multiplication, à droite, par $A_i^{(r)-1}$ (devant être nécessairement inversibles).
- Quelque soit l'étape de réduction, les blocs $A_i^{(r)}$ restent tridiagonaux, les blocs $E_i^{(r)}$ et $F_i^{(r)}$ restent diagonaux.
- Chaque réduction élémentaire externe (au niveau des blocs) fait appel à plusieurs réductions internes (résolution d'un système tridiagonal simple).

Chapitre 3

Méthodes de résolution directes

3.1 Triangularisation de Gauss, Algorithme de base

Objectif : Transformer le système $A.x = b$ en $U.x = b'$ avec U matrice triangulaire supérieure inversible.
Soit $A = [a_{ij}^{(1)}] \quad x = [x_i] \quad b = [b_i^{(1)}] \quad i, j = 1, \ldots, n$

L'opération de triangularisation se compose de $(n-1)$ étapes :

Description de l'étape 1 *:*

Représentation du système à l'état 1 :

$$A^{(1)} = \begin{pmatrix} a_{11}^{(1)} & \ldots & a_{1n}^{(1)} \\ \vdots & & \vdots \\ a_{n1}^{(1)} & \ldots & a_{nn}^{(1)} \end{pmatrix} \qquad b^{(1)} = \begin{pmatrix} b_1^{(1)} \\ \vdots \\ b_n^{(n)} \end{pmatrix}$$

L'équation 1 donne : $x_1 = \dfrac{1}{a_{11}^{(1)}}(b_1^{(1)} - \sum_{j=2}^{n} a_{1j}^{(1)} x_j)$

On remplace x_1 par cette expression dans les équations $i = 2, \ldots, n$ ce qui correspond à la mise à jour (si $a_{11}^{(1)}$ est non nul) :

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

d'où la représentation du système à l'état 2 :

$$A^{(2)} = \begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \ldots & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & \ldots & a_{2n}^{(2)} \\ \vdots & \vdots & & \vdots \\ 0 & a_{n2}^{(2)} & \ldots & a_{nn}^{(2)} \end{pmatrix} \qquad b^{(2)} = \begin{pmatrix} b_1^{(1)} \\ b_2^{(2)} \\ \vdots \\ b_n^{(2)} \end{pmatrix}$$

Etape 2 : Elimination de x_2 dans les équations $3, \ldots, n$ à partir de l'équation 2
. . . . . .

Description de l'étape k :

Représentation du système à l'état k :

$$A^{(k)} = \begin{pmatrix} a_{11}^{(1)} & \dots & \dots & \dots & \dots & \dots & a_{1n}^{(1)} \\ \vdots & \ddots & \vdots & \vdots & \vdots & & \vdots \\ 0 & \dots & a_{k-1k-1}^{(k-1)} & \vdots & \vdots & & \vdots \\ 0 & \dots & 0 & a_{kk}^{(k)} & a_{kk+1}^{(k)} & \dots & a_{kn}^{(k)} \\ \vdots & & \vdots & \vdots & \vdots & & \vdots \\ 0 & \dots & 0 & a_{nk}^{(k)} & a_{nk+1}^{(k)} & \dots & a_{nn}^{(k)} \end{pmatrix}$$

$$^t b^{(k)} = \begin{pmatrix} b_1^{(1)} & \dots & b_{k-1}^{(k-1)} & b_k^{(k)} & \dots & b_n^{(k)} \end{pmatrix}$$

si $a_{kk}^{(k)} \neq 0$:

$$i = k+1, \dots, n \qquad j = k+1, \dots, n \qquad a_{ij}^{(k+1)} = a_{ij}^{(k)} - \frac{a_{ik}^{(k)} a_{kj}^{(k)}}{a_{kk}^{(k)}}$$

$$b_i^{(k+1)} = b_i^{(k)} - \frac{a_{ik}^{(k)} b_k^{(k)}}{a_{kk}^{(k)}}$$

Représentation du système à l'état $k+1$:

$$A^{(k+1)} = \begin{pmatrix} a_{11}^{(1)} & \dots & \dots & \dots & \dots & \dots & a_{1n}^{(1)} \\ \vdots & \ddots & \vdots & \vdots & \vdots & & \vdots \\ 0 & \dots & a_{k-1k-1}^{(k-1} & \vdots & \vdots & & \vdots \\ 0 & \dots & 0 & a_{kk}^{(k)} & a_{kk+1}^{(k)} & \dots & a_{kn}^{(k)} \\ 0 & \dots & 0 & 0 & a_{k+1k+1}^{(k+1)} & \dots & a_{k+1n}^{(k+1)} \\ \vdots & & \vdots & \vdots & \vdots & & \vdots \\ 0 & \dots & 0 & 0 & a_{nk+1}^{(k+1)} & \dots & a_{nn}^{(k+1)} \end{pmatrix}$$

$$^t b^{(k+1)} = \begin{pmatrix} b_1^{(1)} & \dots & b_{k-1}^{(k-1)} & b_k^{(k)} & b_{k+1}^{(k+1)} & \dots & b_n^{(k+1)} \end{pmatrix}$$

A l'issue de l'étape $(n-1)$, on est alors ramené à la résolution d'un système triangulaire supérieur $U.x = b'$ (opération de remontée) avec :

$$U = \begin{pmatrix} a_{11}^{(1)} & \dots & a_{1n}^{(1)} \\ \vdots & \ddots & \vdots \\ 0 & \dots & a_{nn}^{(n)} \end{pmatrix} \qquad b' = \begin{pmatrix} b_1^{(1)} \\ \vdots \\ b_n^{(n)} \end{pmatrix}$$

La matrice U est mémorisée dans la partie triangulaire supérieure de la matrice A, le vecteur b' remplace le second membre b. Les coefficients de la partie triangulaire inférieure de A ne sont pas explicitement mis à zéro.

D'où l'algorithme et le pseudo-code :

$\underline{Pour} \quad k = 1 \quad \underline{à} \quad n-1$

$\quad \underline{Pour} \quad i = k+1 \quad \underline{à} \quad n$

$\quad\quad b_i \quad \longleftarrow \quad b_i - \dfrac{a_{ik} b_k}{a_{kk}}$

$\quad\quad \underline{Pour} \quad j = k+1 \quad \underline{à} \quad n$

$\quad\quad\quad a_{ij} \quad \longleftarrow \quad a_{ij} - \dfrac{a_{ik} a_{kj}}{a_{kk}}$

$\quad\quad \underline{Finpour}$

$\quad \underline{Finpour}$

$\underline{Finpour}$

Exercice 6

On considère le pseudo-code associé à la triangularisation de Gauss qui transforme une système $A.x = b, \quad A \in \mathcal{M}_n(\mathbb{R})$ inversible, en un système $U.x = b'$ avec U matrice triangulaire supérieure inversible. On supposera que cette triangularisation peut être effectuée sans pivotage. Modifiez ce pseudo-code de manière à transformer le système $A.x = b$ en un système $D.x = b''$ où D est une matrice diagonale inversible.

3.2 Stratégies de pivotage

On appelle pivots les coefficients $a_{kk}^{(k)} \quad k = 1, \ldots, n$ intervenant au cours de l'opération de triangularisation.

Si, au cours de l'étape k de la triangularisation $a_{kk}^{(k)} = 0$ et si la matrice A est inversible, alors il existe au moins un coefficient $a_{ik}^{(k)} \neq 0 \quad i = k+1, \ldots, n$. Il suffit alors de permuter les lignes i et k du système (y compris les coefficients du second membre) et de poursuivre l'algorithme initial. Une telle technique s'appelle stratégie de pivotage.

De même si $a_{kk}^{(k)}$ est proche de 0, et compte tenu des opérations associées à l'étape k (mise à jour de coefficients comprenant une division par $a_{kk}^{(k)}$), on peut être trés pessimiste sur le comportement numérique de la méthode. Une stratégie de pivotage peut alors être systématiquement envisagée. Il existe classiquement deux stratégies :

Stratégie de pivotage partiel
Echange systématique, à l'étape k, des lignes i et k, avec i choisi tel que : $|a_{ik}^{(k)}| = \max\limits_{p=k,\ldots,n} |a_{pk}^{(k)}|$

Stratégie de pivotage total
Echange systématique, à l'étape k, des lignes i et k, puis des colonnes j et k, avec i et j choisis tels que : $|a_{ij}^{(k)}| = \max\limits_{p,m=k,\ldots,n} |a_{pm}^{(k)}|$
Dans ce dernier cas (permutation de colonnes), l'ordre des inconnues est modifié et l'ensemble des permutations doit être sauvegardé pour permettre un réordonnancement lors de la remontée (vecteur ou matrice de permutations supplémentaire).

3.3 Application au calcul de A^{-1}

Soit $A^{-1} = (X_1, \ldots, X_n)$ et $\{e_j\} \quad j = 1, \ldots, n$ base canonique de $\mathbb{R}^n$

$$A.A^{-1} = I \;\Leftrightarrow\; A.X_j = e_j \quad j = 1, \ldots, n$$

Soit $I = [a^{(1)}_{in+j}] \qquad i, j = 1, \ldots, n$

On effectue la triangularisation en opérant simultanément sur des seconds membres différents.
Représentation du système au début de l'étape 1 *:*

$$\begin{pmatrix} a^{(1)}_{11} & \ldots & a^{(1)}_{1n} & a^{(1)}_{1n+1} & \ldots & a^{(1)}_{12n} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ a^{(1)}_{n1} & \ldots & a^{(1)}_{nn} & a^{(1)}_{nn+1} & \ldots & a^{(1)}_{n2n} \end{pmatrix} \; i.e. \; \begin{pmatrix} a^{(1)}_{11} & \ldots & a^{(1)}_{1n} & 1 & \ldots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ a^{(1)}_{n1} & \ldots & a^{(1)}_{nn} & 0 & \ldots & 1 \end{pmatrix}$$

Représentation du système en fin de l'étape $n-1$ *:*

$$\begin{pmatrix} a^{(1)}_{11} & \ldots & a^{(1)}_{1n} & a^{(1)}_{1n+1} & \ldots & a^{(1)}_{12n} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \ldots & a^{(n)}_{nn} & a^{(n)}_{nn+1} & \ldots & a^{(n)}_{n2n} \end{pmatrix} \; i.e. \; \begin{pmatrix} a^{(1)}_{11} & \ldots & a^{(1)}_{1n} & 1 & \ldots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \ldots & a^{(1)}_{nn} & a^{(n)}_{nn+1} & \ldots & 1 \end{pmatrix}$$

Il reste alors à effectuer n opérations de remontée :

$$A.X_j = e_j \qquad j = 1, \ldots, n \Leftrightarrow U.X_j = {}^t(0 \; \ldots \; 1 \; a^{(n)}_{j+1n+j} \; \ldots \; a^{(n)}_{nn+j}) \qquad j = 1, \ldots, n$$

3.4 Factorisation de Gauss

Soit $A^{(k)}$ la matrice représentant la matrice A à l'état k. Soit $L^{(k)}$ la matrice définie par :

$$L^{(k)} = \begin{pmatrix} 1 & 0 & \ldots & \ldots & \ldots & \ldots & 0 \\ 0 & \ddots & \ddots & & & & \vdots \\ \vdots & & 1 & \ddots & & & \vdots \\ \vdots & & 0 & 1 & \ddots & & \vdots \\ \vdots & & \vdots & r_{k+1k} & 1 & \ddots & \vdots \\ \vdots & & \vdots & \vdots & 0 & \ddots & 0 \\ 0 & \ldots & 0 & r_{nk} & 0 & \ldots & 1 \end{pmatrix}$$

avec $r_{ik} = -\dfrac{a^{(k)}_{ik}}{a^{(k)}_{kk}} \qquad i = k+1, \ldots, n$

alors $A^{(k+1)} = L^{(k)}.A^{(k)}$ et $U = L^{(n-1)} \ldots L^{(2)}.L^{(1)}.A$ or

- $L^{(n-1)} \ldots L^{(2)}.L^{(1)}$ est une matrice triangulaire inférieure
- L'inverse d'une matrice triangulaire inférieure est triangulaire inférieure.

Soit donc L matrice triangulaire inférieure définie par :

$$L = (L^{(n-1)} \ldots L^{(2)}.L^{(1)})^{-1} = L^{(1)-1}.L^{(2)-1} \ldots L^{(n-1)-1}$$

Par ailleurs $L^{(k)-1}$ possède la même structure que $L^{(k)}$ avec comme $k$ème colonne :

$$^t(0 \ \ldots \ 0 \ 1 \ -r_{k+1k} \ \ldots \ -r_{nk})$$

d'où $L^{-1}.U = A \ \Leftrightarrow \ A = L.U$ et

$$L = \begin{pmatrix} 1 & 0 & \ldots & 0 \\ -r_{21} & 1 & & \vdots \\ \vdots & & \ddots & \vdots \\ -r_{n1} & \ldots & -r_{nn-1} & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & \ldots & 0 \\ \frac{a_{21}^{(1)}}{a_{11}^{(1)}} & 1 & & \vdots \\ \vdots & & \ddots & \vdots \\ \frac{a_{n1}^{(1)}}{a_{11}^{(1)}} & \ldots & \frac{a_{nn-1}^{(n-1)}}{a_{n-1n-1}^{(n-1)}} & 1 \end{pmatrix}$$

Remarque 1 : L est inversible à diagonale unitaire. Si, dans la suite des calculs, la matrice initiale A n'est plus nécessaire, il est donc possible de mémoriser L et U dans la même structure de donnée que A : U dans la partie triangulaire supérieure diagonale comprise, L dans la partie triangulaire inférieure (sa diagonale implicitement unitaire est non mémorisée).

D'où l'algorithme et le pseudo-code :

Pour $k = 1$ *à* $n-1$
  Pour $i = k+1$ *à* n
    $a_{ik} \longleftarrow \dfrac{a_{ik}}{a_{kk}}$
    Pour $j = k+1$ *à* n
      $a_{ij} \longleftarrow a_{ij} - a_{ik}a_{kj}$
    Finpour
  Finpour
Finpour

Remarque 2 : L'équation $A = L.U$ n'a de sens que si aucun pivotage est effectué au cours de la factorisation. Dans le cas contraire, $P.A = L.U$ avec P matrice de permutation. Si toutes les sous-matrices principales de A sont inversibles, la factorisation (ou la triangularisation) de Gauss peut s'appliquer sans pivotage.

Exercice 7

Soit $A \in \mathcal{M}_n(\mathbb{R})$ inversible et factorisable sans pivotage par la méthode de Gauss. Soit le partitionnement de A suivant :

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \text{ où } A_{11} \in \mathcal{M}_q(\mathbb{R}) \quad (q < n)$$

Le complément de Schur de A_{11} dans A est $S = A_{22} - A_{21}.A_{11}^{-1}.A_{12}$

On applique la factorisation de Gauss sans pivotage à la matrice A. Après les q premières étapes de la factorisation, la matrice A_{22} a été mise à jour pour devenir la matrice A'_{22}. Montrer que $A'_{22} = S$.

Exercice 8

Soit $A \in \mathcal{M}_n(\mathbb{R})$ inversible et $b \in \mathbb{R}^n$. Connaissant les coefficients de A, on souhaite résoudre le système $A^2.x = b$ en utilisant la factorisation de Gauss. On supposera que les matrices A et A^2 peuvent être factorisées sans pivotage. La complexité, en nombre d'opérations

virgule flottante, d'une factorisation de Gauss dans $\mathcal{M}_n(\mathbb{R})$ est de l'ordre de $\frac{2n^3}{3}$, celle d'une descente ou d'une remontée de l'ordre de n^2.

Déterminer le nombre d'opérations virgule flottante nécessaire pour obtenir le vecteur x pour chacune des stratégies suivantes :

a) En choisissant de calculer A^2 puis de factoriser cette matrice

b) En choisissant de factoriser A

3.5 Cas d'une matrice symétrique définie positive : factorisation de Cholesky

A symétrique définie positive $\Leftrightarrow$ $\exists L = [l_{ij}]$ matrice réelle d'ordre n, triangulaire inférieure et inversible telle que $A = L.{}^tL$.

Remarque : L n'est pas à diagonale unitaire, la factorisation de Cholesky n'est pas l'équivalent de la factorisation de Gauss dans le cas d'une matrice symétrique définie positive.

3.5.1 Démonstration du caractère défini positif

Soit $A = L.{}^tL$
${}^tA = {}^t(L.{}^tL) = L.{}^tL = A$ donc A est symétrique
$\forall x \in \mathbb{R}^n \quad (A.x|x) = (L.{}^tL.x|x) = ({}^tL.x|{}^tL.x) \geq 0$
$(A.x|x) = 0 \;\Rightarrow\; {}^tL.x = 0 \;\Rightarrow\; x = 0$ $\quad A$ est définie positive

3.5.2 Démonstration de l'existence de la factorisation et algorithme

A symétrique définie positive
Soit $A = M.U \quad (M = [m_{ij}],\ U = [u_{ij}])$, la factorisation de A par la méthode de Gauss (A étant symétrique définie positive, le déterminant des sous-matrices principales est strictement positif, ces sous-matrices sont donc inversibles, et la factorisation de Gauss peut s'appliquer sans pivotage).

On veut trouver $L = [l_{ij}]$ telle que $A = L.{}^tL$. Procédons par identification à partir des équations de l'égalité $A = M.U$.

Calcul de la 1ère colonne

Equation diagonale : $m_{11}u_{11} = a_{11}$
Il est possible de choisir $l_{11} = m_{11} = u_{11}$ car $a_{11} > 0$ (sous-matrice principale de rang 1) : $l_{11} = \sqrt{a_{11}}$

Pour les équations $i = 2, \ldots, n$:

$$l_{11}.l_{i1} = a_{i1} \;\Rightarrow\; l_{i1} = \frac{a_{i1}}{l_{11}}$$

Supposons obtenues les $(j-1)$ premières colonnes de L.

Calcul de la jème colonne

Equation diagonale :

$$l_{j1}^2 + l_{j2}^2 + \ldots + l_{jj-1}^2 + m_{jj}u_{jj} = a_{jj}$$

Est-il possible de choisir $l_{jj} = m_{jj} = u_{jj}$? Soit A_j (resp. L_j) la sous-matrice principale de rang j de A (resp. de L).

$$\begin{aligned} det(A_j) &= m_{jj}.det(L_{j-1}).u_{jj}.det(L_{j-1}) = m_{jj}u_{jj}.[det(L_{j-1})]^2 > 0 \\ &\Rightarrow \quad m_{jj}u_{jj} > 0 \Rightarrow \quad a_{jj} - \sum_{k=1}^{n} l_{jk}^2 > 0 \end{aligned}$$

$$l_{jj} = \sqrt{a_{jj} - \sum_{k=1}^{j-1} l_{jk}^2} > 0$$

Pour les équations $i = j+1, \ldots, n$:

$$l_{i1}l_{j1} + l_{i2}l_{j2} + \ldots + l_{ij}l_{jj} = a_{ij} \;\Rightarrow\; l_{ij} = \frac{1}{l_{jj}}(a_{ij} - \sum_{k=1}^{j-1} l_{jk}l_{ik})$$

Exercice 9

On considère la matrice symétrique, définie positive suivante d'ordre $3n$ (A, B, C, D_1 **et** D_2 **étant des blocs** $n \times n$) :

$$M = \begin{pmatrix} A & 0 & {}^tD_1 \\ 0 & B & {}^tD_2 \\ D_1 & D_2 & C \end{pmatrix}$$

On cherche à implanter l'algorithme de Cholesky, associé à M**, sur un calculateur qui possède une librairie mathématique qui contient les sous-programmes suivants :**

- **Factorisation par Cholesky d'une matrice symétrique définie positive de rang maximal** n
- **Résolution d'un système triangulaire (inférieur ou supérieur) de rang maximal** n
- **Multiplication, addition et soustraction de matrices de dimensions maximales** $n \times n$

En utilisant uniquement cette librairie (aucun calcul flottant effectué en dehors de ses sous-programmes), comment mettre en œuvre l'algorithme ?

Chapitre 4

Méthodes de résolution itératives

4.1 Méthodes itératives de relaxation

On cherche une suite $\{x^{(p)}\}$ de $\mathbb{R}^n$ convergeant vers x, solution de $A.x = b$, quelque soit le vecteur initial $x^{(0)} \in \mathbb{R}^n$.

Algorithme de relaxation : $x^{(p+1)}$ diffère de $x^{(p)}$ par une seule composante.
Algorithme itératif (de relaxation) : $x^{(p+1)}$ diffère de $x^{(p)}$ par plusieurs composantes (en général toutes).

Tous les méthodes itératives de relaxation sont issues de l'**algorithme de relaxation de Southwell** (présenté rapidement ci-dessous). En pratique, cet algorithme n'est pas utilisé compte tenu de sa faible vitesse de convergence.

A la relaxation p : on note $\rho^{(p)}$ le vecteur résidu relatif à $x^{(p)}$ i.e. $\rho^{(p)} = A.x^{(p)} - b$
choix de la composante $n^\circ i$ à relaxer : $1 \leq i \leq n \ \ t.q. \ \ |\rho_i^{(p)}| \ = \max_{j=1,\ldots,n} |\rho_j^{(p)}|$

La relaxation consiste à annuler le résidu sur la composante $n^\circ i$:

$$a_{i1}x_1^{(p)} + \ldots + a_{ii-1}x_{i-1}^{(p)} + a_{ii}x_i^{(p+1)} + a_{ii+1}x_{i+1}^{(p)} + \ldots + a_{in}x_n^{(p)} = b_i$$

d'où $x_i^{(p+1)} = \dfrac{1}{a_{ii}}(b_i - \sum_{j=1, j\neq i}^{n} a_{ij}x_j^{(p)})$ (si $a_{ii} \neq 0$)

$$x_i^{(p+1)} = \frac{1}{a_{ii}}(b_i - \sum_{j=1}^{n} a_{ij}x_j^{(p)} + a_{ii}x_i^{(p)}) = x_i^{(p)} - \frac{\rho_i^{(p)}}{a_{ii}}$$

$$m = 1, \ldots, n \qquad m \neq i \qquad x_m^{(p+1)} = x_m^{(p)}$$

Remarque : A la relaxation suivante $\rho_m^{(p+1)} \quad m \neq i$ se déduit de $\rho_m^{(p)}$ par $\rho_m^{(p+1)} = \rho_m^{(p)} - \dfrac{a_{mi}\rho_i^{(p)}}{a_{ii}}$

Théorème de convergence : Si A est symétrique définie positive alors,
$\forall x^{(0)} \in \mathbb{R}^n$, les relaxations de Southwell convergent vers la solution de $A.x = b$.

4.1.1 Algorithme itératif de Gauss-Seidel

Une itération correspond à n relaxations consécutives (la relaxation $n°i$ tient compte du résultat des relaxations $n°1$ à $n°i-1$) de type Southwell mais il n'y a pas de choix dynamique de la composante à relaxer.

Au cours de l'itération $n°p$, les valeurs $x_i^{(p+1)} \quad i = 1, \ldots, n$ sont obtenues en résolvant successivement les équations (R_i) :

$$(R_i) \qquad a_{i1}x_1^{(p+1)} + \ldots + a_{ii-1}x_{i-1}^{(p+1)} + a_{ii}x_i^{(p+1)} + a_{ii+1}x_{i+1}^{(p)} + \ldots + a_{in}x_n^{(p)} = b_i$$

Soient alors les 3 matrices D, E et F définies par : $A = D - E - F$

- D : matrice diagonale de coefficients $(a_{11}, \ldots, a_{nn})$
- E : matrice triangulaire inférieure à diagonale nulle
  $E = [-a_{ij}] \qquad i = 2, \ldots, n \quad j = 1, \ldots, i-1$
- F : matrice triangulaire supérieure à diagonale nulle
  $F = [-a_{ij}] \qquad i = 1, \ldots, n-1 \quad j = i+1, \ldots, n$

$$\begin{aligned}(R_i) \quad i \quad &= \quad 1, \ldots, n \quad \Leftrightarrow \quad (D-E).x^{(p+1)} - F.x^{(p)} = b \\ &\Leftrightarrow \quad x^{(p+1)} = (D-E)^{-1}.F.x^{(p)} + (D-E)^{-1}.b\end{aligned}$$

(x est solution de l'équation de point fixe : $x = (D-E)^{-1}.F.x + (D-E)^{-1}.b$)

On dit alors que l'algorithme de Gauss-Seidel est issu de la décomposition de A sous la forme $(M-N)$ avec $M = D - E$ et $N = F$. A toute décomposition de ce type est associé un algorithme itératif, si M est inversible, qui s'écrit $x^{(p+1)} = M^{-1}.N.x^{(p)} + M^{-1}.b$.

4.1.2 Algorithme itératif de Jacobi

Une itération correspond à n relaxations indépendantes (une relaxation ne tient pas compte des relaxations qui l'ont précédées au cours de l'itération courante) de type Southwell et il n'y a pas de choix dynamique de la composante à relaxer.

Au cours de l'itération $n°p$, les valeurs $x_i^{(p+1)} \quad i = 1, \ldots, n$ sont obtenues en résolvant parallèlement les équations :

$$(R_i) \qquad a_{i1}x_1^{(p)} + \ldots + a_{ii-1}x_{i-1}^{(p)} + a_{ii}x_i^{(p+1)} + a_{ii+1}x_{i+1}^{(p)} + \ldots + a_{in}x_n^{(p)} = b_i$$

Soient alors les 3 matrices D, E et F définies précédemment $(A = D - E - F)$:

$$\begin{aligned}(R_i) \quad i \quad &= \quad 1, \ldots, n \quad \Leftrightarrow \quad D.x^{(p+1)} - (E+F).x^{(p)} = b \\ &\Leftrightarrow \quad x^{(p+1)} = D^{-1}.(E+F).x^{(p)} + D^{-1}.b\end{aligned}$$

(x est solution de l'équation de point fixe : $x = D^{-1}.(E+F).x + D^{-1}.b$)

On dit alors que l'algorithme de Jacobi est issu de la décomposition de A sous la forme $(M-N)$ avec $M = D$ et $N = E + F$.

4.1.3 Etude de la convergence

Tests d'arrêt

Que ce soit pour l'algorithme de Gauss-Seidel ou de Jacobi, il est nécessaire de mettre en œuvre un test d'arrêt permettant de vérifier la convergence entre deux itérations.
Après choix d'une norme vectorielle et d'une précision ϵ :

- $\|x^{(p)} - x^{(p-1)}\| < \epsilon$ simple à mettre en œuvre mais risque d'arrêt loin de la solution.
- $\|A.x^{(p)} - b\|$ i.e. $\|\rho^{(p)}\| < \epsilon$ numériquement plus correct et le calcul du résidu en fin d'itération p peut-être ré-utilisé pour exprimer l'itération $p+1$:

$$\begin{aligned} x^{(p+1)} &= M^{-1}.N.x^{(p)} + M^{-1}.b = M^{-1}.(M-A).x^{(p)} + M^{-1}.b \\ &= M^{-1}.[b - A.x^{(p)}] + x^{(p)} \end{aligned}$$

Condition nécessaire et suffisante de convergence

Soit un algorithme de résolution itératif issu d'une décomposition de A sous une forme $M - N$:

$$A.x = b \;\Leftrightarrow\; (M-N).x = b \;\Leftrightarrow\; Mx = Nx + b \;\Leftrightarrow\; x = M^{-1}.N.x + M^{-1}.b$$

Algorithme itératif associé : $x^{(p+1)} = M^{-1}.N.x^{(p)} + M^{-1}.b$

Convergence de l'algorithme $\Leftrightarrow \quad \forall x^{(0)} \in \mathbb{R}^n \quad \lim_{p\to+\infty} x^{(p)} = x$
Soit $x^{(p)} = x + \epsilon^{(p)}$
Convergence de l'algorithme $\Leftrightarrow \quad \forall \epsilon^{(0)} \in \mathbb{R}^n \quad \lim_{p\to+\infty} \epsilon^{(p)} = 0$

$$x^{(p+1)} = (M^{-1}.N).x^{(p)} + M^{-1}.b \qquad x = M^{-1}.N.x + M^{-1}.b$$

$$\Rightarrow \quad \epsilon^{(p+1)} = M^{-1}.N.\epsilon^{(p)} \quad \Rightarrow \quad \epsilon^{(p)} = (M^{-1}.N)^p.\epsilon^{(0)}$$

Convergence de l'algorithme $\Leftrightarrow \quad \forall \epsilon^{(0)} \in \mathbb{R}^n \quad \lim_{p\to+\infty} (M^{-1}.N)^p.\epsilon^{(0)} = 0$
$\Leftrightarrow \lim_{p\to+\infty} (M^{-1}.N)^p = 0$ (matrice nulle) $\Leftrightarrow \quad \rho(M^{-1}.N) < 1$ (CNS admise)
ρ représente ici le rayon spectral i.e. pour une matrice A, $\rho(A)$ est la plus grande, en module, des valeurs propres de A.

Autre version de la CNS : $\forall \lambda$ valeur propre de $M^{-1}.N$: $|\lambda| < 1$ ($|.|$ représente ici le module d'un nombre complexe)

Conditions suffisantes de convergence

Théorème de convergence 1 (se déduisant de la CNS précédente) :
Si A est à diagonale dominante stricte i.e. $\quad i = 1, \ldots, n \quad |a_{ii}| > \sum_{j=1, j\neq i}^{n} |a_{ij}|$
les algorithmes de Jacobi et de Gauss-Seidel convergent, quelque soit le vecteur initial.

Théorème 2 (se déduisant de la CNS précédente)

Si A est à diagonale dominante i.e. $\quad i = 1, \ldots, n \quad |a_{ii}| \geq \sum_{j=1, j\neq i}^{n} |a_{ij}|$

et si $\quad \exists k \in \{1, \ldots, n\} \quad |a_{kk}| > \sum_{j=1, j\neq k}^{n} |a_{kj}|$

l'algorithme de Gauss-Seidel converge, quelque soit le vecteur initial.

Théorème 3 (se déduisant de la CNS précédente)

Si A est symétrique définie positive, l'algorithme de Gauss-Seidel converge, quelque soit le vecteur initial.

Exercice 10

Soit le système $A.x = b$ avec $A \in \mathcal{M}_n(\mathbb{R})$ inversible. On considère une décomposition de A sous la forme $M - N$. On supposera (pour simplifier) que toutes les valeurs propres de la matrice $M^{-1}.N$ sont réelles et que tous les vecteurs propres associés appartiennent à $\mathbb{R}^n$. Montrer la propriété suivante (réciproque de la propriété admise) :

La méthode itérative de relaxation associée à la décomposition de A converge vers x quelque soit le vecteur initial $\Longrightarrow \rho(M^{-1}.N) < 1$

Exercice 11

Soit $A \in \mathcal{M}_n(\mathbb{R})$ inversible et à diagonale dominante stricte :
$A = [a_{ij}] \quad i, j = 1, \ldots n$

$$i = 1, \ldots n \qquad |a_{ii}| > \sum_{j=1, j\neq i}^{n} |a_{ij}|$$

On note $A = D - E - F$ la décomposition de A vue dans le cadre des méthodes itératives de relaxation. On supposera (pour simplifier) que toutes les valeurs propres des matrices $D^{-1}.(E + F)$ et $(D - E)^{-1}.F$ sont réelles et que tous les vecteurs propres associés appartiennent à $\mathbb{R}^n$. On choisit comme norme de $\mathbb{R}^n$:

$$\forall x = [x_i] \in \mathbb{R}^n \qquad ||x|| = \max_{i=1,\ldots,n} |x_i|$$

1) Montrez que les matrices D et $(D - E)$ sont effectivement inversibles.

2) Soit λ une valeur propre de $D^{-1}.(E + F)$ et $X = [X_i] \quad i = 1, \ldots, n$ un vecteur propre associé tel que $||X|| = 1$. Montrez qu'il existe $k \in \{1, \ldots, n\}$ tel que $\quad |\lambda a_{kk}| \leqslant \sum_{j=1, j\neq k}^{n} |a_{kj}|.|X_j|$

3) En déduire la convergence de la méthode de Jacobi pour la résolution d'un système $A.x = b$ (résultat admis mais ici à démontrer).

4) Soit λ une valeur propre de $(D - E)^{-1}.F$ et $X = [X_i] \quad i = 1, \ldots, n$ un vecteur propre associé tel que $||X|| = 1$. Montrez qu'il existe $k \in \{1, \ldots, n\}$ tel que $\quad |\lambda a_{kk}| \leqslant |\lambda| \sum_{j=1}^{k-1} |a_{kj}|.|X_j| + \sum_{j=k+1}^{n} |a_{kj}|.|X_j|$

5) En déduire la convergence de la méthode de Gauss-Seidel pour la résolution d'un système $A.x = b$ (résultat admis mais ici à démontrer).

4.1.4 Méthode SOR (Successive Over-Relaxation)

La méthode de Gauss-Seidel, plus efficace que la méthode de Jacobi (caractère plus implicite : on profite des derniers résultats calculés), peut néanmoins converger trés lentement si $\rho(M^{-1}.N)$ est inférieur mais proche de 1. On peut alors introduire un paramètre de relaxation ($\omega \in \mathbb{R}$) pour accélérer la convergence :
A l'itération $n^\circ p$:

$$\begin{aligned} i &= 1,\ldots,n \qquad x_i^{(p+1)} = \frac{\omega}{a_{ii}}(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(p+1)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(p)}) + (1-\omega)x_i^{(p)} \\ &= \omega(Calcul\ de\ x_i^{(p+1)}\ résultant\ de\ Gauss-Seidel) + (1-\omega)x_i^{(p)} \end{aligned}$$

Si $\omega = 1$, la méthode itérative est celle de Gauss-Seidel.

Cette formulation est déduite d'une décomposition de A sous la forme $M_\omega - N_\omega$ avec

$$M_\omega = \frac{1}{\omega}(D - \omega E) \qquad N_\omega = \frac{1}{\omega}[(1-\omega)D + \omega F]$$

Le choix optimal de ω, minimisant $\rho(M_\omega^{-1}.N_\omega)$, est difficile dans le cas général mais certains résultats sont disponibles pour des cas particuliers. Par exemple :

Théorème 1 :
Si A est symétrique définie positive, la méthode SOR converge si et seulement si $0 < \omega < 2$.

Théoréme 2 :
Si A est symétrique définie positive et tridiagonale par blocs, alors il existe un et un seul paramètre de relaxation optimal $\omega^\star$:
$\rho(M_{\omega^\star}^{-1}.N_{\omega^\star}) = \inf_{0<\omega<2} \rho(M_\omega^{-1}.N_\omega) = \omega^\star - 1$

$$\omega^\star = \frac{2}{1+\sqrt{1-\rho(D^{-1}.(E+F))^2}} = \frac{2}{1+\sqrt{1-\rho(M^{-1}.N\ de\ Jacobi)^2}}$$

4.2 Méthodes de gradient dans le cas d'une matrice symétrique définie positive

4.2.1 Méthode de la Steepest Descent

(méthode de la plus grande pente)

Propriété 1

Soit $F : \mathbb{R}^n \longrightarrow \mathbb{R}$ définie par $F(x) = \frac{1}{2}({}^t x.A.x) - {}^t x.b$
Si A est symétrique alors $A.x - b = grad(F(x))$
Rappel : $F : \mathbb{R}^n \longrightarrow \mathbb{R} \qquad grad(F(x)) = {}^t(\frac{\partial f}{\partial x_1}(x) \ \ldots \ \frac{\partial f}{\partial x_n}(x))$

démonstration :

$$\begin{aligned}\frac{\partial F}{\partial x_j}(x) &= \frac{1}{2}\frac{\partial}{\partial x_j}(\sum_{i=1}^{n} x_i(Ax)_i) - \frac{\partial}{\partial x_j}(\sum_{i=1}^{n} x_i b_i) \\ &= \frac{1}{2}\frac{\partial}{\partial x_j}(\sum_{i=1}^{n}(a_{i1}x_1 + \ldots + a_{in}x_n)x_i) - b_j\end{aligned}$$

$$= \frac{1}{2}(\sum_{i=1}^{n}(a_{i1}x_1 + \ldots + a_{in}x_n)\delta_{ij} + \sum_{i=1}^{n} a_{ij}x_i) - b_j$$

$$= \frac{1}{2}(a_{j1}x_1 + \ldots + a_{jn}x_n + \sum_{i=1}^{n} a_{ij}x_i) - b_j = \sum_{i=1}^{n} a_{ij}x_i - b_j = (A.x - b)_j$$

Propriété 2

La solution x de $A.x = b$ est le centre de toutes les hyperquadriques Q_k d'équations $F(x) = k$.

démonstration :
Il faut montrer que si $x' = x + u \in Q_k$ alors $x" = x - u \in Q_k$

$$F(x') = k \Leftrightarrow \frac{1}{2}{}^t(x+u).A.(x+u) - {}^t(x+u).b = k$$

$$\Leftrightarrow \frac{1}{2}{}^t x.A.x + \frac{1}{2}{}^t u.A.u + \frac{1}{2}{}^t x.A.u + \frac{1}{2}{}^t u.A.x - {}^t x.b - {}^t u.b = k$$

Or $\frac{1}{2}{}^t u.A.x + \frac{1}{2}{}^t x.A.u - {}^t u.b = \frac{1}{2}{}^t u.b + \frac{1}{2}{}^t b.u - {}^t u.b = 0$ donc

$$F(x') = k \Leftrightarrow F(x) + \frac{1}{2}{}^t u.A.u = k \Leftrightarrow F(x) + \frac{1}{2}{}^t(-u).A.(-u) = k$$

$$\Leftrightarrow \frac{1}{2}{}^t(x-u).A.(x-u) - {}^t(x-u).b = k \Leftrightarrow x" \in Q_k$$

Propriété 3

x solution de $A.x = b \Leftrightarrow \forall x' \neq x \quad F(x') > F(x)$

démonstration :
i) Soit $A.x = b$ et $x' = x + u$ avec $u \neq 0$: $F(x') = F(x) + \frac{1}{2}{}^t u.A.u$

$$\frac{1}{2}{}^t u.A.u > 0 \Rightarrow F(x') > F(x)$$

ii) Soit $\forall x \neq x' \quad F(x') > F(x) \Rightarrow grad(F(x)) = 0 = A.x - b$

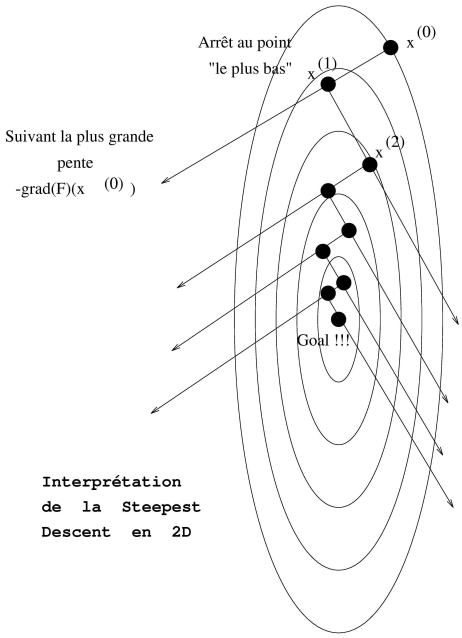


FIG. 4.1 – Comportement de la Steepest Descent en 2D

Algorithme

Soit A symétrique définie positive, $x^{(0)} \in \mathbb{R}^n$ donné, $x^{(p+1)}$ se déduit de $x^{(p)}$ par :

$$x^{(p+1)} = x^{(p)} + \lambda_p r^{(p)}$$

$r^{(p)}$: direction de descente (vecteur)
λ_p : progression dans la direction de descente (scalaire)

Choix de la direction de descente en $x^{(p)}$: sens de la plus forte pente (stratégie du skieur), c'est à dire suivant $r^{(p)} = -(grad(F(x))(x^{(p)}) = b - Ax^{(p)}$

Choix de la progression λ_p en $x^{(p)}$: comme la solution est atteinte au minimum de F, on cherche à minimiser $G(\lambda_p) = F(x^{(p)} + \lambda_p r^{(p)}) - F(x^{(p)})$

$$\begin{aligned} G(\lambda_p) &= \frac{1}{2}\,{}^t(x^{(p)} + \lambda_p r^{(p)}).A.(x^{(p)} + \lambda_p r^{(p)}) - {}^t(x^{(p)} + \lambda_p r^{(p)}).b \\ &\quad -\frac{1}{2}\,{}^t x^{(p)}.A.x^{(p)} + {}^t x^{(p)}.b \end{aligned}$$

$$\begin{aligned} &= \frac{1}{2}\,{}^t x^{(p)}.A.x^{(p)} + \frac{1}{2}\lambda_p^2({}^t r^{(p)}.A.r^{(p)}) + \frac{1}{2}\lambda_p({}^t x^{(p)}.A.r^{(p)} + {}^t r^{(p)}.A.x^{(p)}) - \\ &\quad {}^t x^{(p)}.b - \lambda_p {}^t r^{(p)}.b - \frac{1}{2}\,{}^t x^{(p)}.A.x^{(p)} + {}^t x^{(p)}.b \end{aligned}$$

$$G(\lambda_p) = \frac{1}{2}\lambda_p^2({}^t r^{(p)}.A.r^{(p)}) + \frac{1}{2}\lambda_p({}^t x^{(p)}.A.r^{(p)} + {}^t\, r^{(p)}.A.x^{(p)}) - \lambda_p^t r^{(p)}.b$$

$$= \frac{1}{2}\lambda_p^2({}^t r^{(p)}.A.r^{(p)}) + \lambda_p({}^t r^{(p)}.A.x^{(p)}) - \lambda_p^t r^{(p)}.b \;\; car \;\; {}^t x^{(p)}.A.r^{(p)} \;=\; {}^t r^{(p)}.A.x^{(p)}$$

$$\begin{aligned} &= \frac{1}{2}\lambda_p^2({}^t r^{(p)}.A.r^{(p)}) + \lambda_p[{}^t r^{(p)}.(b - r^{(p)}) - {}^t\, r^{(p)}.b] \\ &= \frac{1}{2}\lambda_p^2({}^t r^{(p)}.A.r^{(p)}) - \lambda_p^t r^{(p)}.r^{(p)} \end{aligned}$$

Le minimum de G est obtenu pour : $\lambda_p = \frac{{}^t r^{(p)}.r^{(p)}}{{}^t r^{(p)}.A.r^{(p)}}$
d'où l'algorithme :

$x^{(0)} = \ldots\ldots$
$r^{(0)} = b - A.x^{(0)}$
$p = 0$
tant que $(\|r^{(p)}\| < \epsilon)$ *faire*

$\lambda_p = \frac{{}^t r^{(p)}.r^{(p)}}{{}^t r^{(p)}.A.r^{(p)}}$
$x^{(p+1)} = x^{(p)} + \lambda_p r^{(p)}$
$r^{(p+1)} = b - A.x^{(p+1)}$
$p = p + 1$
fin

La convergence est assurée si la matrice est symétrique définie positive.

4.2.2 Variante de la Steepest Descent avec paramètre fixe

$x^{(0)}$ donné, $x^{(p+1)}$ se déduit de $x^{(p)}$ par $x^{(p+1)} = x^{(p)} + \alpha r^{(p)}$ avec $\alpha > 0$

$$x^{(p+1)} = (I - \alpha A).x^{(p)} + \alpha I.b$$

Il s'agit donc d'un algorithme de relaxation issu d'une décomposition de A sous la forme $(M - N)$ avec $M^{-1}.N = I - \alpha A$ et $M^{-1} = \alpha I$ i.e.

$$M = \frac{1}{\alpha} I \qquad N = \frac{1}{\alpha} I - A$$

CNS de convergence : $\rho(I - \alpha A) < 1$ (ρ désigne ici le rayon spectral).

Si λ_i est une valeur propre de A, alors $(1 - \alpha\lambda_i)$ est une valeur propre de $(I - \alpha A)$.

$$\begin{aligned} \rho(I - \alpha A) \;<\; 1 \;&\Leftrightarrow\; i = 1, \ldots, n \quad |1 - \alpha\lambda_i| < 1 \\ &\Leftrightarrow\quad i = 1, \ldots, n \quad -1 < 1 - \alpha\lambda_i < 1 \end{aligned}$$

$$\Leftrightarrow \;\; i = 1, \ldots, n \quad 0 < \alpha\lambda_i < 2 \;\; \Leftrightarrow \;\; 0 < \alpha < \frac{2}{\lambda_n}$$

On peut également trouver le paramètre α_{opt} (qui minimise $\rho(I - \alpha A)$) :

$$\rho(I - \alpha A) = \max_{i=1,\ldots,n} |1 - \alpha\lambda_i|$$

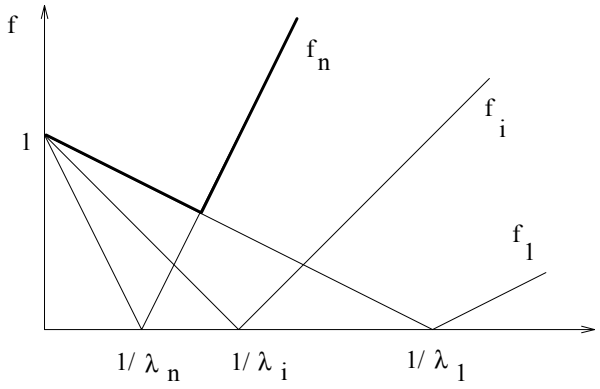


FIG. 4.2 – Représentation des fonctions $f_i(\alpha)$

Soit $f_i(\alpha) = |1 - \alpha\lambda_i|$

Le trait double de la figure 4.2 représente le maximum de $f(\alpha) = \rho(I - \alpha A)$

α_{opt} est tel que $1 - \alpha_{opt}\lambda_1 = -1 + \alpha_{opt}\lambda_n \quad \Rightarrow \quad \alpha_{opt} = \dfrac{2}{\lambda_1 + \lambda_n}$

Exercice 12

Soit le système linéaire $A.x = b$, avec $A \in \mathcal{M}_n(\mathbb{R})$ symétrique définie positive. On considère la résolution de ce système par la méthode de la Steepest Descent : une itération s'écrit $x^{(p+1)} = x^{(p)} + \lambda_p r^{(p)}$ avec $r^{(p)}$ la direction de descente et λ_p la progression dans la direction de descente.

1) Montrez que deux directions de descente successives (par exemple, $r^{(p+1)}$ et $r^{(p)}$) sont orthogonales.

2) Soit λ une valeur propre de A et u un vecteur propre associé à λ. On suppose que le vecteur initial $x^{(0)} \in \mathbb{R}^n$ vérifie :
$\exists \beta \in \mathbb{R} \qquad \beta \neq 0 \qquad tq \qquad x - x^{(0)} = \beta u$
Montrez que la méthode de la Steepest Descent atteint alors la solution exacte en une seule itération.

3) Montrez que, si toutes les valeurs propres de la matrice A sont égales (A possède une seule valeur propre), alors, quelque soit le vecteur $x^{(0)}$, la méthode de la Steepest Descent atteint la solution exacte en une seule itération.

Exercice 13

Soit à résoudre $A.x = b$ avec $A \in \mathcal{M}_n(\mathbb{R})$ symétrique définie positive. On définit le $A-$*produit scalaire* par :
$\forall x \in \mathbb{R}^n \quad \forall y \in \mathbb{R}^n \qquad (x|y)_A = (x|A.y)$
On note $u_i \;\; i = 1, \ldots, n$ les différentes colonnes de A.

1) En adaptant le procédé d'orthogonalisation de Schmidt, expliciter un algorithme permettant de construire une base $\{v_i\}\ \ i = 1, \ldots, n\ \ A-$ *orthogonale* **de** $\mathbb{R}^n$ **à partir des colonnes de** A :

$$i, j = 1, \ldots, n \quad i \neq j \qquad (v_i | v_j)_A = 0$$

2) Exprimer la solution du système x **dans la base** $\{v_i\}\ \ i = 1, \ldots, n$

3) Montrer que : $\forall y \in \mathbb{R}^n \qquad x = y + \sum_{i=1}^{n} \frac{(b - A.y | v_i)}{(v_i | A.v_i)} v_i$

4) On considère l'algorithme itératif suivant :

$$\forall x^{(1)} \in \mathbb{R}^n$$

$$x^{(p+1)} = x^{(p)} + \alpha_p v_p \quad avec \quad \alpha_p = \frac{(v_p | b - A.x^{(p)})}{(v_p | A.v_p)}$$

Montrer que pour $p = 2, 3, \ldots \quad (v_p | A.x^{(p)}) = (v_p | A.x^{(1)})$

5) Montrer que l'algorithme de la question ci-dessus atteint exactement la solution du système $A.x = b$ **en** n **itérations.**

4.2.3 Méthode du Gradient Conjugué

Cette méthode est plus largement détaillée dans des documents annexes. Le principe est similaire à celui de la plus grande pente mais les directions de descente $v^{(p)}$ choisies sont $A-$*orthogonale* (cf. exercice ci-dessus). L'algorithme atteint exactement la solution du système $A.x = b$ en n itérations :

$x^{(0)} = \ldots\ldots$
$r^{(0)} = b - A.x^{(0)}$
$v^{(0)} = r^{(0)}$
$p = 0$
tant que $(\|r^{(p)}\| < \epsilon)$ *faire*

$\lambda_p = \frac{{}^t r^{(p)}.r^{(p)}}{{}^t v^{(p)}.A.v^{(p)}}$
$x^{(p+1)} = x^{(p)} + \lambda_p v^{(p)}$
$r^{(p+1)} = b - A.x^{(p+1)}$
$v^{(p+1)} = r^{(p+1)} + \frac{{}^t r^{(p+1)}.r^{(p+1)}}{{}^t r^{(p)}.r^{(p)}} v^{(p)}$
$p = p + 1$
fin

Chapitre 5

Localisation des valeurs propres et équation caractéristique

5.1 Localisation des valeurs propres

5.1.1 Théorème de Hadamard-Gerchorin

Si $A \in \mathcal{M}_n(\mathbb{C})$, les valeurs propres de A ont des images dans le plan complexe qui appartiennent à l'union des ensembles $D_i \quad i = 1, \ldots, n$ où

$$D_i = \{z \in \mathbb{C} \ / \ |z - a_{ii}| \ \leq \sum_{j=1, j\neq i}^{n} |a_{ij}|\}$$

Démonstration : Soit λ valeur propre de A, $x = [x_i]$ vecteur propre associé. x peut être choisi tel que $|x_r| \ = 1 \qquad r \in \{1, \ldots, n\} \qquad$ et $\qquad |x_i| \ \leq 1 \qquad i \neq r$

$$\sum_{j=1}^{n} a_{rj}x_j = \lambda x_r \ \Rightarrow \ \sum_{j=1, j\neq r}^{n} a_{rj}x_j = (\lambda - a_{rr})x_r$$

$$|\lambda - a_{rr}| \ = | \sum_{j=1, j\neq r}^{n} a_{rj}x_j| \ \leq \sum_{j=1, j\neq r}^{n} |a_{rj}|.|x_j| \ \leq \sum_{j=1, j\neq r}^{n} |a_{rj}| \ \Rightarrow \ \lambda \in D_r$$

5.1.2 Corollaire

i) $\rho(A) \leq \max\limits_{i=1,\ldots,n} \sum\limits_{j=1}^{n} |a_{ij}| \qquad$ *ii)* $\rho(A) \leq \max\limits_{j=1,\ldots,n} \sum\limits_{i=1}^{n} |a_{ij}|$

Démonstration :

i) $D_i \subset D'_i = \{z \in \mathbb{C} \ / \ |z| \ \leq \sum\limits_{j=1}^{n} |a_{ij}|\}$

Soit λ valeur propre quelconque : $\lambda \in \bigcup\limits_{i=1}^{n} D_i \subset \bigcup\limits_{i=1}^{n} D'_i$

$$\bigcup_{i=1}^{n} D'_i = \{z \in \mathbb{C} \ / \ |z| \ \leq \max_{i=1,\ldots,n} \sum_{j=1}^{n} |a_{ij}|\} \ \Rightarrow \ \rho(A) \leq \max_{i=1,\ldots,n} \sum_{j=1}^{n} |a_{ij}|$$

ii) Comme les valeurs propres de A sont également les valeurs propres de ${}^tA \Rightarrow$
•

5.2 Recherche de l'équation caractéristique d'une matrice

Polynôme caractéristique : $x \in \mathbb{R} \quad P(x) = det(A - xI) = |A - xI|$
autre définition $P(x) = |xI - A|$
Equation caractéristique : $|A - xI| = |xI - A| = 0$

5.2.1 Relations de Newton entre les racines d'un polynôme

Remarque préalable : Soit $P(x) = a_0x^n + a_1x^{n-1} + \ldots + a_n = (x-\alpha)Q(x) + \beta_n$
avec $Q(x) = \beta_0x^{n-1} + \beta_1x^{n-2} + \ldots + \beta_{n-1} \qquad \beta_i \in \mathbb{R} \quad i = 1, \ldots, n$
où $\beta_0 = a_0$ et $\beta_p = a_p + \alpha\beta_{p-1} \quad p = 1, \ldots, n$

Soit donc le polynôme P ci-dessus de racines $x_i \quad i = 1, \ldots, p \qquad (p \leq n)$
Soit m_i l'ordre de multiplicité de la racine x_i
Soit $S_k = \sum_{i=1}^{p} m_i x_i^k \quad k = 1, \ldots, n$
Les relations de Newton s'écrivent :

$$a_0S_1 + a_1 = 0$$

$$a_0S_2 + a_1S_1 + 2a_2 = 0$$

$$\ldots\ldots$$

$$a_0S_k + a_1S_{k-1} + \ldots + a_{k-1}S_1 + ka_k = 0$$

$$\ldots\ldots$$

$$a_0S_n + a_1S_{n-1} + \ldots + a_{n-1}S_1 + na_n = 0$$

Démonstration :

Dans le cas de n racines simples (se généralise facilement)

$$P(x) = a_0 \prod_{i=1}^{n} (x - x_i) \qquad P'(x) = \sum_{i=1}^{n} \frac{P(x)}{x - x_i}$$

On utilise la division euclidienne présentée en remarque préalable.
Le reste $\beta_n = 0$ car $P(x)$ est divisible par $(x - x_i)$.

$$\begin{aligned} P'(x) &= x^{n-1} \sum_{i=1}^{n} a_0 + x^{n-2} \sum_{i=1}^{n} (a_1 + a_0x_i) + x^{n-3} \sum_{i=1}^{n} (a_2 + x_i(a_1 + a_0x_i)) + \ldots \\ &\quad + \sum_{i=1}^{n} \beta_{n-1}(x_i) \end{aligned}$$

$$P'(x) = na_0x^{n-1} + (na_1 + a_0S_1)x^{n-2} + (na_2 + a_1S_1 + a_0S_2)x^{n-3} + \ldots + \sum_{i=1}^{n} \beta_{n-1}(x_i)$$

D'autre part, $P'(x) = na_0x^{n-1} + (n-1)a_1x^{n-2} + \ldots + a_{n-1}$

En identifiant les deux formulations de $P'(x)$, on obtient les $(n-1)$ premières relations de Newton. Pour la dernière, on somme les n relations suivantes :

$$P(x_1) = a_0x_1^n + \ldots + a_{n-1}x_1 + a_n = 0$$

$$\ldots\ldots$$

$$P(x_n) = a_0x_n^n + \ldots + a_{n-1}x_n + a_n = 0$$

$$\Rightarrow \quad a_0S_n + \ldots + a_{n-1}S_1 + na_n = 0$$

5.2.2 Méthode de Leverrier

Soit $P(\lambda) = |\lambda I - A| = \lambda^n - a_1\lambda_{n-1} - \ldots - a_{n-1}\lambda - a_n$

Soit $S_p = \sum_{i=1}^{p} m_i\lambda_i^p$

Pour $p = 1, \ldots, n$, S_p peut être calculé par une propriété (admise) :

$$S_1 = trace(A), \quad S_2 = trace(A^2), \quad \ldots, \quad S_n = trace(A^n)$$

Les coefficients $(a_1, \ldots, a_n)$ peuvent alors être calculés à l'aide des relations de Newton, par résolution d'un système triangulaire inférieur :

$S_1 - a_1 = 0$
$S_2 - a_1S_1 - 2a_2 = 0$
$\ldots\ldots$
$S_n - a_1S_{n-1} - \ldots - a_{n-1}S_1 - na_n = 0$

$$\begin{pmatrix} 1 & 0 & \ldots & \ldots & 0 \\ S_1 & 2 & & & \vdots \\ S_2 & S_1 & 3 & & \vdots \\ \vdots & & \ddots & \ddots & \vdots \\ S_{n-1} & S_{n-2} & \ldots & S_1 & n \end{pmatrix} \cdot \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} S_1 \\ S_2 \\ \vdots \\ \vdots \\ S_n \end{pmatrix}$$

Chapitre 6

Recherche des valeurs propres et vecteurs propres d'une matrice symétrique

6.1 Obtentions successives : méthode de la puissance itérée

$A \in \mathcal{M}_n(\mathbb{R})$. L'algorithme permet d'obtenir la plus grande des valeurs propres en module, si celle-ci est réelle, ainsi qu'un vecteur propre associé. Avec une hypothèse supplémentaire (toutes les valeurs propres de A sont réelles, distinctes en valeur absolue et non nulles), on peut obtenir, en appliquant plusieurs fois l'algorithme, toutes les valeurs propres et les vecteurs propres associés. L'algorithme sera présenté avec cette dernière hypothèse.

6.1.1 Algorithme initial

Soient $|\lambda_1| > |\lambda_2| > \ldots > |\lambda_n| > 0$ l'ordonnancement des valeurs propres de A. Soit $(X_1, \ldots, X_n)$ une base de vecteurs propres associés.

Soit $V_0 \in \mathbb{R}^n$ (En pratique le choix de V_0 est arbitraire cf. remarque plus loin).

$$V_0 = \sum_{i=1}^{n} c_i X_i$$

Soit la suite $\{V_p\}$ de vecteurs, définie par $V_{p+1} = \dfrac{A.V_p}{\|A.V_p\|}$
($\|.\|$ norme quelconque de $\mathbb{R}^n$)

$$V_1 = \frac{1}{\|A.V_0\|} A.(\sum_{i=1}^{n} c_i X_i)$$

Soit $\beta_1 = \dfrac{1}{\|A.V_0\|}$ $\qquad V_1 = \beta_1 \sum_{i=1}^{n} c_i \lambda_i X_i$

Soit $\beta_2 = \dfrac{\beta_1}{\|A.V_1\|}$ $\qquad V_2 = \beta_2 \sum_{i=1}^{n} c_i \lambda_i^2 X_i$ $\quad \ldots\ldots$ etc

Soit $\beta_p = \dfrac{\beta_{p-1}}{\|A.V_{p-1}\|}$ $\qquad V_p = \beta_p \sum_{i=1}^{n} c_i \lambda_i^p X_i$

$$= \beta_p \lambda_1^p (c_1 X_1 + c_2 (\frac{\lambda_2}{\lambda_1})^p X_2 + \ldots + c_n (\frac{\lambda_n}{\lambda_1})^p X_n)$$

Soit $\epsilon_p = c_2 (\frac{\lambda_2}{\lambda_1})^p X_2 + \ldots + c_n (\frac{\lambda_n}{\lambda_1})^p X_n$
$V_p = \beta_p \lambda_1^p (c_1 X_1 + \epsilon_p)$ avec $\lim_{p \to +\infty} \epsilon_p = 0$ d'où

$$\lim_{p \to +\infty} \|V_{2p}\| = \lim_{p \to +\infty} \beta_{2p} \lambda_1^{2p} \|c_1 X_1\| = 1$$

$$\Rightarrow \lim_{p \to +\infty} \beta_{2p} \lambda_1^{2p} = \frac{1}{\|c_1 X_1\|} \Rightarrow \lim_{p \to +\infty} V_{2p} = \frac{c_1 X_1}{\|c_1 X_1\|} = W_1$$

avec W_1 vecteur propre normé.

Remarque : l'algorithme n'est applicable que si $c_1 \neq 0$, le choix de V_0 n'est donc pas tout à fait libre. En pratique, le choix de V_0 est arbitraire, la probabilité d'un échec étant faible.

$$\lim_{p \to +\infty} A.V_{2p} = A.W_1 \Rightarrow \lim_{p \to +\infty} \|A.V_{2p}\| = |\lambda_1|$$

Il reste à déterminer le signe de λ_1 par comparaison des signes de V_{2p} et V_{2p+1} :

$$V_{2p} = \beta_{2p} \lambda_1^{2p} (c_1 X_1 + \epsilon_{2p}) \qquad V_{2p+1} = \frac{\beta_{2p}}{\|A.V_{2p}\|} \lambda_1^{2p+1} (c_1 X_1 + \epsilon_{2p+1})$$

(par exemple, sur la composante la plus grande en valeur absolue).
Une autre possibilité consiste à calculer :

$$\alpha_{2p} = (V_{2p} | A.V_{2p}) \qquad \lim_{p \to +\infty} \alpha_{2p} = \lambda_1$$

6.1.2 Opération de déflation

Modification de la matrice initiale pour trouver les autres valeurs propres en appliquant de nouveau l'algorithme.

Cas d'une matrice symétrique

Soit $B = A - \lambda_1 W_1 . {}^t W_1$

Propriétés de la matrice B :
- $B.W_1 = A.W_1 - \lambda_1 W_1 . ({}^t W_1 . W_1) = A.W_1 - \lambda_1 W_1 = 0$
- pour $i = 2, \ldots, n$ $\quad B.X_i = A.X_i - \lambda_1 W_1 . {}^t W_1 . X_i$
  Or : $A.W_1 = \lambda_1 W_1 \Rightarrow {}^t W_1 . A = \lambda_1 {}^t W_1 \Rightarrow {}^t W_1 . A . X_i = \lambda_1 {}^t W_1 . X_i$

  $\Rightarrow \lambda_i {}^t W_1 . X_i = \lambda_1 {}^t W_1 . X_i \Rightarrow (\lambda_i - \lambda_1) {}^t W_1 . X_i = 0 \Rightarrow {}^t W_1 . X_i = 0$

  donc $B.X_i = A.X_i = \lambda_i X_i$

Le rang de B est donc égal à $(n-1)$, B est symétrique, B possède les mêmes valeurs propres $\lambda_i \quad i = 2, \ldots, n$ que A et les mêmes vecteurs propres associés. On peut donc appliquer l'algorithme de la puissance itérée à B pour trouver λ_2 et W_2.

Si le sous-espace propre associé à λ_1 est de dimension > 1, la puissance itérée, appliquée à B, produira de nouveau λ_1 et ${W_1}'$ un vecteur propre associé non colinéaire à W_1.

Exercice 14

Soit $Q \in \mathcal{M}_{m,n}(\mathbb{R})$ $(m \geqslant n)$ matrice orthogonale $({}^tQ.Q = I_n \in \mathcal{M}_n(\mathbb{R}))$. Soit X le sous- espace vectoriel, de dimension n, dont les vecteurs colonnes de la matrice Q forment une base orthogonale. Montrer que $P_X = Q.{}^tQ \in \mathcal{M}_m(\mathbb{R})$ est le projecteur orthogonal sur X.

Exercice 15

Soient $A \in \mathcal{M}_n(\mathbb{R})$, λ_1 une valeur propre de A et W_1 un vecteur propre normé associé à λ_1.
$\forall x \in \mathbb{R}^n$, on notera $x_\perp$ la projection orthogonale de x sur le sous-espace orthogonal à W_1
Soit $B = A - \lambda_1 W_1.{}^tW_1$. Montrer que $\forall x \in \mathbb{R}^n \qquad B.x = A.x_\perp$

Cas d'une matrice non symétrique

(l'algorithme est applicable tant que les valeurs propres restent réelles).

Soit $L \in \mathbb{R}^n$ t.q. ${}^tL.W_1 = 1$ (par exemple $L = W_1$). Soit $B = A - \lambda_1 W_1.{}^tL$

Propriétés de la matrice B :

- $B.W_1 = A.W_1 - \lambda_1 W_1.({}^tL.W_1) = AW_1 - \lambda_1 W_1 = 0$
- pour $i = 2, \ldots, n$ soit $Z_i = X_i - \dfrac{\lambda_1}{\lambda_i}({}^tL.X_i).W_1$

$$B.Z_i = B.X_i - \frac{\lambda_1}{\lambda_i}(L^t.X_i).B.W_1 = B.X_i = (A - \lambda_1 W_1.{}^tL).X_i$$

$$= \lambda_i X_i - \lambda_1 W_1.{}^tL.X_i = \lambda_i(X_i - \frac{\lambda_1}{\lambda_i}({}^tL.X_i)W_1) = \lambda_i Z_i$$

Mêmes conclusions qu'au paragraphe précédent, sauf que les vecteurs propres de B sont les vecteurs $Z_i \quad i = 2, \ldots, n$. En appliquant l'algorithme de la puissance itérée, on obtient λ_2 et Z_2 puis :

$$\begin{aligned} {}^tL.Z_2 &= {}^tL.X_2 - \frac{\lambda_1}{\lambda_2}({}^tL.X_2).({}^tL.W_1) = \frac{\lambda_1 - \lambda_2}{\lambda_2}({}^tL.X_2) \\ &\Rightarrow \quad {}^tL.X_2 = \frac{\lambda_2}{\lambda_2 - \lambda_1}({}^tL.Z_2) \end{aligned}$$

d'où $Z_2 = X_2 - \dfrac{\lambda_1\lambda_2}{\lambda_2(\lambda_1 - \lambda_2)}({}^tL.Z_2).W_1 \quad \Rightarrow \quad X_2 = Z_2 + \dfrac{\lambda_1}{\lambda_2 - \lambda_1}({}^tL.Z_2).W_1$

et $W_2 = \dfrac{X_2}{\|X_2\|}$

Exercice 16

1) Soit $A \in \mathcal{M}_n(\mathbb{R})$. Soient λ une valeur propre de A et u un vecteur propre associé à λ.Soit $\alpha \in \mathbb{R}$ qui n'est pas une valeur propre de A. Montrer que $\mu = \dfrac{1}{\lambda - \alpha}$ est une valeur propre de $(A - \alpha I)^{-1}$ et que, pour cette matrice, u est un vecteur propre associé à μ.

2) On suppose la matrice A symétrique (ou que toutes ses valeurs propres sont réelles)et inversible. Soient $|\lambda_1| \geqslant |\lambda_2| \geqslant \ldots \geqslant |\lambda_n|$ les valeurs propres de A. On note $||.||$ la norme vectorielle euclidienne. Quelles valeurs l'algorithme suivant permet-il d'obtenir ?

$i = 0$, $x_i = \ldots$
Boucler
Résolution du système $A.y_{i+1} = x_i$
$x_{i+1} = \dfrac{y_{i+1}}{||y_{i+1}||}$; $\beta_{i+1} = {}^t x_{i+1}.A.x_{i+1}$
$i = i + 1$
Jusqu'à convergence

3) On suppose la matrice A symétrique (ou que toutes ses valeurs propres sont réelles). Soient $|\lambda_1| \geqslant |\lambda_2| \geqslant \ldots \geqslant |\lambda_n|$ les valeurs propres de A. Soit $\alpha \in \mathbb{R}$ qui n'est pas une valeur propre de A. On note $||.||$ la norme vectorielle euclidienne. Quelles valeurs l'algorithme suivant permet-il d'obtenir ?

$i = 0$, $x_i = \ldots$
Boucler
Résolution du système $(A - \alpha I).y_{i+1} = x_i$
$x_{i+1} = \dfrac{y_{i+1}}{||y_{i+1}||}$; $\beta_{i+1} = {}^t x_{i+1}.A.x_{i+1}$
$i = i + 1$
Jusqu'à convergence

6.2 Obtention simultanée : algorithme de Jacobi

6.2.1 Principe de l'algorithme

C'est un algorithme itératif (génération d'une suite de matrices) permettant d'obtenir simultanément toutes les valeurs propres.

Soient $A_1 = A$ et $A_{k+1} = \Theta_k^{-1}.A_k.\Theta_k$
avec Θ_k matrice orthogonale ($\Theta_k^{-1} = {}^t\Theta_k$)

- ${}^t A_{k+1} = {}^t\Theta_k.{}^t A_k.{}^t\Theta_k^{-1} = \Theta_k^{-1}.A_k.\Theta_k$ A_{k+1} est symétrique
- A_1 possèdant les mêmes valeurs propres que A, supposons la propriété vraie jusqu'à A_k.

$$A_{k+1} = \Theta_k^{-1}.A_k.\Theta_k \Rightarrow A_{k+1}.\Theta_k^{-1} = \Theta_k^{-1}.A_k$$

Soit λ_i une valeur propre de A_k (et de A)
Soit $X_i^{(k)}$ un vecteur propre associé à λ_i pour A_k

$$A_{k+1}.\Theta_k^{-1}.X_i^{(k)} = \Theta_k^{-1}.A_k.X_i^{(k)} = \lambda_i \Theta_k^{-1}.X_i^{(k)}$$

donc λ_i valeur propre de A_{k+1} associée au vecteur $\Theta_k^{-1}.X_i^{(k)}$

A_{k+1} possède donc les mêmes valeurs propres que A.

On voudrait choisir $\{\Theta_k\}$ t.q. $\lim\limits_{n\to+\infty} A_k = D = diag(\lambda_1, \ldots, \lambda_n)$
avec $\lambda_i \quad i = 1, \ldots, n$ les valeurs propres de A.

6.2.2 Détermination des matrices orthogonales

Soit $A_k = [a_{ij}^{(k)}]$

Soit $S_k = \sum\limits_{i=1}^{n} \sum\limits_{j=1}^{n} (a_{ij}^{(k)})^2 - \sum\limits_{i=1}^{n} (a_{ii}^{(k)})^2 > 0$

$$\lim_{k\to+\infty} A_k = D \quad \Leftrightarrow \quad \lim_{k\to+\infty} S_k = 0$$

On choisit alors Θ_k comme la matrice de rotation définie par (i_k, j_k, α_k).
Soient $C = cos(\alpha_k)$ et $S = sin(\alpha_k)$

$$\Theta_k = \begin{pmatrix} 1 & 0 & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & 0 \\ 0 & \ddots & & & & & & & \vdots \\ \vdots & & 1 & & & & & & \vdots \\ \vdots & & & C & \ldots & S & & & \vdots \\ \vdots & & & \vdots & \ddots & \vdots & & & \vdots \\ \vdots & & & -S & \ldots & C & & & \vdots \\ \vdots & & & & & & 1 & & \vdots \\ \vdots & & & & & & & \ddots & 0 \\ 0 & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & 0 & 1 \end{pmatrix} \begin{matrix} \\ \\ \\ \\ \\ i_k \\ \\ j_k \\ \\ \\ \end{matrix}$$

Dans l'expression de $A_{k+1} = [a_{pq}^{(k+1)}]$ en fonction de $A_k = [a_{pq}^{(k)}]$, tous les termes sont inchangés sauf (avec notations simplifiées : i_k noté i, j_k noté j) :

$$p = 1, \ldots, n \quad p \neq i \quad p \neq j \qquad a_{ip}^{(k+1)} = a_{pi}^{(k+1)} = Ca_{ip}^{(k)} + Sa_{jp}^{(k)}$$

$$p = 1, \ldots, n \quad p \neq i \quad p \neq j \qquad a_{jp}^{(k+1)} = a_{pj}^{(k+1)} = -Sa_{ip}^{(k)} + Ca_{jp}^{(k)}$$

$$a_{ii}^{(k+1)} = C^2 a_{ii}^{(k)} + 2CSa_{ij}^{(k)} + S^2 a_{jj}^{(k)}$$

$$a_{jj}^{(k+1)} = S^2 a_{ii}^{(k)} - 2CSa_{ij}^{(k)} + C^2 a_{jj}^{(k)}$$

$$a_{ij}^{(k+1)} = a_{ji}^{(k+1)} = (C^2 - S^2)a_{ij}^{(k)} + CS(a_{jj}^{(k)} - a_{ii}^{(k)})$$

On a donc :

$$(a_{ip}^{(k+1)})^2 + (a_{jp}^{(k+1)})^2 = (a_{ip}^{(k)})^2 + (a_{jp}^{(k)})^2 \qquad S_{k+1} - S_k = 2(a_{ij}^{(k+1)})^2 - 2(a_{ij}^{(k)})^2$$

Pour obtenir une convergence maximale, on cherche à rendre $S_{k+1} - S_k$ le plus négatif possible :

- i et j (ou plus exactement i_k et j_k) sont fixés tels que :
  $|a_{ij}^{(k)}| = \max\limits_{l,m=1,\ldots,n \ \ l\neq m} |a_{lm}^{(k)}|$

– α_k est fixé tel que : $a_{ij}^{(k+1)} = 0 = (C^2 - S^2)a_{ij}^{(k)} + CS(a_{jj}^{(k)} - a_{ii}^{(k)})$
si $a_{jj}^{(k)} = a_{ii}^{(k)}$ alors $\alpha_k = signe(a_{ij}^{(k)})\frac{\pi}{4}$
si $a_{jj}^{(k)} \neq a_{ii}^{(k)}$ alors $a_{ij}^{(k+1)} = 0 \;\Rightarrow\; tg(2\alpha_k) = \frac{2a_{ij}^{(k)}}{a_{ii}^{(k)} - a_{jj}^{(k)}}$ avec $|\alpha_k| < \frac{\pi}{4}$

Vérifions que la suite $\{S_k\}$ converge bien vers 0. Le nombre de termes non nuls contribuant à l'évaluation de S_k est inférieur ou égal à $n^2 - n - 2$ donc :

$$S_k \leq (n^2 - n - 2) \max_{l,m=1,\ldots,n \;\; l \neq m} |a_{lm}^{(k)}|`$$

$$S_{k+1} = S_k - 2 \max_{l,m=1,\ldots,n \;\; l \neq m} |a_{lm}^{(k)}| \;\leq\; S_k(1 - \frac{2}{n^2 - n - 2}) = \epsilon S_k$$

avec $\epsilon < 1$.

6.2.3 Recherche des vecteurs propres

La matrice D fournit les valeurs propres.

$$A_{k+1} = \Theta_k^{-1}.A_k.\Theta_k = \Theta_k^{-1}.(\Theta_{k-1}^{-1}.A_{k-1}.\Theta_{k-1}).\Theta_k = (\Theta_k^{-1} \ldots \Theta_1^{-1}).A.(\Theta_1 \ldots \Theta_k)$$

Soit $\Theta = \lim_{k \to +\infty} \prod_{p=1}^{k} \Theta_k \qquad D = \Theta_{-1}.A.\Theta \;\Rightarrow\; \Theta.D = A.\Theta$

$\Theta(j)$, jème colonne non nulle de Θ est vecteur propre associé à λ_j.

Chapitre 7

Méthode QR pour le calcul des valeurs propres et vecteurs propres d'une matrice non symétrique

7.1 Préambule

7.1.1 Transformations de Householder

Définition 1

$\forall u \in \mathbb{R}^n, u \neq 0$, on définit la matrice de Householder $H \in \mathcal{M}_n(\mathbb{R})$ telle que $H = I - \frac{2}{{}^t u.u} u.{}^t u$.

Propriété 1

H est orthogonale (${}^t H.H = I$) et symétrique.

démonstration : H est évidemment symétrique (${}^t H = H$).

De plus ${}^t H.H = \left(I - \frac{2}{{}^t u.u} u.{}^t u \right)^2 = I + \frac{4}{({}^t u.u)^2} u.{}^t u.u.{}^t u - \frac{4}{{}^t u.u} u.{}^t u$

Donc ${}^t H.H = I + \frac{4}{{}^t u.u} u.{}^t u - \frac{4}{{}^t u.u} u.{}^t u = I$.

Propriété 2

On notera que $Hu = -u$ et, $\forall v \in \mathbb{R}^n$ $H.v = v$ si v est orthogonal à u.

On peut en déduire une interprétation géométrique de la transformation de Householder : l'image $H.v$ d'un vecteur v quelconque non nul est le symétrique par rapport à l'espace orthogonal à u. H est donc un opérateur de réflexion.

Propriété 3

Soit $v \in \mathbb{R}^n$ $(v \neq 0)$, $v = [v_i]$ $i = 1, \ldots, n$ et soit $\sigma = sign(v_1)\|v\|$. Si on définit $u = v + \sigma e_1$ alors $H.v = -\sigma e_1$ (e_1 le premier vecteur de la base canonique).

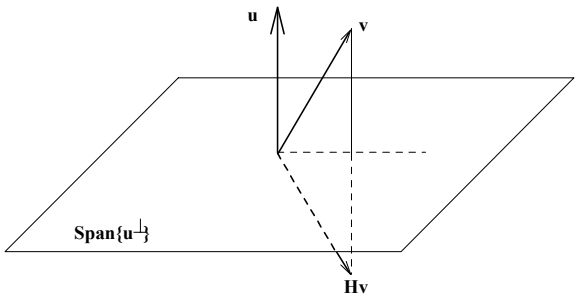


FIG. 7.1 – Réflexion d'un vecteur v

démonstration : Soit $u = v + \sigma e_1$ (donc $u \neq 0$).
$H.v = H.u - \sigma H.e_1 = -u - \sigma \left(e_1 - \frac{2}{{}^t u.u} u.{}^t u.e_1 \right) = -\sigma e_1 + \frac{2\sigma u_1}{{}^t u.u} u - u$
Or ${}^t u.u = {}^t v.u + \sigma {}^t e_1.u = {}^t v\,(v + \sigma e_1) + \sigma\,(\sigma + v_1) = {}^t v.v + 2\sigma v_1 + \sigma^2$
Comme ${}^t v.v = \sigma^2$, et $u_1 = v_1 + \sigma$, on obtient ${}^t u.u = 2\sigma\,(v_1 + \sigma) = 2\sigma u_1$ et donc $H.v = -\sigma e_1$

Remarque : On peut noter le choix de σ dans la Proposition 3. On évite ainsi, lorsque v est proche de e_1, d'avoir ${}^t u.u$ proche de zéro.

7.1.2 Factorisation QR de Householder

Théorème 1
$\forall A \in \mathcal{M}_n(\mathbb{R})$ inversible, il existe $Q \in \mathcal{M}_n(\mathbb{R})$ orthogonale et $R \in M_n(\mathbb{R})$ triangulaire supérieure telles que $A = Q.R$.
démonstration : cf. Exercice 17

Exercice 17
1) Montrer qu'il existe $H_1 \in \mathcal{M}_n(\mathbb{R})$ orthogonale et $\alpha_1 \in \mathbb{R}$ tels que $H_1.A.e_1 = \alpha_1 e_1$(i.e. tous les éléments sous-diagonaux de la colonne 1 de la matrice $H_1.A$ sont nuls).
2) En déduire l'existence d'une suite de matrices orthogonales $H_i \quad i = 1, \ldots, n-1$ telle que $H_{n-1} \ldots H_1.A = R$ avec $R \in \mathcal{M}_n(\mathbb{R})$ matrice triangulaire.

Définition 2
L'algorithme $A = Q.R$ présenté dans l'Exercice 17 décrit la factorisation de Householder.

7.1.3 Complexité de la factorisation de Householder

Mémoire
On ne calcule jamais explicitement les matrices H_i qui sont en générales pleines. On mémorise les vecteurs u_i associés dont la taille est telle que l'on peut utiliser la partie triangulaire de A pour les stocker (plus un vecteur de taille n pour le terme diagonal). Donc l'espace de la matrice intiale (à n près) suffit.

Calcul

Calculons le nombre d'opérations de la factorisation de Householder. La partie coûteuse de l'algorithme est lorsque, à chaque étape k, l'on met à jour la matrice réduite, notée $A_k \in \mathcal{M}_{n-k}(\mathbb{R})$:

$$A_k = A_k - \beta u_k.({}^t u_k.A_k) \quad \text{avec} \quad \beta = \frac{2}{{}^t u_k.u_k} \quad \text{et} \quad u_k \in \mathbb{R}^{n-k}$$

Algorithme 1 : Principales étapes de calcul

Pour $k = 1$ *à* $n-1$
(1) $\quad \beta = \dfrac{2}{{}^t u_k.u_k}$
Pour $j = 1$ *à* $n-k$
/ Soit* Col_j *la colonne* j *de* A_k **/*
(2) $\quad tmp = {}^t u_k \ . \ Col_j$
(3) $\quad tmp = tmp \ . \ \beta$
(4) $\quad Col_j = Col_j - tmp$ x u_k
Finpour
Finpour

Les traitements 2 et 4 coûtent $2(n-k)$ opérations chacune. Les traitementss 1 et 3 sont donc négligeables devant 2 et 4. Le coût de chaque étape k est donc d'environ $2(n-k)^2$ opérations. On peut donc approximer le coût total de l'algorithme par

$$4\sum_{k=0}^{n}(n-k)^2 = 4\sum_{p=0}^{n-1}p^2 \simeq 4\int_0^{n-1}p^2dp = 4\left[p^3/3\right]_0^{n-1} \simeq 4n^3/3$$

L'algorithme est donc de complexité $O(4n^3/3)$.

7.2 Méthode QR de calcul des valeurs et vecteurs propres

La méthode QR de calcul des valeurs propres et des vecteurs propres associés est une des méthodes les plus efficaces et les plus robustes (utilisée dans MATLAB). Nous présentons tout d'abord une premiere version de l'algorithme (Algorithme 2) qui nous servira pour analyser les propriétés de la méthode proposée. Motivés par les résultats d'une analyse de complexité de notre algorithme nous introduirons alors tous les composants permettant de justifer les propriétes d'un nouvel algorithme en Section 7.2.2 (Algorithme 3).

7.2.1 Version initiale de l'algorithme

Soit $A \in \mathcal{M}_n(\mathbb{R})$ une matrice non-symétrique de rang plein.

Algorithme 2 : Version initiale de la méthode QR

$k = 0$; $A_0 = A$
Répéter
  Factoriser $A_k : A_k = Q_k.R_k$ ($^tQ_k.Q_k = I$ *et* R_k *triangulaire)*
  $A_{k+1} = R_k.Q_k$
Jusqu'à convergence

Exercice 18

Montrer, qu'à chaque itération k de la méthode QR, on a
$A_{k+1} = {}^tQ_k.A_k.Q_k$

La méthode QR construit donc une suite de matrices semblables (car $^tQ_k = Q_k^{-1}$) . On peut montrer que cette suite (la suite des A_k) converge globalement (pour les matrices possédant n valeurs propres réelles et distinctes) vers une matrice triangulaire T. On peut alors calculer les valeurs propres et vecteurs propres de T et en deduire (car T est semblable à A) les valeurs propres et vecteurs propres de la matrice A.

Calcul de la complexité de l'algorithme initial

A chaque étape de l'Algorithme 2 une factorisation $Q.R$ de Householder est calculée (coût $O(n^3)$). Si l'on suppose (cas optimiste) que l'on converge en n itérations (1 itération par valeur propre) alors la complexité de l'algorithme complet est $O(n^4)$.

Composants pour un nouvel algorithme

- On peut essayer de transformer la matrice initiale A en une matrice H moins coûteuse à factoriser (structurellement plus simple).
- Afin de préserver les valeurs propres (et être capable de déduire les vecteurs propres), on va chercher à transformer la matrice initiale en une matrice semblable ($H = X^{-1}.A.X$, X inversible).
- De plus pour réduire le coût de la méthode on voudrait X^{-1} facile à calculer (par exemple orthogonale).
- Finalement, pour préserver ces propriétés, il faut que la structure particulière de H soit conservée à chaque itération de l'algorithme lors du produit $H_{k+1} = R_k.Q_k$.

7.2.2 Vers une version plus efficace de l'algorithme

Transformation sous forme Hessenberg

Définition 3
Soit $A \in \mathcal{M}_n(\mathbb{R})$, A est Hessenberg supérieure ssi $i > j + 1 \Longrightarrow a_{ij} = 0$

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

Exemple de matrice Hessenberg supérieure d'ordre 6
('$*$' indique d'un coefficient est non nul)

Afin de préserver les valeurs propres (et être capable de déduire les vecteurs propres), on cherche une matrice orthogonale Q qui transforme la matrice initiale en une matrice Hessenberg et semblable H ($H = Q^{-1}.A.Q$).

Théorème 2

Soit $A \in \mathcal{M}_n(\mathbb{R})$ de rang plein. Il existe une matrice orthogonale Q telle que ${}^tQ.A.Q$ est Hessenberg supérieure.

démonstration : cf. Exercice 19

Exercice 19

Soit $A \in \mathcal{M}_n(\mathbb{R})$ de rang plein.

1) On partitionne la matrice A sous la forme $A = \begin{pmatrix} \alpha_{11} & {}^ta_{12} \\ a_{21} & A_{22} \end{pmatrix}$ avec $\alpha_{11} \in \mathbb{R}$, $A_{22} \in \mathcal{M}_{n-1}(\mathbb{R})$**.**
Soit $\widehat{H}_1 \in \mathcal{M}_{n-1}(\mathbb{R})$ la transformation de Householder telle que $\widehat{H}_1 a_{21} = \beta_1 e_1$.
Soit $H_1 \in M_n(\mathbb{R})$ telle que $H_1 = \begin{pmatrix} 1 & 0 \\ 0 & \widehat{H}_1 \end{pmatrix}$ et soit $B = [b_{ij}] = H_1.A.H_1$.
Montrer que $b_{j1} = 0$ pour $j > 2$

2) Sur le modèle du traitement de la première colonne, construire une suite de matrices de Householder $H_i \quad i = 1, \ldots, n-2$ telle que $H_{n-2} \ldots H_1.A.H_1 \ldots H_{n-2}$ soit une matrice de Hessenberg supérieure.

Rotations de Givens pour factoriser une matrice Hessenberg

Propriété 4

Il suffit de $n-1$ étapes de rotation de Givens pour calculer la factorisation $Q.R$ d'une matrice Hessenberg.

démonstration : Soit $H = [h_{ij}]$ une matrice au format Hessenberg (par exemple supérieur). A chaque étape k ($k = 1, \ldots n-1$) on construit la rotation de Givens $R_k(k, k+1, \theta_k)$ d'angle θ_k permettant d'annuler l'élément $h_{k+1,k}$ de la matrice. Compte tenu des propriétés des matrices de rotations :

- Cette rotation ne modifie que les lignes k et $k+1$ de la matrice.
- De plus une entrée dans la colonne j d'une des deux lignes modifiées est non nulle si et seulement si l'entrée dans la colonne j d'une des deux lignes originales était initialement non nulle.

Aucun nouveau coefficient non nul est donc ajouté, lors de l'étape k, au segment des lignes k et $k+1$ appartenant à la partie triangulaire inférieure de la matrice.

Posons ${}^tQ = R_{n-1}(n-1, n, \theta_{n-1}) \dots R_k(k, k+1, \theta_k) \dots R_1(1, 2, \theta_1)$. Il résulte de la propriété structurelle observée à chaque étape que ${}^tQ.A$ est triangulaire supérieure.

Complexité

A chaque étape k les deux lignes modifées sont combinaisons linéaires des deux lignes initiales (coût $O(n)$). Donc le coût total de l'algorithme de factorisation d'une matrice de Hessenberg est $O(n^2)$.

Algorithme modifié

On suppose qu'à chaque exécution du traitement 3 ci-dessous, la forme Hessenberg supérieure de la matrice est préservée (cf. Exercice 20 pour la démonstration). On peut donc bâtir un nouvel algorithme plus efficace :

Algorithme 3 : Version modifiée de la méthode QR

(1) *Mise sous forme Hessenberg supérieure de* A
$H =^t Q.A.Q$
$k = 0$; $H_0 = H$
Répéter
(2) *Factorisation :* $H_k = Q_k.R_k$ ($n-1$ rotations de Givens)
(3) $H_{k+1} = R_k.Q_k$
Jusqu'à convergence

Complexité

La mise sous forme Hessenberg (traitement 1) a un coût du même ordre qu'une factorisation $Q.R$ de Householder : $O(n^3)$ opérations. De plus, si l'on applique les rotations de Givens, sans construire les matrices orthogonales associées alors la complexité des traitements 2 et 3 est $O(n^2)$. Si l'on considère que l'algorithme a convergé en $O(n)$ itérations alors la complexité est $O(n^3)$ opérations.

L'algorithme complet coûte donc $O(n^3)$ opérations (à comparer avec les $O(n^4)$ opérations de la version initiale).

Préservation de la forme Hessenberg

Exercice 20

Montrer que la forme Hessenberg supérieure de la matrice est préservée à chaque itération de la méthode QR.

1) On pourra établir dans un premier temps que si H_k est Hessenberg supérieure et que $H_k = Q_k.R_k$ (Q_k orthogonale et R_k triangulaire) alors Q_k est Hessenberg supérieure.

2) On pourra alors en déduire que $H_{k+1} = R_k.Q_k$ est aussi Hessenberg supérieure.

Chapitre 8

Représentation et traitement des matrices creuses

Soit $A = [a_{ij}] \quad i, j = 1, \ldots, n$
Matrice pleine : tous les coefficients sont significatifs, représentation sous forme d'un tableau à 2 dimensions.
Matrice creuse : taux élevé de coefficients nuls, représentation sous forme d'une structure de donnée adaptée.

8.1 Représentation par une structure bande

Pour tout coefficient non nul $a_{ij} \quad i < j$, on calcule $l_{ij} = j - i$ puis :
$l = \max_{i=1,\ldots,n \;\; j=i+1,\ldots,n} l_{ij}$ (demie-largeur de bande supérieure)
Pour tout coefficient non nul $a_{ij} \quad i > j$, on calcule $l'_{ij} = i - j$ puis :
$l\,' = \max_{i=1,\ldots,n \;\; j=1,\ldots,i-1} l'_{ij}$ (demie-largeur de bande inférieure)

$L = l + l\,' + 1 =$ largeur de bande
Dans le cas d'une matrice symétrique : $l = l\,' \qquad L = 2l + 1$

La structure de donnée associée, sous forme d'un tableau rectangulaire de L lignes et n colonnes, privilégie un accés par colonnes ou par lignes (cas de la figure ci-dessous).

Exercice 21

Ecrire dans un langage pseudo-algorithmique, le produit d'une matrice bande (accés par lignes) par un vecteur.

8.2 Représentation par une structure profil

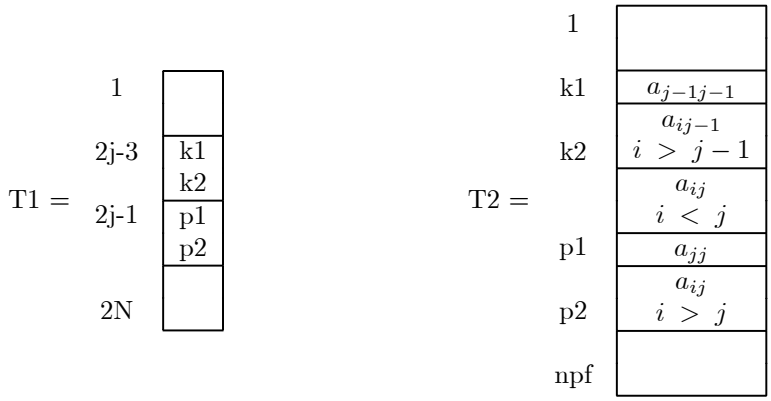
La structure de donnée associée privilégie un accés par lignes ou par colonnes (cas traité dans cette section).

Soient $p = \min_{a_{ij} \neq 0} i \qquad q = \max_{a_{ij} \neq 0} i$

Pour la colonne j, on mémorise tous les coefficients compris entre a_{pj} et a_{qj}.

On surnomme également les matrices bande (resp. les matrices profil) matrices à profil constant (resp. matrices à bande variable).

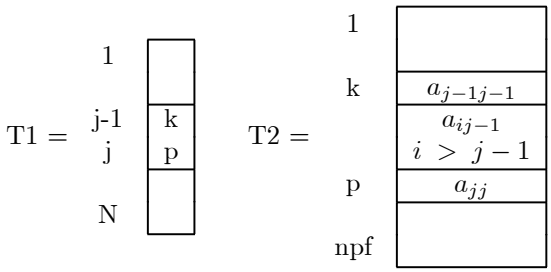
Structure de donnée associée (standard le plus courant) :



$T1$: tableau d'entiers servant à gérer les indirections
$T2$: tableau de réels contenant les coefficients de la matrice.

La colonne j est caractérisée par les informations $T1(2j-1) = p1$ et $T1(2j) = p2$ dans le tableau $T1$. $p1$ est l'indice de $T2$ correspondant au coefficient diagonal a_{jj} (toujours mémorisé), $p2$ est l'indice de $T2$ correspondant au dernier coefficient non nul de la colonne j.

Dans le cas d'une matrice symétrique, il est possible de simplifier la représentation (exemple où on ne mémorise que la matrice triangulaire inférieure).



Exercice 22

Soit $(A, B) \in M_n(\mathbb{R})^2$. A est représentée par une structure profil par lignes. B est représentée par une structure profil par colonnes. Ecrire, dans un langage pseudo-algorithmique le produit de la ligne i de A par la colonne j de B.

8.3 Représentation par une structure creuse

Seuls les coefficients non nuls sont mémorisés.

8.3.1 Langage possédant la notion de type pointeur prédéfini

Représentation par structure de file quadruple : 1 cellule = 1 coefficient réel + 4 pointeurs.

L'accés est indifférencié par lignes ou par colonnes. Il y a possibilité d'insérer facilement un nouveau coefficient non nul mais cette structure est trés coûteuse en espace mémoire et sa manipulation entraîne des temps d'exécution prohibitifs.

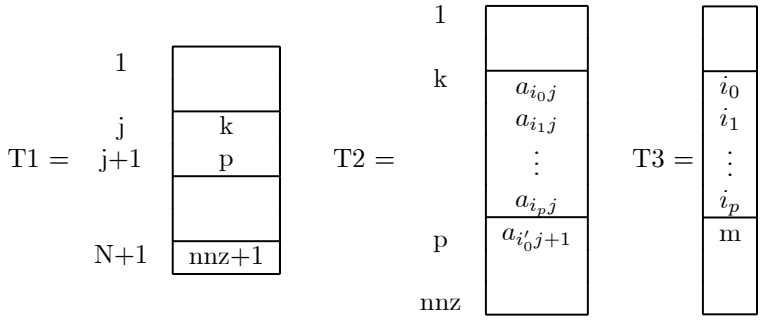
8.3.2 Représentation sans utilisation du type pointeur

L'accés est privilégié par lignes ou par colonnes (cas traité dans cette section).

Exemple de structure pour un accés par colonnes (standard Boeing-Harwell) :

$T1$: tableau d'entiers servant à gérer les indirections
$T2$: tableau de réels contenant les coefficients de la matrice
$T3$: tableau d'entiers contenant les numéros de lignes (même taille que $T2$)

Le premier élément non nul de la colonne j est à la ligne i_0. Le premier élément non nul de la colonne $j+1$ est à la ligne m, le deuxième élément non nul de la colonne j est à la ligne i_1, ..., le dernier à la ligne i_p.



Exercice 23

Ecrire, dans un langage pseudo-algorithmique, la multiplication d'une matrice à structure creuse (accés par lignes) par un vecteur.

Considèrons un calculateur sur lequel les nombres réels sont codés sur 8 octets, les entiers et les objets de type pointeur sur 4 octets. On peut calculer l'espace mémoire nécessaire pour mémoriser une matrice $N \times N$, de bande constante L, de profil moyen L', comportant m coefficients non nuls (par exemple, $N = 10000$, $L = 100$, $L' = 80$, $m = 80000$).

Matrice pleine : $8N^2$ octets (800 Moctets)
Matrice bande : $8NL$ octets ($\approx$ 8 Moctets)
Matrice profil : $8NL' + 8N = 8N(L' + 1)$ octets ($\approx$ 6.5 Moctets)
Matrice creuse par listes : $2N(4 + 4) + m(8 + 4 * 4) = 16N + 24m$ octets ($\approx$ 2 Moctets)
Matrice creuse par tableaux : $4N + 4m + 8m = 4N + 12m$ octets ($\approx$ 1 Moctets)

8.4 Influence des structures de données sur les algorithmes de résolution de systèmes linéaires

8.4.1 Méthodes itératives

Qu'il s'agisse des méthodes de relaxation ou de gradient, la matrice du système n'est pas modifiée et l'algorithmique est basée sur des produits matrice-vecteur. On retiendra donc un mode de stockage par lignes.

8.4.2 Méthodes directes

Elles comportent généralement une phase de factorisation (ou de triangularisation) suivie de la résolution d'un ou plusieurs systèmes triangulaires. On remarquera que la résolution d'un système triangulaire existe sous deux versions, l'une privilégiant un accés par lignes (algorithme sans reports), l'autre un accés par colonnes (algorithme avec reports).

En raisonnant sur la triangularisation ou la factorisation de Gauss (comportement identique pour la factorisation de Cholesky), le processus algorithmique de base comporte, à l'étape k, la mise à jour suivante :

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

Si $a_{ij}^{(k)} = 0$ et $a_{ik}^{(k)} a_{kj}^{(k)} \neq 0$ alors $a_{ij}^{(k+1)} \neq 0$ et il y a donc création d'un nouvel élément non nul : phénomène de fill-in (remplissage).

Sur une matrice à structure bande, le fill-in se produit à l'intérieur de la bande et la structure de donnée n'est pas modifiée. Dans le cas où un fill-in apparaît hors structure, il est nécessaire soit de modifier dynamiquement le mode de stockage, soit de prévoir son apparition par un pré-traitement. Pour une matrice profil, le fill-in se produit à l'intérieur du profil colonne pour la partie triangulaire supérieure, à l'intérieur du profil ligne dans la partie triangulaire inférieure.

Exercice 24

Déterminer le fill-in provoqué, par la factorisation de Gauss, sur la matrice à structure profil (accés par colonnes) suivante :

Le Fill-in modifie-t-il la structure du profil ? Un accés par lignes aurait-il été plus intéressant ?

8.5 Analyse graphique de la topologie

Dans le cas d'une matrice symétrique à diagonale principale strictement dominante (factorisation de Gauss applicable sans pivotage).

8.5.1 Comment anticiper le Fill-in

Un graphe (cf. Cours de théorie des graphes en 2ème année) est un ensemble de noeuds reliés par des arcs.
La topologie de la matrice est représentée sous forme d'un graphe non orienté (les arcs ne sont pas fléchés) :

Un *nœud* est associé à chaque ligne (ou colonne) de la matrice.
Un *arc* est présent entre les nœuds i et j (nœuds adjacents) si $a_{ij} \neq 0$

Interprétation graphique de la factorisation (factorisation symbolique) :

La factorisation symbolique consiste à éliminer successivement tous les noeuds du graphe dans l'ordre croissant de leur numéro.

Elimination de (1) : (2) et (3) étant adjacents à (1), il faut créer un arc entre (2) et (3). Cet arc n'existait pas à l'étape précédente : il y a donc un fill-in en a_{23} (et a_{32}). L'élimination de (2) ne provoque aucun fill-in supplémentaire.

L'élimination d'un noeud est cause de la présence d'un lien entre chaque paire de noeuds de sa liste d'adjacence. Tout nouveau lien est associé à l'apparition de 2 nouveaux coefficients non nuls : *Les noeuds i et j font partie de la liste d'adjacence de k. Il n'y avait pas d'arc entre i et j après l'élimination de $k-1$. Lors de lélimination de k, il y a création d'un arc entre i et j (apparition de $a_{ij} \neq 0$ et $a_{ji} \neq 0$).*

Remarque : Si on élimine d'abord (2) puis (1), il n'y a pas de fill-in.

Eliminer d'abord (2) revient à choisir a_{22} comme premier pivot.

La matrice résultante est factorisable sans Fill-in.

8.5.2 Comment réduire le Fill-in : algorithme du degré minimum

On peut éventuellement réduire le Fill-in en réordonnançant les coefficients de la matrice avant factorisation (qui peut toujours être appliquée sans pivotage). Une technique de réordonnancement consiste à appliquer l'algorithme du degré minimum. Cet algorithme va être appliqué à l'exemple ci-dessous :

Tout le Fill-in apparaît lors de l'étape 1 de la factorisation.

On définit le degré d'un nœud comme étant le nombre de ses voisins. A une étape donnée de la factorisation, on élimine le nœud possédant le plus faible degré (minimum degree) i.e. on traite le pivot possédant le plus faible nombre de coefficients non nuls sur sa ligne (ou colonne) :

- à la première étape, (2) de degré 1 $\Rightarrow$ Pas de Fill-in.

Compte tenu de la disparition d'un noeud dans le graphe, le degré des noeuds doit être mis à jour.

- à la deuxième étape, (3), (4), (5) et (6) sont de degré 2 $\Rightarrow$ choix du numéro le plus faible (dans une version non optimisée) i.e. élimination de (3) $\Rightarrow$ Pas de Fill-in.

- à la troisième étape, élimination de (6) $\Rightarrow$ Pas de Fill-in.

- puis élimination de (1), (4) et (5) toujours sans Fill-in.

On en déduit donc un nouvel ordonnancement $2 \rightarrow 3 \rightarrow 6 \rightarrow 1 \rightarrow 4 \rightarrow 5$ qui va piloter une restructuration de la matrice avant factorisation.

Le réordonnancement terminé, la matrice n'est plus soumise au Fill-in lors de la factorisation.

Exercice 25

Déterminer directement, puis en passant à un modèle graphique, le fill-in induit par la factorisation de Gauss pour la matrice suivante :

Réordonnancer cette matrice en appliquant l'algorithme du degré minimum.

Attention : Cet algorithme n'est qu'une heuristique, il peut parfois accroitre le Fill-in.

Mais l'expérience montre néanmoins son efficacité dans la plupart des cas.

8.6 Autre exemple de réordonnancement : algorithme de Cuthill Mac Kee

L'algorithme de Cuthill Mac Kee existe sous deux versions : algorithme direct et algorithme inverse. Considérons tout d'abord l'algorithme direct.

Cet algorithme possède la propriété de transformer toute matrice symétrique creuse en une matrice symétrique tridiagonale par blocs (chaque bloc pouvant lui-même être creux) tout en réduisant la valeur des demi-largeurs de bande. Son objectif n'est pas la réduction du fill-in.

A partir du graphe de la matrice (construit comme pour l'algorithme du degré minimum), on génère une suite finie d'ensembles $S_i \;\; i = 1, \ldots, p$, composés de nœuds, de la manière suivante :

- S_1 = {parmi l'ensemble des nœuds de degré minimum, celui possédant le plus faible numéro} (S_1 ne contient qu'un seul élément). Dans le cas où plusieurs nœuds possèdent un degré minimum, on retient celui de plus faible numéro.
- $i = 2, \ldots, p \;\; S_i$ = {ensemble des nœuds adjacents à S_{i-1} et n'appartenant à aucun des ensembles $S_j \;\; j = 1, \ldots, i-1$}
  - A l'intérieur de $S_i \;\; i = 2, \ldots, p$, les nœuds sont rangés suivant l'ordre croissant de leur numéro.
  - Si à l'étape i, {ensemble des nœuds adjacents à S_{i-1} et n'appartenant à aucun des ensembles $S_j \;\; j = 1, \ldots, i-1$} $= \emptyset$ et qu'il reste encore des nœuds non sélectionnés, S_i est initialisé par le nœud de degré minimum (et de plus faible numéro) choisi parmi l'ensemble des nœuds non encore sélectionnés.

Lorsque tous les ensembles $S_i \;\; i = 1, \ldots, p$ ont été créés (tous les nœuds ont été sélectionnés), on définit alors le nouvel ordonnancement (algorithme de Cuthill Mac Kee direct) :

$$\text{Le nœud de } S_1 \longrightarrow \text{Les nœuds de } S_2 \text{ (dans l'ordre de leur classement)} \longrightarrow \text{Les nœuds de } S_3 \longrightarrow \ldots \longrightarrow \text{Les nœuds de } S_p$$

L'algorithme de Cuthill Mac Kee inverse consiste à inverser le classement précédent :

Les nœuds de S_p (dans l'ordre de leur classement inverse) $\longrightarrow \ldots \longrightarrow$
Les nœuds de $S_3 \longrightarrow$ Les nœuds de $S_2 \longrightarrow$ Le nœud de S_1

Suivant les cas, l'algorithme inverse peut être plus intéressant que l'algorithme direct car le profil de la matrice réordonnancée possède des propriétés différentes.

Exemple d'application (représentatif du comportement de la méthode mais attention à la notion d'heuristique)
Soit la matrice suivante à profil symétrique :

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

Considérons le profil par colonnes de la partie triangulaire supérieure de cette matrice. Le nombre de coefficients nuls hors profil est 17 avec la présence d'une structure de type puit/cheminée. La demie-largeur de bande est 6.

L'application de l'algorithme de Cuthill Mac Kee direct produit la suite d'ensembles :

$$S_1 = \{9\} \qquad S_2 = \{4\} \qquad S_3 = \{1, 2, 3, 10\}$$

$$S_4 = \{5, 6, 8\} \qquad S_5 = \{7\}$$

d'où le nouvel ordonnancement :

$$9 \rightarrow 4 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 10 \rightarrow 5 \rightarrow 6 \rightarrow 8 \rightarrow 7$$

et la matrice réordonnancée avec sa structure tridiagonale par blocs (dimension du bloc diagonal numéro $i = \mathrm{card}(S_i)$) :

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

Le nombre de coefficients nuls hors profil est 20 et la structure de type puit/cheminée a disparu. La demie-largeur de bande est 4.

Ordonnancement obtenu par application de Cuthill Mac Kee inverse :

$$7 \to 8 \to 6 \to 5 \to 10 \to 3 \to 2 \to 1 \to 4 \to 9$$

et la matrice réordonnancée :

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

Le nombre de coefficients nuls hors profil est 24 (nouvelle optimisation du profil). La version directe reste néanmoins plus intéressante si on considère un profil par lignes. La demie-largeur de bande est 4.

Exercice 26

On considère la matrice symétrique définie positive $12 \star 12$ ci-dessous dont la partie triangulaire supérieure est mémorisée avec un profil par colonnes :

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

1) Déterminer le nombre de coefficients hors profil.

2) Déterminer le fill-in induit par la factorisation de cette matrice par l'algorithme de Cholesky (mémorisation de ${}^tL$).

3) Réordonnancer la matrice initiale en appliquant l'algorithme du degré minimum. Déterminer le nombre de coefficients hors profil. Déterminer le fill-in induit par la factorisation de cette matrice par l'algorithme de Cholesky.

4) Même question avec l'algorithme de Cuthill Mac Kee direct.

5) Même question avec l'algorithme de Cuthill Mac Kee inverse.