
Notes de cours

Calcul scientifique

Jérémy Heleine

Master 2 Recherche et Innovation

Université Toulouse III - Paul Sabatier

Dernière mise à jour : 17 février 2025

Pour toute question ou remarque : jeremy.heleine@math.univ-toulouse.fr

Ce document est mis à disposition selon les termes de la licence [CC BY-NC-SA 4.0](https://creativecommons.org/licenses/by-nc-sa/4.0/).



Table des matières

Table des matières	i
Introduction	1
1 Arithmétique flottante	3
1.1 Un nombre infini de digits	3
1.2 Nombres flottants	6
1.2.1 Représentation	6
1.2.2 Bornes	7
1.2.3 Un exemple de système : la norme IEEE 754	8
1.3 Arrondis	10
1.4 Opérations sur les flottants	12
1.4.1 Modèle standard	12
1.4.2 Perte de propriétés	14
1.4.3 Absorption	14
1.4.4 Annulation catastrophique	14
1.5 Retrouver les paramètres d'un système	15
1.5.1 Base	16
1.5.2 Précision	18
1.5.3 Plage d'exposants	19
2 Conditionnement	21
2.1 Erreurs absolue et relative	21
2.2 Conditionnement	23
2.3 Conditionnement d'un système linéaire	26
2.3.1 Normes matricielles	26
2.3.2 Conditionnement	32
3 Méthodes directes	35
3.1 Une fausse bonne idée : la méthode de Cramer	35
3.2 Systèmes triangulaires	36
3.3 Méthode du pivot de Gauss	42
3.4 Factorisation LU	48
3.5 Factorisation de Cholesky	55

3.6	Systèmes de Hessenberg	58
4	Matrices creuses	64
4.1	L'approche classique	64
4.2	Ignorer les zéros	65
4.3	Formats compressés	67
4.4	Diagonales	68
5	Méthodes itératives	71
5.1	Méthodes classiques	71
5.1.1	Généralités	71
5.1.2	Décomposition de la matrice	73
5.1.3	Méthodes de Jacobi et de Gauss-Seidel	74
5.1.4	Méthode de relaxation	77
5.2	Méthodes de gradient	83
5.2.1	Idée générale	83
5.2.2	Méthode de gradient à pas fixe	85
5.2.3	Méthode de gradient à pas optimal	87
5.2.4	Méthode du gradient conjugué	90
5.2.5	Gradient conjugué préconditionné	101
5.3	Méthodes de Krylov	108
5.3.1	Espaces de Krylov	109
5.3.2	Petit détour par Gram-Schmidt	110
5.3.3	Méthode d'Arnoldi	111
5.3.4	Full Orthogonalization Method (FOM)	114
5.3.5	Méthode Generalized Minimum Residual (GMRES)	118
5.3.6	Préconditionnement	120
5.3.7	Factorisation LU incomplète	121
A	Opérations élémentaires	126
A.1	Ordre et digits	126
A.2	Addition	129
A.3	Soustraction	132
A.4	Multiplication	135
A.5	Division	136
B	Polynômes de Tchebychev	139
B.1	Définition et premières propriétés	139
B.2	Une histoire de cosinus	140
B.3	Et aussi de cosinus hyperbolique	142
B.4	Une expression à base de racines carrées	143
C	Autour d'un mystère de l'arithmétique flottante	146
C.1	Observations	146

C.2	Quelques relations préliminaires	147
C.3	Expression de la somme	148
C.4	Une histoire de parité	149
C.4.1	Premier cas impair	149
C.4.2	Second cas impair	150
C.4.3	Premier cas pair	150
C.4.4	Second cas pair	151
C.5	Dernière remarque	152
D	Factorisation LU des matrices tridiagonales	153
E	Préconditionnement du Laplacien discret	156
E.1	Éléments propres	156
E.2	Un mauvais conditionnement	160
E.3	Préconditionnement	161
F	Gradient biconjugué	166
F.1	Premières propriétés	166
F.2	Système artificiel	168
F.3	Bi-orthogonalité et biconjugaison	168
F.4	Algorithme	171
F.5	Convergence	173
G	Absorption	174
G.1	Un peu de théorie	174
G.2	Une conséquence de l'absorption	175
H	Convergence de la série harmonique ?	178
H.1	Méthode classique	178
H.2	Une autre approche	179
H.3	Stabilité	180
H.4	Phénomènes numériques	181
H.4.1	Absorption	181
H.4.2	Arrondi à zéro	182
H.4.3	Arrondi à l'infini	183
H.4.4	Discussion sur les bornes	183
	Références	iv

Introduction

Il existe différentes méthodes permettant de calculer une approximation numérique de la solution d'une EDP : différences finies, éléments finis ou encore volumes finis sont parmi les plus répandues. Ces méthodes ont en commun le cheminement global : on part de l'équation dite continue, qui est discrétisée de sorte à obtenir un système linéaire qu'il faudra alors résoudre pour obtenir l'approximation souhaitée.

Dans ce cours, on va s'intéresser à cette dernière étape. Plus précisément, on va se pencher sur plusieurs méthodes qui peuvent être « rangées » en deux catégories : les méthodes directes, qui permettent d'obtenir directement la solution d'un système, et les méthodes itératives, qui consistent à construire une suite de vecteurs qui converge vers cette solution.

La notion d'erreur sera centrale tout au long de ce cours. Il ne faut en effet pas oublier que la résolution numérique d'une EDP conduit, dans une écrasante majorité des cas, à une approximation de la solution continue, et non pas à sa connaissance exacte. L'analyse de l'erreur commise par l'étape de discrétisation ne sera pas abordée (elle pourrait facilement faire l'objet de plusieurs cours entiers!). Nous allons plutôt nous concentrer sur l'analyse de l'erreur commise par la résolution du système linéaire qui en résulte.

En effet, un ordinateur ne peut stocker une quantité infinie d'informations. Une conséquence évidente est qu'il n'est pas possible de résoudre directement une EDP qui serait définie sur un domaine non borné : il faut trouver un moyen de tronquer le domaine. Mais le problème se pose aussi sur un domaine borné : un simple intervalle réel, aussi petit soit-il, contient toujours un nombre infini de points, et il est donc impossible d'obtenir la valeur d'une fonction sur tous les points qu'il contient. Là encore, il faut trouver des stratégies permettant de discrétiser le domaine. Ce passage de l'infini au fini amène à ce qu'on appelle des erreurs d'approximation.

Une autre conséquence, peut-être moins immédiate, est que tous les nombres ne peuvent pas être stockés de manière exacte. De la même façon qu'il ne nous est pas possible d'écrire toutes les décimales de π sur une feuille, un ordinateur ne pourra pas connaître tous les nombres de manière exacte. C'est évident pour les nombres qui n'admettent pas une représentation décimale finie, mais c'est aussi vrai pour d'autres, et les nombres qu'il est possible de stocker de manière exacte peuvent bien sûr changer selon l'ordinateur et les systèmes utilisés. Par exemple, une calculatrice utilisant la base 10 pourra sans problème représenter 0.1, alors qu'un ordinateur utilisant la base 2 ne pourra jamais réaliser cet exploit. Quand on stocke un nombre sur un ordinateur et quand on effectue des calculs numériques, on commet donc des erreurs dites d'arrondi. L'un des objectifs de ce cours est de mieux comprendre ces erreurs.

On peut également citer un troisième type d'erreur : les erreurs sur les données. Une EDP est souvent le résultat de la modélisation d'un problème physique, que l'on peut chercher à résoudre pour répondre à une question concrète. Quelle est la répartition de la chaleur dans une pièce ? Comment une onde acoustique se propage-t-elle dans un guide ? Y a-t-il une tumeur dans un cerveau, et si oui où est-elle ? À quel moment un patient doit-il s'attendre à sa prochaine crise d'épilepsie ? Il ne s'agit que d'un petit nombre de questions qui se posent, et qui ont toute un point commun : pour y répondre, il faut des données. La répartition des radiateurs, la fréquence de l'onde envoyée, des mesures effectuées sur un patient, ou même le résultat d'un calcul numérique précédent. Toutes ces données ne sont pas toujours connues de manière exacte (c'est même très rarement le cas) : il faut donc le prendre en compte lorsque l'on résout une EDP, et toujours avoir en tête que les quantités que l'on manipule ne sont pas le reflet exact de la réalité.

Dans la pratique, ces trois types d'erreurs se mélangent, ont parfois un effet amplificateur l'une sur l'autre. Elles ont des origines et des méthodes d'analyse différentes, mais elles ont toutes en commun un fait : elles sont inévitables, et il est important de les avoir en tête pour mieux les comprendre et d'établir des stratégies adaptées afin de les contrôler. L'augmentation de la précision utilisée pour stocker des nombres est bien sûr une réponse évidente aux erreurs d'arrondis, mais elle n'est que partielle : on pourra stocker plus de nombres, mais il y en aura toujours une infinité d'autres qui ne pourront jamais l'être. Utiliser des milliards de décimales pour représenter π sera bien sûr plus précis que le classique 3.14, mais si les opérations qui sont réalisées sur cette approximation sont instables, cette erreur microscopique se propagera et s'amplifiera. Au final, on aura une solution de l'EDP qui n'aura d'approximation de la solution exacte que le nom.

Chapitre 1

Arithmétique flottante

Considérons un nombre décimal x , c'est-à-dire un réel dont on connaît une représentation exacte tel que -92 ou 2016.0201 . Une façon de représenter ces nombres est d'utiliser la notation scientifique :

$$x = \pm a \cdot 10^e,$$

où $a \in [1, 10[$ et $e \in \mathbb{Z}$. Avec cette notation, les exemples précédents s'écrivent $-9.2 \cdot 10^1$ ($a = 9.2$, $e = 1$) et $2.0160201 \cdot 10^3$ ($a = 2.0160201$, $e = 3$). On peut écrire un certain $a \in [1, 10[$ donné en faisant explicitement intervenir les $N \in \mathbb{N}^*$ différentes décimales qui le composent :

$$a = \sum_{n=0}^N \frac{d_n}{10^n}, \quad \text{où } \forall n \in \llbracket 0, N \rrbracket, \quad d_n \in \llbracket 0, 9 \rrbracket,$$

et ainsi :

$$x = \pm 10^e \sum_{n=0}^N \frac{d_n}{10^n}.$$

Cette écriture, relativement naturelle, peut être généralisée avec d'autres bases. L'objectif de ce chapitre est de comprendre comment et de répondre à la question qui va réellement nous intéresser : comment représenter un réel quelconque avec un nombre fini d'informations ?

1.1 Un nombre infini de digits

Commençons par un résultat permettant, en quelque sorte, de représenter un réel quelconque dans une certaine base.

Théorème 1.1.1. Soit $b \in \mathbb{N}$, $b \geq 2$. Soit $a \in [1, b[$. Il existe une suite $(d_n)_{n \in \mathbb{N}}$ prenant ses valeurs dans $\llbracket 0, b-1 \rrbracket$ telle que :

$$a = \sum_{n=0}^{+\infty} \frac{d_n}{b^n}.$$

Démonstration. Soit la suite $(a_n)_{n \in \mathbb{N}}$ définie par :

$$\forall n \in \mathbb{N}, \quad a_n := \frac{\lfloor b^n a \rfloor}{b^n}.$$

La suite $(a_n)_{n \in \mathbb{N}}$ est croissante. En effet, par définition de la partie entière, pour tout $n \in \mathbb{N}$:

$$\lfloor b^n a \rfloor \leq b^n a,$$

d'où :

$$b \lfloor b^n a \rfloor \leq b^{n+1} a.$$

Or, $b \lfloor b^n a \rfloor \in \mathbb{Z}$ et $\lfloor b^{n+1} a \rfloor$ est le plus grand entier inférieur ou égal à $b^{n+1} a$. Ainsi :

$$b \lfloor b^n a \rfloor \leq \lfloor b^{n+1} a \rfloor.$$

Il suffit alors de diviser par b^{n+1} pour obtenir :

$$a_n = \frac{\lfloor b^n a \rfloor}{b^n} \leq \frac{\lfloor b^{n+1} a \rfloor}{b^{n+1}} = a_{n+1}.$$

Montrons maintenant que la suite $(a_n)_{n \in \mathbb{N}}$ permet d'encadrer a . En effet, pour tout $n \in \mathbb{N}$, la définition de la partie entière donne :

$$\lfloor b^n a \rfloor \leq b^n a < \lfloor b^n a \rfloor + 1,$$

d'où l'on déduit, en divisant par b^n :

$$a_n \leq a < a_n + \frac{1}{b^n}. \quad (1.1)$$

La partie gauche de cette inégalité permet de montrer que la suite $(a_n)_{n \in \mathbb{N}}$ est majorée par a . Comme elle est croissante, on en déduit qu'elle converge. On passe alors à la limite dans (1.1) pour obtenir :

$$\lim_{n \rightarrow +\infty} a_n = a.$$

Définissons maintenant la suite $(d_n)_{n \in \mathbb{N}}$ par :

$$\forall n \in \mathbb{N}, \quad d_n := \lfloor b^n a \rfloor - b^n a_{n-1},$$

où l'on a posé, pour simplifier la définition, $a_{-1} := 0$. Commençons par remarquer que cela définit une suite d'entiers. En effet, $d_0 = \lfloor a \rfloor \in \mathbb{Z}$ et :

$$\forall n \in \mathbb{N}^*, \quad d_n = \lfloor b^n a \rfloor - b^n \frac{\lfloor b^{n-1} a \rfloor}{b^{n-1}} = \lfloor b^n a \rfloor - b \lfloor b^{n-1} a \rfloor \in \mathbb{Z}.$$

Rappelons que la suite $(a_n)_{n \in \mathbb{N}}$ est croissante. Ainsi, pour tout $n \in \mathbb{N}$, on a :

$$a_{n-1} \leq a_n = \frac{\lfloor b^n a \rfloor}{b^n},$$

d'où :

$$d_n = \lfloor b^n a \rfloor - b^n a_{n-1} \geq 0.$$

Par ailleurs, $d_0 = \lfloor a \rfloor < b$ car $a < b$. Pour $n \in \mathbb{N}^*$, on reprend la partie droite de (1.1) :

$$a < a_{n-1} + \frac{1}{b^{n-1}},$$

d'où :

$$b^n a < b^n a_{n-1} + b.$$

Ainsi :

$$\lfloor b^n a \rfloor \leq b^n a < b^n a_{n-1} + b,$$

d'où l'on déduit :

$$d_n = \lfloor b^n a \rfloor - b^n a_{n-1} < b.$$

Donc la suite $(d_n)_{n \in \mathbb{N}}$ est bien à valeurs dans $\llbracket 0, b-1 \rrbracket$.

Pour tout $p \in \mathbb{N}$ on écrit la somme partielle :

$$\sum_{n=0}^p \frac{d_n}{b^n} = \sum_{n=0}^p \frac{\lfloor b^n a \rfloor - b^n a_{n-1}}{b^n} = \sum_{n=0}^p (a_n - a_{n-1}) = a_p.$$

Comme la limite de $(a_n)_{n \in \mathbb{N}}$ n'est autre que a , on en déduit le résultat. ■

Remarque 1.1.2. La suite $(a_n)_{n \in \mathbb{N}}$ définie dans la preuve ci-dessus n'est en réalité rien d'autre que la troncature de a . Plus précisément, pour tout $n \in \mathbb{N}$, a_n est l'écriture de a en base b tronquée à n chiffres après la virgule. Prenons par exemple $a := 2.0160201$ et $b = 10$. Alors :

$$a_0 = 2, \quad a_1 = 2.0, \quad a_2 = 2.01, \quad a_3 = 2.016, \quad \text{etc.}$$

La suite $(d_n)_{n \in \mathbb{N}}$, quant à elle, est la suite des chiffres de a dans son écriture en base b . Toujours avec le même exemple, on obtient :

$$d_0 = 2, \quad d_1 = 0, \quad d_2 = 1, \quad d_3 = 6, \quad \text{etc.}$$

Une autre façon d'écrire le résultat du [Théorème 1.1.1](#), un peu moins formelle, serait donc :

$$a = d_0.d_1d_2d_3\dots$$

Corollaire 1.1.3. Soit $b \in \mathbb{N}$, $b \geq 2$. Soit $x \in \mathbb{R}$. Il existe $s \in \{0, 1\}$, $e \in \mathbb{Z}$ et une suite $(d_n)_{n \in \mathbb{N}}$ prenant ses valeurs dans $\llbracket 0, b-1 \rrbracket$ telle que :

$$x = (-1)^s b^e \sum_{n=0}^{+\infty} \frac{d_n}{b^n}.$$

Démonstration. Si $x = 0$, $s = 0$, $e = 0$ et la suite constante égale à 0 conviennent. On peut donc désormais supposer $x \neq 0$. Commençons par écrire :

$$x = (-1)^s |x|$$

avec $s = 0$ si $x > 0$ et $s = 1$ si $x < 0$. Comme la fonction $\log_b$ est une bijection de $\mathbb{R}_*^+$ vers $\mathbb{R}$, on trouve un unique $e \in \mathbb{Z}$ tel que :

$$e \leq \log_b(|x|) < e + 1.$$

Autrement dit :

$$b^e \leq |x| < b^{e+1}.$$

Soit alors $a := b^{-e}|x|$. L'inégalité ci-dessus implique :

$$1 \leq a < b,$$

et on peut alors appliquer le [Théorème 1.1.1](#) pour conclure. ■

Ce résultat ne permet bien sûr pas de remplir notre objectif, à savoir représenter un réel quelconque à l'aide d'un nombre fini d'informations : il faudrait pour cela que la suite $(d_n)_{n \in \mathbb{N}}$ soit elle-même finie (c'est-à-dire nulle à partir d'un certain rang), ce qui ne se produit pas pour tous les réels. Cette représentation a cependant le bon goût d'être naturelle et nous allons, dans la prochaine section, définir un système qui s'en inspire très largement.

1.2 Nombres flottants

1.2.1 Représentation

Définition 1.2.1. Soient $b \in \mathbb{N}$ tel que $b \geq 2$, $p \in \mathbb{N}^*$ et $\bar{e} = (e_{\min}, e_{\max}) \in \mathbb{Z}^2$. On définit l'ensemble :

$$F(b, p, \bar{e}) := \left\{ x \in \mathbb{R} ; \exists (e, d) \in \llbracket e_{\min}, e_{\max} \rrbracket \times \llbracket 0, b-1 \rrbracket^{p+1} / x = \pm b^e \sum_{n=0}^p \frac{d_n}{b^n} \right\}.$$

Le paramètre b est la base de représentation, p est la précision et $\bar{e}$ est la plage d'exposants. Tout élément de $F(b, p, \bar{e})$ est appelé nombre à virgule flottante, nombre flottant, ou encore simplement flottant. À chaque flottant est associé un entier $e \in \llbracket e_{\min}, e_{\max} \rrbracket$ appelé exposant et un $(p+1)$ -uplet $d \in \llbracket 0, b-1 \rrbracket^{p+1}$ appelé le significande (ou la mantisse), contenant les digits, ou chiffres, du flottant.

Dans toute la suite du cours, lorsque l'on parlera d'un ensemble de flottants F , on sous-entendra la donnée d'une base b , d'une précision p et d'une plage d'exposants $\bar{e}$ tels que $F = F(b, p, \bar{e})$. De la même façon, en parlant d'un flottant, il est sous-entendu la donnée de son signe, son exposant et son significande.

Définition 1.2.2. Soit F un ensemble de nombres flottants. Les flottants dont le premier digit est non nul (c'est-à-dire $d_0 \geq 1$) forment le sous-ensemble des nombres flottants normalisés. Les flottants dont le premier digit est nul ($d_0 = 0$) forment le sous-ensemble des nombres flottants dénormalisés.

Remarque 1.2.3. Les nombres flottants normalisés ont une grande importance : ils profitent d'une représentation unique, ainsi que d'une précision égale à p . Les nombres dénormalisés permettent de représenter des réels beaucoup plus proches de zéro, mais avec une précision amoindrie.

Remarque 1.2.4. Il peut parfois être utile de représenter autrement les flottants. De manière équivalente, on peut ainsi définir :

$$x = \pm b^{e-p} m, \quad \text{avec } m \in \llbracket 0, b^{p+1} - 1 \rrbracket.$$

Le significande (ou la mantisse) est alors représenté par un entier dont les chiffres ne sont autres que les d_n . Les nombres normalisés correspondent à $m \geq b^p$.

1.2.2 Bornes

Contrairement à la droite réelle, un ensemble de nombres flottants est borné : pour la même raison que l'on ne peut pas stocker un nombre infini de chiffres, on ne peut pas représenter des nombres arbitrairement grands (ou petits) dans un système de nombres flottants F fixé. Intéressons-nous de plus près aux différentes bornes qui peuvent être définies.

Plus grande valeur positive. Notée $F_{\max}$, elle est obtenue en considérant le plus grand exposant possible ($e = e_{\max}$), avec tous les digits égaux à la plus grande valeur possible (tous les d_n égaux à $b - 1$). Un rapide calcul donne :

$$F_{\max} = b^{e_{\max}} \sum_{n=0}^p \frac{b-1}{b^n} = b^{e_{\max}} (b - b^{-p}).$$

Plus petit nombre normalisé positif. Noté F_{ε} , il est obtenu en considérant le plus petit exposant possible ($e = e_{\min}$), et en prenant tous les digits nuls, sauf le premier (pour avoir un nombre normalisé) pour lequel on considère la plus petite valeur non nulle possible. On obtient :

$$F_{\varepsilon} = b^{e_{\min}} \left(\frac{1}{b^0} + \sum_{n=1}^p \frac{0}{b^n} \right) = b^{e_{\min}}.$$

Plus petit nombre dénormalisé positif. Noté F_{ν} , il est obtenu en considérant le plus petit exposant possible ($e = e_{\min}$), et en prenant tous les digits nuls, sauf le dernier :

$$F_{\nu} = b^{e_{\min}} \left(\sum_{n=0}^{p-1} \frac{0}{b^n} + \frac{1}{b^p} \right) = b^{e_{\min}-p}.$$

Epsilon machine. Noté ε_M , le epsilon machine est défini comme la distance entre 1 et le plus petit nombre flottant strictement supérieur à 1. Ce flottant est obtenu en passant le dernier digit de la représentation de 1 de 0 à 1, ce qui revient à ajouter b^{-p} . Ainsi :

$$\varepsilon_M = b^{-p}.$$

Remarque 1.2.5. On a ici utilisé la représentation du significande par un $(p+1)$ -uplet, mais on obtient bien sûr les mêmes résultats en représentant le significande par un entier.

Si la définition du epsilon machine peut paraître quelque peu artificielle au premier coup d'œil, il s'agit en réalité d'une valeur très utile lorsque l'on s'intéresse à la précision d'un système de nombres flottants.

Proposition 1.2.6. Soit $x \in F$ normalisé, et soit $x^\pm$ les deux nombres flottants adjacents à x dans F . Alors :

$$|x| \frac{\varepsilon_M}{b - \varepsilon_M} \leq |x^\pm - x| \leq |x| \varepsilon_M.$$

Démonstration. Supposons pour simplifier $x > 0$ (le cas $x < 0$ se traite de la même manière). On écrit $x = b^{e-p}m$ où e est l'exposant de x et $m \in \llbracket b^p, b^{p+1} - 1 \rrbracket$ son significande. On obtient ainsi :

$$x^\pm = b^{e-p}(m \pm 1),$$

d'où :

$$|x^\pm - x| = b^{e-p} = b^e \varepsilon_M.$$

Comme $m \in \llbracket b^p, b^{p+1} - 1 \rrbracket$, on en déduit :

$$b^e \leq x \leq b^e(b - b^{-p}),$$

d'où le résultat. ■

1.2.3 Un exemple de système : la norme IEEE 754

L'Institute of Electrical and Electronics Engineers a publié en 1985 [Ins85] une norme d'arithmétique flottante. Connu sous le nom de IEEE 754, ce standard décrit un système de nombres flottants, la manière d'y représenter un réel quelconque, ainsi que la façon dont doivent se comporter les opérations élémentaires. Différents formats y ont été initialement décrits, et d'autres ont été ajoutés avec le temps. On y trouve ainsi des formats binaires ($b = 2$) mais aussi décimaux ($b = 10$), avec différentes précisions p et différentes plages d'exposants ($e_{\min}, e_{\max}$). Cette norme est aujourd'hui très largement répandue : la plupart des ordinateurs utilisent maintenant le format dit en « double précision ». Le [Tableau 1.1](#) liste quelques formats décrit dans la norme, parmi les plus utilisés.

Format	Nom usuel	p	$e_{\min}$	$e_{\max}$
binary32	Simple précision	23	-126	127
binary64	Double précision	52	-1022	1023
binary128	Quadruple précision	112	-16 382	16 383

TABLE 1.1 – Paramètres de formats définis dans la IEEE 754.

Concrètement, un nombre est stocké sur un ensemble de 32 bits en simple précision, 64 bits en double précision, ou encore 128 bits en quadruple précision. Cet ensemble est découpé en trois espaces (voir la [Figure 1.1](#) pour avoir une idée des proportions).

Bit de signe. Un seul bit est utilisé pour le signe : il est fixé à 0 si le nombre est positif, et 1 si le nombre est négatif.

Exposant biaisé. L'exposant est stocké sur une séquence d'un nombre fixé de bits. Plus précisément, on y stocke un exposant dit « biaisé » : c'est un entier positif, auquel on retranche une valeur fixée (le biais) pour obtenir l'exposant du nombre qui sera alors compris entre $e_{\min} - 1$ et $e_{\max} + 1$. Ces deux valeurs extrémales sont cependant réservées (on y reviendra juste après).

Significande. Les bits restants servent à stocker le significande : un bit correspond à un digit du nombre. Un total de p bits est utilisé : on ne stocke que les digits d_1 à d_p , en supposant que $d_0 = 1$ (sauf cas particulier, précisé juste après).

Exemple 1.2.7. En double précision, $p = 52$ bits sont utilisés pour le significande, sur un total de 64 bits dont un est réservé pour le signe. Il reste donc 11 bits pour stocker l'exposant biaisé qui peut ainsi prendre toutes les valeurs entières comprises entre 0 et $2^{11} - 1 = 2047$. Le biais est fixé à 1023.

FIGURE 1.1 – Stockage d'un flottant selon la norme IEEE 754 : un bit pour le signe s , une séquence pour l'exposant biaisé $\tilde{e}$, le reste pour le significande m .

Des cas particuliers se présentent donc si l'exposant est en dehors de la plage indiquée dans le [Tableau 1.1](#). Si $e = e_{\min} - 1$, deux cas se présentent :

1. le significande est nul, auquel cas le flottant représente zéro ;
2. le significande est non nul, auquel cas le flottant est dénormalisé : d_0 est implicitement fixé à 0 et e est fixé à $e_{\min}$.

Le cas $e = e_{\max} + 1$ permet de représenter des valeurs spéciales. Là encore, le significande permet de distinguer deux cas :

1. s'il est nul, on représente $\pm\infty$ (selon le bit de signe) ;

2. s'il est non nul, on représente une erreur : NaN (« Not a Number »), qui est le plus souvent résultat d'une opération dont le résultat ne peut être connu ($0 \times \infty$ ou $\infty - \infty$ par exemple).

Pour représenter un réel quelconque $x \in \mathbb{R}$ dans l'ensemble des nombres flottants, différentes stratégies d'arrondis sont définies. La plus répandue est celle de l'arrondi au chiffre pair le plus proche, décrite dans la prochaine section. Si $|x| > F_{\max}$, c'est un cas de dépassement (ou overflow) : on le représente par $\pm\infty$. Au contraire, si $|x|$ est trop petit, c'est un cas de sous-passement (ou underflow) : x est arrondi au nombre dénormalisé le plus proche si $|x| \in]F_{\nu}, F_{\varepsilon}[$, ou devient zéro si $|x| < F_{\nu}$.

1.3 Arrondis

Revenons à un système de nombres flottants F défini par une base b , une plage d'exposants $(e_{\min}, e_{\max})$ et une précision p et intéressons-nous de plus près à ce qu'on appelle communément les erreurs d'arrondis. Pour simplifier les notations dans la suite, on définit

$$G := \left\{ x \in \mathbb{R} ; \exists (e, d) \in \mathbb{Z} \times \llbracket 0, b-1 \rrbracket^{p+1} / x = \pm b^e \sum_{n=0}^p \frac{d_n}{b^n} \right\},$$

l'ensemble de tous les nombres flottants avec précision p , sans restriction sur l'exposant. On définit l'application arrondi

$$\begin{aligned} \text{fl} : \mathbb{R} &\rightarrow G \\ x &\mapsto \text{fl}(x) \end{aligned}$$

où l'image d'un réel est définie par :

$$\forall x \in \mathbb{R}, \quad |x - \text{fl}(x)| = \min_{y \in G} |x - y|.$$

Si deux éléments de G vérifient ce minimum, on choisit celui dont le dernier digit est nul : $d_p = 0$.

Définition 1.3.1. On appelle plage de F l'ensemble des réels $x \in \mathbb{R}$ pour lesquels $\text{fl}(x) \in F$ et $\text{fl}(x)$ est normalisé.

Remarque 1.3.2. Si $x \in \mathbb{R}$ est dans la plage de F , on peut construire son représentant flottant explicitement. Considérons pour simplifier $x > 0$. Comme $\text{fl}(x)$ sera un flottant normalisé, on a en particulier :

$$b^{e_{\min}} = F_{\varepsilon} \leq x \leq F_{\max} < b^{e_{\max}+1}.$$

Ainsi :

$$e_{\min} \leq \log_b x < e_{\max} + 1.$$

Soit $e := \lfloor \log_b x \rfloor$ et soit $f := \log_b x - e$. Alors $x = b^e b^f$. Comme $f \in [0, 1[$, on

remarque que $s := b^f \in [1, b[$. En posant $\mu := b^p s$, on a obtenu :

$$x = b^{e-p}\mu, \quad \text{avec } \mu \in [b^p, b^{p+1}[.$$

On obtient alors

$$\text{fl}(x) = b^{e-p}m, \quad \text{avec } m \in \{\lfloor \mu \rfloor, \lfloor \mu \rfloor + 1\},$$

le choix de m permettant de minimiser la distance entre x et $\text{fl}(x)$.

Définition 1.3.3. On définit l'unité d'arrondi comme le réel

$$u := \frac{\varepsilon_M}{2}.$$

Théorème 1.3.4. Soit $x \in \mathbb{R}$ dans la plage de F . Alors :

$$\exists \delta \in]-u, u[\quad \text{fl}(x) = x(1 + \delta).$$

Démonstration. Supposons pour simplifier $x > 0$. Comme x est supposé être dans la plage de F , on peut écrire :

$$x = b^{e-p}\mu, \quad \text{avec } \mu \in [b^p, b^{p+1}[.$$

On peut toujours écrire :

$$\lfloor \mu \rfloor \leq \mu < \lfloor \mu \rfloor + 1,$$

soit :

$$y_1 := b^{e-p}\lfloor \mu \rfloor \leq x \leq b^{e-p}(\lfloor \mu \rfloor + 1) =: y_2.$$

Ainsi, y_1 et y_2 sont les deux éléments adjacents de F les plus proches de x . Autrement dit, $\text{fl}(x) \in \{y_1, y_2\}$, et donc :

$$|x - \text{fl}(x)| \leq \frac{y_2 - y_1}{2} = \frac{b^{e-p}}{2}.$$

On a donc obtenu :

$$\frac{|x - \text{fl}(x)|}{|x|} \leq \frac{\frac{b^{e-p}}{2}}{b^{e-p}\mu} = \frac{1}{2\mu}.$$

On se souvient alors que $\mu \geq b^p$, d'où :

$$\frac{|x - \text{fl}(x)|}{|x|} \leq \frac{1}{2b^p} = u.$$

Si $\mu > b^p$, alors l'inégalité ci-dessus est trivialement stricte. Sinon, $\mu = b^p$. Dans ce cas, $x = b^e$ est représenté de manière exacte : $x = \text{fl}(x)$, d'où l'inégalité stricte également, puisque le membre de droite est nul. ■

Remarque 1.3.5. Dans la preuve, on pourrait aussi estimer $\frac{|x - \text{fl}(x)|}{|\text{fl}(x)|}$. On obtiendrait la même borne. Ainsi, on peut proposer une reformulation du résultat du [Théorème 1.3.4](#) :

$$\exists \delta \in]-u, u[\quad \text{fl}(x) = \frac{x}{1 + \delta}.$$

L'erreur relative commise lorsque l'on stocke un réel est donc bornée par u . Il s'agit d'une borne supérieure : il existe bien entendu des nombres pouvant être représentés de manière exacte, et pour laquelle cette erreur sera donc nulle. Le résultat suivant établit par exemple que les entiers (s'ils ne sont pas trop grands) peuvent être représentés de manière exacte (dans la pratique, si on sait que l'on travaille avec des entiers, il serait tout de même plus judiciable d'utiliser les types dédiés).

Proposition 1.3.6. Soit $m \in \llbracket 1, b^{p+1} - 1 \rrbracket$. Alors $\text{fl}(m) = m$.

Démonstration. Comme $1 \leq m < b^{p+1}$, alors en particulier $0 \leq \log_b m < p + 1$ et ainsi $0 \leq \lfloor \log_b m \rfloor \leq p$. Soit alors $e := \lfloor \log_b m \rfloor$. On note $f := \log_b m - e$, de sorte que $\log_b m = e + f$ et ainsi $m = b^e b^f$. Par définition de la partie entière, on a :

$$e \leq \log_b m < e + 1,$$

d'où l'on déduit $f \in [0, 1[$. En notant $s := b^f$, on obtient alors $s \in [1, b[$. On écrit alors :

$$m = b^e s = b^{e-p} b^p s = b^{e-p} \mu,$$

où $\mu := b^p s \in [b^p, b^{p+1}[$. Or :

$$\mu = b^{p-e} m.$$

Comme $e \leq p$, $p - e \geq 0$, d'où $b^{p-e} \in \mathbb{N}$. Comme $m \in \mathbb{N}$, on en déduit $\mu \in \mathbb{N}$ et ainsi

$$m = b^{e-p} \mu \quad \text{avec } \mu \in \llbracket b^p, b^{p+1} - 1 \rrbracket,$$

ce qui n'est autre que la représentation de m dans la base b avec une précision p . ■

1.4 Opérations sur les flottants

1.4.1 Modèle standard

Le modèle standard de l'arithmétique des flottants impose :

$$\forall (x, y) \in \mathbb{R}^2, \quad \text{fl}(x \text{ op } y) = (x \text{ op } y)(1 + \delta), \quad \text{avec } |\delta| \leq u,$$

où op désigne les opérations élémentaires : $+$, $-$, $\times$ et $/$. Ce modèle est respecté par la norme IEEE 754 qui indique que ces opérations élémentaires doivent être réalisées comme si on avait accès à une « précision infinie » avant d'arrondir le résultat. Concrètement, cela est rendu possible en réalisant les opérations avec des précisions étendues, en augmentant temporairement le nombre de chiffres.

Remarque 1.4.1. Tout comme pour le [Théorème 1.3.4](#), il est équivalent d'imposer :

$$\forall (x, y) \in \mathbb{R}^2, \quad \text{fl}(x \text{ op } y) = \frac{x \text{ op } y}{1 + \delta}, \quad \text{avec } |\delta| \leq u.$$

Exemple 1.4.2. L'augmentation temporaire de la précision est importante. Prenons par exemple, en base 2 et avec une précision $p = 2$: $x = 1.00 \cdot 2^0$ et $y = 1.11 \cdot 2^{-1}$. Pour effectuer l'opération $x - y$, il faut commencer par transformer les opérandes de sorte qu'elles partagent le même exposant (le plus grand des deux). Ici, il faut donc travailler avec $y = 0.111 \cdot 2^0$. On voit tout de suite le problème : une précision de $p = 3$ serait nécessaire. Si on utilise une telle précision, on effectue l'opération

$$1.000 \cdot 2^0 - 0.111 \cdot 2^0 = 0.001 \cdot 2^0 = 1.00 \cdot 2^{-3}.$$

Si au contraire, on reste en précision $p = 2$, y devra être tronqué et on effectue l'opération :

$$1.00 \cdot 2^0 - 0.11 \cdot 2^0 = 0.01 \cdot 2^0 = 1.00 \cdot 2^{-2}.$$

On obtient donc un résultat deux fois plus grand que celui attendu !

Pour un couple $(x, y) \in \mathbb{R}^2$, l'opération $x \text{ op } y$ est réalisée en trois étapes :

1. $\text{fl}(x)$ et $\text{fl}(y)$ sont calculés,
2. l'opération $\text{fl}(x) \text{ op } \text{fl}(y)$ est réalisée en « précision infinie »,
3. le résultat est arrondi pour être stocké ou réutilisé.

Remarque 1.4.3. Le fait que les représentants de x et y soient calculés avant toute autre opération peut paraître naturel : après tout, l'ordinateur ne pourra pas manipuler des quantités abstraites pour lui. Ce simple fait a pourtant quelques conséquences, parfois surprenantes. Il n'est ainsi pas toujours vrai que $\text{fl}(\text{fl}(x) + \text{fl}(y)) = \text{fl}(x + y)$. Par exemple, en base $b = 2$, les réels 0.1, 0.2 et 0.3 ne sont pas représentables en précision finie :

$$\text{fl}(0.1) \neq 0.1, \quad \text{fl}(0.2) \neq 0.2, \quad \text{fl}(0.3) \neq 0.3.$$

On peut montrer avec quelques calculs que :

$$\text{fl}(\text{fl}(0.1) + \text{fl}(0.2)) \neq \text{fl}(0.3).$$

En particulier, dans un code, le test d'égalité $0.1 + 0.2 == 0.3$ est faux ! C'est une des raisons pour lesquelles, de manière générale, il est souvent préférable de ne pas tester l'égalité entre deux flottants, mais plutôt de tester s'ils sont suffisamment proches (pour une tolérance fixée).

Avec pas moins de trois arrondis qui sont effectués, une opération élémentaire n'est pas aussi anodine qu'elle pourrait paraître. Il peut donc être intéressant, quand on s'intéresse

à un algorithme, de « compter » le nombre d'opérations élémentaires qui seront nécessaires à sa réalisation (ou tout du moins en avoir une estimation). Ce nombre peut être exprimé en flops.

| **Définition 1.4.4.** Un flop représente une opération élémentaire.

Plus un algorithme nécessite de flops, plus la machine qui l'exécutera mettra du temps à arriver au bout, et plus elle risque de multiplier les erreurs.

1.4.2 Perte de propriétés

Les opérandes d'une opération élémentaire étant arrondies de manière indépendante, la commutativité de l'addition et de la multiplication est conservée. En revanche, l'associativité est perdue : il n'est pas toujours vrai que $(a + b) + c = (a + (b + c))$, et de même pour la multiplication.

| **Exemple 1.4.5.** Avec $a = \text{fl}(0.1)$, $b = \text{fl}(0.2)$ et $c = \text{fl}(0.3)$, on a :

$$\text{fl}((a + b) + c) \neq \text{fl}(a + (b + c)) \quad \text{et} \quad \text{fl}((ab)c) \neq \text{fl}(a(bc)).$$

De la même façon, on peut trouver des exemples qui montrent que, dans le monde des flottants, l'addition n'est plus distributive par rapport à la multiplication : il n'est pas toujours vrai que $a(b + c) = ab + ac$.

| **Exemple 1.4.6.** Avec $a = \text{fl}(10^{-10})$ et $b = c = \text{fl}(10^{308})$, la somme $b + c$ mène à un dépassement, évité si l'on fait $ab + bc$. Ainsi :

$$\text{fl}(a(b + c)) \neq \text{fl}(ab + ac).$$

1.4.3 Absorption

Le phénomène d'absorption intervient dans la somme de deux nombres dont les exposants sont trop éloignés l'un de l'autre. Lorsque les opérandes sont ramenées au même exposant (le plus grand des deux), il peut arriver que les bits soient décalés « trop loin », menant à une perte d'information pure et simple.

| **Exemple 1.4.7.** Avec $a = \text{fl}(10^9)$ et $b = \text{fl}(10^{-8})$, en double précision, on obtient $a + b = a$.

1.4.4 Annulation catastrophique

Le phénomène d'annulation catastrophique intervient dans la soustraction de deux nombres trop proches : la plupart des chiffres significatifs s'annulent les uns après les autres, et il n'en reste alors que très peu, menant à un résultat de faible précision.

Exemple 1.4.8. Considérons un exemple complètement artificiel, pour comprendre de quoi il retourne. On travaille en base 10, avec une précision $p = 10$. Soient $x = \pi$ et $y = \pi(1 + 10^{-10})$. En arithmétique exacte, on aurait :

$$y - x = \pi \cdot 10^{-10} \approx 3.1415926536 \cdot 10^{-10},$$

où le résultat a été arrondi à 10 chiffres après la virgule. Avec la précision choisie, on a cependant :

$$\text{fl}(x) = 3.1415926536 \quad \text{et} \quad \text{fl}(y) = 3.1415926539.$$

Les deux opérandes ont le même exposant, donc la soustraction peut se faire sans étape préliminaire. Cependant, tous les chiffres de $\text{fl}(x)$ et $\text{fl}(y)$ coïncident, sauf la dernière décimale. Il ne reste donc plus que :

$$\text{fl}(y) - \text{fl}(x) = 3 \cdot 10^{-10},$$

ce qui est une approximation assez grossière du résultat exact, quand on s'attend à avoir une précision à 10 chiffres.

Exemple 1.4.9. Soit $(a, b, c) \in \mathbb{R}^3$ tel que $a \neq 0$ et $b^2 - 4ac > 0$. Il est bien connu que le trinôme $ax^2 + bx + c$ admet alors deux racines réelles données par :

$$x^{\pm} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}.$$

Cette formule peut parfois poser quelques problèmes numériques. Considérons $a = c = 1$ et, pour $n \in \mathbb{N}^*$, $b = -(10^n + 10^{-n})$, de sorte que $x^+ = 10^n$ et $x^- = 10^{-n}$. Avec $n = 8$, en double précision, la formule ci-dessus donne bien $x^+ = 10^8$, mais on obtient $x^- \approx 1.49 \cdot 10^{-8}$. L'erreur diminue si on considère des valeurs de n plus petites. Le problème ici est que $b^2 \gg 4|ac|$, de sorte que $\sqrt{b^2 - 4ac} \approx |b|$. Comme b est négatif, le numérateur de x^- est ainsi une soustraction entre deux valeurs très proches, ce qui explique l'erreur importante. On peut contourner ce problème en calculant d'abord la racine où n'interviendra pas la soustraction problématique, puis en utilisant la relation $x^+ x^- = \frac{c}{a}$. À noter que si l'annulation catastrophique apparaît dans la soustraction $b^2 - 4ac$ (ce qui arrive si les deux racines sont proches), il n'existe aucune solution permettant d'arranger les choses (à part l'utilisation d'une plus grande précision).

1.5 Retrouver les paramètres d'un système

Tout ce qui précède s'applique à un système de nombres flottants, fixé par une base b , avec une précision p et une plage d'exposants $(e_{\min}, e_{\max})$. Pour mieux comprendre les erreurs d'arrondi, il est donc important de connaître les paramètres du système dans lequel on effectue les calculs. Nous allons nous intéresser ici à des méthodes permettant

de retrouver ces paramètres, dans le cas où ceux-ci ne seraient pas accessibles dans une quelconque documentation.

1.5.1 Base

Commençons par nous pencher sur l'[Algorithme 1.5.1](#).

Algorithme 1.5.1 : Déterminer la base d'un système

```

 $\alpha := 1$ 
tant que  $((\alpha + 1) - \alpha) = 1$ 
   $\alpha := 2\alpha$ 

 $\beta := 1$ 
tant que  $((\alpha + \beta) - \alpha) \neq \beta$ 
   $\beta := \beta + 1$ 

```

Du point de vue de l'arithmétique exacte, cet algorithme semble quelque peu absurde : a priori, la condition de la première boucle semble éternellement vérifiée, tandis que la condition de la seconde boucle semble ne jamais l'être. Le monde de l'arithmétique flottante est cependant différent, et il se trouve que, dans la pratique, la première boucle va effectivement s'arrêter, et la seconde boucle fera au moins un tour, le tout pour finalement donner une valeur bien particulière à β .

La preuve de l'arrêt en temps fini de cet algorithme est l'objet du résultat qui suit. L'idée est d'exploiter le phénomène d'absorption : à chaque tour de la première boucle, la valeur de α est augmentée de sorte que, au bout d'un certain temps, son exposant (en tant que flottant) dépassera la précision p . Lorsque l'addition $\alpha + 1$ est réalisée, le décalage nécessaire pour aligner les digits de α et de 1 fera que ce dernier sera finalement représenté par zéro : on obtiendra alors $\text{fl}(\alpha + 1) = \alpha$, d'où la sortie de la boucle. Cette absorption continuera alors dans la seconde boucle, tant que β ne sera pas suffisamment grand pour qu'elle n'ait plus lieu.

Proposition 1.5.2. L'[Algorithme 1.5.1](#) se termine en temps fini avec pour valeur finale $\beta = b$.

Démonstration. La première boucle construit une suite de réels strictement croissante $(\alpha_k)_k$ définie par $\alpha_k = 2^k$. Supposons que $\alpha_k \leq b^{p+1} - 1$. Comme α_k est entier, on a directement $\text{fl}(\alpha_k) = \alpha_k$. Si $\alpha_k + 1 \leq b^{p+1} - 1$, on a de la même façon $\text{fl}(\alpha_k + 1) = \alpha_k + 1$. Sinon, $\alpha_k + 1 = b^{p+1} = b^{e-p}b^p$ avec $e = p + 1$, donc $\text{fl}(\alpha_k + 1) = \alpha_k + 1$. En résumé, dans tous les cas, dès que $\alpha_k \leq b^{p+1} - 1$, on a $\text{fl}(\alpha_k + 1) = \alpha_k + 1$, d'où $\text{fl}((\alpha_k + 1) - \alpha_k) = 1$, et on reste dans la boucle.

Soit maintenant $k \in \mathbb{N}$ le premier entier pour lequel $\alpha_k > b^{p+1} - 1$, c'est-à-dire $\alpha_k \geq b^{p+1}$ puisque $\alpha_k \in \mathbb{N}$. Par minimalité de k , $\alpha_{k-1} \leq b^{p+1} - 1$ donc, par ce qui précède,

$\text{fl}(\alpha_{k-1}) = \alpha_{k-1}$. Ainsi

$$\text{fl}(\alpha_k) = \text{fl}(2 \text{fl}(\alpha_{k-1})) = \text{fl}(2\alpha_{k-1}) = \text{fl}(2^k).$$

Comme $\alpha_{k-1} \leq b^{p+1} - 1$, on a :

$$\alpha_k = 2\alpha_{k-1} \leq 2b^{p+1} - 2.$$

Or, $b \geq 2$, donc $2b^{p+1} \leq b \cdot b^{p+1} = b^{p+2}$. Autrement dit :

$$b^{p+1} \leq \alpha_k \leq b^{p+2} - 2 < b^{p+2}.$$

Ainsi :

$$p + 1 \leq \log_b \alpha_k < p + 2.$$

On a donc obtenu $e := \lfloor \log_b \alpha_k \rfloor = p + 1$. On en déduit :

$$\text{fl}(\alpha_k) = b^e \sum_{n=0}^p \frac{d_n}{b^n} = b^{p+1} \sum_{n=0}^p \frac{d_n}{b^n}.$$

Ainsi :

$$\text{fl}(\alpha_k) + 1 = b^{p+1} \sum_{n=0}^p \frac{d_n}{b^n} + b^{p+1} \frac{1}{b^{p+1}} = b^{p+1} \left(\sum_{n=0}^p \frac{d_n}{b^n} + \frac{1}{b^{p+1}} \right).$$

Autrement dit :

$$\text{fl}(\text{fl}(\alpha_k) + 1) = b^{p+1} \sum_{n=0}^p \frac{d_n}{b^n} = \text{fl}(\alpha_k),$$

d'où :

$$\text{fl}(\text{fl}(\alpha_k) + 1) - \text{fl}(\alpha_k) = 0,$$

et la boucle s'arrêtera alors avec $\alpha = \text{fl}(\alpha_k)$.

Intéressons-nous à la seconde boucle. Supposons $\beta < b$. Comme β est entier et $\beta \in \llbracket 0, b-1 \rrbracket$, on peut directement écrire

$$\text{fl}(\beta) = \beta = b^{p+1} \frac{\beta}{b^{p+1}},$$

d'où :

$$\alpha + \text{fl}(\beta) = b^{p+1} \left(\sum_{n=0}^p \frac{d_n}{b^n} + \frac{\beta}{b^{p+1}} \right),$$

et ainsi :

$$\text{fl}(\alpha + \text{fl}(\beta)) = b^{p+1} \sum_{n=0}^p \frac{d_n}{b^n} = \alpha,$$

donc $\text{fl}(\alpha + \text{fl}(\beta)) - \alpha = 0$ et on reste dans la boucle.

Considérons alors $\beta = b$. Alors $\text{fl}(\beta) = \beta = b = b^{p+1} \frac{1}{b^p}$. Ainsi :

$$\alpha + \beta = b^{p+1} \left(\sum_{n=0}^p \frac{d_n}{b^n} + \frac{1}{b^p} \right) = b^{p+1} \left(\sum_{n=0}^{p-1} \frac{d_n}{b^n} + \frac{d_p + 1}{b^p} \right).$$

Si $d_p \in \llbracket 0, b-2 \rrbracket$, on a donc directement obtenu la représentation de $\alpha + \beta$. Sinon, $d_p + 1 = b$ et on écrit :

$$\frac{d_p + 1}{b^p} = \frac{d_p + 1 - b + b}{b^p} = \frac{1}{b^{p-1}},$$

d'où :

$$\alpha + \beta = b^{p+1} \left(\sum_{n=0}^{p-2} \frac{d_n}{b^n} + \frac{d_{p-1} + 1}{b^{p-1}} \right)$$

et on continue ainsi jusqu'à atteindre un rang $d_n \in \llbracket 0, b-2 \rrbracket$. Si un tel rang n'existe pas, cela signifie que :

$$\alpha = b^{p+1} \sum_{n=0}^p \frac{b-1}{b^n} = b^{p+2} - b,$$

donc $\alpha + \beta = b^{p+2} - b + b = b^{p+2} = b^{e-p} b^p$ avec $e = p+2$. Dans tous les cas, on a obtenu :

$$\text{fl}(\alpha + \beta) = \alpha + \beta$$

donc on sort de la boucle avec la valeur finale $\beta = b$. ■

1.5.2 Précision

Une fois la base connue, il peut être intéressant de récupérer la précision autorisée par le système. On pourrait l'obtenir à partir de l'algorithme précédent (cachée dans la valeur finale de α), mais nous allons voir ici une autre approche qui consiste à calculer la valeur de l'épsilon machine. Là encore, tout repose sur le phénomène d'absorption qui nous permet de détecter le moment à partir duquel il n'est plus possible de dissocier 1 de $1 + \varepsilon$.

Algorithme 1.5.3 : Déterminer la précision d'un système

```

 $\varepsilon := 1$ 
 $k := 0$ 
tant que  $1 + \varepsilon > 1$ 
   $\varepsilon := \frac{\varepsilon}{b}$ 
   $k := k + 1$ 

```

Proposition 1.5.4. L'Algorithme 1.5.3 se termine en temps fini avec pour valeur finale $k = p + 1$.

Démonstration. L'Algorithme 1.5.3 construit une suite $(\varepsilon_k)_k$ avec :

$$\forall k \in \mathbb{N}, \quad \varepsilon_k = b^{-k} = b^0 \frac{1}{b^k}.$$

Pour tout $k \in \llbracket 0, k \rrbracket$, on aura donc :

$$1 + \varepsilon_k = b^0 \left(1 + \sum_{n=1}^{k-1} \frac{0}{b^n} + \frac{1}{b^k} + \sum_{n=k+1}^p \frac{0}{b^n} \right),$$

d'où $\text{fl}(1 + \varepsilon_k) = 1 + \varepsilon_k > 1$. Tant que $k \in \llbracket 0, k \rrbracket$, on reste donc dans la boucle. Considérons maintenant $k = p + 1$. Alors :

$$1 + \varepsilon_{p+1} = b^0 \left(1 + \sum_{n=1}^p \frac{0}{b^n} + \frac{1}{b^{p+1}} \right),$$

d'où $\text{fl}(1 + \varepsilon_{p+1}) = 1$, donc la condition n'est plus vérifiée : on sort de la boucle, avec $k = p + 1$. ■

1.5.3 Plage d'exposants

Il reste à déterminer la plage d'exposants autorisés par le système. Pour déterminer l'exposant maximal, on peut utiliser l'Algorithme 1.5.5 dont l'idée est d'exploiter le dépassement : on construit un nombre sous la forme b^k , avec k de plus en plus grand, jusqu'à atteindre les limites du système, où ce nombre sera alors considéré comme infini.

Algorithme 1.5.5 : Déterminer l'exposant maximal d'un système

```

y := 1
k := 0
tant que by > y
    y := by
    k := k + 1

```

Proposition 1.5.6. L'Algorithme 1.5.5 se termine en temps fini avec pour valeur finale $k = e_{\max} + 1$.

Démonstration. L'Algorithme 1.5.5 construit la suite des

$$y_k = b^k.$$

Si $k < e_{\max}$, $y_k = b^k < b^{e_{\max}}$ donc $\text{fl}(y_k) = y_k$. De plus, $by_k = b^{k+1} \leq b^{e_{\max}}$ donc $\text{fl}(by_k) = by_k$. Ainsi $\text{fl}(by_k) > \text{fl}(y_k)$ et on reste dans la boucle. Si $k = e_{\max}$, $y_k = b^{e_{\max}}$ donc $\text{fl}(y_k) = y_k$. Cependant, $by_k = b^{e_{\max}+1}$ donc $\text{fl}(by_k) = +\infty$. Dans ce cas encore, $\text{fl}(by_k) > \text{fl}(y_k)$ donc on reste dans la boucle. Si $k = e_{\max} + 1$, $y_k = b^{e_{\max}+1}$ donc $\text{fl}(y_k) = +\infty$. De même, $\text{fl}(by_k) = +\infty$. Ainsi, $\text{fl}(by_k) = \text{fl}(y_k)$ donc on sort de la boucle. ■

L'exposant minimal peut être obtenu par une approche similaire, qui est décrite dans l'[Algorithme 1.5.7](#). La preuve peut être facilement adaptée de celle du résultat précédent. L'idée est ici d'exploiter le sous-passement : on construit b^{-k} , jusqu'à ce que ce nombre atteigne les limites du système et soit donc considéré comme 0.

Algorithme 1.5.7 : Déterminer l'exposant minimal d'un système

```

 $y := 1$ 
 $k := 0$ 
tant que  $\frac{y}{b} < y$ 
     $y := \frac{y}{b}$ 
     $k := k - 1$ 

```

Proposition 1.5.8. L'[Algorithme 1.5.7](#) se termine en temps fini avec pour valeur finale $k = e_{\min} - p - 1$.

On remarque la nécessité d'avoir accès à la précision, contrairement à la détermination de l'exposant maximal qui n'en avait pas besoin. Cela est dû au sous-passement graduel : quand y devient plus petit que le plus petit flottant normalisé, il n'est pas encore arrondi à zéro. Cela n'arrivera que quand il deviendra plus petit que le plus petit flottant dénormalisé, d'où les p itérations en plus.

Chapitre 2

Propagation des erreurs et conditionnement

Le chapitre précédent montre que les erreurs sont inévitables : qu'elles viennent de mesures imprécises, d'erreurs dues à des calculs précédents, ou simplement d'arrondis lors du stockage, on ne peut pas supposer que les données d'un calcul sont parfaites. On peut en revanche chercher à faire en sorte que ces erreurs ne soient pas amplifiées.

2.1 Erreurs absolue et relative

Il existe différentes manières de mesurer l'écart entre deux objets. Une approche plutôt naturelle consiste à calculer l'erreur absolue.

Définition 2.1.1. Soit x un élément d'un espace vectoriel normé. Soit $\tilde{x}$ un autre élément dont on s'attend à ce qu'il « approche » x . L'erreur absolue entre x et $\tilde{x}$ est définie par :

$$\|x - \tilde{x}\|.$$

En théorie, on va donc chercher à avoir la plus petite erreur possible. En pratique, cette mesure n'est pas toujours la plus pertinente. Considérons un exemple où x est une certaine longueur, exprimée en mètres, que l'on cherche à mesurer, le résultat de la mesure étant stocké dans $\tilde{x}$. Imaginons que l'on obtienne une erreur absolue égale à 10 : l'écart entre la mesure et la longueur réelle est donc de 10 m. À première vue, cela peut paraître beaucoup, et il est vrai que si l'on cherche à mesurer, par exemple, la longueur d'une étagère d'une bibliothèque, une erreur de 10 m semble importante. Mais si la longueur à approcher est le rayon de la Terre, une erreur de 10 m est plus qu'acceptable. Même si on aimerait que l'erreur absolue soit toujours la plus petite possible, il faut donc bien garder en tête que ce qu'on appelle une erreur « acceptable » dépend fortement du contexte.

La différence entre la longueur d'une étagère et le rayon de la Terre est l'échelle : la première longueur est de l'ordre du mètre, tandis que la seconde est de l'ordre de plusieurs milliers de kilomètres. Pour tenir compte de l'échelle, on peut utiliser l'erreur relative.

Définition 2.1.2. Soit $x \neq 0$ un élément d'un espace vectoriel normé. Soit $\tilde{x}$ un autre élément dont on s'attend à ce qu'il « approche » x . L'erreur relative entre x et $\tilde{x}$ est définie par :

$$\frac{\|x - \tilde{x}\|}{\|x\|}.$$

Pour différencier l'erreur absolue de l'erreur relative, on choisit assez souvent d'exprimer cette dernière en pourcentage. Reprenons notre exemple de mesure d'une longueur. Imaginons que la longueur réelle de l'étagère soit de 3 m et que notre mesure soit de 13 m. On obtient donc une erreur absolue égale à 10, et une erreur relative de plus de 333.33 % (et on devra acheter plus de livres). Le rayon moyen de la Terre est de 6371.008 km = 6 371 008 m [Wik23]. Avec une estimation à 6 371 018 m, on obtient là aussi une erreur absolue égale à 10, mais une erreur relative de 0.0002 %, ce qui semble beaucoup plus acceptable.

Remarque 2.1.3. Quand on cherche à comparer deux réels (une valeur « exacte » et une approximation obtenue par un calcul), il est souvent tentant de comparer leurs chiffres significatifs. Cependant, il ne s'agit pas toujours de la mesure la plus pertinente. Considérons par un exemple un réel $\tilde{x}$ qui doit approcher 1. Si $\tilde{x} = 1.00015$, on obtient une approximation exacte à 3 chiffres après la virgule, ce qui à première vue peut sembler meilleur que $\tilde{x} = 0.99999$ qui ne partage aucun chiffre avec la valeur exacte. Pourtant, cette deuxième approximation est meilleure que la première : l'erreur est inférieure à 0.001 %, là où le premier cas donnait une erreur de 0.015 %.

Le résultat suivant établit la manière dont se propage une erreur sur les opérandes d'une opération élémentaire.

Proposition 2.1.4. Soit $(x, y) \in \mathbb{R}^2$ tel que $x, y \neq 0$. Soit $(\delta_x, \delta_y) \in \mathbb{R}^2$ tel que $\frac{|\delta_x|}{|x|} \ll 1$ et $\frac{|\delta_y|}{|y|} \ll 1$. Alors :

$$\begin{aligned} (x + \delta_x)(y + \delta_y) &= xy \left(1 + \frac{\delta_x}{x} + \frac{\delta_y}{y} + \mathcal{O}\left(\frac{\delta_x \delta_y}{x y}\right) \right), \\ \frac{x + \delta_x}{y + \delta_y} &= \frac{x}{y} \left(1 + \frac{\delta_x}{x} - \frac{\delta_y}{y} + \mathcal{O}\left(\frac{\delta_x \delta_y}{x y}\right) \right), \\ (x + \delta_x) + (y + \delta_y) &= (x + y) \left(1 + \frac{x}{x + y} \frac{\delta_x}{x} + \frac{y}{x + y} \frac{\delta_y}{y} \right). \end{aligned}$$

Démonstration. Multiplication. Il suffit de développer :

$$(x + \delta_x)(y + \delta_y) = xy + x\delta_y + y\delta_x + \delta_x\delta_y,$$

puis on factorise par xy pour obtenir le résultat.

Division. On commence par un développement limité :

$$\frac{1}{y + \delta_y} = \frac{1}{y} - \frac{1}{y} \frac{\delta_y}{y} + \mathcal{O}\left(\left(\frac{\delta_y}{y}\right)^2\right).$$

Ainsi :

$$\frac{x + \delta_x}{y + \delta_y} = \frac{x + \delta_x}{y} - \frac{x + \delta_x}{y} \frac{\delta_y}{y} + \mathcal{O}\left(\frac{\delta_x}{x} \frac{\delta_y}{y}\right).$$

Or :

$$\frac{x + \delta_x}{y} = \frac{x}{y} + \frac{\delta_x}{y} = \frac{x}{y} \left(1 + \frac{\delta_x}{x}\right).$$

Il suffit ensuite de développer.

Addition. Comme pour la multiplication, l'idée est de factoriser par $x + y$ et de faire apparaître l'erreur relative sur les données. ■

En notant que $\delta_x = (x + \delta_x) - x$, la quantité $\frac{\delta_x}{x}$ n'est autre que l'erreur relative obtenue en approchant x par $x + \delta_x$. On peut bien sûr faire la même remarque pour y . Ainsi, $\frac{\delta_x}{x}$ et $\frac{\delta_y}{y}$ représentent les erreurs relatives sur les données. Ce résultat nous dit que la multiplication et la division sont des opérations stables : si les erreurs d'entrée sont petites, les erreurs de sortie le seront aussi.

On ne peut pas en dire autant de l'addition, ou plus exactement de la soustraction. Comme les signes de x et y ne sont pas précisés, le résultat s'applique aux deux opérations. Si x et y sont de même signe, on peut montrer que $\frac{x}{x+y} < 1$, ce qui fait que l'erreur reste contrôlée par les erreurs relatives sur les données : tout va bien. En revanche, si x et y sont de signes opposés, alors l'un des deux quotients $\frac{x}{x+y}$ ou $\frac{y}{x+y}$ sera strictement supérieur à 1 : l'erreur n'est plus contrôlée, ce qui est d'autant plus vrai si $|x|$ et $|y|$ sont proches.

2.2 Conditionnement

On s'intéresse à un calcul écrit sous la forme $y = f(x)$ où f est une fonction entre deux espaces vectoriels normés. Plus précisément, on cherche à déterminer la « qualité » d'une certaine approximation $\hat{y}$ de y en distinguant deux types d'erreurs (voir [Hig02]). Pour simplifier la définition des erreurs considérées, on suppose $x \neq 0$ et $y \neq 0$.

Définition 2.2.1. On appelle erreur aval (ou erreur « forward ») l'erreur (relative) commise sur la sortie :

$$\frac{\|y - \hat{y}\|}{\|y\|}.$$

On appelle erreur amont (ou erreur « backward ») la quantité :

$$\frac{\|x - \hat{x}\|}{\|x\|} \quad \text{où } f(\hat{x}) = \hat{y}.$$

Si la définition de l'erreur aval semble plutôt naturelle, puisqu'elle revient à mesurer la précision avec laquelle on s'est approchés de la solution exacte, celle de l'erreur amont l'est un peu moins. Pour comprendre son intérêt, il faut rappeler le contexte du numérique : on ne dispose, en général, pas de l'entrée exacte x , au sens où la quantité qui sera réellement envoyée à la fonction sera probablement différente, à cause d'imprécisions dans des mesures,

ou à cause d'erreurs d'arrondis, par exemple. L'idée derrière l'analyse de l'erreur amont est de considérer que $\hat{y}$ est le résultat exact de l'exécution de f , mais à partir d'une entrée qui est elle perturbée.

Exemple 2.2.2. On considère un système de nombres flottants F en base $b = 2$ et en précision $p = 2$ (sans limite d'exposants pour simplifier). Soit $f: x \mapsto 2x$. On cherche à déterminer les erreurs commises lors de l'évaluation de f au point $x = 0.2$. On commence par calculer le représentant $\hat{x}$ de x dans F :

$$\hat{x} = 2^{-3} \left(1 + \frac{1}{2} + \frac{0}{4} \right) = \frac{3}{16}.$$

Ainsi on aura :

$$\hat{y} = 2 \times \hat{x} = \frac{3}{8}.$$

Or, $x = 0.2 = \frac{1}{5}$, donc $y = 2x = \frac{2}{5}$. On peut calculer l'erreur aval :

$$\frac{|y - \hat{y}|}{|y|} = \frac{\frac{2}{5} - \frac{3}{8}}{\frac{2}{5}} = \frac{1}{16},$$

ainsi que l'erreur amont :

$$\frac{|x - \hat{x}|}{|x|} = \frac{\frac{1}{5} - \frac{3}{16}}{\frac{1}{5}} = \frac{1}{16}.$$

Les erreurs amont et aval sont liées par une quantité appelée le conditionnement, que l'on définit ici dans le cas réel uniquement.

Définition 2.2.3. Soit $f: \mathbb{R} \rightarrow \mathbb{R}$ une fonction de classe $\mathcal{C}^1$. Le conditionnement de f en un point $x \in \mathbb{R}$ tel que $f(x) \neq 0$ est la quantité :

$$\text{cond}_f(x) := \frac{|xf'(x)|}{|f(x)|}.$$

Proposition 2.2.4. Soit $f: I \rightarrow \mathbb{R}$ de classe $\mathcal{C}^2$ sur I . On suppose qu'il existe deux constantes positives m et M telles que :

$$\forall x \in I, \quad |f''(x)| \leq M \text{ et } |f(x)| \geq m.$$

Soit $x \in I \setminus \{0\}$ tel que $y := f(x) \neq 0$. Soient $\hat{x} \in I$ et $\hat{y} \in \mathbb{R}$ tels que $\hat{y} = f(\hat{x})$. On note $\delta_x := \hat{x} - x$. Alors :

$$\frac{|\hat{y} - y|}{|y|} \leq \frac{|\hat{x} - x|}{|x|} \text{cond}_f(x) + \frac{M}{2m} \delta_x^2.$$

Démonstration. On commence par un développement de Taylor :

$$\hat{y} = f(\hat{x}) = f(x + \delta_x) = f(x) + \delta_x f'(x) + \frac{\delta_x^2}{2} f''(\xi_x), \quad \text{avec } \xi_x \in]\min(x, \hat{x}), \max(x, \hat{x})[.$$

Comme $f(x) = y$, on peut réorganiser :

$$\hat{y} - y = (\hat{x} - x) f'(x) + \frac{\delta_x^2}{2} f''(\xi_x).$$

On divise par $y = f(x)$ pour obtenir :

$$\frac{\hat{y} - y}{y} = (\hat{x} - x) \frac{f'(x)}{f(x)} + \frac{\delta_x^2}{2} \frac{f''(\xi_x)}{f(x)}.$$

On fait apparaître le conditionnement :

$$(\hat{x} - x) \frac{f'(x)}{f(x)} = \frac{\hat{x} - x}{x} \frac{x f'(x)}{f(x)},$$

et le résultat se déduit en passant à la valeur absolue. ■

Exemple 2.2.5. On peut calculer le conditionnement des opérations élémentaires. Pour $x \in \mathbb{R}$ fixé, soit :

$$p_x: y \mapsto xy$$

la multiplication par x . Un calcul rapide montre que $\text{cond}_{p_x}(y) = 1$: on retrouve le fait, établi dans la [Proposition 2.1.4](#), que la multiplication est stable. De même, en considérant

$$d_x: y \mapsto \frac{y}{x}$$

la division par x , on retrouve un conditionnement égal à 1. On retrouve également le fait que l'addition n'est pas toujours stable avec

$$s_x: y \mapsto x + y$$

qui mène à l'expression suivante du conditionnement : $\text{cond}_{s_x}(y) = \frac{|y|}{|x+y|}$, d'autant plus grand que y est proche de $-x$.

Remarque 2.2.6. Le fait qu'une fonction ait un « bon » conditionnement ne suffit pas à garantir un résultat précis : celui-ci dépend en effet de la méthode utilisée pour le calcul de f . Prenons par exemple :

$$f: x \mapsto \sqrt{x+1} - \sqrt{x},$$

dont le conditionnement est égal à :

$$\text{cond}_f(x) = \frac{1}{2} \sqrt{\frac{x}{x+1}}.$$

Pour x grand, le conditionnement est donc proche de $\frac{1}{2}$, ce qui est plutôt bon. Cependant, pour x grand, l'évaluation de f renverra un résultat imprécis. En reformulant l'expression de f , on arrive à des résultats beaucoup plus précis :

$$f(x) = \frac{1}{\sqrt{x+1} + \sqrt{x}}$$

est ainsi plus stable, pour (bien sûr) le même conditionnement. Le problème dans la première expression est la soustraction : on a vu que cette opération était instable si les opérandes étaient proches, ce qui est le cas lorsque x est grand.

2.3 Conditionnement d'un système linéaire

2.3.1 Normes matricielles

Définition 2.3.1. Une norme matricielle sur $M_n(\mathbb{R})$ est une norme qui vérifie :

$$\forall (A, B) \in M_n(\mathbb{R})^2, \quad \|AB\| \leq \|A\| \cdot \|B\|.$$

Exemple 2.3.2. L'application

$$\begin{aligned} \|\cdot\| : M_n(\mathbb{R}) &\rightarrow \mathbb{R}^+ \\ A &\mapsto \max_{1 \leq i, j \leq n} |A_{ij}| \end{aligned}$$

est une norme sur $M_n(\mathbb{R})$ mais pas une norme matricielle. En effet, considérons $A = B = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$. Alors $\|A\| = \|B\| = 1$, mais $AB = \begin{pmatrix} 2 & 2 \\ 2 & 2 \end{pmatrix}$, d'où $\|AB\| = 2 > \|A\| \cdot \|B\|$.

Proposition 2.3.3. Soit $\|\cdot\|_{\mathbb{R}^n}$ une norme sur $\mathbb{R}^n$. Alors l'application

$$\begin{aligned} \|\cdot\| : M_n(\mathbb{R}) &\rightarrow \mathbb{R}^+ \\ A &\mapsto \sup_{x \in \mathbb{R}^n \setminus \{0\}} \frac{\|Ax\|_{\mathbb{R}^n}}{\|x\|_{\mathbb{R}^n}} \end{aligned}$$

définit une norme matricielle sur $M_n(\mathbb{R})$, appelée norme subordonnée à la norme vectorielle $\|\cdot\|_{\mathbb{R}^n}$.

Démonstration. Norme. Vérifions dans un premier temps que cette application définit bien une norme. Si $A = 0$, alors pour tout $x \in \mathbb{R}^n$, $Ax = 0$, et on obtient bien $\|A\| = 0$. Soit maintenant $A \in M_n(\mathbb{R})$ telle que $\|A\| = 0$. On a :

$$\forall x \in \mathbb{R}^n \setminus \{0\}, \quad \frac{\|Ax\|_{\mathbb{R}^n}}{\|x\|_{\mathbb{R}^n}} \leq \|A\| = 0,$$

ce qui implique :

$$\forall x \in \mathbb{R}^n, \quad \|Ax\|_{\mathbb{R}^n} = 0,$$

et ainsi :

$$\forall x \in \mathbb{R}^n, \quad Ax = 0$$

car $\|\cdot\|_{\mathbb{R}^n}$ est une norme vectorielle. En choisissant par exemple $x := e_j$ le j -ème vecteur de la base canonique de $\mathbb{R}^n$, on obtient que la j -ème colonne de A est identiquement nulle, et ce pour tout $j \in \llbracket 1, n \rrbracket$, ce qui montre bien que $A = 0$.

Soit $(A, B) \in M_n(\mathbb{R})^2$. En utilisant l'inégalité triangulaire sur $\|\cdot\|_{\mathbb{R}^n}$, on obtient :

$$\forall x \in \mathbb{R}^n \setminus \{0\}, \quad \frac{\|(A+B)x\|_{\mathbb{R}^n}}{\|x\|_{\mathbb{R}^n}} \leq \frac{\|Ax\|_{\mathbb{R}^n}}{\|x\|_{\mathbb{R}^n}} + \frac{\|Bx\|_{\mathbb{R}^n}}{\|x\|_{\mathbb{R}^n}} \leq \|A\| + \|B\|.$$

En passant au sup, on obtient l'inégalité triangulaire sur la norme subordonnée.

Soient $A \in M_n(\mathbb{R})$ et $\lambda \in \mathbb{R}$. Toujours en utilisant le fait que $\|\cdot\|_{\mathbb{R}^n}$ est une norme, on trouve :

$$\forall x \in \mathbb{R}^n \setminus \{0\}, \quad \sup_{x \in \mathbb{R}^n \setminus \{0\}} \frac{\|(\lambda A)x\|_{\mathbb{R}^n}}{\|x\|_{\mathbb{R}^n}} = \sup_{x \in \mathbb{R}^n \setminus \{0\}} |\lambda| \frac{\|Ax\|_{\mathbb{R}^n}}{\|x\|_{\mathbb{R}^n}} = |\lambda| \|A\|.$$

Norme matricielle. Soit $(A, B) \in M_n(\mathbb{R})^2$. Alors :

$$\begin{aligned} \|AB\| &= \sup_{x \in \mathbb{R}^n \setminus \{0\}} \frac{\|(AB)x\|_{\mathbb{R}^n}}{\|x\|_{\mathbb{R}^n}} = \sup_{x \notin \ker B} \frac{\|ABx\|_{\mathbb{R}^n}}{\|x\|_{\mathbb{R}^n}} \\ &= \sup_{x \notin \ker B} \frac{\|ABx\|_{\mathbb{R}^n}}{\|Bx\|_{\mathbb{R}^n}} \frac{\|Bx\|_{\mathbb{R}^n}}{\|x\|_{\mathbb{R}^n}} \\ &\leq \sup_{x \notin \ker B} \frac{\|ABx\|_{\mathbb{R}^n}}{\|Bx\|_{\mathbb{R}^n}} \times \sup_{x \in \mathbb{R}^n \setminus \{0\}} \frac{\|Bx\|_{\mathbb{R}^n}}{\|x\|_{\mathbb{R}^n}} \\ &\leq \sup_{y \in \mathbb{R}^n \setminus \{0\}} \frac{\|Ay\|_{\mathbb{R}^n}}{\|y\|_{\mathbb{R}^n}} \times \sup_{x \in \mathbb{R}^n \setminus \{0\}} \frac{\|Bx\|_{\mathbb{R}^n}}{\|x\|_{\mathbb{R}^n}} = \|A\| \cdot \|B\|, \end{aligned}$$

ce qui termine la preuve. ■

Ce résultat permet notamment de définir des normes matricielles à partir de normes vectorielles bien connues, comme par exemple :

- $\|\cdot\|_p : \mathbf{v} \mapsto \left(\sum_{i=1}^n |v_i|^p \right)^{1/p}$, pour $p \in \mathbb{N}^*$;
- $\|\cdot\|_\infty : \mathbf{v} \mapsto \max_{1 \leq i \leq n} |v_i|$.

Pour certaines de ces normes, on peut cependant expliciter un peu plus précisément leurs expressions.

Théorème 2.3.4. Les normes matricielles subordonnées aux normes vectorielles $\|\cdot\|_1$, $\|\cdot\|_2$ et $\|\cdot\|_\infty$ sont données par les expressions :

$$\begin{aligned} \text{— } \|A\|_1 &= \sup_{x \in \mathbb{R}^n \setminus \{0\}} \frac{\|Ax\|_1}{\|x\|_1} = \max_{1 \leq j \leq n} \sum_{i=1}^n |A_{ij}|; \\ \text{— } \|A\|_\infty &= \sup_{x \in \mathbb{R}^n \setminus \{0\}} \frac{\|Ax\|_\infty}{\|x\|_\infty} = \max_{1 \leq i \leq n} \sum_{j=1}^n |A_{ij}|; \\ \text{— } \|A\|_2 &= \sup_{x \in \mathbb{R}^n \setminus \{0\}} \frac{\|Ax\|_2}{\|x\|_2} = \sqrt{\rho(A^*A)}; \end{aligned}$$

pour $A \in M_n(\mathbb{R})$.

Démonstration. Norme 1. Pour tout $x \in \mathbb{R}^n$, on a :

$$\begin{aligned} \|Ax\|_1 &= \sum_{i=1}^n |(Ax)_i| = \sum_{i=1}^n \left| \sum_{j=1}^n a_{ij} x_j \right| \leq \sum_{i=1}^n \sum_{j=1}^n |a_{ij}| \cdot |x_j| \\ &\leq \sum_{j=1}^n |x_j| \sum_{i=1}^n |a_{ij}| \\ &\leq \left(\max_{1 \leq j \leq n} \sum_{i=1}^n |a_{ij}| \right) \underbrace{\sum_{j=1}^n |x_j|}_{\|x\|_1}. \end{aligned}$$

Ainsi :

$$\|A\|_1 = \sup_{x \in \mathbb{R}^n \setminus \{0\}} \frac{\|Ax\|_1}{\|x\|_1} \leq \max_{1 \leq j \leq n} \sum_{i=1}^n |a_{ij}|.$$

Pour montrer l'égalité, il suffit alors de trouver un vecteur permettant d'atteindre le max. Soit pour cela $j_0 \in \llbracket 1, n \rrbracket$ tel que :

$$\sum_{i=1}^n |a_{ij_0}| = \max_{1 \leq j \leq n} \sum_{i=1}^n |a_{ij}|.$$

On pose $x \in \mathbb{R}^n$ défini par :

$$x_j := \begin{cases} 1, & \text{si } j = j_0, \\ 0, & \text{sinon.} \end{cases}$$

Dans ce cas :

$$\|Ax\|_1 = \sum_{i=1}^n \left| \sum_{j=1}^n a_{ij} x_j \right| = \sum_{i=1}^n |a_{ij_0}| = \max_{1 \leq j \leq n} \sum_{i=1}^n |a_{ij}|,$$

par définition de x et de j_0 .

Norme ∞ . La preuve est assez similaire au cas de la norme 1. Pour tout $x \in \mathbb{R}^n$, on a :

$$\begin{aligned} \|Ax\|_\infty &= \max_{1 \leq i \leq n} |(Ax)_i| = \max_{1 \leq i \leq n} \left| \sum_{j=1}^n a_{ij} x_j \right| \leq \max_{1 \leq i \leq n} \sum_{j=1}^n |a_{ij}| \cdot \underbrace{|x_j|}_{\leq \|x\|_\infty} \\ &\leq \left(\max_{1 \leq i \leq n} \sum_{j=1}^n |a_{ij}| \right) \|x\|_\infty. \end{aligned}$$

On en déduit :

$$\|A\|_\infty \leq \max_{1 \leq i \leq n} \sum_{j=1}^n |a_{ij}|$$

et il suffit donc de trouver un vecteur permettant d'atteindre le maximum pour démontrer l'égalité. Soit pour cela $i_0 \in \llbracket 1, n \rrbracket$ tel que :

$$\sum_{j=1}^n |a_{i_0 j}| = \max_{1 \leq i \leq n} \sum_{j=1}^n |a_{ij}|.$$

On considère $x \in \mathbb{R}^n$ défini par :

$$x_j := \begin{cases} 1, & \text{si } a_{i_0 j} \geq 0, \\ -1, & \text{sinon.} \end{cases}$$

Dans ce cas :

$$\|Ax\|_\infty = \max_{1 \leq i \leq n} \left| \sum_{j=1}^n a_{ij} x_j \right| \geq \left| \sum_{j=1}^n a_{i_0 j} x_j \right| = \left| \sum_{j=1}^n |a_{i_0 j}| \right| = \max_{1 \leq i \leq n} \sum_{j=1}^n |a_{ij}|$$

par définition de x et de i_0 . Comme ici $\|x\|_\infty = 1$, on a ainsi :

$$\max_{1 \leq i \leq n} \sum_{j=1}^n |a_{ij}| \leq \frac{\|Ax\|_\infty}{\|x\|_\infty} \leq \|A\|_\infty \leq \max_{1 \leq i \leq n} \sum_{j=1}^n |a_{ij}|,$$

d'où l'égalité.

Norme 2. La matrice A^*A est auto-adjointe positive. Ses valeurs propres sont donc dans $\mathbb{R}^+$. On les note :

$$0 \leq \lambda_1 \leq \dots \leq \lambda_n.$$

Il existe $\{e_1, \dots, e_n\}$ une base orthonormée de vecteurs propres associés. Soit $x \in \mathbb{R}^n$. On peut écrire :

$$x = \sum_{i=1}^n \alpha_i e_i, \quad \text{avec } \alpha_i \in \mathbb{R}.$$

Ainsi :

$$\begin{aligned}
\|Ax\|_2^2 &= (Ax, Ax) = (A^*Ax, x) = \left(A^*A \sum_{i=1}^n \alpha_i e_i, \sum_{j=1}^n \alpha_j e_j \right) \\
&= \sum_{i=1}^n \sum_{j=1}^n \alpha_i \overline{\alpha_j} \underbrace{((A^*A)e_i, e_j)}_{\alpha_i e_i} \\
&= \sum_{i=1}^n \sum_{j=1}^n \lambda_i \alpha_i \overline{\alpha_j} \underbrace{(e_i, e_j)}_{\delta_{ij}} \\
&= \sum_{i=1}^n \lambda_i |\alpha_i|^2,
\end{aligned}$$

d'où :

$$\frac{\|Ax\|_2^2}{\|x\|_2^2} = \frac{\sum_{i=1}^n \overbrace{\lambda_i}^{\leq \lambda_n} |\alpha_i|^2}{\sum_{i=1}^n |\alpha_i|^2} \leq \lambda_n \frac{\sum_{i=1}^n |\alpha_i|^2}{\sum_{i=1}^n |\alpha_i|^2} = \lambda_n = \rho(A^*A).$$

On a donc bien $\|A\|_2 \leq \sqrt{\rho(A^*A)}$. Pour obtenir l'égalité, on choisit $x \in \mathbb{R}^n$ tel que :

$$\alpha_i = \begin{cases} 0, & \text{si } i < n, \\ 1, & \text{si } i = n. \end{cases}$$

Dans ce cas :

$$\|Ax\|_2^2 = \lambda_n |\alpha_n|^2 \quad \text{et} \quad \|x\|_2^2 = |\alpha_n|^2,$$

d'où :

$$\frac{\|Ax\|_2^2}{\|x\|_2^2} = \lambda_n = \rho(A^*A),$$

et on obtient le résultat voulu. ■

Un exemple de norme matricielle, mais dont on montre aisément qu'elle n'est subordonnée à aucune norme vectorielle, est donné par la norme de Frobenius.

Définition 2.3.5. L'application

$$\|\cdot\|_F : A \mapsto \sqrt{\sum_{i=1}^n \sum_{j=1}^n |A_{ij}|^2}$$

définit une norme matricielle, appelée norme de Frobenius.

Proposition 2.3.6. Soit $A \in M_n(\mathbb{R})$. Alors :

$$\|A\|_2 \leq \|A\|_F \leq \sqrt{n}\|A\|_2.$$

Démonstration. Contrôle de la norme 2. Soit $x \in \mathbb{R}^n$, écrit sous la forme

$$x = \sum_{j=1}^n x_j e_j$$

avec e_j les vecteurs de la base canonique. On peut ainsi écrire :

$$\begin{aligned} \|Ax\|_2 &= \left\| \sum_{j=1}^n x_j A e_j \right\|_2 \\ &\leq \sum_{j=1}^n |x_j| \|A e_j\|_2 \\ &\leq \underbrace{\left(\sum_{j=1}^n |x_j|^2 \right)^{1/2}}_{\|x\|_2} \left(\sum_{j=1}^n \|A e_j\|_2^2 \right)^{1/2}. \end{aligned}$$

On en déduit :

$$\frac{\|Ax\|_2}{\|x\|_2} \leq \left(\sum_{j=1}^n \|A e_j\|_2^2 \right)^{1/2}.$$

Or,

$$\sum_{j=1}^n \|A e_j\|_2^2 = \sum_{j=1}^n \sum_{i=1}^n |A_{ij}|^2 = \|A\|_F^2,$$

d'où :

$$\|A\|_2 \leq \|A\|_F,$$

puisque l'inégalité a été obtenue pour x quelconque.

Contrôle de la norme de Frobenius. Réciproquement, on écrit

$$\|A\|_F^2 = \sum_{j=1}^n \|A e_j\|_2^2.$$

Or, $\|A e_j\|_2 \leq \|A\|_2 \|e_j\|_2 = \|A\|_2$, donc :

$$\|A\|_F^2 \leq \sum_{j=1}^n \|A\|_2^2 = n \|A\|_2^2,$$

d'où l'inégalité recherchée. ■

2.3.2 Conditionnement

Définition 2.3.7. Soit $A \in M_n(\mathbb{R})$ inversible. On appelle conditionnement de A le réel :

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

où $\|\cdot\|$ est une norme subordonnée.

Le conditionnement d'une matrice dépend de la norme considérée. Lorsque cela est nécessaire, on pourra rappeler la norme utilisée, en utilisant par exemple la notation cond_2 pour indiquer l'utilisation de la norme 2.

Proposition 2.3.8. Soit $A \in M_n(\mathbb{R})$ inversible. Alors :

1. $\text{cond}(A) \geq 1$;
2. $\forall \alpha \in \mathbb{R}^*, \text{cond}(\alpha A) = \text{cond}(A)$.
3. Si, de plus, A est normale (c'est-à-dire, pour rappel, $A^*A = AA^*$), alors :

$$\text{cond}_2(A) = \frac{\max_{\lambda \in \text{Sp}(A)} |\lambda|}{\min_{\lambda \in \text{Sp}(A)} |\lambda|}.$$

Démonstration. Commençons par les deux premiers points.

1. Comme $\|\cdot\|$ est subordonnée, on peut écrire :

$$1 = \|I\| = \|AA^{-1}\| \leq \|A\| \cdot \|A^{-1}\| = \text{cond}(A).$$

2. Soit $\alpha \in \mathbb{R}^*$. On écrit la définition du conditionnement :

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

3. Intéressons-nous maintenant à la norme 2. Par définition :

$$\text{cond}_2(A) = \|A\|_2 \cdot \|A^{-1}\|_2.$$

On applique le [Théorème 2.3.4](#) :

$$\text{cond}_2(A) = \sqrt{\rho(A^*A)} \sqrt{\rho((A^{-1})^*A^{-1})} = \sqrt{\rho(A^*A)} \sqrt{\rho((A^*A)^{-1})}.$$

Les valeurs propres de A^*A sont réelles. Comme A est inversible, elles sont de plus strictement positives. On peut donc les ordonner :

$$0 < \lambda_1^2 \leq \lambda_2^2 \leq \dots \leq \lambda_n^2,$$

où les λ_i sont les valeurs propres de A . Ainsi :

$$\rho(A^*A) = \lambda_n^2 \quad \text{et} \quad \rho((A^*A)^{-1}) = \frac{1}{\lambda_1^2},$$

d'où l'égalité.

■

Tout comme dans le cas des fonctions réelles, le conditionnement permet de relier les erreurs amont et aval lors de la résolution d'un système linéaire.

Théorème 2.3.9. Soient $A \in M_n(\mathbb{R})$ inversible et $b \in \mathbb{R}^n \setminus \{0\}$. Soit $x \in \mathbb{R}^n$ la solution du système $Ax = b$. Soient $\tilde{A} \in M_n(\mathbb{R})$ inversible une « perturbation » de A et $\tilde{b} \in \mathbb{R}^n$ une « perturbation » de b . On note $\delta_A := \tilde{A} - A$ et $\delta_b := \tilde{b} - b$. Soit $\tilde{x} \in \mathbb{R}^n$ la solution du système perturbé $\tilde{A}\tilde{x} = \tilde{b}$. Supposons que $\|\delta_A\| < \|A^{-1}\|^{-1}$. Alors l'erreur relative est contrôlée :

$$\frac{\|x - \tilde{x}\|}{\|x\|} \leq \frac{\text{cond}(A)}{1 - \text{cond}(A) \frac{\|\delta_A\|}{\|A\|}} \left(\frac{\|\delta_b\|}{\|b\|} + \frac{\|\delta_A\|}{\|A\|} \right).$$

Démonstration. Par définition de x et $\tilde{x}$, on peut écrire :

$$\begin{aligned} \delta_b &= \tilde{b} - b = \tilde{A}\tilde{x} - Ax \\ &= \tilde{A}\tilde{x} - Ax + A\tilde{x} - A\tilde{x} \\ &= A(\tilde{x} - x) + \delta_A\tilde{x} \end{aligned}$$

et ainsi :

$$\tilde{x} - x = A^{-1}\delta_b + A^{-1}\delta_A\tilde{x},$$

d'où :

$$\|\tilde{x} - x\| \leq \|A^{-1}\| \cdot \|\delta_b\| + \|A^{-1}\| \cdot \|\delta_A\| \cdot \|\tilde{x}\|.$$

Par ailleurs, on peut toujours écrire

$$\tilde{x} = \tilde{x} - x + x,$$

que l'on injecte dans l'inégalité précédente :

$$\|\tilde{x} - x\| \leq \|A^{-1}\| \cdot \|\delta_b\| + \|A^{-1}\| \cdot \|\delta_A\| \cdot \|\tilde{x} - x\| + \|A^{-1}\| \cdot \|\delta_A\| \cdot \|x\|.$$

On en déduit :

$$\frac{\|x - \tilde{x}\|}{\|x\|} \leq \frac{\|A^{-1}\|}{1 - \|A^{-1}\| \cdot \|\delta_A\|} \left(\frac{\|\delta_b\|}{\|x\|} + \|\delta_A\| \right) = \frac{\|A\| \cdot \|A^{-1}\|}{1 - \|A^{-1}\| \cdot \|\delta_A\|} \left(\frac{\|\delta_b\|}{\|A\| \cdot \|x\|} + \frac{\|\delta_A\|}{\|A\|} \right).$$

Or, $b = Ax$, donc $\|b\| \leq \|A\| \cdot \|x\|$, d'où $\frac{1}{\|A\| \cdot \|x\|} \leq \frac{1}{\|b\|}$, d'où le résultat en utilisant de plus la définition du conditionnement. ■

Théorème 2.3.10. Soient $A \in M_n(\mathbb{R})$ inversible et $b \in \mathbb{R}^n \setminus \{0\}$. Soit $x \in \mathbb{R}^n$ la solution du système $Ax = b$. Soit $\tilde{b} \in \mathbb{R}^n$. On note $\delta_b := \tilde{b} - b$. Soit $\tilde{x} \in \mathbb{R}^n$ la solution

du système $A\tilde{x} = \tilde{b}$. Alors :

$$\frac{1}{\text{cond}(A)} \frac{\|\delta_b\|}{\|b\|} \leq \frac{\|\tilde{x} - x\|}{\|x\|} \leq \text{cond}(A) \frac{\|\delta_b\|}{\|b\|}.$$

Démonstration. La partie droite de l'inégalité est une conséquence directe du résultat précédent (en considérant $\delta_A = 0$). Intéressons-nous à la partie gauche. On commence par écrire :

$$\delta_b = \tilde{b} - b = A(\tilde{x} - x).$$

Ainsi :

$$\|\delta_b\| \leq \|A\| \cdot \|\tilde{x} - x\|,$$

d'où l'on déduit :

$$\frac{\|\delta_b\|}{\|b\|} \leq \|A\| \cdot \|\tilde{x} - x\| \cdot \frac{1}{\|b\|}.$$

Or, $x = A^{-1}b$, donc $\|x\| \leq \|A^{-1}\| \cdot \|b\|$ et ainsi :

$$\frac{\|\delta_b\|}{\|b\|} \leq \|A\| \cdot \|\tilde{x} - x\| \cdot \frac{\|A^{-1}\|}{\|x\|},$$

et on obtient le résultat en réorganisant les termes. ■

Ce dernier résultat nous permet d'encadrer l'erreur relative assez précisément. En particulier, il nous donne une sorte de condition d'optimalité : si le conditionnement de la matrice est égal à 1, l'erreur relative commise sur la solution du système est exactement égale à l'erreur relative commise sur le terme source. Autrement dit, on a donc tout intérêt à chercher à avoir un conditionnement le plus proche possible de 1.

Chapitre 3

Méthodes directes

Dans cette partie, $A \in M_n(\mathbb{R})$ désigne une matrice inversible. On s'intéresse à la résolution du système linéaire

$$Ax = b$$

avec $b \in \mathbb{R}^n$.

3.1 Une fausse bonne idée : la méthode de Cramer

La méthode de Cramer permet d'avoir une expression explicite des composantes de la solution du système.

Théorème 3.1.1. On note $(C^j)_{j \in \llbracket 1, n \rrbracket}$ les colonnes de A :

$$\forall j \in \llbracket 1, n \rrbracket, \quad \forall i \in \llbracket 1, n \rrbracket, \quad C_i^j := A_{ij}.$$

Pour $i \in \llbracket 1, n \rrbracket$, on note A_i la matrice obtenue en remplaçant la i -ème colonne de A par b . Alors :

$$\forall i \in \llbracket 1, n \rrbracket, \quad x_i = \frac{\det(A_i)}{\det(A)}.$$

Démonstration. Un rapide calcul donne :

$$b = \sum_{j=1}^n C^j x_j.$$

Soit $i \in \llbracket 1, n \rrbracket$. Par linéarité du déterminant par rapport à la i -ème variable, on a :

$$\det(A_i) = \sum_{j=1}^n x_j \det(C^1, \dots, C^{i-1}, C^j, C^{i+1}, \dots, C^n).$$

Pour $j \neq i$, on retrouve deux fois la colonne C^j donc :

$$\det(C^1, \dots, C^{i-1}, C^j, C^{i+1}, \dots, C^n) = 0.$$

Il reste alors :

$$\det(A_i) = x_i \det(C^1, \dots, C^{i-1}, C^i, C^{i+1}, \dots, C^n) = x_i \det(A),$$

d'où le résultat. ■

On obtient une expression simple de la solution du système, et plutôt simple à implémenter. Le problème, c'est qu'il n'est pas anodin de calculer un déterminant, et que cette méthode nécessite d'en calculer $n + 1$. Sauf pour un très petit nombre d'inconnues, le temps de calcul devient vite énorme. On peut ajouter à cela le fait que la méthode de Cramer est généralement instable [Hig02]. En résumé : il s'agit d'un résultat intéressant d'un point de vue théorique, mais à bannir en pratique.

3.2 Systèmes triangulaires

Dans cette section, A désigne une matrice triangulaire (supérieure ou inférieure).

Remarque 3.2.1. La matrice A étant toujours supposée inversible, sa structure triangulaire permet d'assurer que ses coefficients diagonaux sont tous non nuls.

Observons ce qu'il se passe dans le cas où A est triangulaire inférieure, c'est-à-dire de la forme :

$$A = \begin{pmatrix} a_{11} & & & & \\ a_{21} & a_{22} & & & 0 \\ a_{31} & a_{32} & a_{33} & & \\ \vdots & \vdots & \vdots & \ddots & \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} \end{pmatrix},$$

ou, un peu plus formellement, vérifiant $a_{ij} = 0$ dès que $j > i$. On peut écrire le système correspondant :

$$\begin{cases} a_{11}x_1 & = b_1 \\ a_{21}x_1 + a_{22}x_2 & = b_2 \\ \vdots & \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \dots + a_{nn}x_n & = b_n \end{cases}$$

qui peut être résolu assez facilement : la première ligne permet d'obtenir x_1 , et ainsi la seconde ligne permet d'obtenir x_2 , etc., jusqu'à atteindre la dernière ligne qui donne x_n . Il s'agit de la méthode de descente, décrite dans l'Algorithme 3.2.2.

Un système triangulaire supérieur pourra être résolu de manière similaire, à ceci près qu'on partira cette fois-ci de x_n pour remonter jusqu'à x_1 : on obtient la méthode de remontée, décrite dans l'Algorithme 3.2.3.

Proposition 3.2.4. Les méthodes de descente et de remontée ont une complexité en $\mathcal{O}(n^2)$ flops.

Algorithme 3.2.2 : Méthode de descente

```

 $x_1 := \frac{b_1}{a_{11}}$ 
pour  $i$  de 2 à  $n$ 
   $\left[ \begin{array}{l} x_i := \frac{1}{a_{ii}} \left( b_i - \sum_{j=1}^{i-1} a_{ij}x_j \right) \end{array} \right.$ 

```

Algorithme 3.2.3 : Méthode de remontée

```

 $x_n := \frac{b_n}{a_{nn}}$ 
pour  $i$  de  $n-1$  à 1
   $\left[ \begin{array}{l} x_i := \frac{1}{a_{ii}} \left( b_i - \sum_{j=i+1}^n a_{ij}x_j \right) \end{array} \right.$ 

```

Démonstration. La première étape de la méthode de descente nécessite :

— 1 division.

Pour $i \in \llbracket 2, n \rrbracket$, on a ensuite :

— 1 division,

— $i-1$ multiplications,

— $i-1$ additions.

Au total, on peut donc compter :

— n divisions,

— $\sum_{i=2}^n (i-1) = \frac{n(n-1)}{2}$ multiplications,

— autant d'additions.

On a donc exactement :

$$n + 2 \frac{n(n-1)}{2} = n^2$$

flops. Le calcul pour la méthode de remontée est similaire. ■

On se propose ici d'étudier la stabilité d'une implémentation de la méthode de la descente (la remontée ayant les mêmes propriétés), consistant à calculer les x_i de manière assez naturelle. Nous aurons pour cela besoin des deux résultats suivants.

Lemme 3.2.5. Soit $n \in \mathbb{N}^*$ tel que $nu < 1$ où u est l'unité d'arrondi. Soient $(\delta_i)_{i=1}^n \in [-u, u]^n$ et $(\rho_i)_{i=1}^n \in \{\pm 1\}^n$. Alors :

$$\prod_{i=1}^n (1 + \delta_i)^{\rho_i} = 1 + \theta_n, \quad \text{où } |\theta_n| \leq \frac{nu}{1 - nu} =: \gamma_n.$$

Démonstration. On procède par récurrence sur n . Commençons par le cas $n = 1$. On suppose $\rho_1 = 1$. Vérifions qu'alors $\theta_1 := \delta_1$ convient. De l'inégalité $0 < u < 1$, on tire deux choses : $1 - u > 0$ et $1 - u < 1$. Ainsi :

$$|\theta_1| = |\delta_1| \leq u < \frac{u}{1 - u} = \gamma_1.$$

Supposons maintenant $\rho_1 = -1$. On va vérifier que

$$\theta_1 := -\frac{\delta_1}{1 + \delta_1}$$

convient. En effet :

$$|\theta_1| = \frac{|\delta_1|}{|1 + \delta_1|}.$$

Or, $-u \leq \delta_1 \leq u$, donc en particulier $1 + \delta_1 \geq 1 - u > 0$ et ainsi $|1 + \delta_1| = 1 + \delta_1 \geq 1 - u$. Ainsi :

$$\frac{1}{1 + \delta_1} \leq \frac{1}{1 - u}$$

et donc :

$$|\theta_1| = \frac{|\delta_1|}{1 + \delta_1} \leq \frac{|\delta_1|}{1 - u} \leq \frac{u}{1 - u} = \gamma_1.$$

Soit $n \in \mathbb{N}^*$, $n \geq 2$ tel que le résultat soit vrai jusqu'au rang $n - 1$. Autrement dit :

$$\prod_{i=1}^n (1 + \delta_i)^{\rho_i} = \left(\prod_{i=1}^{n-1} (1 + \delta_i)^{\rho_i} \right) (1 + \delta_n)^{\rho_n} = (1 + \theta_{n-1})(1 + \delta_n)^{\rho_n}$$

avec $|\theta_{n-1}| \leq \gamma_{n-1}$. Étudions là encore les deux cas possibles. Commençons par supposer $\rho_n = 1$. On pose $\theta_n := \delta_n + \theta_{n-1}(1 + \delta_n)$. Alors :

$$\begin{aligned} |\theta_n| &\leq |\delta_n| + |\theta_{n-1}|(1 + \delta_n) \\ &\leq u + \frac{(n-1)u}{1 - (n-1)u}(1 + u) \\ &= \frac{nu}{1 - (n-1)u}. \end{aligned}$$

Comme $nu < 1$, $1 - nu > 0$ et il en va de même pour $1 - (n-1)u$. Ainsi :

$$\frac{1}{1 - (n-1)u} \leq \frac{1}{1 - nu}$$

et on en déduit $|\theta_n| \leq \gamma_n$. Supposons maintenant $\rho_n = -1$. On pose cette fois-ci

$$\theta_n := \frac{\theta_{n-1} - \delta_n}{1 + \delta_n}.$$

On a alors :

$$\begin{aligned}
 |\theta_n| &\leq \frac{|\theta_{n-1}| + |\delta_n|}{|1 + \delta_n|} \\
 &\leq \frac{\frac{(n-1)u}{1-(n-1)u} + u}{1 + \delta_n} \\
 &= \frac{nu}{(1 - (n-1)u)(1 + \delta_n)} - \frac{(n-1)u^2}{(1 - (n-1)u)(1 + \delta_n)} \\
 &\leq \frac{nu}{(1 - (n-1)u)(1 + \delta_n)}.
 \end{aligned}$$

Or, $-u \leq \delta_n \leq u$ donc, en particulier, $1 + \delta_n \geq 1 - u$ et ainsi $(1 - (n-1)u)(1 + \delta_n) \geq (1 - (n-1)u)(1 - u) \geq 1 - nu$. ■

Algorithme 3.2.6 : Retranchement d'un produit scalaire partiel

Entrée : $a \in \mathbb{R}^*$, $c \in \mathbb{R}$, $(u_i)_{i=1}^{k-1} \in \mathbb{R}^{k-1}$ et $(v_i)_{i=1}^{k-1} \in \mathbb{R}^{k-1}$

Sortie : $y := \frac{1}{a} \left(c - \sum_{i=1}^{k-1} u_i v_i \right)$

$s := c$

pour i de 1 à $k-1$

$s := s - u_i v_i$

$y := \frac{s}{a}$

Lemme 3.2.7. Soient $a \in \mathbb{R}^*$, $c \in \mathbb{R}$, $(u_i)_{i=1}^{k-1} \in \mathbb{R}^{k-1}$ et $(v_i)_{i=1}^{k-1} \in \mathbb{R}^{k-1}$. Soit

$$y := \frac{1}{a} \left(c - \sum_{i=1}^{k-1} u_i v_i \right).$$

On note $\hat{y}$ l'approximation numérique de y calculée par l'Algorithme 3.2.6. Alors :

$$a\hat{y}(1 + \theta_k) = c - \sum_{i=1}^{k-1} u_i v_i (1 + \theta_i).$$

Démonstration. On commence par la somme

$$s := c - \sum_{i=1}^{k-1} u_i v_i.$$

Celle-ci est initialisée à c où on suppose l'absence de perturbations :

$$\hat{s}^{(0)} := c.$$

On retranche ensuite le premier terme du produit scalaire :

$$\hat{s}^{(1)} := \text{fl}(s^{(0)} - u_1 v_1).$$

Deux opérations sont réalisées ici : le produit $u_1 v_1$, qui mène à une erreur d'arrondi qui sera notée ε_1 , et la soustraction $s^{(0)} - u_1 v_1$, qui mène à une seconde erreur d'arrondi qui sera notée δ_1 . On obtient :

$$\hat{s}^{(1)} = (\hat{s}^{(0)} - u_1 v_1(1 + \varepsilon_1))(1 + \delta_1) = c(1 + \delta_1) - u_1 v_1(1 + \varepsilon_1)(1 + \delta_1).$$

Vient ensuite le second terme du produit scalaire :

$$\hat{s}^{(2)} := \text{fl}(\hat{s}^{(1)} - u_2 v_2).$$

On trouve les mêmes opérations que précédemment :

$$\begin{aligned} \hat{s}^{(2)} &= (\hat{s}^{(1)} - u_2 v_2(1 + \varepsilon_2))(1 + \delta_2) \\ &= c(1 + \delta_1)(1 + \delta_2) - u_1 v_1(1 + \varepsilon_1)(1 + \delta_1)(1 + \delta_2) - u_2 v_2(1 + \varepsilon_2)(1 + \delta_2). \end{aligned}$$

On montre ainsi par récurrence que :

$$\forall j \in \llbracket 1, k-1 \rrbracket, \quad \hat{s}^{(j)} = c \prod_{i=1}^j (1 + \delta_i) - \sum_{i=1}^j u_i v_i (1 + \varepsilon_i) \prod_{m=i}^j (1 + \delta_m).$$

En particulier, la valeur finale de la somme est donnée par $\hat{s} := \hat{s}^{(k-1)}$ qui peut être explicitée :

$$\hat{s} = c \prod_{i=1}^{k-1} (1 + \delta_i) - \sum_{i=1}^{k-1} u_i v_i (1 + \varepsilon_i) \prod_{j=i}^{k-1} (1 + \delta_j).$$

Reste la division par a :

$$\hat{y} = \text{fl}\left(\frac{\hat{s}}{a}\right) = \frac{\hat{s}}{a} \frac{1}{1 + \delta_k},$$

d'où :

$$a\hat{y}(1 + \delta_k) = \hat{s} = c \prod_{i=1}^{k-1} (1 + \delta_i) - \sum_{i=1}^{k-1} u_i v_i (1 + \varepsilon_i) \prod_{j=i}^{k-1} (1 + \delta_j).$$

On divise cette égalité par le produit des $(1 + \delta_i)$:

$$a\hat{y} \frac{1 + \delta_k}{\prod_{i=1}^{k-1} (1 + \delta_i)} = c - \sum_{i=1}^{k-1} u_i v_i \frac{1 + \varepsilon_i}{\prod_{j=1}^{i-1} (1 + \delta_j)}.$$

On applique alors le lemme précédent pour conclure. ■

Théorème 3.2.8. Soit $L \in M_n(\mathbb{R})$ une matrice triangulaire inférieure et inversible. Soit $b \in \mathbb{R}^n$. Soit $\tilde{x} \in \mathbb{R}^n$ la solution du système $Lx = b$ obtenue par la méthode de descente. Alors $\tilde{x}$ est en réalité solution de :

$$\tilde{L}\tilde{x} = \tilde{b},$$

où $\tilde{L} = L + \delta_L$ et $\tilde{b} = b + \delta_b$ sont tels que :

$$\forall (i, j) \in \llbracket 1, n \rrbracket^2, \quad |(\delta_L)_{i,j}| \leq \begin{cases} \gamma_j |L_{ij}|, & \text{si } i \geq j, \\ 0, & \text{sinon} \end{cases} \quad \text{et} \quad |(\delta_b)_i| \leq u|b_i|.$$

Démonstration. Soit $i \in \llbracket 1, n \rrbracket$. La i -ème composante de x est donnée par :

$$x_i = \frac{1}{L_{ii}} \left(b_i - \sum_{j=1}^{i-1} L_{ij} x_j \right).$$

On reconnaît l'expression du [Lemme 3.2.7](#). Ainsi :

$$L_{ii}\hat{x}_i(1 + \theta_i^{(i)}) = b_i(1 + \delta_0^{(i)}) - \sum_{j=1}^{i-1} L_{ij}\hat{x}_j(1 + \theta_j^{(i)}),$$

où on prend en compte le fait de ne pas nécessairement connaître b de manière exacte. On réorganise les termes :

$$\sum_{j=1}^i L_{ij}\hat{x}_j(1 + \theta_j^{(i)}) = b_i(1 + \delta_0^{(i)}).$$

On définit δ_L par :

$$\forall (i, j) \in \llbracket 1, n \rrbracket^2, \quad (\delta_L)_{ij} = \theta_j^{(i)} L_{ij}.$$

Comme L et δ_L sont triangulaires inférieures, on a ainsi obtenu :

$$((L + \delta_L)\hat{x})_i = b_i(1 + \delta_0^{(i)}),$$

ce qui est bien le résultat annoncé. ■

Avec les mêmes arguments, on peut obtenir un résultat similaire pour les systèmes triangulaires supérieurs.

Théorème 3.2.9. Soit $U \in M_n(\mathbb{R})$ une matrice triangulaire supérieure et inversible. Soit $b \in \mathbb{R}^n$. Soit $\tilde{x} \in \mathbb{R}^n$ la solution du système $Ux = b$ obtenue par la méthode de remontée. Alors $\tilde{x}$ est en réalité solution de :

$$\tilde{U}\tilde{x} = \tilde{b},$$

où $\tilde{U} = U + \delta_U$ et $\tilde{b} = b + \delta_b$ sont tels que :

$$\forall (i, j) \in \llbracket 1, n \rrbracket^2, \quad |(\delta_U)_{i,j}| \leq \begin{cases} \gamma_{j-i}|U_{ij}|, & \text{si } j > i, \\ \gamma_{n-i+1}|U_{ii}|, & \text{si } j = i, \\ 0, & \text{sinon} \end{cases} \quad \text{et} \quad |(\delta_b)_i| \leq u|b_i|.$$

Corollaire 3.2.10. Soit $L \in M_n(\mathbb{R})$ une matrice triangulaire inférieure inversible. Soit $b \in \mathbb{R}^n$. Soit $\tilde{x} \in \mathbb{R}^n$ la solution du système $Lx = b$ obtenue par la méthode de descente. Alors :

$$\frac{\|x - \tilde{x}\|_2}{\|x\|_2} \leq \frac{\text{cond}_2(L)}{1 - \gamma_n \sqrt{n} \text{cond}_2(L)} (u + \gamma_n \sqrt{n}).$$

Démonstration. On part de :

$$\frac{\|x - \tilde{x}\|_2}{\|x\|_2} \leq \frac{\text{cond}_2(L)}{1 - \text{cond}_2(L) \frac{\|\delta_L\|_2}{\|L\|_2}} \left(\frac{\|\delta_b\|_2}{\|b\|_2} + \frac{\|\delta_L\|_2}{\|L\|_2} \right).$$

Or :

$$\forall i \in \llbracket 1, n \rrbracket, \quad |(\delta_b)_i| \leq u|b_i|,$$

d'où l'on tire facilement :

$$\|\delta_b\|_2 \leq u\|b\|_2.$$

De même :

$$\forall (i, j) \in \llbracket 1, n \rrbracket^2, \quad |(\delta_L)_{ij}| \leq \gamma_i |L_{ij}|.$$

Or :

$$\forall i \in \llbracket 1, n \rrbracket, \quad \gamma_i \leq \gamma_n$$

et on obtient :

$$\|\delta_L\|_F \leq \gamma_n \|L\|_F.$$

Ainsi :

$$\|\delta_L\|_2 \leq \|\delta_L\|_F \leq \gamma_n \|L\|_F \leq \gamma_n \sqrt{n} \|L\|_2,$$

d'où le résultat. ■

3.3 Méthode du pivot de Gauss

Les systèmes triangulaires sont simples à inverser, mais tous les systèmes linéaires ne sont bien sûr pas triangulaires. Dans le cas général, une idée pourrait être de transformer le système, en le multipliant par exemple par une matrice M inversible :

$$Ax = b \iff MAx = Mb.$$

L'intérêt est de faire en sorte que la matrice MA ait une « meilleure structure ». Par exemple, si elle est triangulaire, on pourra résoudre facilement le second système. C'est justement le principe de la méthode du pivot de Gauss qui consiste à triangulariser un système linéaire par des opérations sur ses lignes.

Exemple 3.3.1. Soit

$$A := \begin{pmatrix} 2 & 1 & 3 \\ 4 & 5 & 7 \\ 2 & 4 & 1 \end{pmatrix}.$$

On cherche à donner à cette matrice une structure triangulaire supérieure, colonne par colonne. Pour la première colonne, on ne touche pas au premier coefficient qui doit rester non nul. On doit annuler le premier coefficient de la deuxième ligne, ce qui peut se faire en retranchant deux fois la première ligne. On annule également le premier coefficient de la troisième ligne en retranchant la première :

$$A^{(1)} = \begin{pmatrix} 2 & 1 & 3 \\ 0 & 3 & 1 \\ 0 & 3 & -2 \end{pmatrix}.$$

Il reste maintenant le dernier coefficient de la seconde colonne à annuler. On peut pour cela retrancher la deuxième ligne.

$$A^{(2)} = \begin{pmatrix} 2 & 1 & 3 \\ 0 & 3 & 1 \\ 0 & 0 & -3 \end{pmatrix}.$$

Triangulariser la matrice ne suffit pas : il faut également effectuer les mêmes opérations sur les coefficients du second membre. Cela fonctionne car les opérations effectuées peuvent être traduites matriciellement, en utilisant des matrices de permutation et de transvection. Dans le contexte de ce cours, une matrice de permutation entre p et q est la matrice identité dans laquelle les lignes p et q ont été échangées. Multiplier à gauche une matrice A par cette matrice de permutation échange les lignes p et q de A . Multiplier à droite échange les colonnes p et q . On appellera matrice de transvection (agissant sur la colonne m) la matrice identité, avec les éléments situés strictement sous la diagonale, et sur la m -ème colonne, non nécessairement nuls. Multiplier à gauche une matrice A par une matrice de transvection permet d'effectuer des combinaisons linéaires de lignes de A .

Lemme 3.3.2. Soit $B \in M_k(\mathbb{R})$ inversible, pour $k \in \mathbb{N}^*$ quelconque. Alors il existe une matrice de transposition P et une matrice de transvection T telles que $\tilde{B} := TPB$ vérifie :

$$\tilde{B}_{11} \neq 0, \quad \forall i \in \llbracket 2, k \rrbracket, \tilde{B}_{i1} = 0, \quad \det \tilde{B} = \pm \det B.$$

Démonstration. Comme B est inversible, en particulier sa première colonne est non nulle et il existe au moins un $p \in \llbracket 1, k \rrbracket$ tel que $B_{p,1} \neq 0$. Soit P la matrice de transposition entre 1 et p . Alors $B' := PB$ est la matrice B où ont été échangées la première et la p -ème lignes. En particulier, $\rho := B'_{1,1} = B_{p,1} \neq 0$. On peut alors définir une matrice de transvection T dont tous les éléments diagonaux sont égaux à 1 et dont la première colonne est donnée

par :

$$\forall i \in \llbracket 2, k \rrbracket, \quad T_{i,1} = -\frac{B'_{i,1}}{\rho},$$

tous les autres coefficients étant nuls. Alors $\tilde{B} := TPB$ satisfait les propriétés voulues. En effet :

$$\forall i \in \llbracket 2, n \rrbracket, \quad \tilde{B}_{i,1} = \sum_{j=1}^k T_{i,j} B'_{j,1} = T_{i,1} B'_{1,1} + T_{i,i} B'_{i,1} = -\frac{B'_{i,1}}{\rho} \rho + 1 \cdot B'_{i,1} = 0,$$

et :

$$\tilde{B}_{1,1} = \sum_{j=1}^k T_{1,j} B'_{j,1} = B'_{1,1} \neq 0.$$

Par ailleurs :

$$\det \tilde{B} = \det T \det P \det B.$$

La matrice T est triangulaire inférieure de diagonale égale à 1, donc $\det T = 1$. Si $p = 1$, alors $P = I$ et $\det P = 1$ et ainsi $\det \tilde{B} = \det B$. Si $p > 1$, alors $\det P = -1$ et $\det \tilde{B} = -\det B$. ■

Théorème 3.3.3. Soit $A \in M_n(\mathbb{R})$ une matrice inversible. Alors il existe $M \in M_n(\mathbb{R})$ une matrice inversible telle que la matrice MA soit triangulaire supérieure.

Démonstration. On va montrer qu'il est possible de construire une suite $(A^{(m)})_{m=1}^{n-1}$ telle que, pour tout $m \in \llbracket 1, n-1 \rrbracket$, pour tout $j \in \llbracket 1, m \rrbracket$:

$$A_{jj}^{(m)} \neq 0 \quad \text{et} \quad \forall i \in \llbracket j+1, n \rrbracket, A_{ij}^{(m)} = 0.$$

Autrement dit, à chaque étape m , les m premières colonnes de $A^{(m)}$ ont la structure d'une matrice triangulaire supérieure inversible. On va également montrer que :

$$\forall m \in \llbracket 1, n-1 \rrbracket, \quad \det A^{(m)} = \pm \det A.$$

Le cas $m = 1$ est donné par le lemme précédent : on trouve une matrice de permutation $P^{(1)}$ et une matrice de transvection $T^{(1)}$ telles que $A^{(1)} := T^{(1)} P^{(1)} A$ vérifie les propriétés souhaitées. Supposons donc cette suite construite jusqu'à un certain rang $m \in \llbracket 1, n-2 \rrbracket$. On peut en particulier écrire :

$$A^{(m)} = \begin{pmatrix} U & F \\ 0 & B \end{pmatrix}$$

avec $U \in M_m(\mathbb{R})$ triangulaire supérieure à coefficients diagonaux non nuls et $B \in M_{n-m}(\mathbb{R})$. Comme $A^{(m)}$ est triangulaire par blocs, on a directement :

$$\det A^{(m)} = (\det U)(\det B).$$

Par hypothèse, $\det A^{(m)} = \pm \det A \neq 0$ car A est inversible. De plus, les coefficients diagonaux de la matrice triangulaire U sont non nuls, donc $\det U \neq 0$. On en déduit $\det B \neq 0$. En appliquant à nouveau le lemme, on trouve une matrice de permutation $P \in M_{n-m}(\mathbb{R})$ et une matrice de transvection $T \in M_{n-m}(\mathbb{R})$ telles que tous les coefficients, sauf le premier, de la première colonne de $\tilde{B} := TPB$ sont nuls, avec de plus $\det \tilde{B} = \pm \det B$. On pose :

$$P^{(m+1)} := \begin{pmatrix} I & 0 \\ 0 & P \end{pmatrix} \quad \text{et} \quad T^{(m+1)} := \begin{pmatrix} I & 0 \\ 0 & T \end{pmatrix},$$

de taille $n \times n$. Ainsi :

$$A^{(m+1)} := P^{(m+1)} T^{(m+1)} A^{(m)} = \begin{pmatrix} I & 0 \\ 0 & P \end{pmatrix} \begin{pmatrix} I & 0 \\ 0 & T \end{pmatrix} \begin{pmatrix} U & F \\ 0 & B \end{pmatrix} = \begin{pmatrix} U & F \\ 0 & \tilde{B} \end{pmatrix}$$

vérifie les propriétés voulues.

La matrice $A^{(n-1)}$ est donc triangulaire supérieure inversible et peut s'écrire comme $A^{(n-1)} = MA$ avec

$$M := \prod_{m=1}^{n-1} T^{(n-m)} P^{(n-m)}$$

qui est bien inversible. ■

Dans la pratique, on n'a pas besoin de calculer la matrice M explicitement pour résoudre un système linéaire : on réalise les différentes opérations (échanges et combinaisons linéaires de lignes) directement sur A et b . En absence de changement de pivot (dans la preuve précédente, cela revient à avoir toutes les matrices de permutation égales à l'identité), on peut écrire l'[Algorithme 3.3.4](#).

Algorithme 3.3.4 : Méthode du pivot de Gauss

```

pour k de 1 à n - 1
  pour i de k + 1 à n
    Mik := Aik / Akk
    pour j de k à n
      Aij = Aij - Mik Akj
    bi = bi - Mik bk

```

Proposition 3.3.5. La méthode du pivot de Gauss est en $\mathcal{O}\left(\frac{2n^3}{3}\right)$ flops.

Démonstration. Pour $k \in \llbracket 1, n-1 \rrbracket$ et $i \in \llbracket k+1, n \rrbracket$, on compte :

— 1 division

pour calculer le pivot. On compte également :

- 1 multiplication,
- 1 addition

pour la modification du vecteur. Enfin, toujours pour toutes ces valeurs de k et i , pour $j \in \llbracket k, n \rrbracket$, on compte :

- 1 multiplication,
- 1 addition

pour la modification de la matrice. On obtient alors un total de :

- $\sum_{k=1}^{n-1} \sum_{i=k+1}^n 1 = \frac{n(n-1)}{2}$ divisions pour le pivot,
- $\frac{n(n-1)}{2}$ multiplications et additions pour le vecteur,
- $\sum_{k=1}^{n-1} \sum_{i=k+1}^n \sum_{j=k}^n 1 = \frac{n(n-1)(n+1)}{3}$ multiplications et additions pour la matrice.

On a donc exactement :

$$3 \frac{n(n-1)}{2} + 2 \frac{n(n-1)(n+1)}{3} = \frac{n(n-1)(4n+13)}{6}$$

flops. Il faut ajouter à cela la méthode de remontée pour résoudre le système triangulaire final, qui est en $\mathcal{O}(n^2)$. ■

Théoriquement, il suffit qu'un coefficient soit non nul pour qu'il puisse être utilisé comme pivot. En pratique, les erreurs d'arrondis peuvent nous éloigner fortement de la solution. Un exemple simple est donné par le système :

$$\begin{pmatrix} \varepsilon & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \end{pmatrix},$$

avec $\varepsilon > 0$. Comme $\varepsilon \neq 0$, on peut appliquer le pivot de Gauss sans changement de pivot pour obtenir le système équivalent :

$$\begin{pmatrix} \varepsilon & 1 \\ 0 & 1 - \varepsilon^{-1} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 1 \\ 2 - \varepsilon^{-1} \end{pmatrix}.$$

On obtient alors la solution théorique :

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \frac{1}{1-\varepsilon} \\ \frac{2\varepsilon-1}{\varepsilon-1} \end{pmatrix}.$$

Lorsque ε tend vers 0, la solution de ce système tend donc vers $(1, 1)$. Cependant, numériquement, si ε est trop petit, un phénomène d'absorption va survenir et on aura $\text{fl}(1 - \varepsilon^{-1}) = -\text{fl}(\varepsilon^{-1})$ et $\text{fl}(2 - \varepsilon^{-1}) = -\text{fl}(\varepsilon^{-1})$, ce qui donnera $y = 1$, et donc $x = 0$. On a donc tout intérêt à changer de pivot, même si le coefficient « naturel » est non nul. Plusieurs stratégies sont alors possibles.

Changement partiel. À l'étape $k \in \llbracket 1, n-1 \rrbracket$, on recherche le pivot de plus grand module dans la sous-colonne à traiter :

$$|a_{pk}| = \max_{i \in \llbracket k, n \rrbracket} |a_{ik}|,$$

ce qui assure que les coefficients m_{ik} seront bornés par 1. Une fois le pivot trouvé, on échange les lignes p et k de la matrice et du vecteur. Matriciellement, cela revient à multiplier le système à gauche par la matrice de permutation P entre k et p :

$$PAx = Pb.$$

Changement total. À l'étape $k \in \llbracket 1, n-1 \rrbracket$, on recherche le pivot de plus grand module dans la sous-matrice restante :

$$|a_{pq}| = \max_{(i,j) \in \llbracket k, n \rrbracket^2} |a_{ij}|.$$

On échange alors les lignes p et k de la matrice et du vecteur, ainsi que les colonnes q et k de la matrice, et les lignes q et k du vecteur inconnu. Matriciellement, en notant P la matrice de permutation entre k et p , et Q la matrice de permutation entre k et q , on obtient :

$$P(AQ)(Qx) = Pb.$$

Changement « tour ». À l'étape $k \in \llbracket 1, n-1 \rrbracket$, on recherche un pivot de plus grand module sur toute sa ligne et toute sa colonne :

$$|a_{pq}| = \max_{i \in \llbracket k, n \rrbracket} |a_{iq}| = \max_{j \in \llbracket k, n \rrbracket} |a_{pj}|.$$

On procède ensuite comme pour le changement total.

Remarque 3.3.6. Un pivot trouvé par changement « tour » sera forcément de module inférieur ou égal à celui trouvé par changement total. L'intérêt est dans le nombre d'opérations effectuées. Alors qu'un changement partiel nécessite $\mathcal{O}(n^2)$ comparaisons, un changement total va en nécessiter $\mathcal{O}(n^3)$. Il apparaît que, en pratique, le changement « tour » s'approche plus de $\mathcal{O}(n^2)$ que de $\mathcal{O}(n^3)$, ce qui justifie qu'il peut être vu comme un compromis entre les deux autres changements.

Remarque 3.3.7. Il n'y a pas toujours unicité du pivot : deux coefficients peuvent être de module maximal dans la sous-colonne ou la sous-matrice. C'est encore plus vrai dans le cas du changement « tour », pour lequel on n'utilisera pas nécessairement un coefficient de module maximal. On peut imaginer plusieurs façons d'implémenter un changement « tour ». Partant du coefficient en haut à gauche de la sous-matrice, on peut par exemple rechercher dans la colonne le coefficient de plus grand module. On arrive sur une nouvelle ligne, dans laquelle on recherche le coefficient de plus grand module. On recherche alors dans cette nouvelle colonne, et ainsi de suite. Cette méthode est illustrée graphiquement par la [Figure 3.1](#) dans laquelle on peut retrouver le nom de ce type de changement : les « mouvements » sont ceux d'une tour

aux échecs.

3	2	3	8	4
4	4	7	1	8
2	9	4	5	5
5	3	6	2	1
1	2	5	3	9

3	2	3	8	4
4	4	7	1	8
2	9	4	5	5
5	3	6	2	1
1	2	5	3	9

FIGURE 3.1 – Illustration du changement « tour ». Dans les deux cas, on alterne la recherche sur les lignes et les colonnes. À gauche, on commence par la colonne. À droite, on commence par la ligne.

3.4 Factorisation LU

La méthode du pivot de Gauss est en quelque sorte une « préparation » du système avant sa résolution effective : on transforme le système tout entier avant de le résoudre par une méthode de remontée. Une autre façon de préparer le système est de factoriser la matrice : c'est l'idée de la décomposition LU.

Théorème 3.4.1. Soit $A \in M_n(\mathbb{R})$ inversible. Si, pour tout $k \in \llbracket 1, n \rrbracket$, $\Delta_k := (A_{ij})_{(i,j) \in \llbracket 1, k \rrbracket^2}$ est inversible, alors il existe un unique couple de matrices (L, U) tel que L est triangulaire inférieure à diagonale uniquement constituée de 1, U est triangulaire supérieure et :

$$A = LU.$$

Démonstration. Existence. Reprenons la preuve du [Théorème 3.3.3](#). Pour $k \in \llbracket 1, n-1 \rrbracket$, on a

$$A^{(k)} = \prod_{m=1}^k T^{(k+1-m)} P^{(k+1-m)} A.$$

Nous allons montrer que toutes les matrices de permutation peuvent être choisies comme étant l'identité. Comme Δ_1 est inversible, en particulier $A_{11} \neq 0$, donc on peut choisir $P^{(1)} = I$. Soit $k \in \llbracket 1, n-2 \rrbracket$ tel que, pour tout $m \in \llbracket 1, k \rrbracket$, $P^{(m)} = I$. Ainsi :

$$A^{(k)} = \prod_{m=1}^k T^{(k+1-m)} A.$$

On note

$$T := \prod_{m=1}^k T^{(k+1-m)} = \begin{pmatrix} \tilde{T} & 0 \\ \star & I \end{pmatrix}$$

avec $\tilde{T} \in M_{k+1}(\mathbb{R})$ triangulaire inférieure dont les coefficients diagonaux sont tous égaux à 1. De manière similaire, on note

$$A^{(k)} = \begin{pmatrix} \tilde{A} & \star \\ 0 & \star \end{pmatrix}$$

avec $\tilde{A} \in M_{k+1}(\mathbb{R})$ triangulaire supérieure. On a ainsi :

$$\begin{pmatrix} \tilde{A} & \star \\ 0 & \star \end{pmatrix} = A^{(k)} = TA = \begin{pmatrix} \tilde{T} & 0 \\ \star & I \end{pmatrix} \begin{pmatrix} \Delta_{k+1} & \star \\ \star & \star \end{pmatrix},$$

d'où l'on déduit $\tilde{A} = \tilde{T}\Delta_{k+1}$. En particulier, $\det \tilde{A} = (\det \tilde{T})(\det \Delta_{k+1})$. Or, $\det \tilde{T} = 1$ et $\det \Delta_{k+1} \neq 0$ par hypothèse, donc $\det \tilde{A} \neq 0$. Comme $\tilde{A}$ est triangulaire, on a cependant :

$$\det \tilde{A} = \prod_{m=1}^{k+1} A_{mm}^{(k)}.$$

On en déduit que $A_{k+1,k+1}^{(k)} \neq 0$, ce qui implique que l'on peut choisir $P^{(k+1)} = I$.

On obtient ainsi $U := A^{(n-1)} = TA$ triangulaire supérieure. Comme T est triangulaire inférieure à diagonale uniquement constituée de 1, il en va de même pour $L := T^{-1}$, d'où le résultat.

Unicité. Soient L_1 et L_2 deux matrices triangulaires inférieures à diagonales uniquement constituées de 1, et U_1 et U_2 deux matrices triangulaires supérieures telles que :

$$A = L_1 U_1 = L_2 U_2.$$

Ainsi :

$$L_2^{-1} L_1 = U_2 U_1^{-1}.$$

On a donc égalité entre une matrice triangulaire inférieure et une matrice triangulaire supérieure. On peut donc écrire $L_2^{-1} L_1 = U_2 U_1^{-1} = D$ avec D diagonale. Or, les diagonales de L_1 et L_2 sont uniquement constituées de 1, donc il en est de même pour celle de $L_2^{-1} L_1$. Ainsi, $D = I$, et on en déduit $L_1 = L_2$ et $U_1 = U_2$. ■

Si on connaît la décomposition LU de la matrice, on peut reformuler le système :

$$Ax = b \iff (LU)x = b \iff L(Ux) = b \iff \begin{cases} Ly = b \\ Ux = y. \end{cases}$$

Ce système peut être résolu en deux étapes. On calcule d'abord y en résolvant $Ly = b$ via une méthode de descente. On peut ensuite calculer x en résolvant $Ux = y$ via une méthode

de remontée. Au total, il faudra donc de l'ordre de $\mathcal{O}(2n^2)$ flops pour résoudre le système, ce qui est plus avantageux que de faire appel à la méthode du pivot de Gauss. Bien sûr, c'est sans compter le calcul de la décomposition LU elle-même qui est, elle, équivalente à la méthode du pivot de Gauss. S'il s'agit de ne résoudre qu'un seul système, il est donc contre-productif de calculer la décomposition LU. L'avantage de ce type de décomposition apparaît lorsque l'on cherche à résoudre plusieurs fois le « même » système linéaire : la matrice reste la même, mais on change le membre de droite. On calcule la décomposition LU à la première inversion, puis les résolutions suivantes seront beaucoup plus rapides.

Concrètement, on peut voir dans la preuve de l'existence que la matrice U n'est autre que celle construite par la méthode du pivot de Gauss, tandis que la matrice L est l'inverse du produit de toutes les matrices de transvection utilisées. Il semble donc naturel d'utiliser le même algorithme pour calculer la décomposition LU. Une option intéressante du point de vue du coût en mémoire consiste à remplacer la matrice A par sa décomposition, stockée dans une seule et même matrice : la partie triangulaire supérieure (diagonale comprise) correspondra à U , tandis que la partie triangulaire inférieure (diagonale exclue) correspondra à L (au lieu de ne contenir que des zéros). Ne pas stocker la diagonale de L n'est pas un problème, puisque l'on sait qu'elle ne contient que des 1. On obtient l'[Algorithme 3.4.2](#).

Algorithme 3.4.2 : Décomposition LU

```

pour  $k$  de 1 à  $n - 1$ 
  pour  $i$  de  $k + 1$  à  $n$ 
     $\alpha := \frac{A_{ik}}{A_{kk}}$ 
    pour  $j$  de  $k$  à  $n$ 
       $A_{ij} = A_{ij} - \alpha A_{kj}$ 
     $A_{ik} = \alpha$ 

```

Une autre manière d'obtenir un algorithme de calcul de factorisation LU consiste à expliciter un tel produit. Comme L est triangulaire inférieure et U est triangulaire supérieure, on trouve facilement :

$$\forall (i, j) \in \llbracket 1, n \rrbracket^2, \quad A_{ij} = \sum_{p=1}^{\min(i,j)} L_{ip} U_{pj}.$$

Ceci nous donne pour la première ligne :

$$\forall j \in \llbracket 1, n \rrbracket, \quad A_{1j} = L_{11} U_{1j}.$$

Or, L_{11} est un élément diagonal de L et est donc égal à 1 : on obtient la première ligne de U . Écrivons la relation sur la première colonne :

$$\forall i \in \llbracket 2, n \rrbracket, \quad A_{i1} = L_{i1} U_{11}.$$

Comme U_{11} fait partie de la première ligne de U , désormais connue, on obtient ainsi toute la première colonne de L . On peut généraliser le procédé. Soit $k \in \llbracket 2, n \rrbracket$. Supposons connues

les $k-1$ premières lignes de U ainsi que les $k-1$ premières colonnes de L . Comme $L_{kk} = 1$, on a :

$$\forall j \in \llbracket k, n \rrbracket, \quad A_{kj} = \sum_{i=1}^{k-1} L_{ki} u_{ij} + U_{kj}.$$

La somme utilise des termes des lignes et colonnes connues de U et L , donc on peut en déduire les coefficients de la k -ème ligne de U . De même :

$$\forall i \in \llbracket k+1, n \rrbracket, \quad A_{ik} = \sum_{j=1}^{k-1} L_{ij} U_{jk} + L_{ik} U_{kk}.$$

Comme U_{kk} a été calculé juste avant, on obtient la k -ème colonne de L . On obtient l'[Algorithme 3.4.3](#).

Algorithme 3.4.3 : Méthode de Doolittle

```

pour  $k$  de 1 à  $n$ 
  pour  $j$  de  $k$  à  $n$ 
     $U_{kj} := A_{kj} - \sum_{i=1}^{k-1} L_{ki} U_{ij}$ 
  pour  $i$  de  $k+1$  à  $n$ 
     $L_{ik} := \frac{1}{U_{kk}} \left( A_{ik} - \sum_{j=1}^{k-1} L_{ij} U_{jk} \right)$ 

```

Proposition 3.4.4. La méthode de Doolittle nécessite $\mathcal{O}\left(\frac{n^3}{3}\right)$ flops.

Démonstration. On trouve autant de multiplications que d'additions :

$$\sum_{k=1}^n \left(\sum_{j=k}^n (k-1) + \sum_{i=k+1}^n (k-1) \right) = \frac{n(n-1)(2n-1)}{6}.$$

Il faut également ajouter :

$$\sum_{k=1}^n \sum_{i=k+1}^n 1 = \frac{n(n-1)}{2}$$

divisions, soit un total de :

$$\frac{n(n-1)(n+1)}{3}$$

opérations élémentaires. ■

Proposition 3.4.5. Soit $A \in M_n(\mathbb{R})$ admettant une décomposition LU. Soient $\hat{L}$ et $\hat{U}$ les approximations numériques de L et U . Alors :

$$\hat{L}\hat{U} = A + \delta_A, \quad \text{avec } \forall (i, j) \in \llbracket 1, n \rrbracket^2, |(\delta_A)_{ij}| \leq \gamma_n \sum_{k=1}^n |\hat{L}_{ik}| |\hat{U}_{kj}|.$$

Démonstration. Soit $k \in \llbracket 1, n \rrbracket$. Soit $i \in \llbracket k+1, n \rrbracket$. On a :

$$L_{ik} = \frac{1}{U_{kk}} \left(A_{ik} - \sum_{j=1}^{k-1} L_{ij} U_{jk} \right).$$

En appliquant le [Lemme 3.2.7](#), on trouve alors $\theta_k^{i,k}$ et des $\theta_j^{i,k}$ tels que :

$$\hat{L}_{ik} \hat{U}_{kk} (1 + \theta_k^{i,k}) = A_{ik} - \sum_{j=1}^{k-1} \hat{L}_{ij} \hat{U}_{jk} (1 + \theta_j^{i,k}).$$

En réarrangeant les termes, on obtient :

$$\sum_{j=1}^k \hat{L}_{ij} \hat{U}_{jk} = A_{ik} - \underbrace{\sum_{j=1}^k \hat{L}_{ij} \hat{U}_{jk} \theta_j^{i,k}}_{(\delta_A)_{ik}}.$$

Comme $\hat{L}$ est triangulaire inférieure et $\hat{U}$ est triangulaire supérieure, le membre de gauche de cette égalité n'est autre que $(\hat{L}\hat{U})_{ik}$. Par ailleurs, chaque $\theta_j^{i,k}$ est contrôlé par $\gamma_j \leq \gamma_n$.

Soit maintenant $j \in \llbracket k, n \rrbracket$. On utilise le même lemme :

$$\hat{U}_{kj} (1 + \theta_{k-1}^{k,j}) = A_{kj} - \sum_{i=1}^{k-1} \hat{L}_{ki} \hat{U}_{ij} (1 + \theta_i^{k,j}).$$

Ainsi :

$$\sum_{i=1}^{k-1} \hat{L}_{ki} \hat{U}_{ij} + \hat{U}_{kj} = A_{kj} - \sum_{i=1}^{k-1} \hat{L}_{ki} \hat{U}_{ij} \theta_i^{k,j} - \hat{U}_{kj} \theta_{k-1}^{k,j}.$$

On utilise $\hat{L}_{kk} = 1$ et on peut alors conclure de la même façon que le cas précédent. ■

Dans les cas où la condition suffisante du [Théorème 3.4.1](#) n'est pas satisfaite, mais que la matrice reste inversible, tout espoir n'est pas perdu : il est toujours possible d'effectuer une permutation des lignes de A pour retrouver une factorisation LU. On s'appuie pour cela sur un lemme technique, qui lui-même s'appuie sur une remarque : en considérant une transvection T agissant sur une colonne m et une permutation P agissant sur deux indices $m < p < q$, on peut noter que le produit PTP est encore une transvection agissant sur la colonne m . On va ainsi pouvoir « déplacer » les permutations et transvections dans le

produit obtenu dans la méthode du pivot de Gauss. Prenons un exemple avec trois transvections et permutations : $T^{(3)}P^{(3)}T^{(2)}P^{(2)}T^{(1)}P^{(1)}$. L'idée est de laisser $T^{(3)}$ telle quelle et de « ramener » les transvections $T^{(2)}$ et $T^{(1)}$ à ses côtés, pour isoler les permutations. Comme l'inverse de $P^{(3)}$ est elle-même, on peut toujours écrire :

$$P^{(3)}T^{(2)} = P^{(3)}T^{(2)}P^{(3)}P^{(3)}.$$

Le produit $\tilde{T}^{(2)} := P^{(3)}T^{(2)}P^{(3)}$ est une nouvelle transvection agissant sur la colonne 2. En remplaçant dans le produit précédent on a donc obtenu :

$$T^{(3)}\tilde{T}^{(2)}P^{(3)}P^{(2)}T^{(1)}P^{(1)}.$$

On recommence alors pour $T^{(1)}$:

$$P^{(3)}P^{(2)}T^{(1)} = P^{(3)}P^{(2)}T^{(1)}P^{(2)}P^{(3)}P^{(3)}P^{(2)}.$$

Le produit $P^{(2)}T^{(1)}P^{(2)}$ définit une nouvelle transvection T' agissant sur la colonne 1, donc le produit $P^{(3)}T'P^{(3)}$ définit une nouvelle transvection $\tilde{T}^{(1)}$ agissant elle aussi sur la colonne 1. On obtient finalement :

$$T^{(3)}P^{(3)}T^{(2)}P^{(2)}T^{(1)}P^{(1)} = T^{(3)}\tilde{T}^{(2)}\tilde{T}^{(1)}P^{(3)}P^{(2)}P^{(1)}.$$

Cette idée est généralisée dans le lemme suivant.

Lemme 3.4.6. Soit $M \in \mathbb{N}^*$, $M \geq 2$. Soient $(P^{(m)})_{m=2}^M$ des permutations telles que, pour tout m , $P^{(m)}$ agit sur des indices p et q tels que $m \leq p < q$. Soient $(T^{(m)})_{m=1}^{M-1}$ des transvections telles que, pour tout m , $T^{(m)}$ agit sur la colonne m . Alors il existe $(\tilde{T}^{(m)})_{m=1}^{M-1}$ des transvections telles que :

$$\prod_{m=1}^{M-1} P^{(M-m+1)}T^{(M-m)} = \left(\prod_{m=1}^{M-1} \tilde{T}^{(M-m)} \right) \left(\prod_{m=1}^{M-1} P^{(M-m+1)} \right).$$

Démonstration. On procède comme dans l'exemple : toute l'astuce consiste à écrire l'identité comme le produit des toutes les permutations suivant la transvection que l'on cherche à transformer. Plus précisément :

$$\begin{aligned} & \prod_{m=1}^{M-1} P^{(M-m+1)}T^{(M-m)} \\ &= \prod_{m=1}^{M-1} P^{(M-m+1)}T^{(M-m)} \left(\prod_{p=M-m+1}^M P^{(p)} \right) \left(\prod_{p=0}^{m-1} P^{(M-p)} \right) \\ &= \left(\prod_{m=1}^{M-1} \left(\prod_{p=0}^{m-2} P^{(M-p)} \right) P^{(M-m+1)}T^{(M-m)} \left(\prod_{p=M-m+1}^M P^{(p)} \right) \right) \left(\prod_{p=0}^{M-2} P^{(M-p)} \right), \end{aligned}$$

où on a simplement déplacé le dernier produit vers la « prochaine itération ». On remarque maintenant :

$$\left(\prod_{p=0}^{m-2} P^{(M-p)} \right) P^{(M-m+1)} = \prod_{p=0}^{m-1} P^{(M-p)}.$$

Les produits de permutations entourant $T^{(M-m)}$ sont les mêmes, dans des ordres opposés. On en déduit donc l'existence d'une transvection $\tilde{T}^{(M-m)}$ telle que :

$$\tilde{T}^{(M-m)} = \left(\prod_{p=0}^{m-1} P^{(M-p)} \right) T^{(M-m)} \left(\prod_{p=M-m+1}^M P^{(p)} \right).$$

Le résultat s'ensuit alors. ■

Théorème 3.4.7. Soit $A \in M_n(\mathbb{R})$ inversible. Alors il existe une matrice de permutation P , une matrice triangulaire inférieure L à diagonale unité, et une matrice triangulaire supérieure U telles que :

$$PA = LU.$$

Démonstration. Comme A est inversible, il existe $M \in M_n(\mathbb{R})$ inversible telle que $U := MA$ est triangulaire supérieure. De plus, on trouve $n-1$ matrices de permutation $P^{(m)}$ et de transvection $T^{(m)}$ telles que :

$$M = \prod_{m=1}^{n-1} T^{(n-m)} P^{(n-m)} = T^{(n-1)} \left(\prod_{m=1}^{n-2} P^{(n-m)} T^{(n-m-1)} \right) P^{(1)}.$$

On applique alors le [Lemme 3.4.6](#) : il existe $n-2$ matrices de transvection $\tilde{T}^{(n-m)}$ telles que :

$$\prod_{m=1}^{n-2} P^{(n-m)} T^{(n-m-1)} = \left(\prod_{m=2}^{n-1} \tilde{T}^{(n-m)} \right) \left(\prod_{m=1}^{n-2} P^{(n-m)} \right).$$

On a obtenu :

$$M = \left(\prod_{m=1}^{n-1} \tilde{T}^{(n-m)} \right) \left(\prod_{m=1}^{n-1} P^{(n-m)} \right),$$

avec $\tilde{T}^{(n-1)} := T^{(n-1)}$. On note $T := \prod_{m=1}^{n-1} \tilde{T}^{(n-m)}$. C'est une matrice de transvection, dont

l'inverse est également une matrice de transvection, notée L . On note $P := \prod_{m=1}^{n-1} P^{(n-m)}$.

On a donc obtenu :

$$PA = LU.$$
■

Remarque 3.4.8. Comme on l'a vu, il existe plusieurs stratégies de changement de pivot : il n'y a donc pas unicité de la décomposition sous la forme $PA = LU$! Cependant, à permutation fixée P , la décomposition LU de PA , si elle existe, est bien unique.

3.5 Factorisation de Cholesky

La factorisation LU s'applique à des matrices inversibles quelconques. Si la matrice à inverser possède des propriétés particulières, on peut chercher à les exploiter pour obtenir d'autres décompositions.

Théorème 3.5.1. Soit $A \in M_n(\mathbb{R})$ symétrique définie positive. Alors il existe une matrice triangulaire inférieure B telle que :

$$A = BB^T.$$

Si on suppose de plus que les coefficients diagonaux de B sont strictement positifs, alors cette décomposition est unique.

Démonstration. Existence. On va montrer que la matrice A admet une décomposition LU. Soit pour cela $k \in \llbracket 1, n \rrbracket$. On pose $\Delta_k := (A_{ij})_{(i,j) \in \llbracket 1, k \rrbracket^2}$. Soit $w \in \mathbb{R}^k$. On définit $v \in \mathbb{R}^n$ par :

$$v_i := \begin{cases} w_i, & \text{si } i \in \llbracket 1, k \rrbracket, \\ 0, & \text{sinon.} \end{cases}$$

Ainsi $w^T \Delta_k w = v^T A v$. Comme A est symétrique définie positive, cette égalité permet de montrer que les Δ_k sont également symétriques définies positives, et donc inversibles. Ainsi A admet une décomposition LU : $A = LU$. Toujours pour $k \in \llbracket 1, n \rrbracket$, on peut écrire cette égalité sous forme de blocs :

$$\begin{pmatrix} \Delta_k & \star \\ \star & \star \end{pmatrix} = \begin{pmatrix} L_k & 0 \\ \star & \star \end{pmatrix} \begin{pmatrix} U_k & \star \\ 0 & \star \end{pmatrix},$$

où L_k et U_k sont définies similairement à Δ_k . En particulier, on obtient $\Delta_k = L_k U_k$, d'où $\det \Delta_k = \det L_k \det U_k$. Or, la diagonale de L_k est unitaire, et U_k est triangulaire, donc :

$$0 < \det \Delta_k = \prod_{i=1}^k U_{ii}.$$

Cette inégalité étant valable pour tout $k \in \llbracket 1, n \rrbracket$, on en déduit que les coefficients diagonaux de U sont tous strictement positifs. On peut donc définir $\Lambda := \text{diag}(\sqrt{U_{ii}})$, inversible. Ainsi :

$$A = LU = L(\Lambda \Lambda^{-1})U = BC$$

avec $B := L\Lambda$ et $C := \Lambda^{-1}U$. La matrice B est triangulaire inférieure, C est triangulaire supérieure, et leurs diagonales sont toutes les deux égales à $\sqrt{U_{ii}}$. Comme A est symétrique, on obtient :

$$A = A^T \iff BC = (BC)^T = C^T B^T,$$

soit :

$$C(B^T)^{-1} = B^{-1}C^T.$$

La matrice $C(B^T)^{-1}$ est triangulaire supérieure à diagonale unité, et $B^{-1}C^T$ est triangulaire inférieure à diagonale unité. Donc :

$$C(B^T)^{-1} = B^{-1}C^T = I$$

et on en déduit $C = B^T$.

Unicité. Soient B_1 et B_2 triangulaires inférieures à diagonales strictement positives et telles que :

$$A = B_1 B_1^T = B_2 B_2^T.$$

Soit Λ_1 la matrice diagonale de B_1 . Alors :

$$A = (B_1 \Lambda_1^{-1})(\Lambda_1 B_1^T)$$

et ainsi A est factorisée en un produit d'une matrice triangulaire inférieure à diagonale unité et d'une matrice triangulaire supérieure. On reconnaît la factorisation LU de A , qui apparaît aussi, avec des notations similaires, dans :

$$A = (B_2 \Lambda_2^{-1})(\Lambda_2 B_2^T).$$

De l'unicité de la décomposition LU, on déduit :

$$B_1 \Lambda_1^{-1} = B_2 \Lambda_2^{-1} \quad \text{et} \quad \Lambda_1 B_1^T = \Lambda_2 B_2^T.$$

De $\Lambda_1 B_1^T = \Lambda_2 B_2^T$, on déduit l'égalité des carrés des éléments diagonaux de B_1 et B_2 . Comme ils sont supposés strictement positifs, ils sont alors égaux. Autrement dit, $\Lambda_1 = \Lambda_2$. Comme $B_1 \Lambda_1^{-1} = B_2 \Lambda_2^{-1}$, on a donc obtenu $B_1 = B_2$. ■

Pour calculer concrètement la matrice B d'une décomposition de Cholesky, on peut commencer par écrire le produit $A = BB^T$:

$$\forall (i, j) \in \llbracket 1, n \rrbracket^2, \quad A_{ij} = \sum_{k=1}^n B_{ik}(B^T)_{kj} = \sum_{k=1}^n B_{ik}B_{jk} = \sum_{k=1}^{\min(i,j)} B_{ik}B_{jk}$$

car B est triangulaire inférieure. Écrivons cette égalité pour $i = j = 1$:

$$A_{11} = B_{11}^2,$$

ce qui donne :

$$B_{11} = \sqrt{A_{11}}.$$

Comme B est triangulaire inférieure, tous les autres coefficients de la première ligne sont nuls. Continuons avec la deuxième ligne ($i = 2$) :

$$A_{21} = B_{21}B_{11} \quad \text{et} \quad A_{22} = B_{21}^2 + B_{22}^2.$$

Comme on connaît B_{11} , on peut obtenir :

$$B_{21} = \frac{1}{B_{11}}A_{21},$$

ce qui permet de calculer :

$$B_{22} = \sqrt{A_{22} - B_{21}^2}.$$

On a ainsi déterminé tous les coefficients de la deuxième ligne de B . L'idée est donc la suivante : on construit B ligne par ligne, en utilisant à chaque fois les lignes précédemment calculées. Considérons $i \in \llbracket 2, n \rrbracket$ quelconque. Le premier terme de la ligne se calcule en considérant $j = 1$:

$$A_{i1} = B_{i1}B_{11}, \quad \text{d'où} \quad B_{i1} = \frac{1}{B_{11}}A_{i1}.$$

Considérons maintenant $j \in \llbracket 2, i-1 \rrbracket$:

$$A_{ij} = \sum_{k=1}^j B_{ik}B_{jk} = \sum_{k=1}^{j-1} B_{ik}B_{jk} + B_{ij}B_{jj}.$$

Si on calcule tous les coefficients de gauche à droite, on peut en déduire :

$$B_{ij} = \frac{1}{B_{jj}} \left(A_{ij} - \sum_{k=1}^{j-1} B_{ik}B_{jk} \right).$$

Reste le coefficient diagonal :

$$A_{ii} = \sum_{k=1}^i B_{ik}^2 = \sum_{k=1}^{i-1} B_{ik}^2 + B_{ii}^2,$$

d'où :

$$B_{ii} = \sqrt{A_{ii} - \sum_{k=1}^{i-1} B_{ik}^2}.$$

On résume cette méthode de calcul dans l'[Algorithme 3.5.2](#).

Proposition 3.5.3. La décomposition de Cholesky est en $\mathcal{O}\left(\frac{n^3}{3}\right)$ flops.

Démonstration. Pour chaque $i \in \llbracket 1, n \rrbracket$, on compte, pour les coefficients non diagonaux, et donc chaque $j \in \llbracket 1, i-1 \rrbracket$:

— 1 division,

Algorithme 3.5.2 : Décomposition de Cholesky

```

pour  $i$  de 1 à  $n$ 
  pour  $j$  de 1 à  $i - 1$ 
     $B_{ij} := \frac{1}{B_{jj}} \left( A_{ij} - \sum_{k=1}^{j-1} B_{ik} B_{jk} \right)$ 
   $B_{ii} := \sqrt{A_{ii} - \sum_{k=1}^{i-1} B_{ik}^2}$ 

```

— $j - 1$ multiplications,

— $j - 1$ soustractions,

et pour le coefficient diagonal :

— $i - 1$ multiplications,

— $i - 1$ soustractions,

— 1 racine carrée.

Le nombre total de multiplications est donc donné par :

$$\sum_{i=1}^n \left(\sum_{j=1}^{i-1} (j - 1) + (i - 1) \right) = \frac{n^3 - n}{6}.$$

On compte autant de soustractions. Le nombre de divisions est donné par :

$$\sum_{i=1}^n \sum_{j=1}^{i-1} 1 = \frac{n(n - 1)}{2},$$

et le nombre de racines est donné par :

$$\sum_{i=1}^n 1 = n.$$

On a donc un total de :

$$2 \frac{n^3 - n}{6} + \frac{n(n - 1)}{2} + n$$

opérations, d'où le résultat. ■

3.6 Systèmes de Hessenberg

Définition 3.6.1. Soit $H \in M_{m \times n}(\mathbb{R})$, $m \geq n$. On dit que H est une matrice de Hessenberg si :

$$\forall j \in \llbracket 1, n \rrbracket, \quad \forall i \in \llbracket j+1, m \rrbracket, \quad H_{ij} = 0.$$

Dans le cas des matrices carrées, une matrice de Hessenberg peut être vue comme une matrice triangulaire supérieure pour laquelle la première sous-diagonale n'est pas nécessairement nulle. Des systèmes linéaires définis par des matrices de Hessenberg sont donc assez « proches » de systèmes triangulaires. L'objectif de cette section est de définir une méthode permettant de triangulariser de tels systèmes. On rappelle pour cela une notation pour désigner le cercle unité qui sera utile dans la suite :

$$S^1 := \{(x, y) \in \mathbb{R}^2 ; x^2 + y^2 = 1\}.$$

Lemme 3.6.2. Soit $H \in M_2(\mathbb{R})$. Alors il existe $(c, s) \in S^1$ tel que, en notant

$$Q := \begin{pmatrix} c & s \\ -s & c \end{pmatrix},$$

la matrice QH est triangulaire supérieure.

Démonstration. On note

$$H = \begin{pmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{pmatrix}.$$

Si $h_{21} = 0$, alors H est déjà triangulaire supérieure : le couple $(c, s) := (1, 0)$ convient. Sinon, $h_{11}^2 + h_{21}^2 > 0$ et on peut alors définir :

$$c := \frac{h_{11}}{\sqrt{h_{11}^2 + h_{21}^2}} \quad \text{et} \quad s := \frac{h_{21}}{\sqrt{h_{11}^2 + h_{21}^2}}.$$

On peut alors vérifier que le couple (c, s) ainsi défini convient. ■

Remarque 3.6.3. Si ce résultat peut paraître quelque peu sorti de nulle part, il est en réalité assez simple de l'interpréter géométriquement. Les colonnes de la matrice H définissent deux vecteurs de $\mathbb{R}^2$. Comme $(c, s) \in S^1$, la matrice Q n'est autre qu'une matrice de rotation d'angle $-\theta$, avec $c = \cos \theta$ et $s = \sin \theta$. L'idée est donc simplement d'appliquer une rotation à ces vecteurs de telle sorte que le premier se retrouve colinéaire à l'axe des abscisses.

Théorème 3.6.4. Soit $H \in M_n(\mathbb{R})$ une matrice de Hessenberg. Alors il existe

$((c_k, s_k))_{k \in \llbracket 1, n-1 \rrbracket} \in (S^1)^{n-1}$ tel que, en notant

$$\forall k \in \llbracket 1, n-1 \rrbracket, \quad Q_k := \begin{pmatrix} I_{k-1} & & \\ & c_k & s_k \\ & -s_k & c_k \\ & & & I_{n-k-1} \end{pmatrix},$$

et en définissant la suite

$$\forall k \in \llbracket 0, n-2 \rrbracket, \quad H^{(k+1)} := Q_{k+1} H^{(k)}$$

à partir de $H^{(0)} := H$, la matrice $H^{(n-1)}$ est triangulaire supérieure.

Démonstration. L'idée est de montrer par récurrence sur $k \in \llbracket 0, n-1 \rrbracket$ que $H^{(k)}$ est une matrice de Hessenberg dont les k premiers éléments sous-diagonaux sont nuls. C'est trivialement vrai pour $k = 0$. À l'étape k , on peut donc écrire

$$H^{(k)} = \begin{pmatrix} H_{11} & H_{12} & H_{13} \\ 0 & H_{22} & H_{23} \\ 0 & H_{32} & H_{33} \end{pmatrix}$$

avec $H_{11} \in M_k(\mathbb{R})$ triangulaire supérieure, $H_{22} \in M_2(\mathbb{R})$ et $H_{33} \in M_{n-k-2}(\mathbb{R})$. Comme $H^{(k)}$ est une matrice de Hessenberg, on peut vérifier que la première colonne de $H_{32} \in M_{(n-k-2) \times 2}(\mathbb{R})$ est nulle.

En appliquant le [Lemme 3.6.2](#), on trouve $(c_{k+1}, s_{k+1}) \in S^1$ tel que $\tilde{Q}_{k+1} H_{22}$ est triangulaire supérieure avec

$$\tilde{Q}_{k+1} = \begin{pmatrix} c_{k+1} & s_{k+1} \\ -s_{k+1} & c_{k+1} \end{pmatrix}.$$

Posons

$$Q_{k+1} := \begin{pmatrix} I_k & 0 & 0 \\ 0 & \tilde{Q}_{k+1} & 0 \\ 0 & 0 & I_{n-k-2} \end{pmatrix}.$$

Ainsi :

$$H^{(k+1)} = Q_{k+1} H^{(k)} = \begin{pmatrix} H_{11} & H_{12} & H_{13} \\ 0 & \tilde{Q}_{k+1} H_{22} & \tilde{Q}_{k+1} H_{23} \\ 0 & H_{32} & H_{33} \end{pmatrix}.$$

On a donc annulé le $(k+1)$ -ème élément sous-diagonal. ■

En notant Q le produit de toutes les matrices Q_k utilisées, on obtient un nouveau système linéaire :

$$Ax = b \iff (QA)x = Qb,$$

triangulaire supérieur, qui peut donc facilement être inversé par une méthode de remontée. Comme pour le pivot de Gauss, en pratique, on ne calcule pas les matrices de rotation pour ensuite effectuer la multiplication : on effectue directement les changements. On peut d'ailleurs remarquer qu'à chaque étape, seules deux lignes sont affectées : les lignes k et

$k + 1$. Les lignes précédentes restent inchangées par rapport à l'étape précédente, tandis que les lignes suivantes restent celles de H . On obtient l'[Algorithme 3.6.5](#).

Algorithme 3.6.5 : Résolution d'un système de Hessenberg

Entrée : $H \in M_n(\mathbb{R})$ matrice de Hessenberg inversible, $b \in \mathbb{R}^n$

Sortie : $x \in \mathbb{R}^n$ solution de $Hx = b$

pour k de 1 à $n - 1$

$$\left[\begin{array}{l} n_k = \sqrt{\left(H_{kk}^{(k)}\right)^2 + \left(H_{k+1,k}^{(k)}\right)^2} \\ c_k := \frac{H_{kk}^{(k)}}{n_k} \\ s_k := \frac{H_{k+1,k}^{(k)}}{n_k} \\ \textbf{pour } j \text{ de } k \text{ à } n \\ \quad \left[\begin{array}{l} H_{kj}^{(k+1)} := c_k H_{kj}^{(k)} + s_k H_{k+1,j}^{(k)} \\ H_{k+1,j}^{(k+1)} := c_k H_{k+1,j}^{(k)} - s_k H_{kj}^{(k)} \end{array} \right. \\ b_k^{(k+1)} := c_k b_k^{(k)} + s_k b_{k+1}^{(k)} \\ b_{k+1}^{(k+1)} := c_k b_{k+1}^{(k)} - s_k b_k^{(k)} \end{array} \right.$$

Résoudre le système $H^{(n-1)}x = b^{(n)}$ par remontée

Proposition 3.6.6. La résolution d'un système de Hessenberg par l'[Algorithme 3.6.5](#) nécessite $\mathcal{O}(4n^2)$ flops.

Démonstration. Pour tout $k \in \llbracket 1, n-1 \rrbracket$, le calcul de n_k nécessite 2 multiplications, 1 addition et 1 racine carrée. Les coefficients c_k et s_k s'en déduisent avec 2 divisions. La transformation des deux lignes de la matrice nécessite, pour chaque $j \in \llbracket k, n \rrbracket$, 4 multiplications et 2 additions. La transformation du vecteur nécessite 4 multiplications et 2 additions. On obtient au total :

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

divisions,

$$\sum_{k=1}^{n-1} 1 = n-1$$

calculs de racine carrée,

$$\sum_{k=1}^{n-1} \left(2 + \sum_{j=k}^n 4 + 4 \right) = 2(n-1)(n+5)$$

multiplications, et

$$\sum_{k=1}^{n-1} \left(1 + \sum_{j=k}^n 2 + 2 \right) = (n-1)(n+5)$$

additions. Au total, on a donc :

$$2(n-1) + (n-1) + 2(n-1)(n+5) + (n-1)(n+5) = 3(n-1)(n+6) = \mathcal{O}(3n^2)$$

opérations élémentaires pour la transformation du système. Il faut ajouter à cela $\mathcal{O}(n^2)$ flops pour la résolution du système triangulaire. ■

Il est intéressant de noter que la transformation d'un système de Hessenberg s'effectue colonne par colonne : la transformation des coefficients d'une colonne donnée n'affecte en rien les autres colonnes. Cela indique qu'il peut être « facile » d'agrandir le système : on peut passer d'un système de taille n à un système de taille $n+1$ sans pour autant avoir à refaire tous les calculs de transformation, puisque seule la dernière colonne devrait avoir à être transformée, a priori.

Considérons une matrice de Hessenberg $H \in M_n(\mathbb{R})$. On définit $\tilde{H} \in M_{n+1}(\mathbb{R})$ comme étant une « extension » de la matrice H , et $\tilde{b} \in \mathbb{R}^{n+1}$ comme étant une extension de b :

$$\tilde{H} := \begin{pmatrix} H & h \\ \tilde{h} & h_{n+1,n+1} \end{pmatrix}, \quad \tilde{b} := \begin{pmatrix} b \\ b_{n+1} \end{pmatrix}$$

où $h := (h_{1,n+1}, h_{2,n+1}, \dots, h_{n,n+1})^T$ et $\tilde{h} = (0, \dots, 0, h_{n+1,n})$. La transformation du système de taille n peut s'écrire sous la forme $QHx = Qb$. On définit

$$\tilde{Q} := \begin{pmatrix} Q & 0 \\ 0 & 1 \end{pmatrix}.$$

Ainsi :

$$\tilde{Q}\tilde{H} = \begin{pmatrix} QH & Qh \\ \tilde{h} & h_{n+1,n+1} \end{pmatrix}, \quad \tilde{Q}\tilde{b} = \begin{pmatrix} Qb \\ b_{n+1} \end{pmatrix}.$$

Les transformations QH et Qb ayant déjà été calculées, il reste à calculer Qh avant de triangulariser le système, ce qui ne nécessitera qu'une seule ligne de transformation supplémentaire (celle ayant été ajoutée). On obtient l'[Algorithme 3.6.7](#).

On peut calculer que l'[Algorithme 3.6.7](#) nécessite $6(n-1) + 24$ opérations élémentaires, auxquelles il faut bien sûr ajouter $\mathcal{O}((n+1)^2)$ pour la résolution du système triangulaire.

Algorithme 3.6.7 : Augmentation de la taille d'un système de Hessenberg

pour k de 1 à $n - 1$

$$\left[\begin{array}{l} H_{k,n+1}^{(k)} := c_k H_{k,n+1}^{(k-1)} + s_k H_{k+1,n+1}^{(k-1)} \\ H_{k+1,n+1}^{(k)} := c_k H_{k+1,n+1}^{(k-1)} - s_k H_{k,n+1}^{(k-1)} \end{array} \right.$$

$$n_n := \sqrt{(H_{nn}^{(n)})^2 + (H_{n+1,n}^{(n)})^2}$$

$$c_n := \frac{H_{nn}^{(n)}}{n_n}$$

$$s_n := \frac{H_{n+1,n}^{(n)}}{n_n}$$

pour j de n à $n + 1$

$$\left[\begin{array}{l} H_{nj}^{(n)} := c_n H_{nj}^{(n-1)} + s_n H_{n+1,j}^{(n-1)} \\ H_{n+1,j}^{(n)} := c_n H_{n+1,j}^{(n-1)} - s_n H_{nj}^{(n-1)} \end{array} \right.$$

$$b_n^{(n)} := c_n b_n^{(n-1)} + s_n b_{n+1}^{(n-1)}$$

$$b_{n+1}^{(n)} := c_n b_{n+1}^{(n-1)} - s_n b_n^{(n-1)}$$

Chapitre 4

Matrices creuses

Les matrices issues de la discrétisation d'une EDP contiennent souvent un grand nombre de coefficients nuls. Il peut être utile (voire indispensable dans de nombreux cas) de prendre en compte cet aspect.

Dans tout ce chapitre, M désigne une matrice de taille $m \times n$. Ses mn coefficients seront notés M_{ij} , pour $(i, j) \in \llbracket 0, m-1 \rrbracket \times \llbracket 0, n-1 \rrbracket$.

4.1 L'approche classique

Commençons par nous intéresser au stockage de matrices en général. On peut par exemple commencer par stocker tous les coefficients de la matrice M dans un tableau V de taille mn . En faisant cela, on « applatit » la matrice en un vecteur : il nous faut un moyen de retrouver la structure initiale, ce qui revient à se demander comment accéder à un certain coefficient M_{ij} . Sans hypothèse sur l'ordre dans lequel ont été stockés les valeurs dans V , deux autres tableaux seront nécessaires, tous deux de taille mn , pour stocker les coordonnées des coefficients (un tableau pour la ligne, un autre pour la colonne). Cette approche est bien évidemment loin d'être optimale.

Une meilleure approche, toujours en utilisant un tableau de taille mn , consiste à définir un ordre dans lequel sont stockés les coefficients : on remplit V avec les éléments « lus » de gauche à droite, de haut en bas. On obtient la représentation :

$$V = (M_{0,0}, M_{0,1}, \dots, M_{0,n-1}, M_{1,0}, \dots, M_{1,n-1}, \dots, M_{m-1,0}, \dots, M_{m-1,n-1}).$$

L'avantage de cette approche est qu'elle ne nécessite pas de stocker les coordonnées des coefficients qui peuvent être déterminées implicitement via la connaissance du nombre de lignes m et du nombre de colonnes n :

$$M_{ij} = V_{im+j}.$$

Exemple 4.1.1. La matrice

$$M := \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \end{pmatrix}$$

sera stockée dans un « vecteur »

$$V = (1, 2, 3, 4, 5, 6, 7, 8).$$

S'il est possible d'utiliser des tableaux bidimensionnels, une autre approche consiste à représenter la matrice de la même façon que celle utilisée quand on l'écrit sur le papier. Un tableau à m éléments est défini. Chaque élément est lui-même un tableau à n éléments, listant les coefficients de la m -ème ligne de M .

Proposition 4.1.2. Soit $M \in M_{m \times n}(\mathbb{R})$ et $x \in \mathbb{R}^n$. Le produit Mx nécessite $m(2n - 1)$ opérations élémentaires.

Démonstration. Le résultat y de l'opération Mx est un vecteur de $\mathbb{R}^m$ dont chaque coefficient est calculé de la même façon. Soit $i \in \llbracket 0, m - 1 \rrbracket$. Alors :

$$y_i = \sum_{j=0}^{n-1} M_{ij}x_j.$$

Pour chaque coefficient, on compte ainsi n multiplications et $n - 1$ additions. On obtient donc un total de :

$$m(n + n - 1) = m(2n - 1)$$

opérations élémentaires. ■

4.2 Ignorer les zéros

Dans les cas où aucune hypothèse n'est faite sur la matrice, il n'est pas possible de faire mieux que l'approche décrite précédemment en terme d'espace occupé en mémoire : on a besoin de stocker les mn éléments de M pour la représenter complètement. À partir de maintenant, on supposera donc que M est une matrice dite creuse, c'est-à-dire contenant « beaucoup » de zéros. On note N_{nz} le nombre de coefficients non nuls de M . On suppose ainsi :

$$N_{\text{nz}} \ll mn.$$

Il n'existe pas de définition stricte de matrice creuse : la proportion « limite » à partir de laquelle on passe de matrice dense à matrice creuse est floue. L'idée est simplement d'utiliser les techniques d'optimisation du stockage et du calcul dès que cela devient intéressant. On peut en effet citer deux inconvénients majeurs au fait de traiter une matrice creuse comme étant dense.

Coût mémoire. Le premier est probablement le plus évident : la place occupée en mémoire. Imaginons l'exemple (plutôt extrême) d'une matrice diagonale de taille $n \times n$, qui contient donc n^2 coefficients mais seulement n coefficients non nuls. On compte ainsi $n^2 - n = n(n - 1)$ zéros : on va ainsi réclamer n^2 places mémoire, là où n suffissent pour stocker toute l'information. Si n est grand (ce qui arrive souvent en pratique), selon la mémoire disponible, il se peut qu'il soit tout simplement impossible de stocker n^2 coefficients : on ne pourra donc pas réaliser le calcul, alors que l'on aurait pu si on avait seulement stocké nos n coefficients non nuls.

Coût de calcul. Le second inconvénient est le temps de calcul : une opération couramment utilisée est le produit matrice-vecteur. Chaque coefficient du résultat est le produit scalaire d'une certaine ligne de la matrice avec le vecteur. Or, si ces lignes contiennent un grand nombre de zéros, on va passer la majeure partie du temps à effectuer des calculs qui n'influenceront pas le résultat final. L'exemple de la matrice diagonale est là encore assez parlant : chaque produit scalaire n'est finalement qu'un seul produit. Si on ne prend pas en compte cette structure, on va effectuer n multiplications et $n - 1$ additions, soit $2n - 1$ opérations, là où une seule suffisait !

Différentes approches existent pour tirer parti de la présence de nombreux zéros, avec pour point commun l'objectif de stocker le moins de zéros possible. L'approche la plus générique, appelée COO (pour « COOrdinate format ») consiste à utiliser trois tableaux de taille N_{nz} . Le premier, **V**, liste les coefficients de la matrice. Les deux autres, **R** et **C**, servent à stocker les coordonnées de ces coefficients, dans le même ordre. Ce format nécessite donc $3N_{\text{nz}}$ places mémoire.

Exemple 4.2.1. La matrice

$$M := \begin{pmatrix} 1 & 0 & 2 & 0 & 3 \\ 0 & 4 & 0 & 0 & 5 \\ 6 & 0 & 0 & 7 & 0 \\ 0 & 0 & 8 & 0 & 9 \end{pmatrix}$$

pourra être représentée avec les trois tableaux suivants.

$$\mathbf{V} = [1, 4, 6, 2, 7, 8, 3, 5, 9]$$

$$\mathbf{R} = [0, 1, 2, 0, 2, 3, 0, 1, 3]$$

$$\mathbf{C} = [0, 1, 0, 2, 3, 2, 4, 4, 4]$$

Dans l'exemple précédent, les coefficients de M sont stockés dans un ordre quelconque, ce qui est possible puisque les coordonnées de chaque coefficient sont également stockées. Dans les implémentations pratiques, il peut même être possible d'avoir des doublons dans les coordonnées : les entrées ayant les mêmes coordonnées sont alors sommées avant d'être utilisées. Le format COO est assez largement répandu, notamment pour sa facilité d'utilisation : il est en effet très simple de construire une matrice creuse avec ce format. L'accès à des coefficients spécifiques, et donc le calcul effectif avec la matrice représentée, n'est cependant pas immédiat : il faut rechercher dans les tableaux **R** et **C** le couple voulu pour en déduire la position de l'élément dans **V**.

En pratique, le format COO est donc assez souvent une « porte d'entrée » : on l'utilise pour construire facilement la matrice, qui est ensuite convertie en un autre format facilitant les calculs.

4.3 Formats compressés

Comme aucun ordre n'est imposé dans le format COO, il n'y a pas unicité de la représentation d'une matrice donnée. L'[Exemple 4.2.1](#) pourrait par exemple devenir

$$\begin{aligned} V &= [1, 2, 3, 4, 5, 6, 7, 8, 9] \\ R &= [0, 0, 0, 1, 1, 2, 2, 3, 3] \\ C &= [0, 2, 4, 1, 4, 0, 3, 2, 4] \end{aligned}$$

si on stocke les éléments de gauche à droite et de haut en bas. En regroupant les coefficients ligne par ligne, on fait apparaître une redondance dans R . Naturellement, on cherche donc à compresser les informations contenues dans R : c'est l'idée du format CSR (pour « Compressed Sparse Row »). Dans ce format, les définitions de V et C ne changent pas. En revanche, on impose que les coefficients soient stockés ligne par ligne, colonne par colonne. Le tableau R devient un tableau à $m + 1$ éléments. Pour $i \in \llbracket 0, m - 1 \rrbracket$, le i -ème élément de R est l'indice dans V (et donc dans C) à partir duquel commence la i -ème ligne de M . Le dernier élément de R est le nombre de coefficients non nuls contenus dans M . De manière équivalente, le i -ème élément de R indique le nombre de coefficients non nuls avant la ligne i .

Exemple 4.3.1. La matrice de l'[Exemple 4.2.1](#) pourra être représentée avec les trois tableaux suivants.

$$\begin{aligned} V &= [1, 2, 3, 4, 5, 6, 7, 8, 9] \\ C &= [0, 2, 4, 1, 4, 0, 3, 2, 4] \\ R &= [0, 3, 5, 7, 9] \end{aligned}$$

Avec ce format, on stocke donc moins d'éléments dans R . Plus précisément, on a toujours besoin de $2N_{\text{nz}}$ éléments pour V et C , tandis que R contient $m + 1$ éléments, ce qui donne un total de $2N_{\text{nz}} + m + 1$ places mémoire.

Remarque 4.3.2. Le premier élément de R sera toujours 0, et le dernier sera toujours N_{nz} . Il peut sembler assez inutile de les stocker. Leur présence permet cependant de faciliter les calculs. En stockant 0, on évite de devoir gérer l'exception que deviendrait la première ligne. De même, en stockant N_{nz} , on sait à quel indice s'arrête la dernière ligne sans avoir à gérer l'exception que celle-ci représenterait. La présence de ces deux constantes permet aussi que la formule

$$\text{nombre de coefficients non nuls sur la ligne } i = R[i + 1] - R[i]$$

soit vraie quel que soit $i \in \llbracket 0, m - 1 \rrbracket$. Ne pas les stocker nécessiterait des conditions

supplémentaires dans les calculs effectués, pour n'économiser que deux cases mémoire.

On peut assez facilement écrire une formule permettant d'effectuer un produit matrice-vecteur dans le cas où la matrice est stockée sous le format CSR :

$$\forall i \in \llbracket 0, n-1 \rrbracket, \quad (Ax)_i = \sum_{j=R_i}^{R_{i+1}-1} V_j x_{C_j}.$$

On effectue ainsi un total de N_{nz} multiplications et $N_{\text{nz}} - m$ additions.

Remarque 4.3.3. Comme on s'intéresse aux lignes de la matrice, une question naturelle peut venir : comment traiter le cas où une ligne est entièrement nulle ? Ce cas particulier ne nécessite en réalité aucun traitement spécifique. Une définition équivalente de $\mathbf{R}$ consiste à dire que le i -ème élément est le nombre de coefficients non nuls avant la i -ème ligne. Si une ligne est nulle, ce nombre ne bougera pas et on observera simplement une répétition dans $\mathbf{R}$. Par exemple, la matrice

$$M := \begin{pmatrix} 1 & 0 & 2 & 0 & 3 \\ 0 & 0 & 0 & 0 & 0 \\ 6 & 0 & 0 & 7 & 0 \\ 0 & 0 & 8 & 0 & 9 \end{pmatrix}$$

sera représentée par

$$\begin{aligned} \mathbf{V} &= [1, 2, 3, 6, 7, 8, 9] \\ \mathbf{C} &= [0, 2, 4, 0, 3, 2, 4] \\ \mathbf{R} &= [0, 3, 3, 5, 7] \end{aligned}$$

Remarque 4.3.4. Il existe également le format CSC (pour « Compressed Sparse Column »), similaire au format CSR, à ceci près que les coefficients sont regroupés par colonne et non plus par ligne.

4.4 Diagonales

Dans le cas de matrices carrées, une alternative au format CSR propose de séparer la diagonale principale de M . Il s'agit du format MSR (pour « Modified Sparse Row »). Supposons que cette diagonale ne contienne aucun zéro. Le tableau des valeurs contient alors $N_{\text{nz}} + 1$ éléments : les n coefficients diagonaux, un élément séparateur (généralement 0), puis tous les autres coefficients extra-diagonaux non nuls de M , ordonnés de la même façon que dans le format CSR. Il n'y a plus qu'un seul tableau de coordonnées qui revient à concaténer les tableaux $\mathbf{R}$ et $\mathbf{C}$ définis dans le format CSR.

Exemple 4.4.1. La matrice

$$M := \begin{pmatrix} 1 & 0 & 0 & 2 \\ 0 & 3 & 4 & 0 \\ 5 & 0 & 6 & 0 \\ 7 & 0 & 0 & 8 \end{pmatrix}$$

est représentée dans le format MSR par les deux tableaux suivants.

$$\begin{aligned} \mathbf{V} &= [1, 3, 6, 8, 0, 2, 4, 5, 7] \\ \mathbf{I} &= [5, 6, 7, 8, 9, 3, 2, 0, 0] \end{aligned}$$

Lorsque l'on sait qu'une matrice est nulle en dehors de quelques diagonales, il peut être intéressant de ne stocker que celles-ci. L'idée est alors de « réduire » la matrice carrée M en une matrice rectangulaire D ne contenant que les diagonales non nulles. Dans un tableau à part J , on stocke les positions de ces diagonales : la diagonale principale est à la position 0, les surdiagonales ont des positions strictement positives, et les sous-diagonales ont des positions strictement négatives.

En notant N_d le nombre de diagonales stockées, la matrice D sera de taille $n \times N_d$. Ainsi, seule la diagonale principale peut remplir complètement sa colonne. Pour les autres, une idée est de les décaler : on ajoute autant de zéros que nécessaire pour « aligner » une sous-diagonale en bas et une surdiagonale en haut.

Exemple 4.4.2. On considère la matrice tridiagonale

$$M := \begin{pmatrix} 1 & 0 & 2 & 0 & 0 \\ 3 & 4 & 0 & 5 & 0 \\ 0 & 6 & 7 & 0 & 8 \\ 0 & 0 & 9 & 10 & 0 \\ 0 & 0 & 0 & 11 & 12 \end{pmatrix},$$

qui peut être représentée avec :

$$D := \begin{pmatrix} 1 & 0 & 2 \\ 4 & 3 & 5 \\ 7 & 6 & 8 \\ 10 & 9 & 0 \\ 12 & 11 & 0 \end{pmatrix}, \quad J := (0, -1, 2).$$

On peut réaliser un produit matrice vecteur avec l'[Algorithme 4.4.3](#).

Cet algorithme sous-entend que les coefficients « inutiles » dans D (ceux servant à remplir les colonnes) sont des zéros, ce qui permet de ne pas altérer le résultat final. Si on tient à éviter complètement ces coefficients, il suffit de modifier les bornes de la boucle sur i : le premier indice est $-\min(0, \delta)$, le dernier est $n - 1 - \max(0, \delta)$.

Algorithme 4.4.3 : Produit matrice-vecteur avec un stockage diagonal

Entrée : (D, J) une représentation diagonale de M , $x \in \mathbb{R}^n$ **Sortie** : $y := Mx$ $y := 0$ **pour** j de 0 à $N_d - 1$ $\delta := J(j)$ **pour** i de 0 à $n - 1$ $y_i := y_i + D_{ij}x_{i+\delta}$

Chapitre 5

Méthodes itératives

On considère toujours une matrice $A \in M_n(\mathbb{R})$ inversible et un vecteur $b \in \mathbb{R}^n$ en vue de la résolution du système linéaire

$$Ax = b.$$

On va cette fois-ci s'intéresser à des méthodes dites « itératives » : plutôt que de chercher à résoudre directement le système pour obtenir sa solution, on va construire une suite qui va (on l'espère !) converger vers cette solution.

5.1 Méthodes classiques

5.1.1 Généralités

Soit $B \in M_n(\mathbb{R})$ telle que la matrice $(I - B)$ est inversible et telle que :

$$x = Bx + c$$

pour un certain vecteur $c \in \mathbb{R}^n$, x étant toujours la solution du système $Ax = b$.

Remarque 5.1.1. Une telle matrice B existe toujours. On peut par exemple considérer $B := I - A$, de sorte que $I - B = A$, donc inversible. Avec ce choix, on peut vérifier que $c = b$.

On définit une méthode itérative par la donnée d'un vecteur initial $x^{(0)} \in \mathbb{R}^n$ et par la relation de récurrence :

$$\forall k \in \mathbb{N}, \quad x^{(k+1)} = Bx^{(k)} + c.$$

On dira que la méthode itérative converge si :

$$\forall x^{(0)} \in \mathbb{R}^n, \quad \lim_{k \rightarrow +\infty} x^{(k)} = x.$$

Le résultat suivant établit une condition nécessaire et suffisante à la convergence d'une telle méthode itérative (voir par exemple [Cia82]).

Théorème 5.1.2. Soit une méthode itérative définie par une matrice B . Les propositions suivantes sont équivalentes :

1. la méthode itérative converge ;
2. $\rho(B) < 1$;
3. il existe une norme matricielle $\|\cdot\|$ telle que $\|B\| < 1$.

Démonstration. On note

$$\forall k \in \mathbb{N}, \quad e^{(k)} := x^{(k)} - x.$$

Alors on montre facilement que

$$\forall k \in \mathbb{N}, \quad e^{(k+1)} = Be^{(k)},$$

d'où :

$$\forall k \in \mathbb{N}, \quad e^{(k)} = B^k e_0.$$

1 $\implies$ 2. On procède par contraposée. Supposons que $\rho(B) \geq 1$. En particulier, il existe $p \in \mathbb{R}^n \setminus \{0\}$ tel que $Bp = \lambda p$ avec $|\lambda| \geq 1$ et :

$$\forall k \in \mathbb{N}, \quad \|B^k p\| = |\lambda|^k \|p\|$$

qui ne peut pas tendre vers 0 par hypothèse sur λ , donc $(B^k p)_{k \in \mathbb{N}}$ ne tend pas vers 0. Ainsi, en considérant $x^{(0)} := p + x$, la suite $(x^{(k)})_{k \in \mathbb{N}}$ ne converge pas vers x , donc la méthode ne converge pas.

2 $\implies$ 3. Comme $\rho(B) < 1$, on peut définir

$$\varepsilon := \frac{1 - \rho(B)}{2} > 0.$$

Il existe une norme subordonnée $\|\cdot\|$ telle que

$$\|B\| \leq \rho(B) + \varepsilon = \frac{1 + \rho(B)}{2} < 1.$$

3 $\implies$ 1. Comme $\|\cdot\|$ est une norme subordonnée, on a :

$$\forall k \in \mathbb{N}, \quad \|B^k e^{(0)}\| \leq \|B\|^k \|e^{(0)}\|.$$

Or, $\|B\| < 1$, donc

$$\lim_{k \rightarrow +\infty} B^k e^{(0)} = 0,$$

ce qui montre que la méthode converge. ■

5.1.2 Décomposition de la matrice

On se propose de construire la matrice B à partir d'une décomposition particulière de la matrice A . Soient pour cela $M \in M_n(\mathbb{R})$, inversible, et $N \in M_n(\mathbb{R})$ telles que $A = M - N$. On a alors :

$$\begin{aligned} Ax = b &\iff (M - N)x = b \\ &\iff Mx = Nx + b \\ &\iff x = M^{-1}Nx + M^{-1}b, \end{aligned}$$

ce qui permet naturellement de définir une méthode itérative par la matrice $B := M^{-1}N$.

Remarque 5.1.3. Comme $B = M^{-1}N = M^{-1}(M - A) = I - M^{-1}A$, on en déduit que la matrice $I - B = M^{-1}A$ est toujours inversible.

Théorème 5.1.4. Supposons que A soit symétrique définie positive. Si la matrice (symétrique) $M^T + N$ est définie positive, alors $\rho(M^{-1}N) < 1$ et, en particulier, la méthode itérative associée converge.

Démonstration. Commençons par vérifier que la matrice $M^T + N$ est bien symétrique. En effet :

$$\begin{aligned} (M^T + N)^T &= M + N^T \\ &= A + N + N^T \\ &= A^T + N + N^T && \text{car } A = A^T \\ &= (A + N)^T + N \\ &= M^T + N. \end{aligned}$$

Soit $\|\cdot\|$ la norme subordonnée définie par la matrice A : $\|\cdot\|^2 = (\cdot)^T A \cdot$. Alors :

$$\|M^{-1}N\| = \|M^{-1}(M - A)\| = \|I - M^{-1}A\| = \sup_{\substack{v \in \mathbb{R}^n \\ \|v\|=1}} \|v - M^{-1}Av\|.$$

Soit alors $v \in \mathbb{R}^n$ tel que $\|v\| = 1$. On note $w := M^{-1}Av$. Alors :

$$\|v - w\|^2 = (v - w)^T A(v - w) = v^T Av - v^T Aw - w^T Av + w^T Aw.$$

Or, $v^T Av = \|v\|^2 = 1$. De plus, $w = M^{-1}Av$, donc $Av = Mw$. On déduit également de la définition de w que $v = A^{-1}Mw$, donc $v^T = w^T M^T (A^{-1})^T = w^T M^T A^{-1}$ car $A = A^T$. Ainsi :

$$\|v - w\|^2 = 1 - w^T M^T w - w^T Mw + w^T Aw = 1 - w^T (M^T + M - A)w = 1 - w^T (M^T + N)w.$$

Comme $v \neq 0$, en particulier $w \neq 0$ et ainsi $w^T(M^T + N)w > 0$ car la matrice $M^T + N$ est supposée définie positive. Autrement dit :

$$\forall v \in \mathbb{R}^n, \quad \|v\| = 1 \implies \|v - M^{-1}Av\| < 1.$$

L'application $v \mapsto \|v - M^{-1}Av\|$ est continue sur le compact $\{v \in \mathbb{R}^n ; \|v\| = 1\}$. Elle y est donc bornée et atteint ses bornes. En particulier, on obtient $\|M^{-1}N\| < 1$ et le résultat s'ensuit car $\rho(M^{-1}N) \leq \|M^{-1}N\|$. ■

5.1.3 Méthodes de Jacobi et de Gauss-Seidel

Dans tout le reste de cette section, on supposera que tous les éléments diagonaux de A sont non nuls :

$$\forall i \in \llbracket 1, n \rrbracket, \quad A_{ii} \neq 0.$$

On définit trois matrices à partir de A : D , sa diagonale, E , l'opposée de sa partie triangulaire inférieure stricte, et F , l'opposée de sa partie triangulaire supérieure stricte. On obtient graphiquement :

$$A = D - E - F = \begin{pmatrix} \ddots & & -F \\ & D & \\ -E & & \ddots \end{pmatrix}.$$

Ces matrices peuvent bien sûr être définies de manière un peu plus formelle :

$$D_{ij} = \begin{cases} A_{ii}, & \text{si } i = j, \\ 0, & \text{sinon,} \end{cases} \quad E_{ij} = \begin{cases} -A_{ij}, & \text{si } i > j, \\ 0, & \text{sinon,} \end{cases} \quad F_{ij} = \begin{cases} -A_{ij}, & \text{si } i < j, \\ 0, & \text{sinon,} \end{cases}$$

pour tout $(i, j) \in \llbracket 1, n \rrbracket^2$.

La méthode de Jacobi entre dans le cadre décrit précédemment en définissant $M := D$ (invertible par hypothèse sur les coefficients diagonaux de A) et $N := E + F$. On obtient la matrice de Jacobi :

$$J := M^{-1}N = D^{-1}(E + F).$$

D'autres choix de combinaison des matrices D , E et F sont possibles. Ainsi la méthode de Gauss-Seidel consiste à choisir $M := D - E$ (invertible également pour les mêmes raisons) et $N := F$ pour définir la matrice de Gauss-Seidel :

$$R_1 := M^{-1}N = (D - E)^{-1}F.$$

Ces deux méthodes sont liées. Pour le montrer, nous aurons besoin du lemme technique suivant.

Lemme 5.1.5. Pour $\mu \in \mathbb{R}^*$, on définit la matrice :

$$A(\mu) := \begin{pmatrix} b_1 & \mu^{-1}c_1 & & & \\ \mu a_2 & b_2 & \mu^{-1}c_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \mu a_{n-1} & b_{n-1} & \mu^{-1}c_{n-1} \\ & & & \mu a_n & b_n \end{pmatrix}.$$

Alors :

$$\forall \mu \in \mathbb{R}^*, \quad \det(A(\mu)) = \det(A(1)).$$

Démonstration. Soit $\mu \in \mathbb{R}^*$. On pose :

$$Q(\mu) := \begin{pmatrix} \mu & & & \\ & \mu^2 & & \\ & & \ddots & \\ & & & \mu^n \end{pmatrix}.$$

Un simple calcul permet de montrer :

$$A(\mu) = Q(\mu)A(1)Q(\mu)^{-1}.$$

Ainsi :

$$\det(A(\mu)) = \det(Q(\mu)) \det(A(1)) \det(Q(\mu)^{-1}),$$

et le résultat s'ensuit car $\det(Q(\mu)^{-1}) = \det(Q(\mu))^{-1}$. ■

Théorème 5.1.6. Supposons que A soit tridiagonale. Alors :

$$\rho(R_1) = \rho(J)^2.$$

Démonstration. Rappelons les expressions des matrices de Jacobi et de Gauss-Seidel :

$$J = D^{-1}(E + F) \quad \text{et} \quad R_1 = (D - E)^{-1}F.$$

Les valeurs propres de J sont les racines de son polynôme caractéristique :

$$\lambda \mapsto \det(J - \lambda I).$$

On rappelle que les éléments diagonaux de A sont supposés non nuls. En particulier, $\det(-D) \neq 0$. Ainsi :

$$\det(J - \lambda I) = 0 \iff \det(-D) \det(J - \lambda I) = 0.$$

Les valeurs propres de J sont donc également les racines du polynôme :

$$p_J: \lambda \mapsto \det(-D) \det(J - \lambda I) = \det(-D(J - \lambda I)) = \det(\lambda D - (E + F)).$$

De la même façon, $\det(E - D) \neq 0$, donc les valeurs propres de R_1 sont les racines du polynôme :

$$p_{R_1}: \lambda \mapsto \det(\lambda D - \lambda E - F).$$

Intéressons-nous aux valeurs propres λ non nulles. Pour commencer :

$$p_{R_1}(\lambda^2) = \det(\lambda^2 D - \lambda^2 E - F).$$

La matrice A étant tridiagonale, on peut par exemple écrire :

$$A = \begin{pmatrix} d_1 & -f_1 & & & \\ -e_2 & d_2 & -f_2 & & \\ & \ddots & \ddots & \ddots & \\ & & -e_{n-1} & d_{n-1} & -f_{n-1} \\ & & & -e_n & d_n \end{pmatrix}.$$

Ainsi :

$$\begin{aligned} \lambda^2 D - \lambda^2 E - F &= \begin{pmatrix} \lambda^2 d_1 & -f_1 & & & \\ -\lambda^2 e_2 & \lambda^2 d_2 & -f_2 & & \\ & \ddots & \ddots & \ddots & \\ & & -\lambda^2 e_{n-1} & \lambda^2 d_{n-1} & -f_{n-1} \\ & & & -\lambda^2 e_n & \lambda^2 d_n \end{pmatrix} \\ &= \lambda \begin{pmatrix} \lambda d_1 & -\lambda^{-1} f_1 & & & \\ -\lambda e_2 & \lambda d_2 & -\lambda^{-1} f_2 & & \\ & \ddots & \ddots & \ddots & \\ & & -\lambda e_{n-1} & \lambda d_{n-1} & -\lambda^{-1} f_{n-1} \\ & & & -\lambda e_n & \lambda d_n \end{pmatrix}. \end{aligned}$$

On en déduit :

$$p_{R_1}(\lambda^2) = \det(\lambda^2 D - \lambda^2 E - F) = \lambda^n \det(\lambda D - \lambda E - \lambda^{-1} F).$$

On applique le lemme technique :

$$\det(\lambda D - \lambda E - \lambda^{-1} F) = \det(\lambda D - E - F).$$

On a donc obtenu :

$$p_{R_1}(\lambda^2) = \lambda^n p_J(\lambda).$$

En utilisant cette relation, on montre facilement que :

$$\lambda \in \text{Sp}(J) \setminus \{0\} \implies \lambda^2 \in \text{Sp}(R_1) \setminus \{0\}.$$

Et réciproquement :

$$\lambda \in \text{Sp}(R_1) \setminus \{0\} \implies \pm \ell \in \text{Sp}(J) \setminus \{0\}$$

où $\ell \in \mathbb{C}$ désigne une racine carrée de λ . ■

Remarque 5.1.7. En particulier, ce résultat implique que, au moins dans les cas des matrices tridiagonales, les méthodes de Jacobi et de Gauss-Seidel convergent (ou divergent) simultanément.

Remarque 5.1.8. On peut également définir les méthodes de Jacobi et de Gauss-Seidel « par blocs » (ce qui consiste moralement à définir D non pas comme une diagonale mais des blocs diagonaux). Avec une approche similaire, on peut montrer que le résultat précédent est alors valable pour des matrices tridiagonales par blocs.

5.1.4 Méthode de relaxation

La méthode de relaxation consiste à introduire un paramètre $\omega \neq 0$ perturbant la définition de la matrice de Gauss-Seidel. On fixe ainsi :

$$M := \frac{D}{\omega} - E \quad \text{et} \quad N := \frac{1-\omega}{\omega}D + F,$$

de sorte que l'on obtient :

$$R_\omega := M^{-1}N = \left(\frac{D}{\omega} - E\right)^{-1} \left(\frac{1-\omega}{\omega}D + F\right).$$

Remarque 5.1.9. En fixant $\omega = 1$, on retrouve bien la méthode de Gauss-Seidel, de matrice associée R_1 , d'où la notation qui avait été choisie à ce moment-là.

Théorème 5.1.10. Pour tout $\omega \neq 0$:

$$\rho(R_\omega) \geq |\omega - 1|.$$

Démonstration. On commence par utiliser l'expression de R_ω :

$$\det(R_\omega) = \det\left(\left(\frac{D}{\omega} - E\right)^{-1}\right) \det\left(\frac{1-\omega}{\omega}D + F\right) = \frac{\det\left(\frac{1-\omega}{\omega}D + F\right)}{\det\left(\frac{D}{\omega} - E\right)}.$$

On se retrouve alors à calculer des déterminants de matrices triangulaires. Après simplification, on obtient :

$$\det(R_\omega) = (1 - \omega)^n.$$

Par ailleurs, en notant $\lambda_i(R_\omega)$ les valeurs propres de R_ω , on peut écrire :

$$\det(R_\omega) = \prod_{i=1}^n \lambda_i(R_\omega).$$

Or :

$$\forall i \in \llbracket 1, n \rrbracket, \quad \rho(R_\omega) \geq |\lambda_i(R_\omega)|.$$

On combine le tout :

$$\rho(R_\omega)^n \geq \left| \prod_{i=1}^n \lambda_i(R_\omega) \right| = |\det(R_\omega)| = |1 - \omega|^n,$$

et on obtient le résultat en appliquant la racine n -ième. ■

Remarque 5.1.11. En particulier, si la méthode de relaxation converge, alors on a nécessairement $\omega \in]0, 2[$.

Le résultat donne ainsi une condition nécessaire à la convergence de la méthode de relaxation, valable pour toute matrice A . Dans le cas où l'on se donne un peu plus d'hypothèses sur A , on peut cependant montrer qu'il s'agit également d'une condition suffisante.

Théorème 5.1.12. Supposons que A soit symétrique définie positive. Si $\omega \in]0, 2[$, alors la méthode de relaxation converge.

Démonstration. On rappelle que $A = M - N$ avec :

$$M = \frac{D}{\omega} - E \quad \text{et} \quad N = \frac{1 - \omega}{\omega} D + F.$$

Ainsi :

$$M^T + N = \left(\frac{D}{\omega} - E \right)^T + \left(\frac{1 - \omega}{\omega} D + F \right) = \frac{D^T}{\omega} - E^T + \frac{1 - \omega}{\omega} D + F.$$

Or, $A = A^T$, donc $E^T = F$, et D est une matrice diagonale donc, en particulier, $D = D^T$. D'où :

$$M^T + N = \frac{D}{\omega} - F + \frac{1 - \omega}{\omega} D + F = \frac{2 - \omega}{\omega} D.$$

Nous allons montrer que D est définie positive. Soit pour cela $v \in \mathbb{R}^n$. En utilisant la structure de D , on calcule rapidement :

$$v^T D v = \sum_{i=1}^n v_i A_{ii} v_i.$$

Pour tout $k \in \llbracket 1, n \rrbracket$, on définit un vecteur w^k par :

$$w_i^k = \begin{cases} v_k, & \text{si } i = k, \\ 0, & \text{sinon,} \end{cases}$$

de sorte que :

$$(w^k)^T A w^k = v_k A_{kk} v_k.$$

Or, A est définie positive, donc $(w^k)^T A w^k > 0$ dès que $w^k \neq 0$. Ainsi, pour tout $i \in \llbracket 1, n \rrbracket$, $v_i A_{ii} v_i > 0$ si $v_i \neq 0$ (et est nul sinon). Comme $v \neq 0$, il existe au moins une composante de v qui ne s'annule pas, ce qui montre que :

$$v^T D v = \sum_{i=1}^n v_i A_{ii} v_i > 0,$$

donc D est définie positive. La matrice $M^T + N = \frac{2-\omega}{\omega} D$ est donc définie positive si et seulement si $\frac{2-\omega}{\omega} > 0$, ce qui est équivalent à imposer $\omega \in]0, 2[$. ■

Les définitions des méthodes de Jacobi, Gauss-Seidel et de relaxation étant proches, il semble naturel de penser qu'elles sont liées d'une certaine façon. C'est effectivement ce qu'indique le résultat suivant.

Théorème 5.1.13. Supposons que A soit tridiagonale et que toutes les valeurs propres de la matrice de Jacobi J soient réelles. Alors la méthode de Jacobi et la méthode de relaxation, pour $\omega \in]0, 2[$, convergent ou divergent simultanément. Dans le cas où elles convergent, les rayons spectraux de leurs matrices sont reliés :

$$\rho(R_\omega) = \begin{cases} \frac{\rho(J)^2 \omega^2 - 2(\omega-1) + \rho(J)\omega \sqrt{\rho(J)^2 \omega^2 - 4(\omega-1)}}{2}, & \text{si } \omega \in]0, \omega_0], \\ \omega - 1, & \text{si } \omega \in]\omega_0, 2[, \end{cases}$$

$$\text{où } \omega_0 := \frac{2}{1 + \sqrt{1 - \rho(J)^2}}.$$

Démonstration. Relation fonctionnelle. On a déjà montré précédemment que les valeurs propres de J sont racines du polynôme

$$p_J: \lambda \mapsto \det(\lambda D - E - F).$$

On montre de la même façon que les valeurs propres de R_ω sont les racines du polynôme

$$p_{R_\omega}: \lambda \mapsto \det\left(E - \frac{D}{\omega}\right) \det(R_\omega - \lambda I) = \det\left(\frac{\lambda + \omega - 1}{\omega} D - \lambda E - F\right).$$

Soit $\lambda \neq 0$. Alors :

$$p_{R_\omega}(\lambda^2) = \det\left(\frac{\lambda^2 + \omega - 1}{\omega} D - \lambda^2 E - F\right).$$

En utilisant la structure tridiagonale de A et le lemme technique précédemment utilisé, on obtient :

$$\begin{aligned} p_{R_\omega}(\lambda^2) &= \det\left(\frac{\lambda^2 + \omega - 1}{\omega} D - \lambda E - \lambda F\right) \\ &= \det\left(\lambda \left(\frac{\lambda^2 + \omega - 1}{\lambda \omega} D - E - F\right)\right) \\ &= \lambda^n p_J\left(\frac{\lambda^2 + \omega - 1}{\lambda \omega}\right). \end{aligned}$$

Lien entre les valeurs propres. Soit $\beta \in \text{Sp}(R_\omega) \setminus \{0\}$. Alors :

$$0 = p_{R_\omega}(\beta) = p_{R_\omega}((\pm\sqrt{\beta})^2),$$

d'où, par la relation qui précède :

$$p_J\left(\frac{\beta + \omega - 1}{\pm\sqrt{\beta}\omega}\right) = 0$$

et ainsi :

$$\pm\frac{\beta + \omega - 1}{\sqrt{\beta}\omega} \in \text{Sp}(J).$$

Réciproquement, soit $\alpha \in \text{Sp}(J)$. Alors $p_J(\alpha) = 0$, d'où $p_{R_\omega}(\lambda^2) = 0$ où λ est tel que :

$$\frac{\lambda^2 + \omega - 1}{\lambda\omega} = \alpha \iff \lambda^2 - \alpha\omega\lambda + \omega - 1 = 0.$$

On s'intéresse alors au discriminant de ce polynôme :

$$\Delta(\omega) := \alpha^2\omega^2 - 4(\omega - 1).$$

On obtient ainsi deux racines (distinctes ou non, réelles ou complexes) :

$$\lambda^\pm = \frac{\alpha\omega \pm \sqrt{\Delta(\omega)}}{2}.$$

On pose :

$$\mu^\pm(\alpha, \omega) := (\lambda^\pm)^2 = \frac{\alpha^2\omega^2 - 2(\omega - 1) \pm \alpha\omega\sqrt{\Delta(\omega)}}{2}.$$

Comme $p_{R_\omega}(\lambda^2) = 0$, on a donc obtenu $\mu^\pm \in \text{Sp}(R_\omega)$.

Un rayon spectral explicite. De ce qui précède, on déduit :

$$\forall \beta \in \text{Sp}(R_\omega) \setminus \{0\}, \exists \alpha \in \text{Sp}(J) / \beta = \mu^\pm(\alpha, \omega).$$

Ainsi :

$$\rho(R_\omega) = \max_{\beta \in \text{Sp}(R_\omega)} |\beta| = \max_{\alpha \in \text{Sp}(J)} (|\mu^-(\alpha, \omega)|, |\mu^+(\alpha, \omega)|).$$

On s'intéresse alors à la fonction

$$\begin{aligned} M: \quad \mathbb{R} \times]0, 2[&\rightarrow \mathbb{R}^+ \\ (\alpha, \omega) &\mapsto \max(|\mu^-(\alpha, \omega)|, |\mu^+(\alpha, \omega)|) \end{aligned}$$

dont on peut restreindre l'étude aux $\alpha \in \mathbb{R}^+$ puisque l'on peut montrer

$$\alpha \in \text{Sp}(J) \iff -\alpha \in \text{Sp}(J).$$

On en déduit au passage que, puisque l'on a supposé que les valeurs propres de J étaient réelles, alors $\rho(J)$ est valeur propre de J .

Le cas $\alpha = 0$. On obtient $\mu^\pm = (\lambda^\pm)^2$ avec

$$(\lambda^\pm)^2 + \omega - 1 = 0.$$

Autrement dit :

$$M(0, \omega) = |\mu^\pm(0, \omega)| = |1 - \omega|.$$

Le cas $\alpha \in]0, 1[$. Revenons à

$$\Delta(\omega) = \alpha^2 \omega^2 - 4\omega + 4.$$

C'est là encore un polynôme du second degré par rapport à ω donc le discriminant est donné par

$$16(1 - \alpha^2) > 0.$$

On obtient alors deux racines :

$$\omega^\pm = \frac{2 \pm 2\sqrt{1 - \alpha^2}}{\alpha^2}.$$

On note $\omega_1(\alpha) := \omega^+$. Ainsi :

$$\omega_1(\alpha) = \frac{2 + 2\sqrt{1 - \alpha^2}}{\alpha^2} > \frac{2}{\alpha^2} > 2.$$

Intéressons-nous alors à :

$$\omega_0(\alpha) := \omega^- = \frac{2 - 2\sqrt{1 - \alpha^2}}{\alpha^2} = \frac{(2 - 2\sqrt{1 - \alpha^2})(1 + \sqrt{1 - \alpha^2})}{\alpha^2(1 + \sqrt{1 - \alpha^2})} = \frac{2}{1 + \sqrt{1 - \alpha^2}}.$$

On a alors :

$$1 < \omega_0(\alpha) < 2 < \omega_1(\alpha).$$

Comme $\omega \in]0, 2[$, deux cas se présentent. Supposons dans un premier temps $\omega \in]\omega_0(\alpha), 2[$. En particulier, $\omega \in]\omega_0(\alpha), \omega_1(\alpha)[$ et ainsi $\Delta(\omega) < 0$. Alors $\lambda^- = \overline{\lambda^+}$, d'où $\mu^- = \overline{\mu^+}$. On obtient ainsi :

$$M(\alpha, \omega) = |\mu^-| = |\mu^+| = |\lambda^+|^2 = \lambda^- \lambda^+ = \omega - 1.$$

Supposons maintenant $\omega \in]0, \omega_0(\alpha)[$. Alors $\Delta(\omega) \geq 0$ et $\lambda^\pm \in \mathbb{R}$. On peut alors écrire $\lambda^- \leq \lambda^+$, d'où $\mu^- \leq \mu^+$. Comme de plus $\lambda^+ > 0$, on en déduit $\mu^+ > 0$ et ainsi :

$$M(\alpha, \omega) = \mu^+(\alpha, \omega) =: \nu(\alpha, \omega)^2.$$

Comme $0 < \alpha < 1$, on trouve :

$$\nu^2 = \frac{\alpha^2 \omega^2 - 2(\omega - 1) + \alpha \omega \sqrt{\alpha^2 \omega^2 - 4(\omega - 1)}}{2} < \frac{\omega^2 - 2(\omega - 1) + \omega \sqrt{\omega^2 - 4(\omega - 1)}}{2}.$$

Or, $\omega^2 - 4(\omega - 1) = (\omega - 2)^2$, donc $\sqrt{\omega^2 - 4(\omega - 1)} = |\omega - 2| = 2 - \omega$ car $0 < \omega < 2$. On en déduit :

$$M(\alpha, \omega) = \nu^2 < 1.$$

Par ailleurs, de l'expression de ν , on tire facilement :

$$\lim_{\omega \rightarrow 0} \nu(\alpha, \omega) = 1,$$

d'où :

$$\lim_{\omega \rightarrow 0} M(\alpha, \omega) = 1.$$

De même :

$$\lim_{\omega \rightarrow \omega_0} \nu(\alpha, \omega) = \frac{\alpha\omega_0}{2},$$

d'où :

$$\lim_{\omega \rightarrow \omega_0} M(\alpha, \omega) = \left(\frac{\alpha\omega_0}{2} \right)^2 = \omega_0 - 1,$$

et on en déduit en particulier la continuité de $\omega \mapsto M(\alpha, \omega)$ en ω_0 à α fixé.

Le cas $\alpha \geq 1$. Dans ce cas, $\Delta(\omega) \geq 0$ et :

$$\begin{aligned} M(\alpha, \omega) = \mu^+(\alpha, \omega) &= \frac{\alpha^2\omega^2 - 2(\omega - 1) + \alpha\omega\sqrt{\alpha^2\omega^2 - 4(\omega - 1)}}{2} \\ &\geq \frac{\omega^2 - 2(\omega - 1) + \omega\sqrt{\omega^2 - 4(\omega - 1)}}{2} = 1. \end{aligned}$$

Convergence simultanée. Supposons que la méthode de Jacobi converge. Alors $\rho(J) < 1$ et :

$$\forall \omega \in]0, 2[, \quad \rho(R_\omega) = M(\rho(J), \omega) < 1$$

et ainsi la méthode de relaxation converge. Réciproquement, supposons que la méthode de Jacobi diverge. Alors $\rho(J) \geq 1$ et, de même, $\rho(R_\omega) = M(\rho(J), \omega) \geq 1$ et ainsi la méthode de relaxation diverge. ■

On obtient ainsi un résultat intéressant dans le cas des matrices tridiagonales symétriques définies positives, pour lesquelles le rayon spectral de R_ω aura l'allure de la [Figure 5.1](#).

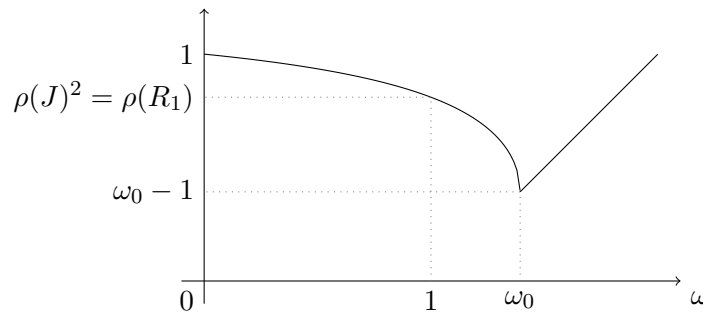


FIGURE 5.1 – Allure du graphe de $\rho(R_\omega)$ en fonction de ω .

Théorème 5.1.14. Supposons que A soit symétrique définie positive et tridiagonale. Alors les méthodes de Jacobi, de Gauss-Seidel, et de relaxation pour $\omega \in]0, 2[$, convergent. Il existe de plus un paramètre optimal pour la relaxation :

$$\omega_0 = \frac{2}{1 + \sqrt{1 - \rho(J)^2}},$$

qui est tel que :

$$\rho(R_{\omega_0}) = \inf_{\omega \in]0, 2[} \rho(R_\omega) = \omega_0 - 1 < \rho(R_1) = \rho(J)^2 < \rho(J).$$

Démonstration. Soit λ une valeur propre de $J = D^{-1}(E + F)$, et soit $v \in \mathbb{R}^n \setminus \{0\}$ un vecteur propre associé. Alors :

$$\begin{aligned} Jv = \lambda v &\implies (E + F)v = \lambda Dv \\ &\implies Av = (1 - \lambda)Dv \end{aligned}$$

où l'on rappelle $A = D - (E + F)$. Ainsi :

$$v^T Av = (1 - \lambda)v^T Dv.$$

Or, A est symétrique définie positive, donc il en va de même pour D , et on a ainsi $v^T Av \in \mathbb{R}_*^+$ et $v^T Dv \in \mathbb{R}_*^+$ car $v \neq 0$. On en déduit $\lambda \in \mathbb{R}$. Ainsi toutes les valeurs propres de J sont réelles. Comme A est tridiagonale, on peut appliquer le théorème précédent. ■

5.2 Méthodes de gradient

On suppose dans cette section que $A \in M_n(\mathbb{R})$ est symétrique définie positive. On pose :

$$\begin{aligned} J: \mathbb{R}^n &\rightarrow \mathbb{R} \\ y &\mapsto \frac{1}{2}(Ay, y) - (b, y). \end{aligned}$$

5.2.1 Idée générale

Lemme 5.2.1. Pour tous $(y, z) \in (\mathbb{R}^n)^2$ et $t \in \mathbb{R}$:

$$J(y + tz) = J(y) + t(Ay - b, z) + \frac{t^2}{2}(Az, z).$$

Démonstration. Soient $(y, z) \in (\mathbb{R}^n)^2$ et $t \in \mathbb{R}$. Alors :

$$\begin{aligned} J(y + tz) &= \frac{1}{2}(A(y + tz), y + tz) - (b, y + tz) \\ &= \frac{1}{2}(Ay, y) + \frac{t}{2}((Ay, z) + (Az, y)) + \frac{t^2}{2}(Az, z) - (b, y) - t(b, z) \end{aligned}$$

et on obtient le résultat en utilisant la symétrie de A . ■

| **Lemme 5.2.2.** Soit $y \in \mathbb{R}^n$. Alors $\nabla J(y) = Ay - b$.

Démonstration. Soient $d \in \mathbb{R}^n$ et $h \in \mathbb{R}$. Alors, en utilisant le [Lemme 5.2.1](#) :

$$J(y + hd) = J(y) + h(Ay - b, d) + \frac{h^2}{2}(Ad, d),$$

et ainsi :

$$\lim_{h \rightarrow 0} \frac{J(y + hd) - J(y)}{h} = (Ay - b, d),$$

d'où le résultat puisque d est quelconque. ■

| **Proposition 5.2.3.** Supposons A symétrique positive. Alors $x \in \mathbb{R}^n$ est solution du système $Ax = b$ si et seulement si x est un minimiseur de J .

Démonstration. Toute solution est un minimiseur. Soit $x \in \mathbb{R}^n$ tel que $Ax = b$. Soit $y \in \mathbb{R}^n$. On pose $d := y - x$. Alors :

$$\begin{aligned} J(y) &= J(x + d) \\ &= J(x) + \underbrace{(Ax - b, d)}_{=0} + \frac{1}{2} \underbrace{(Ad, d)}_{\geq 0} \\ &\geq J(x) \end{aligned}$$

donc x est un minimiseur de J .

Tout minimiseur est une solution. Réciproquement, soit $x \in \mathbb{R}^n$ minimisant J . Soit $y \in \mathbb{R}^n$. On définit

$$\begin{aligned} j: \mathbb{R} &\rightarrow \mathbb{R} \\ t &\mapsto J(x + ty) \end{aligned}$$

dérivable avec :

$$\forall t \in \mathbb{R}, \quad j'(t) = (\nabla J(x + ty), y).$$

Pour tout $t \in \mathbb{R}$, on peut écrire :

$$j(t) = J(x + ty) \geq J(x) = j(0)$$

car x minimise J . En considérant $t > 0$ dans cette inégalité, on obtient :

$$\forall t \in \mathbb{R}_*^+, \quad \frac{j(t) - j(0)}{t} \geq 0,$$

d'où :

$$j'(0) = \lim_{t \rightarrow 0^+} \frac{j(t) - j(0)}{t} \geq 0.$$

De même, avec $t < 0$, on obtient :

$$j'(0) = \lim_{t \rightarrow 0^-} \frac{j(t) - j(0)}{t} \leq 0.$$

D'où $0 = j'(0) = (\nabla J(x), y)$. Comme y est quelconque, on en déduit $\nabla J(x) = 0$ et on conclut avec le lemme précédent. ■

Dans le cas où A est symétrique définie positive, il est donc équivalent de résoudre le système $Ax = b$ et de minimiser la fonctionnelle J . On souhaite maintenant construire une suite $(x^{(k)})_{k \in \mathbb{N}}$ telle que :

$$\forall k \in \mathbb{N}, \quad J(x^{(k+1)}) < J(x^{(k)}).$$

Pour chaque $k \in \mathbb{N}$, on va définir une direction de descente $d^{(k)} \in \mathbb{R}^n$ ainsi qu'un pas $\alpha^{(k)} \in \mathbb{R}$ afin de pouvoir définir :

$$\forall k \in \mathbb{N}, \quad x^{(k+1)} = x^{(k)} + \alpha^{(k)} d^{(k)}.$$

Avec ce choix, la condition $J(x^{(k+1)}) < J(x^{(k)})$ devient :

$$\forall k \in \mathbb{N}, \quad -\alpha^{(k)} (\nabla J(x^{(k)}), d^{(k)}) > \frac{(\alpha^{(k)})^2}{2} (Ad^{(k)}, d^{(k)}).$$

Comme A est symétrique définie positive, le membre de droite est strictement positif. Une condition nécessaire sur $\alpha^{(k)}$ et $d^{(k)}$ pour que $(x^{(k)})_k$ soit une suite minimisante est donc :

$$\forall k \in \mathbb{N}, \quad \alpha^{(k)} (\nabla J(x^{(k)}), d^{(k)}) < 0.$$

Un choix plutôt naturel est alors de considérer $d^{(k)} = -\nabla J(x^{(k)})$ et $\alpha^{(k)} > 0$. C'est l'idée des méthodes de gradient qui consistent, à partir d'un $x^{(0)} \in \mathbb{R}^n$ donné, à construire les itérés :

$$\forall k \in \mathbb{N}, \quad x^{(k+1)} = x^{(k)} + \alpha^{(k)} r^{(k)}$$

où $r^{(k)} := -\nabla J(x^{(k)}) = b - Ax^{(k)}$ désigne le résidu à l'étape k .

5.2.2 Méthode de gradient à pas fixe

Comme son nom l'indique, la méthode de gradient à pas fixe consiste à choisir :

$$\forall k \in \mathbb{N}, \quad \alpha^{(k)} = \alpha$$

où $\alpha > 0$ est donné. On obtient ainsi :

$$\begin{aligned} \forall k \in \mathbb{N}, \quad x^{(k+1)} &= x^{(k)} + \alpha r^{(k)} \\ &= x^{(k)} + \alpha (b - Ax^{(k)}) \\ &= (I - \alpha A)x^{(k)} + \alpha b. \end{aligned}$$

Théorème 5.2.4. La méthode de gradient à pas fixe converge si et seulement si

$$0 < \alpha < \frac{2}{\rho(A)}.$$

Le paramètre optimal est donné par $\alpha_{\text{opt}} = \frac{2}{\lambda_1 + \lambda_n}$ pour lequel $\rho(I - \alpha A) = \frac{\lambda_n - \lambda_1}{\lambda_n + \lambda_1}$.

Démonstration. En posant $B := I - \alpha A$, on reconnaît une méthode itérative telle que définie dans la section précédente. Cette méthode converge si et seulement si $\rho(B) < 1$. Or, on peut montrer que $\mu \in \mathbb{R}$ est valeur propre de B si et seulement si $\mu = 1 - \alpha\lambda$ où $\lambda > 0$ est valeur propre de A . On obtient donc la condition :

$$\forall \lambda \in \text{Sp}(A), |1 - \alpha\lambda| < 1 \iff \forall \lambda \in \text{Sp}(A), 0 < \alpha < \frac{2}{\lambda} \iff 0 < \alpha < \frac{2}{\rho(A)}.$$

Le paramètre optimal est celui qui permet de minimiser $\rho(B)$. Notons

$$0 < \lambda_1 \leq \dots \leq \lambda_n$$

les valeurs propres de A . Les valeurs propres de B sont données par :

$$\forall i \in \llbracket 1, n \rrbracket, \quad \mu_i = 1 - \alpha\lambda_i.$$

Ainsi :

$$\rho(B) = \max_{i \in \llbracket 1, n \rrbracket} |\mu_i| = \max_{i \in \llbracket 1, n \rrbracket} |1 - \alpha\lambda_i|.$$

De l'ordre des valeurs propres de A , on obtient :

$$\forall i \in \llbracket 1, n \rrbracket, \quad |1 - \alpha\lambda_i| \leq \max(|1 - \alpha\lambda_1|, |1 - \alpha\lambda_n|),$$

d'où :

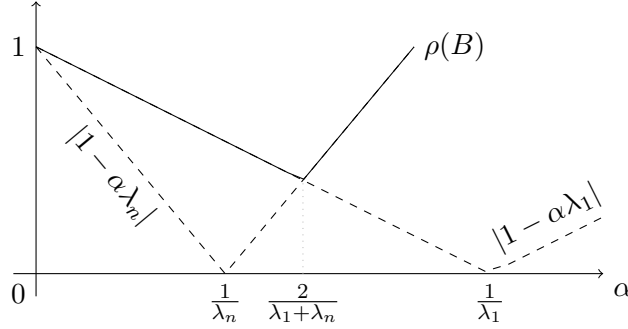
$$\rho(B) = \max(|1 - \alpha\lambda_1|, |1 - \alpha\lambda_n|).$$

On en déduit notamment l'allure du graphe de $\rho(B)$ en fonction de α (voir [Figure 5.2](#)). Le paramètre optimal est le minimum de cette fonction, donné par :

$$1 - \alpha_{\text{opt}}\lambda_1 = \alpha_{\text{opt}}\lambda_n - 1 \iff \alpha_{\text{opt}} = \frac{2}{\lambda_1 + \lambda_n}.$$

La valeur de $\rho(B)$ pour ce paramètre optimal s'en déduit. ■

Remarque 5.2.5. La méthode de gradient à pas fixe « optimale » donne une vitesse de convergence en $\rho(B) = \frac{\lambda_n - \lambda_1}{\lambda_n + \lambda_1}$. Or, A est symétrique, donc $\text{cond}_2(A) = \frac{\lambda_n}{\lambda_1}$ et ainsi $\rho(B) = \frac{\text{cond}_2(A) - 1}{\text{cond}_2(A) + 1}$, qui sera d'autant plus proche de 1 que le conditionnement de A est grand.

FIGURE 5.2 – Allure du graphe de $\rho(B)$ en fonction de α .

5.2.3 Méthode de gradient à pas optimal

Reprenons une suite de pas $(\alpha^{(k)})_{k \in \mathbb{N}}$ non constante. Rappelons

$$x^{(k+1)} = x^{(k)} + \alpha^{(k)} r^{(k)},$$

d'où :

$$J(x^{(k+1)}) = J(x^{(k)}) - \alpha^{(k)} \|r^{(k)}\|^2 + \frac{(\alpha^{(k)})^2}{2} (Ar^{(k)}, r^{(k)}).$$

Comme on cherche à minimiser J , l'idée consiste à choisir $\alpha^{(k)}$ tel que $J(x^{(k+1)})$ soit le plus petit possible. On recherche donc le minimum de la fonction

$$\alpha \mapsto J(x^{(k)}) - \alpha \|r^{(k)}\|^2 + \frac{\alpha^2}{2} (Ar^{(k)}, r^{(k)}).$$

Il s'agit d'une simple fonction polynomiale en α , dont on détermine facilement le minimum :

$$\alpha^{(k)} = \frac{\|r^{(k)}\|^2}{(Ar^{(k)}, r^{(k)})}.$$

Remarque 5.2.6. La matrice A étant définie positive, on a $(Ar^{(k)}, r^{(k)}) > 0$ donc, en particulier, le dénominateur dans la définition de $\alpha^{(k)}$ ne s'annule pas.

A priori, la méthode requiert ainsi le calcul de deux produits matrice-vecteur à chaque étape : $Ax^{(k)}$ pour calculer le nouveau résidu, et $Ar^{(k)}$ pour calculer le nouveau pas. On peut cependant reformuler :

$$\begin{aligned} r^{(k+1)} &= b - Ax^{(k+1)} \\ &= b - A(x^{(k)} + \alpha^{(k)} r^{(k)}) \\ &= r^{(k)} - \alpha^{(k)} Ar^{(k)}. \end{aligned}$$

De cette manière, il n'y a plus qu'un seul produit matrice-vecteur : $Ar^{(k)}$. On obtient l'[Algorithme 5.2.7](#).

Algorithme 5.2.7 : Méthode de gradient à pas optimal

$x^{(0)} \in \mathbb{R}^n$ donné
 $r^{(0)} := b - Ax^{(0)}$
pour $k \in \mathbb{N}$
 $y^{(k)} := Ar^{(k)}$
 $\alpha^{(k)} := \frac{\|r^{(k)}\|^2}{(y^{(k)}, r^{(k)})}$
 $x^{(k+1)} := x^{(k)} + \alpha^{(k)} r^{(k)}$
 $r^{(k+1)} := r^{(k)} - \alpha^{(k)} y^{(k)}$

Proposition 5.2.8. Deux résidus successifs obtenus par la méthode de gradient à pas optimal sont orthogonaux :

$$\forall k \in \mathbb{N}, \quad (r^{(k+1)}, r^{(k)}) = 0.$$

Démonstration. Le calcul est immédiat en utilisant $r^{(k+1)} = r^{(k)} - \alpha^{(k)} Ar^{(k)}$ et la définition des $\alpha^{(k)}$. ■

Théorème 5.2.9. La méthode de gradient à pas optimal converge et on a :

$$\forall k \in \mathbb{N}, \quad \|x^{(k)} - x\| \leq \sqrt{\text{cond}_2(A)} \left(\frac{\text{cond}_2(A) - 1}{\text{cond}_2(A) + 1} \right)^k \|x^{(0)} - x\|.$$

Démonstration. Soit $k \in \mathbb{N}$. Commençons par :

$$\begin{aligned}
 \|x^{(k+1)} - x\|_A^2 &= (A(x^{(k+1)} - x), x^{(k+1)} - x) \\
 &= (A(x^{(k+1)} - x), x^{(k+1)} - x^{(k)} + x^{(k)} - x) \\
 &= (A(x^{(k+1)} - x), x^{(k+1)} - x^{(k)}) + (A(x^{(k+1)} - x), x^{(k)} - x).
 \end{aligned}$$

Or, $x^{(k+1)} - x^{(k)} = \alpha^{(k)} r^{(k)}$ et $A(x^{(k+1)} - x) = Ax^{(k+1)} - b = -r^{(k+1)}$, d'où :

$$(A(x^{(k+1)} - x), x^{(k+1)} - x^{(k)}) = -\alpha^{(k)} (r^{(k+1)}, r^{(k)}) = 0.$$

On obtient ainsi :

$$\begin{aligned}
 \|x^{(k+1)} - x\|_A^2 &= (A(x^{(k+1)} - x), x^{(k)} - x) \\
 &= (A(x^{(k)} + \alpha^{(k)} r^{(k)} - x), x^{(k)} - x) \\
 &= (A(x^{(k)} - x), x^{(k)} - x) + \alpha^{(k)} (Ar^{(k)}, x^{(k)} - x).
 \end{aligned}$$

Or, $(Ar^{(k)}, x^{(k)} - x) = (r^{(k)}, A(x^{(k)} - x))$ car A est symétrique. Comme $A(x^{(k)} - x) = -r^{(k)}$, on en déduit :

$$\|x^{(k+1)} - x\|_A^2 = \|x^{(k)} - x\|_A^2 - \alpha^{(k)} \|r^{(k)}\|^2.$$

Intéressons-nous maintenant à :

$$\begin{aligned}\|x^{(k)} - x\|_A^2 &= (A(x^{(k)} - x), x^{(k)} - x) \\ &= (A(x^{(k)} - x), A^{-1}A(x^{(k)} - x)) \\ &= (-r^{(k)}, A^{-1}(-r^{(k)})) \\ &= \|r^{(k)}\|_{A^{-1}}^2.\end{aligned}$$

Ainsi :

$$\begin{aligned}\|x^{(k+1)} - x\|_A^2 &= \|r^{(k)}\|_{A^{-1}}^2 - \alpha^{(k)}\|r^{(k)}\|^2 \\ &= \|r^{(k)}\|_{A^{-1}}^2 - \frac{\|r^{(k)}\|^4}{\|r^{(k)}\|_A^2} \\ &= \|r^{(k)}\|_{A^{-1}}^2 \left(1 - \frac{\|r^{(k)}\|^4}{\|r^{(k)}\|_A^2 \|r^{(k)}\|_{A^{-1}}^2} \right).\end{aligned}$$

On utilise alors l'inégalité de Kantorovitch :

$$\forall y \in \mathbb{R}^n, \quad \|y\|_A^2 \|y\|_{A^{-1}}^2 \leq \frac{1}{4} \frac{(\lambda_1 + \lambda_n)^2}{\lambda_1 \lambda_n} \|y\|^4,$$

où λ_1 et λ_n désignent la plus petite et la plus grande valeur propre de A . En l'appliquant à notre cas, on obtient :

$$\|x^{(k+1)} - x\|_A^2 \leq \frac{(\lambda_n - \lambda_1)^2}{(\lambda_1 + \lambda_n)^2} \|x^{(k)} - x\|_A^2.$$

Ainsi :

$$\forall k \in \mathbb{N}, \quad \|x^{(k)} - x\|_A \leq \left(\frac{\lambda_n - \lambda_1}{\lambda_1 + \lambda_n} \right)^k \|x^{(0)} - x\|_A.$$

Le résultat s'ensuit alors en utilisant :

$$\forall y \in \mathbb{R}^n, \quad \sqrt{\lambda_1} \|y\| \leq \|y\|_A \leq \sqrt{\lambda_n} \|y\|$$

et en utilisant l'expression du conditionnement en norme 2. ■

Considérons un exemple simple :

$$A := \begin{pmatrix} a_1 & 0 \\ 0 & a_2 \end{pmatrix}, \quad b := 0,$$

avec $(a_1, a_2) \in (\mathbb{R}^*)^2$ tel que $a_1 \neq a_2$. La solution du système $Ax = b$ est évidemment $x = (0, 0)$. Pour un $x^{(0)} \in \mathbb{R}^2$ donné, on peut expliciter la suite construite par la méthode de gradient à pas optimal :

$$\forall k \in \mathbb{N}, \quad x_1^{(k+1)} = \frac{a_2^2(a_2 - a_1)x_1^{(k)}(x_2^{(k)})^2}{a_1^3(x_1^{(k)})^2 + a_2^3(x_2^{(k)})^2} \quad \text{et} \quad x_2^{(k+1)} = \frac{a_1^2(a_1 - a_2)(x_1^{(k)})^2 x_2^{(k)}}{a_1^3(x_1^{(k)})^2 + a_2^3(x_2^{(k)})^2}.$$

Autrement dit, si on choisit la donnée initiale $x^{(0)}$ telle que $x_1^{(0)} \neq 0$ et $x_2^{(0)} \neq 0$, alors :

$$\forall k \in \mathbb{N}, \quad x_1^{(k)} \neq 0 \quad \text{et} \quad x_2^{(k)} \neq 0,$$

ce qui signifie que la méthode ne convergera jamais en un nombre fini d'itérations. Il s'agit pourtant d'un exemple simple, et on se dit qu'il devrait être possible de trouver mieux. Comme le nom de la méthode l'indique, le pas est déjà optimal, mais il n'en est pas forcément de même pour la direction de descente. Reprenons $x^{(1)}$ le premier itéré construit par la méthode de gradient à pas optimal :

$$x^{(1)} = x^{(0)} + \frac{\|r^{(0)}\|^2}{(Ar^{(0)}, r^{(0)})} r^{(0)}.$$

On souhaite construire

$$x^{(2)} = x^{(1)} + \alpha^{(1)} d^{(1)}.$$

La solution du système étant l'origine du plan, il semble naturel de penser que, dans ce cas, la direction optimale $d^{(1)}$ n'est autre que $-x^{(1)}$: de cette façon, il existera un $\alpha^{(1)}$ tel que $x^{(2)} = 0$. Ce cas particulier nous permet de mettre en lumière une condition intéressante puisque, avec ces définitions, on a :

$$(Ad^{(1)}, d^{(0)}) = 0,$$

où on a noté $d^{(0)} := r^{(0)}$ la première direction de descente.

5.2.4 Méthode du gradient conjugué

En revenant à la définition de la méthode de gradient à pas optimal ([Algorithme 5.2.7](#)), on peut montrer par récurrence que :

$$\forall k \in \mathbb{N}^*, \quad x^{(k)} - x^{(0)} \in \text{vect}\{r^{(0)}, \dots, r^{(k-1)}\}.$$

L'idée de la méthode du gradient conjugué consiste à conserver cette propriété, tout en considérant d'autres directions de descente. À chaque étape $k \in \mathbb{N}$, on recherche donc

$$x^{(k+1)} = x^{(0)} + \sum_{j=0}^k \alpha_j^{(k)} r^{(j)}$$

avec $(\alpha_j^{(k)})_{j \in \llbracket 0, k \rrbracket} \in \mathbb{R}^{k+1}$ minimisant $J(x^{(k+1)})$. Si, avec la méthode de gradient à pas optimal, deux résidus successifs étaient orthogonaux, cette nouvelle approche permet d'aller plus loin.

Proposition 5.2.10. Les résidus obtenus par la méthode du gradient conjugué sont deux à deux orthogonaux :

$$\forall k \in \mathbb{N}, \forall i \in \llbracket 0, k \rrbracket, \quad (r^{(k+1)}, r^{(i)}) = 0.$$

Démonstration. On note

$$g: (\alpha_0, \dots, \alpha_k) \mapsto J \left(x^{(0)} + \sum_{j=0}^k \alpha_j r^{(j)} \right).$$

Soit $\alpha := (\alpha_0, \dots, \alpha_k)$ minimisant g . Alors $\nabla g(\alpha) = 0$. Ainsi :

$$\forall i \in \llbracket 0, k \rrbracket, \quad 0 = \frac{\partial g}{\partial \alpha_i}(\alpha) = (\nabla J(x^{(k+1)}), r^{(i)}).$$

Or $\nabla J(x^{(k+1)}) = -r^{(k+1)}$, ce qui permet de conclure. ■

Une itération consiste à déterminer un pas $\gamma^{(k)}$ ainsi qu'une direction de descente $d^{(k)}$ afin d'écrire :

$$x^{(k+1)} = x^{(k)} + \gamma^{(k)} d^{(k)}.$$

Cherchons à mieux comprendre $d^{(k)}$. Pour cela, on peut décomposer $x^{(k)}$ de la même façon que $x^{(k+1)}$:

$$x^{(k)} = x^{(0)} + \sum_{j=0}^{k-1} \alpha_j^{(k-1)} r^{(j)}.$$

Ainsi :

$$\gamma^{(k)} d^{(k)} = x^{(k+1)} - x^{(k)} = \alpha_k^{(k)} r^{(k)} + \sum_{j=0}^{k-1} (\alpha_j^{(k)} - \alpha_j^{(k-1)}) r^{(j)},$$

et on en déduit :

$$d^{(k)} \in \text{vect}\{r^{(0)}, \dots, r^{(k)}\}.$$

Remarque 5.2.11. Quitte à multiplier $d^{(k)}$ par une constante, on pourra parfois considérer

$$d^{(k)} \in r^{(k)} + \text{vect}\{r^{(0)}, \dots, r^{(k-1)}\}.$$

On notera que ceci n'a de sens que si $\alpha_k^{(k)} \neq 0$. Cependant, si $\alpha_k^{(k)} = 0$, cela signifie que $r^{(k)}$ n'a aucune contribution dans la nouvelle direction de descente où seuls les précédents résidus auront été utilisés : autrement dit, on aurait pu trouver « mieux » pour $x^{(k)}$, ce qui n'est pas possible par définition.

Proposition 5.2.12. Les directions de descente sont deux à deux A -conjuguées :

$$\forall k \in \mathbb{N}^*, \forall j \in \llbracket 0, k-1 \rrbracket, \quad (Ad^{(k)}, d^{(j)}) = 0.$$

Démonstration. On écrit

$$x^{(k+1)} = x^{(k)} + \gamma^{(k)} d^{(k)}.$$

Ainsi :

$$r^{(k+1)} = b - Ax^{(k+1)} = r^{(k)} - \gamma^{(k)} Ad^{(k)}.$$

Soit $i \in \llbracket 0, k-1 \rrbracket$. Comme les résidus sont deux à deux orthogonaux et $i \neq k+1$, on a en particulier $(r^{(k+1)}, r^{(i)}) = 0$, d'où :

$$0 = (r^{(k+1)}, r^{(i)}) = (r^{(k)}, r^{(i)}) - \gamma^{(k)}(Ad^{(k)}, r^{(i)}).$$

Or, $i \neq k$ donc $(r^{(k)}, r^{(i)}) = 0$ et on a ainsi montré :

$$\forall i \in \llbracket 0, k-1 \rrbracket, \quad (Ad^{(k)}, r^{(i)}) = 0.$$

Soit $j \in \llbracket 0, k-1 \rrbracket$. On trouve $(\beta_0, \dots, \beta_j) \in \mathbb{R}^{j+1}$ tel que :

$$d^j = \sum_{i=0}^j \beta_i r^{(i)}.$$

Ainsi :

$$(Ad^{(k)}, d^j) = \sum_{i=0}^j \beta_i (Ad^{(k)}, r^{(i)}).$$

Par ce qui précède, on obtient le résultat. ■

Proposition 5.2.13. Pour $k \in \mathbb{N}$, on note :

$$\begin{aligned} R_k &:= \text{vect}\{r^{(0)}, \dots, r^{(k)}\}, \\ D_k &:= \text{vect}\{d^{(0)}, \dots, d^{(k)}\}, \\ K_k &:= \text{vect}\{r^{(0)}, Ar^{(0)}, \dots, A^k r^{(0)}\}. \end{aligned}$$

Si $d^{(0)} = r^{(0)}$, alors $R_k = D_k = K_k$.

Démonstration. On procède par récurrence. Le cas $k = 0$ est évident. Soit donc $k \in \mathbb{N}$ tel que $R_k = D_k = K_k$. On a déjà vu que l'on pouvait écrire :

$$d^{(k+1)} = r^{(k+1)} + \sum_{j=0}^k \alpha_j r^{(j)},$$

ce qui montre $d^{(k+1)} \in R_{k+1}$ et donc $D_{k+1} \subset R_{k+1}$. De cette égalité, on tire également :

$$r^{(k+1)} = d^{(k+1)} - \underbrace{\sum_{j=0}^k \alpha_j r^{(j)}}_{\in R_k = D_k} \in D_{k+1},$$

d'où $R_{k+1} \subset D_{k+1}$ et ainsi $R_{k+1} = D_{k+1}$. Par ailleurs :

$$x^{(k+1)} = x^{(k)} + \gamma^{(k)} d^{(k)}, \text{ avec } \gamma^{(k)} \neq 0.$$

Ainsi :

$$r^{(k+1)} = r^{(k)} - \gamma^{(k)} Ad^{(k)}.$$

Comme $d^{(k)} \in D_k = K_k$ par hypothèse de récurrence, on peut écrire :

$$d^{(k)} = \sum_{j=0}^k \beta_j A^j r^{(0)}, \quad \text{d'où } Ad^{(k)} = \sum_{j=0}^k \beta_j A^{j+1} r^{(0)}.$$

On en déduit $Ad^{(k)} \in K_{k+1}$. Comme $r^{(k)} \in R_k = K_k$, on a ainsi $r^{(k+1)} \in K_{k+1}$, ce qui montre $R_{k+1} \subset K_{k+1}$. Isolons le terme $A^{k+1}r^{(0)}$:

$$Ad^{(k)} = \sum_{j=1}^k \beta_{j-1} A^j r^{(0)} + \beta_k A^{k+1} r^{(0)},$$

d'où :

$$\gamma^{(k)} \sum_{j=1}^k \beta_{j-1} A^j r^{(0)} + \gamma^{(k)} \beta_k A^{k+1} r^{(0)} = \gamma^{(k)} Ad^{(k)} = r^{(k)} - r^{(k+1)}$$

et ainsi :

$$\gamma^{(k)} \beta_k A^{k+1} r^{(0)} = \underbrace{r^{(k)} - r^{(k+1)}}_{\in R_{k+1}} - \underbrace{\gamma^{(k)} \sum_{j=1}^k \beta_{j-1} A^j r^{(0)}}_{\in K_k = R_k} \in R_{k+1},$$

ce qui permet de conclure. ■

Nous allons maintenant exploiter ce qui a été obtenu afin d'écrire l'algorithme du gradient conjugué. Soit $x^{(0)} \in \mathbb{R}^n$ donné. On peut alors calculer $r^{(0)} := b - Ax^{(0)}$, et on définit $d^{(0)} := r^{(0)}$. Supposons que, pour $k \in \mathbb{N}$, on dispose des $(x^{(0)}, \dots, x^{(k)})$, $(r^{(0)}, \dots, r^{(k)})$ et $(d^{(0)}, \dots, d^{(k)})$. On cherche à en déduire $x^{(k+1)}$, $r^{(k+1)}$ et $d^{(k+1)}$. On écrit pour cela :

$$x^{(k+1)} = x^{(k)} + \gamma^{(k)} d^{(k)},$$

avec $\gamma^{(k)}$ à définir de sorte à minimiser $J(x^{(k+1)})$. On écrit alors :

$$J(x^{(k+1)}) = J(x^{(k)}) - \gamma^{(k)} (r^{(k)}, d^{(k)}) + \frac{(\gamma^{(k)})^2}{2} (Ad^{(k)}, d^{(k)}).$$

On se retrouve à minimiser une fonction quadratique en γ , ce qui donne :

$$\gamma^{(k)} = \frac{(r^{(k)}, d^{(k)})}{(Ad^{(k)}, d^{(k)})}.$$

On peut aller plus loin en rappelant :

$$d^{(k)} \in r^{(k)} + \text{vect}\{r^{(0)}, \dots, r^{(k-1)}\},$$

ce qui permet d'écrire :

$$d^{(k)} = r^{(k)} + \sum_{j=0}^{k-1} \delta_j r^{(j)}.$$

Ainsi :

$$(r^{(k)}, d^{(k)}) = \|r^{(k)}\|^2 + \sum_{j=0}^{k-1} \delta_j \underbrace{(r^{(k)}, r^{(j)})}_{=0},$$

d'où :

$$\gamma^{(k)} = \frac{\|r^{(k)}\|^2}{(Ad^{(k)}, d^{(k)})}.$$

On peut ainsi directement calculer

$$x^{(k+1)} = x^{(k)} + \gamma^{(k)} d^{(k)}.$$

On en déduit :

$$r^{(k+1)} = b - Ax^{(k+1)} = r^{(k)} - \gamma^{(k)} Ad^{(k)}.$$

On pose maintenant :

$$d^{(k+1)} := r^{(k+1)} + \sum_{j=0}^k \alpha_j d^{(j)}.$$

On va déterminer les α_i en imposant :

$$\forall i \in \llbracket 0, k \rrbracket, \quad (d^{(k+1)}, Ad^{(i)}) = 0.$$

Soit $i \in \llbracket 0, k \rrbracket$. Alors :

$$0 = (d^{(k+1)}, Ad^{(i)}) = (r^{(k+1)}, Ad^{(i)}) + \sum_{j=0}^k \alpha_j \underbrace{(d^{(j)}, Ad^{(i)})}_{=0 \text{ si } j \neq i} = (r^{(k+1)}, Ad^{(i)}) + \alpha_i (d^{(i)}, Ad^{(i)}).$$

On en déduit :

$$\forall i \in \llbracket 0, k \rrbracket, \quad \alpha_i = -\frac{(r^{(k+1)}, Ad^{(i)})}{(d^{(i)}, Ad^{(i)})}.$$

On va montrer que la plupart de ces coefficients sont en réalité nuls. Soit $i \in \llbracket 0, k-1 \rrbracket$. Comme $d^{(i)} \in D_i = K_i$, on en déduit $Ad^{(i)} \in K_{i+1} = R_{i+1}$, donc on peut écrire :

$$Ad^{(i)} = \sum_{j=0}^{i+1} \delta_j r^{(j)}.$$

Ainsi :

$$(r^{(k+1)}, Ad^{(i)}) = \sum_{j=0}^{i+1} \delta_j (r^{(k+1)}, r^{(j)}) = 0$$

Algorithme 5.2.14 : Méthode du gradient conjugué

```

 $x^{(0)} \in \mathbb{R}^n$  donné
 $r^{(0)} := b - Ax^{(0)}$ 
 $d^{(0)} := r^{(0)}$ 
pour  $k \in \mathbb{N}$ 
     $z^{(k)} := Ad^{(k)}$ 
     $p_k := (z^{(k)}, d^{(k)})$ 
     $\gamma^{(k)} := \frac{\|r^{(k)}\|^2}{p_k}$ 
     $x^{(k+1)} := x^{(k)} + \gamma^{(k)} d^{(k)}$ 
     $r^{(k+1)} := r^{(k)} - \gamma^{(k)} z^{(k)}$ 
     $\alpha_k := -\frac{(r^{(k+1)}, z^{(k)})}{p_k}$ 
     $d^{(k+1)} := r^{(k+1)} + \alpha_k d^{(k)}$ 

```

car les résidus sont deux à deux orthogonaux. Autrement dit, $\alpha_i = 0$, et il reste :

$$d^{(k+1)} = r^{(k+1)} + \alpha_k d^{(k)} \quad \text{avec} \quad \alpha_k = -\frac{(r^{(k+1)}, Ad^{(k)})}{(d^{(k)}, Ad^{(k)})}.$$

On a ainsi obtenu l'[Algorithme 5.2.14](#).

Le lemme suivant établit une propriété intéressante des directions de descente : en un nombre suffisant, elles forment une base de l'espace.

| **Lemme 5.2.15.** La famille $\{d^{(0)}, \dots, d^{(n-1)}\}$ forme une base de $\mathbb{R}^n$.

Démonstration. Soit $(\alpha_0, \dots, \alpha_{n-1}) \in \mathbb{R}^n$ tel que :

$$\sum_{i=0}^{n-1} \alpha_i d^{(i)} = 0.$$

Soit $j \in \llbracket 0, n-1 \rrbracket$. Alors :

$$0 = \left(\sum_{i=0}^{n-1} \alpha_i d^{(i)}, Ad^{(j)} \right) = \sum_{i=0}^{n-1} \alpha_i \underbrace{(d^{(i)}, Ad^{(j)})}_{=0 \text{ si } i \neq j} = \alpha_j (d^{(j)}, Ad^{(j)}).$$

Or, A est définie positive donc $(d^{(j)}, Ad^{(j)}) > 0$, d'où $\alpha_j = 0$. La famille $\{d^{(0)}, \dots, d^{(n-1)}\}$ est donc libre dans $\mathbb{R}^n$ et est composée de n vecteurs : il s'agit donc d'une base. ■

Théorème 5.2.16. L'algorithme du gradient conjugué converge en au plus n itérations.

Démonstration. Comme les résidus sont deux à deux orthogonaux, en particulier :

$$\forall i \in \llbracket 0, n-1 \rrbracket, \quad (r^{(n)}, r^{(i)}) = 0.$$

Ainsi $r^{(n)}$ est orthogonal à $R_{n-1} = D_{n-1} = \mathbb{R}^n$ par le lemme qui précède, donc $r^{(n)} = 0$. ■

Proposition 5.2.17. Soit $N \in \mathbb{N}^*$. Effectuer N itérations du gradient conjugué nécessite $\mathcal{O}(2(N+1)n^2)$ flops.

Démonstration. Un produit matrice-vecteur nécessite $2n^2 - n$ opérations, et le calcul d'un produit scalaire (donc également celui du carré d'une norme) en nécessite $2n - 1$.

L'initialisation requiert ainsi $2n^2 - n$ opérations pour le calcul de $Ax^{(0)}$, puis n opérations pour la soustraction des deux vecteurs, soit un total de $2n^2$ opérations.

À chaque itération, on a besoin de $2n^2 + 11n - 1$ opérations.

On obtient alors :

$$2n^2 + (2n^2 + 11n - 1)N = 2(N+1)n^2 + 11Nn - N$$

opérations élémentaires. ■

La méthode du gradient conjugué converge en un nombre fini d'itérations, c'est la raison pour laquelle elle est parfois considérée comme une méthode directe de résolution d'un système linéaire. Il s'agit cependant d'un résultat théorique : numériquement, les erreurs d'arrondis peuvent faire en sorte que la convergence ne soit pas atteinte en n itérations. Par ailleurs, en comptant le nombre d'opérations élémentaires qu'elle nécessite, on remarque qu'elle n'est pas si avantageuse que cela : avec n itérations, on atteint $\mathcal{O}(2n^3)$ flops, ce qui est supérieur à la méthode du pivot de Gauss ! Ce calcul est fait en supposant que tous les coefficients de la matrice sont utilisés : si la matrice est creuse, les produits matrice-vecteur seront beaucoup plus rapides, ce qui devrait redonner l'avantage à la méthode du gradient conjugué.

On s'intéresse maintenant à la vitesse de convergence de la méthode. Pour cela, on introduit l'ensemble :

$$\forall k \in \mathbb{N}, \quad G_k := \{P \in \mathbb{R}_k[X] ; P(0) = 1\}.$$

Lemme 5.2.18. Si $Ax = b$, alors :

$$\forall y \in \mathbb{R}^n, \quad J(y) = \frac{1}{2}\|x - y\|_A^2 - \frac{1}{2}\|x\|_A^2.$$

Démonstration. Soit $y \in \mathbb{R}^n$. Comme $b = Ax$, on peut écrire :

$$\begin{aligned}
 J(y) &= \frac{1}{2}(Ay, y) - (b, y) \\
 &= \frac{1}{2}(Ay, y) - (Ax, y) \\
 &= \frac{1}{2}(Ay - Ax, y) - \frac{1}{2}(Ax, y) \\
 &= \frac{1}{2}(Ay - Ax, y) - \frac{1}{2} \underbrace{(Ax, y)}_{=(Ay, x)} + \frac{1}{2}(Ax, x) - \frac{1}{2}(Ax, x) \\
 &= \frac{1}{2}(Ay - Ax, y) - \frac{1}{2}(Ay - Ax, x) - \frac{1}{2}(Ax, x),
 \end{aligned}$$

d'où le résultat. ■

Lemme 5.2.19. Soit $k \in \mathbb{N}$. Alors le k -ème itéré de la méthode du gradient conjugué vérifie :

$$\|x - x^{(k)}\|_A = \min_{Q \in G_k} \|Q(A)(x - x^{(0)})\|_A.$$

Démonstration. Partant de

$$x^{(k)} - x^{(0)} \in R_{k-1} = K_{k-1},$$

on écrit :

$$x^{(k)} - x^{(0)} = P_{k-1}(A)r^{(0)}$$

avec $P_{k-1} \in \mathbb{R}_{k-1}[X]$ vérifiant :

$$J(x^{(0)} + P_{k-1}(A)r^{(0)}) = \min_{P \in \mathbb{R}_{k-1}[X]} J(x^{(0)} + P(A)r^{(0)}).$$

Du lemme précédent, on déduit que ce problème de minimisation est équivalent à :

$$\|x - x^{(0)} - P_{k-1}(A)r^{(0)}\|_A = \min_{P \in \mathbb{R}_{k-1}[X]} \|x - x^{(0)} - P(A)r^{(0)}\|_A.$$

Or, pour tout $P \in \mathbb{R}_{k-1}[X]$:

$$\begin{aligned}
 x - x^{(0)} - P(A)r^{(0)} &= x - x^{(0)} - P(A)(b - Ax^{(0)}) \\
 &= x - x^{(0)} - P(A)(Ax - Ax^{(0)}) \\
 &= (I - P(A)A)(x - x^{(0)}) \\
 &= Q(A)(x - x^{(0)})
 \end{aligned}$$

avec $Q := 1 - PX \in G_k$. ■

Lemme 5.2.20. Soient $0 < a < b$. Soit $k \in \mathbb{N}$. Alors :

$$\min_{Q \in G_k} \max_{t \in [a, b]} |Q(t)| = \frac{1}{T_k \left(\frac{a+b}{b-a} \right)},$$

où T_k est le k -ème polynôme de Tchebychev. En particulier :

$$\min_{Q \in G_k} \max_{t \in [a, b]} |Q(t)| \leq 2 \left(\frac{\sqrt{b} - \sqrt{a}}{\sqrt{a} + \sqrt{b}} \right)^k.$$

Démonstration. Majoration du minimum. On commence par introduire $\phi: t \mapsto \frac{a+b-2t}{b-a}$ qui définit notamment une bijection entre $[a, b]$ et $[-1, 1]$. On définit alors le polynôme :

$$q_k(X) := \frac{T_k(\phi(X))}{T_k(\phi(0))}.$$

On vérifie aisément que $q_k \in G_k$. Ainsi :

$$\min_{Q \in G_k} \max_{t \in [a, b]} |Q(t)| \leq \max_{t \in [a, b]} |q_k(t)|.$$

Or :

$$\forall u \in [-1, 1], \quad T_k(u) = \cos(k \arccos u).$$

Ainsi :

$$\forall t \in [a, b], \quad |T_k(\phi(t))| \leq 1,$$

l'égalité étant atteinte en (par exemple) $t = b$. On a donc obtenu :

$$\min_{Q \in G_k} \max_{t \in [a, b]} |Q(t)| \leq \max_{t \in [a, b]} |q_k(t)| = \frac{1}{|T(\phi(0))|}.$$

Majoration explicite. On considère $t := \frac{a+b}{b-a}$. Comme $0 < a < b$, alors $t \geq 1$ et ainsi :

$$T_k(t) = \cosh(k \operatorname{arccosh} t).$$

On peut montrer :

$$T_k(t) = \frac{1}{2} \left(\left(t + \sqrt{t^2 - 1} \right)^k + \left(t - \sqrt{t^2 - 1} \right)^k \right).$$

Or, $t - \sqrt{t^2 - 1} \geq 0$ donc

$$T_k(t) \geq \frac{1}{2} \left(t + \sqrt{t^2 - 1} \right)^k.$$

On calcule alors :

$$\begin{aligned}
 t + \sqrt{t^2 - 1} &= \frac{a+b}{b-a} + \sqrt{\frac{(a+b)^2}{(b-a)^2} - 1} \\
 &= \frac{a+b}{b-a} + \sqrt{\frac{4ab}{(b-a)^2}} \\
 &= \frac{a+b+2\sqrt{ab}}{b-a} \\
 &= \frac{(\sqrt{a} + \sqrt{b})^2}{(\sqrt{b} - \sqrt{a})(\sqrt{b} + \sqrt{a})} \\
 &= \frac{\sqrt{a} + \sqrt{b}}{\sqrt{b} - \sqrt{a}},
 \end{aligned}$$

d'où le résultat (en particulier, $T_k(\phi(0)) > 0$).

Égalité. Soit $Q \in G_k$ tel que

$$\max_{t \in [a,b]} |Q(t)| \leq \max_{t \in [a,b]} |q_k(t)|.$$

On pose $\tilde{Q} := q_k - Q$. Pour tout $m \in \llbracket 0, k \rrbracket$, on note

$$u_m := \cos\left(\frac{m\pi}{k}\right)$$

les points tels que $T_k(u_m) = (-1)^m$. En notant $t_m := \phi^{-1}(u_m)$, on a alors :

$$q_k(t_m) = \frac{(-1)^m}{T_k(\phi(0))}.$$

Par ailleurs :

$$|Q(t_m)| \leq \max_{t \in [a,b]} |q_k(t)| = \frac{1}{T_k(\phi(0))}.$$

Ainsi :

$$-\frac{1}{T_k(\phi(0))} \leq \tilde{Q}(t_m) \leq \frac{1}{T_k(\phi(0))},$$

donc :

$$\frac{(-1)^m - 1}{T_k(\phi(0))} \leq Q(t_m) \leq \frac{(-1)^m + 1}{T_k(\phi(0))}.$$

En particulier, si m est pair, alors $\tilde{Q}(t_m) \geq 0$ et, si m est impair, $\tilde{Q}(t_m) \leq 0$. Comme $\tilde{Q}$ est un polynôme, en particulier, $\tilde{Q}$ est continu. Par le théorème des valeurs intermédiaires, pour tout $m \in \llbracket 0, k-1 \rrbracket$, on trouve donc $\alpha_m \in [t_m, t_{m+1}]$ tel que $\tilde{Q}(\alpha_m) = 0$. On a ainsi déterminé k racines distinctes de $\tilde{Q}$ dans $[a, b]$. Ou presque. Les α_m étant dans les sous-intervalles fermés, il est toujours possible qu'ils soient égaux à l'un des t_m . Si tel est le cas,

alors il est possible que deux sous-intervalles consécutifs ne contiennent finalement qu'un seul α_m . Cependant, si l'un des t_m est racine de $\tilde{Q}$, cela signifie :

$$|q_k(t_m)| = |Q(t_m)| = \frac{1}{T_k(\phi(0))}.$$

Donc t_m est un extremum de q_k et de Q : ainsi $\tilde{Q}'(t_m) = 0$, donc t_m est une racine double. Au final, on obtient donc bien au moins k racines.

Par ailleurs, $q_k \in G_k$ et $Q \in G_k$ donc :

$$\tilde{Q}(0) = p_k(0) - Q(0) = 1 - 1 = 0.$$

Comme $0 \notin [a, b]$, on obtient donc $k+1$ racines distinctes de $\tilde{Q}$. Or, $\tilde{Q}$ est de degré au plus k , donc $\tilde{Q} \equiv 0$ et ainsi $Q = q_k$. ■

Théorème 5.2.21. La méthode du gradient conjugué converge et on a :

$$\forall k \in \mathbb{N}, \quad \|x - x^{(k)}\| \leq 2\sqrt{\text{cond}_2(A)} \left(\frac{\sqrt{\text{cond}_2(A)} - 1}{\sqrt{\text{cond}_2(A)} + 1} \right)^k \|x - x^{(0)}\|.$$

Démonstration. On note les valeurs propres de A :

$$0 < \lambda_1 \leq \dots \leq \lambda_n.$$

Soit $\{u_1, \dots, u_n\}$ une base orthonormée de vecteurs propres associés aux valeurs propres de A . On peut alors décomposer selon cette base :

$$x - x^{(0)} = \sum_{i=1}^n \alpha_i u_i.$$

On a alors directement :

$$\|x - x^{(0)}\|_A^2 = (A(x - x^{(0)}), x - x^{(0)}) = \left(\sum_{i=1}^n \alpha_i \lambda_i u_i, \sum_{i=1}^n \alpha_i u_i \right) = \sum_{i=1}^n \alpha_i^2 \lambda_i.$$

Soit $k \in \mathbb{N}$. Soit $Q \in G_k$. Alors :

$$Q(A)(x - x^{(0)}) = \sum_{i=1}^n \alpha_i Q(\lambda_i) u_i,$$

d'où :

$$\begin{aligned} \|Q(A)(x - x^{(0)})\|_A^2 &= \sum_{i=1}^n \alpha_i^2 \lambda_i Q(\lambda_i)^2 \\ &\leq \sum_{i=1}^n \alpha_i^2 \lambda_i \max_{\lambda \in [\lambda_1, \lambda_n]} |Q(\lambda)|^2 \\ &= \max_{\lambda \in [\lambda_1, \lambda_n]} |Q(\lambda)|^2 \|x - x^{(0)}\|_A^2. \end{aligned}$$

On en déduit :

$$\|Q(A)(x - x^{(0)})\|_A \leq \max_{\lambda \in [\lambda_1, \lambda_n]} |Q(\lambda)| \|x - x^{(0)}\|_A.$$

On applique le [Lemme 5.2.19](#) :

$$\|x - x^{(k)}\|_A = \min_{Q \in G_k} \|Q(A)(x - x^{(0)})\|_A \leq \min_{Q \in G_k} \max_{\lambda \in [\lambda_1, \lambda_n]} |Q(\lambda)| \|x - x^{(0)}\|_A.$$

Le [Lemme 5.2.20](#) donne alors :

$$\|x - x^{(k)}\|_A \leq 2 \left(\frac{\sqrt{\text{cond}_2(A)} - 1}{\sqrt{\text{cond}_2(A)} + 1} \right)^k \|x - x^{(0)}\|_A,$$

d'où l'on déduit le résultat en utilisant les inégalités entre les normes $\|\cdot\|$ et $\|\cdot\|_A$. ■

5.2.5 Gradient conjugué préconditionné

La vitesse de convergence de la méthode du gradient conjugué dépend du conditionnement de la matrice. Si celui-ci est trop grand, le taux sera proche de 1 et la méthode convergera plus lentement. On cherche à améliorer cela. Dans ce but, une piste est de transformer le système initial, un peu comme on le faisait pour les méthodes directes : on sait très bien résoudre des systèmes triangulaires, donc on transforme le système pour obtenir un système triangulaire. Ici, l'idée est donc de se donner une matrice inversible $C \in M_n(R)$ et de résoudre le système

$$C^{-1}Ax = C^{-1}b,$$

qui est bien équivalent au système initial $Ax = b$. Si le conditionnement de $C^{-1}A$ est meilleur que celui de A , la méthode convergera alors plus rapidement, et ce vers la solution recherchée.

Les résultats obtenus ne sont cependant valables que dans le cas où la matrice du système est symétrique définie positive. En général, rien ne garantit que $C^{-1}A$ l'est :

$$(C^{-1}A)^T = A^T C^{-T} = AC^{-T}.$$

Même si on considère C symétrique, alors $C^{-T} = C^{-1}$, mais on n'aura toujours pas l'égalité tant que A et C^{-1} ne commutent pas, donc la symétrie n'est pas conservée. Si on considère C symétrique définie positive, cependant, on peut remarquer que la matrice

$$C^{-1/2}AC^{-1/2}$$

est encore symétrique définie positive. On va donc s'intéresser au système

$$C^{-1/2}AC^{-1/2}C^{1/2}x = C^{-1/2}b$$

dont on vérifie aisément qu'il est toujours équivalent au système initial. On va donc appliquer la méthode du gradient conjugué au système défini par $\tilde{A} := C^{-1/2}AC^{-1/2}$ et $\tilde{b} :=$

Algorithme 5.2.22 : Méthode du gradient conjugué préconditionné, première version

```

 $y^{(0)} \in \mathbb{R}^n$  donné
 $\tilde{r}^{(0)} := \tilde{b} - \tilde{A}y^{(0)}$ 
 $\tilde{d}^{(0)} := \tilde{r}^{(0)}$ 
pour  $k \in \mathbb{N}$ 
     $\tilde{z}^{(k)} := \tilde{A}\tilde{d}^{(k)}$ 
     $\tilde{p}_k := (\tilde{z}^{(k)}, \tilde{d}^{(k)})$ 
     $\tilde{\gamma}^{(k)} := \frac{\|\tilde{r}^{(k)}\|^2}{\tilde{p}_k}$ 
     $y^{(k+1)} := y^{(k)} + \tilde{\gamma}^{(k)}\tilde{d}^{(k)}$ 
     $\tilde{r}^{(k+1)} := \tilde{r}^{(k)} - \tilde{\gamma}^{(k)}\tilde{z}^{(k)}$ 
     $\tilde{\alpha}_k := -\frac{(\tilde{r}^{(k+1)}, \tilde{z}^{(k)})}{\tilde{p}_k}$ 
     $\tilde{d}^{(k+1)} := \tilde{r}^{(k+1)} + \tilde{\alpha}_k\tilde{d}^{(k)}$ 

```

$C^{-1/2}b$, avec pour inconnue $y := C^{1/2}x$. On peut facilement réécrire l'Algorithme 5.2.14 pour résoudre ce nouveau système. On construira alors une suite $(y^{(k)})_k$, en notant $\tilde{r}^{(k)}$ les résidus correspondants et $\tilde{d}^{(k)}$ les directions de descente. On obtient l'Algorithme 5.2.22.

Une fois la convergence atteinte, on peut retrouver $x = C^{-1/2}y$. Il semblerait cependant un peu plus satisfaisant de construire directement la suite $(x^{(k)})_k$. À chaque étape k , on peut simplement calculer :

$$x^{(k)} = C^{-1/2}y^{(k)}.$$

Au cours d'une itération, on calcule $y^{(k+1)}$. À partir de la relation précédente, on peut obtenir $x^{(k+1)}$:

$$\begin{aligned}
 x^{(k+1)} &= C^{-1/2}y^{(k+1)} \\
 &= C^{-1/2}(y^{(k)} + \tilde{\gamma}^{(k)}\tilde{d}^{(k)}) \\
 &= x^{(k)} + \tilde{\gamma}^{(k)}C^{-1/2}\tilde{d}^{(k)}.
 \end{aligned}$$

Un choix plutôt naturel si l'on souhaite construire directement la suite des $x^{(k)}$ serait alors de considérer

$$d^{(k)} := C^{-1/2}\tilde{d}^{(k)}.$$

On peut également exprimer le résidu en y en fonction du résidu en x :

$$\begin{aligned}
 \tilde{r}^{(k)} &= \tilde{b} - \tilde{A}y^{(k)} \\
 &= C^{-1/2}b - C^{-1/2}AC^{-1/2}C^{1/2}x^{(k)} \\
 &= C^{-1/2}(b - Ax^{(k)}) \\
 &= C^{-1/2}r^{(k)}.
 \end{aligned}$$

En particulier :

$$d^{(0)} = C^{-1/2} \tilde{d}^{(0)} = C^{-1/2} \tilde{r}^{(0)} = C^{-1} r^{(0)}.$$

Voyons comment les autres variables se comportent. Pour commencer :

$$\begin{aligned} \tilde{z}^{(k)} &= \tilde{A} \tilde{d}^{(k)} \\ &= C^{-1/2} A C^{-1/2} C^{1/2} d^{(k)} \\ &= C^{-1/2} A d^{(k)}. \end{aligned}$$

Ainsi :

$$\tilde{p}_k = (C^{-1/2} A d^{(k)}, C^{1/2} d^{(k)}) = (A d^{(k)}, d^{(k)})$$

car $C^{1/2}$ est symétrique. On peut par ailleurs calculer :

$$\|\tilde{r}^{(k)}\| = (C^{-1/2} r^{(k)}, C^{-1/2} r^{(k)}) = (C^{-1} r^{(k)}, r^{(k)}).$$

On obtient alors :

$$\tilde{\gamma}^{(k)} = \frac{(C^{-1} r^{(k)}, r^{(k)})}{(A d^{(k)}, d^{(k)})}.$$

Avec tout cela, on peut donc calculer directement $x^{(k+1)}$ sans passer par $y^{(k+1)}$. À ce stade, on a cependant toujours besoin de $\tilde{r}^{(k)}$ et surtout de $\tilde{d}^{(k)}$. Voyons déjà comment mettre à jour le résidu :

$$\begin{aligned} r^{(k+1)} &= C^{1/2} \tilde{r}^{(k+1)} \\ &= C^{1/2} (\tilde{r}^{(k)} - \tilde{\gamma}^{(k)} \tilde{z}^{(k)}) \\ &= C^{1/2} (C^{-1/2} r^{(k)} - \tilde{\gamma}^{(k)} C^{-1/2} A d^{(k)}) \\ &= r^{(k)} - \tilde{\gamma}^{(k)} A d^{(k)}. \end{aligned}$$

On peut également exprimer $\tilde{\alpha}_k$ directement en fonction de ce résidu :

$$\alpha_k = -\frac{(C^{-1} r^{(k+1)}, A d^{(k)})}{(A d^{(k)}, d^{(k)})}.$$

Pour finir, mettons à jour directement la nouvelle direction de descente :

$$\begin{aligned} d^{(k+1)} &= C^{-1/2} \tilde{d}^{(k)} \\ &= C^{-1/2} (\tilde{r}^{(k+1)} + \tilde{\alpha}_k \tilde{d}^{(k)}) \\ &= C^{-1/2} (C^{-1/2} r^{(k+1)} + \tilde{\alpha}_k C^{1/2} d^{(k)}) \\ &= C^{-1} r^{(k+1)} + \tilde{\alpha}_k d^{(k)}. \end{aligned}$$

On a alors tout ce qu'il nous faut pour nous débarrasser de la variable y , ce qui est fait dans l'[Algorithme 5.2.23](#). Au-delà de l'aspect esthétique de ne plus passer par la variable y , on remarque surtout que cette approche permet de ne plus avoir à calculer la racine de C .

Algorithme 5.2.23 : Méthode du gradient conjugué préconditionné

```

 $x^{(0)} \in \mathbb{R}^n$  donné
 $r^{(0)} := b - Ax^{(0)}$ 
Résoudre le système  $Cy^{(0)} = r^{(0)}$ 
 $d^{(0)} := y^{(0)}$ 
pour  $k \in \mathbb{N}$ 
     $z^{(k)} := Ad^{(k)}$ 
     $p_k := (z^{(k)}, d^{(k)})$ 
     $\gamma^{(k)} := \frac{(y^{(k)}, r^{(k)})}{p_k}$ 
     $x^{(k+1)} := x^{(k)} + \gamma^{(k)}d^{(k)}$ 
     $r^{(k+1)} := r^{(k)} - \gamma^{(k)}z^{(k)}$ 
    Résoudre le système  $Cy^{(k+1)} = r^{(k+1)}$ 
     $\alpha_k := -\frac{(y^{(k+1)}, z^{(k)})}{p_k}$ 
     $d^{(k+1)} := y^{(k+1)} + \alpha_k d^{(k)}$ 

```

Le choix d'un préconditionneur est déjà un problème difficile en soi. Il est guidé par différentes contraintes, notamment C doit être symétrique définie positive et telle que

$$\text{cond}_2(C^{-1/2}AC^{-1/2}) \ll \text{cond}_2(A)$$

pour que l'ajout de complexité soit justifié. L'[Algorithme 5.2.23](#) indique qu'il va falloir résoudre un système défini par la matrice C à chaque itération : il faut donc également que C soit « facile à inverser ». C'est par exemple le cas si C est une matrice diagonale, ou si on en connaît une décomposition particulière. Dans ce dernier cas, on peut d'ailleurs simplifier légèrement l'estimation du conditionnement.

Proposition 5.2.24. Soit $C \in M_n(\mathbb{R})$ symétrique définie positive. Soit $T \in M_n(\mathbb{R})$ inversible telle que $C = TT^T$. Alors

$$\text{cond}_2(C^{-1/2}AC^{-1/2}) = \text{cond}_2(T^{-1}AT^{-T}).$$

Démonstration. Commençons par introduire T :

$$C^{-1/2}AC^{-1/2} = C^{-1/2}TT^{-1}AT^{-T}T^TC^{-1/2}.$$

Soit $P := C^{-1/2}T$. Alors :

$$P^{-1} = T^{-1}C^{1/2} = T^{-1}CC^{-1/2} = T^{-1}(TT^T)C^{-1/2} = T^TC^{-1/2}.$$

On obtient ainsi :

$$C^{-1/2}AC^{-1/2} = PT^{-1}AT^{-T}P^{-1}.$$

Donc $C^{-1/2}AC^{-1/2}$ et $T^{-1}AT^{-T}$ sont semblables. Elles ont donc les mêmes valeurs propres, et donc le même conditionnement en norme 2. ■

Remarque 5.2.25. Comme C est supposée symétrique définie positive, la matrice T existe nécessairement : la décomposition de Cholesky peut par exemple être appliquée.

Avec une telle décomposition, on peut ainsi obtenir une estimation du conditionnement de la matrice qui nous intéresse, sous certaines conditions. Avant de l'énoncer, rappelons la définition du coefficient de Rayleigh :

$$R(M, x) := \frac{(Mx, x)}{(x, x)},$$

pour une certaine matrice symétrique M et un vecteur non nul x . Ce coefficient fait l'objet d'un certain nombre de résultats. Citons donc seulement ceux qui vont nous intéresser :

$$\sup_{x \neq 0} R(M, x) = \lambda_{\max}(M), \quad \inf_{x \neq 0} R(M, x) = \lambda_{\min}(M),$$

où $\lambda_{\max}(M)$ (respectivement $\lambda_{\min}(M)$) désigne la plus grande (respectivement la plus petite) valeur propre de M .

Proposition 5.2.26. Soit $C \in M_n(\mathbb{R})$ symétrique définie positive. Soit $T \in M_n(\mathbb{R})$ inversible telle que $C = TT^T$. Soient $\alpha > 0$ et $\beta > 0$ des constantes telles que :

$$\forall x \in \mathbb{R}^n \setminus \{0\}, \quad \alpha \leq \frac{(Ax, x)}{(Cx, x)} \leq \beta.$$

Alors :

$$\text{cond}_2(T^{-1}AT^{-T}) \leq \frac{\beta}{\alpha}.$$

Démonstration. Soit $x \neq 0$. D'une part :

$$(Cx, x) = (TT^T x, x) = (T^T x, T^T x) = (y, y),$$

où l'on a posé $y := T^T x$. D'autre part :

$$\begin{aligned} (Ax, x) &= (TT^{-1}AT^{-T}T^T x, x) \\ &= (T^{-1}AT^{-T}T^T x, T^T x) \\ &= (T^{-1}AT^{-T}y, y). \end{aligned}$$

Ainsi :

$$\frac{(Ax, x)}{(Cx, x)} = \frac{(T^{-1}AT^{-T}y, y)}{(y, y)} = R(T^{-1}AT^{-T}, y).$$

De l'hypothèse, on tire donc :

$$\alpha \leq R(T^{-1}AT^{-T}, y) \leq \beta.$$

Ainsi :

$$\lambda_{\max}(T^{-1}AT^{-T}) = \sup_{y \neq 0} R(T^{-1}AT^{-T}, y) \leq \beta,$$

et :

$$\lambda_{\min}(T^{-1}AT^{-T}) = \inf_{y \neq 0} R(T^{-1}AT^{-T}, y) \geq \alpha.$$

On conclut alors :

$$\text{cond}_2(T^{-1}AT^{-T}) = \frac{\lambda_{\max}(T^{-1}AT^{-T})}{\lambda_{\min}(T^{-1}AT^{-T})} \leq \frac{\beta}{\alpha}.$$

■

Remarque 5.2.27. Les matrices A et C étant symétriques définies positives, elles définissent une norme sur $\mathbb{R}^n$. Toutes les normes étant équivalentes, l'existence des constantes α et β est immédiate.

Plus les normes définies par les matrices A et C sont proches, plus le préconditionnement sera efficace. On pouvait s'y attendre : plus C est « proche » de A , plus $C^{-1}A$ sera proche de l'identité, qui définit un système somme toute plutôt facile à inverser.

Un préconditionneur assez facile à tester est tout simplement la diagonale de A : comme A est symétrique définie positive, sa diagonale également. Ce préconditionneur est souvent référencé sous le nom de préconditionneur de Jacobi, en référence à la méthode itérative du même nom (si on se souvient de la définition de la méthode, le lien est évident !).

Nous allons nous intéresser à un autre préconditionneur : SSOR (pour « Symmetric Successive Over-Relaxation »). Reprenons la décomposition $A = D - E - F$ (diagonale, partie triangulaire inférieure stricte et partie triangulaire supérieure stricte) et définissons :

$$C := \frac{1}{\omega(2-\omega)}(D - \omega E)D^{-1}(D - \omega F),$$

avec $\omega \in]0, 2[$. Comme A est symétrique, on peut déjà remarquer $F = E^T$. Ainsi, en posant

$$T := \frac{1}{\sqrt{\omega}\sqrt{2-\omega}}(D - \omega E)D^{-1/2},$$

on obtient $C = TT^T$. Ce préconditionneur a donc la structure nécessaire à l'application des résultats précédents. De plus, T étant triangulaire inférieure, le système $Cz = d$ à résoudre dans la méthode du gradient conjugué préconditionné pourra être facilement résolu par remontée/descente.

Proposition 5.2.28. Le préconditionneur SSOR C vérifie :

$$\forall x \in \mathbb{R}^n \setminus \{0\}, \quad \frac{(Ax, x)}{(Cx, x)} \leq 1.$$

Si de plus il existe des constantes $\mu > 0$ et $\delta \geq 0$ telles que :

$$\max_{x \neq 0} \frac{(Dx, x)}{(Ax, x)} \leq \mu \quad \text{et} \quad \max_{x \neq 0} \frac{(x, ED^{-1}E^Tx) - \frac{1}{4}(Dx, x)}{(Ax, x)} \leq \delta,$$

alors :

$$\forall x \in \mathbb{R}^n \setminus \{0\}, \quad \frac{(Ax, x)}{(Cx, x)} \geq \frac{2 - \omega}{1 + \mu \frac{(2-\omega)^2}{4\omega} + \omega\delta}.$$

Démonstration. Majoration. Commençons par le premier point. Dans ce but, on pose :

$$\tilde{D} := \frac{2 - \omega}{\omega} D \quad \text{et} \quad \tilde{E} := \frac{\omega - 1}{\omega} D - E,$$

de telle sorte que $\tilde{D} + \tilde{E} = \frac{D}{\omega} - E$. Avec ces notations, on a alors :

$$\begin{aligned} C &= \frac{1}{\omega(2 - \omega)} (D - \omega E) D^{-1} (D - \omega E)^T \\ &= \frac{1}{2 - \omega} \left(\frac{D}{\omega} - E \right) D^{-1} \omega \left(\frac{D}{\omega} - E \right)^T \\ &= (\tilde{D} + \tilde{E}) \tilde{D}^{-1} (\tilde{D} + \tilde{E}^T) \\ &= (I + \tilde{E} \tilde{D}^{-1}) (\tilde{D} + \tilde{E}^T) \\ &= \underbrace{\tilde{D} + \tilde{E}^T + \tilde{E}}_{=A} + \tilde{E} \tilde{D}^{-1} \tilde{E}^T. \end{aligned}$$

Ainsi, pour $x \neq 0$, on a :

$$\begin{aligned} (Cx, x) &= (Ax, x) + (\tilde{E} \tilde{D}^{-1} \tilde{E}^T x, x) \\ &= (Ax, x) + (\tilde{D}^{-1} \tilde{E}^T x, \tilde{E}^T x). \end{aligned}$$

Comme A est symétrique définie positive, il en va de même pour D et donc pour $\tilde{D}$. Ainsi $(\tilde{D}^{-1} \tilde{E}^T x, \tilde{E}^T x) \geq 0$. Autrement dit, $(Cx, x) \geq (Ax, x)$.

Minoration. Pour le deuxième point, commençons par un petit calcul :

$$\begin{aligned}
(2 - \omega)C &= \frac{1}{\omega}(D - \omega E)D^{-1}(D - \omega E^T) \\
&= \frac{1}{\omega}(I - \omega ED^{-1})(D - \omega E^T) \\
&= \frac{1}{\omega}(D - \omega E^T - \omega E + \omega^2 ED^{-1}E^T) \\
&= \frac{D}{\omega} - E^T - E + \omega ED^{-1}E^T \\
&= \frac{D}{\omega} - D + \underbrace{D - E^T - E}_{=A} + \omega ED^{-1}E^T \\
&= A + \frac{1 - \omega}{\omega}D + \omega ED^{-1}E^T \\
&= A + \omega \left(ED^{-1}E^T - \frac{1}{4}D \right) + \frac{\omega}{4}D + \frac{1 - \omega}{\omega}D \\
&= A + \omega \left(ED^{-1}E^T - \frac{1}{4}D \right) + \frac{(2 - \omega)^2}{4\omega}D.
\end{aligned}$$

Ainsi :

$$(2 - \omega)(Cx, x) = (Ax, x) + \omega \left(\left(ED^{-1}E^T - \frac{1}{4}D \right) x, x \right) + \frac{(2 - \omega)^2}{4\omega}(Dx, x).$$

On obtient l'inégalité en factorisant par $\frac{(Ax, x)}{2 - \omega}$ et en utilisant les hypothèses. ■

5.3 Méthodes de Krylov

On suppose dans cette section que $A \in M_n(\mathbb{R})$ est inversible. Les méthodes de Krylov peuvent être vues comme une généralisation de la méthode du gradient conjugué [Saa03]. L'idée de départ s'appuie sur une simple observation. Considérons p le polynôme caractéristique de A :

$$p: \lambda \mapsto \det(A - \lambda I) =: \sum_{k=0}^n a_k \lambda^k.$$

Le théorème de Cayley-Hamilton affirme :

$$p(A) = 0.$$

Autrement dit :

$$0 = \sum_{k=0}^n a_k A^k = a_0 I + \sum_{k=1}^n a_k A^k.$$

La matrice A étant inversible, on peut multiplier cette égalité par A^{-1} :

$$0 = a_0 A^{-1} + \sum_{k=1}^n a_k A^{k-1}.$$

Or, $a_0 = \det A \neq 0$ car A est inversible, donc on obtient :

$$A^{-1} = - \sum_{k=1}^n \frac{a_k}{a_0} A^{k-1}.$$

Soit $x \in \mathbb{R}^n$ la solution du système $Ax = b$. Pour une certaine donnée initiale $x^{(0)} \in \mathbb{R}^n$, on écrit

$$v := x - x^{(0)},$$

un vecteur « erreur » à rechercher pour pouvoir retrouver x à partir de $x^{(0)}$. Notons $r^{(0)} := b - Ax^{(0)}$ le premier résidu. Par définition de v , on doit avoir $Av = r^{(0)}$, soit $v = A^{-1}r^{(0)}$ et ainsi :

$$v = - \sum_{k=1}^n \frac{a_k}{a_0} A^{k-1} r^{(0)}.$$

Autrement dit, le vecteur « erreur » vit dans l'espace $\text{vect}\{r^{(0)}, Ar^{(0)}, \dots, A^{n-1}r^{(0)}\}$.

5.3.1 Espaces de Krylov

Définition 5.3.1. Soit $v \in \mathbb{R}^n$. Soit $m \in \mathbb{N}^*$. L'espace de Krylov de degré m de v associé à A est l'espace vectoriel

$$K_m(A, v) := \text{vect}\{v, Av, \dots, A^{m-1}v\}.$$

Quand il n'y aura pas d'ambiguïté, on notera simplement $K_m = K_m(A, v)$.

Définition 5.3.2. Soit $v \in \mathbb{R}^n$. On appelle degré de v associé à A l'entier

$$\deg(A, v) := \min\{\deg P ; P \in \mathbb{R}[X] \setminus \{0\} \text{ et } P(A)v = 0\}.$$

Remarque 5.3.3. Comme le polynôme caractéristique p de A est tel que $p(A) = 0$, cette définition est bien posée et on a $\deg(A, v) \leq n$.

Proposition 5.3.4. Soit $v \in \mathbb{R}^n$. Soit $\mu := \deg(A, v)$. Alors K_μ est invariant par A et :

$$\forall m \geq \mu, \quad K_m = K_\mu.$$

Démonstration. Par définition du degré, il existe $P \in \mathbb{R}_\mu[X]$ tel que $P(A)v = 0$. Quitte à normaliser P , on peut le supposer unitaire et on note :

$$P(X) = \sum_{k=0}^{\mu} a_k X^k, \quad \text{avec } a_\mu = 1.$$

Par définition de P , on a :

$$0 = P(a)v = \sum_{k=0}^{\mu} a_k A^k v = \sum_{k=0}^{\mu-1} a_k A^k v + A^{\mu} v.$$

Ainsi :

$$A^{\mu} v = - \sum_{k=0}^{\mu-1} a_k A^k v \in K_{\mu}.$$

On peut notamment en déduire :

$$A^{\mu+1} v = - \sum_{k=0}^{\mu-1} a_k A^{k+1} v = - \underbrace{\sum_{k=1}^{\mu-1} a_{k-1} A^k v}_{\in K_{\mu}} - \underbrace{a_{\mu-1} A^{\mu} v}_{\in K_{\mu}} \in K_{\mu},$$

et ainsi de suite pour obtenir :

$$\forall m \geq \mu, \quad A^m v \in K_{\mu}.$$

Le résultat s'ensuit alors. ■

Proposition 5.3.5. Soient $v \in \mathbb{R}^n$ et $m \in \mathbb{N}^*$. Alors :

$$\dim K_m(A, v) = m \iff \deg(A, v) \geq m.$$

Démonstration. L'ensemble $\{v, Av, \dots, A^{m-1}v\}$ forme une base de K_m si et seulement si le seul polynôme p de degré inférieur ou égal à $m-1$ tel que $p(A)v = 0$ est le polynôme nul, ce qui est équivalent à $\deg(A, v) \geq m$. ■

Corollaire 5.3.6. Soient $v \in \mathbb{R}^n$ et $m \in \mathbb{N}^*$. Alors :

$$\dim K_m(A, v) = \min(m, \deg(A, v)).$$

Démonstration. Si $m \leq \deg(A, v)$, alors $\dim K_m = m = \min(m, \deg(A, v))$. Sinon, soit $\mu := \deg(A, v)$. Alors $\dim K_{\mu} = \mu$. Or, comme $m > \mu$, $K_m = K_{\mu}$, ce qui termine la preuve. ■

5.3.2 Petit détour par Gram-Schmidt

Soit $m \in \mathbb{N}^*$. On considère une famille libre de vecteurs $\{u_1, \dots, u_m\}$ de $\mathbb{R}^n$. On peut vérifier que la famille $\{v_1, \dots, v_m\}$ construite par l'algorithme de Gram-Schmidt (décrit dans l'[Algorithme 5.3.7](#)) est orthonormale.

Algorithme 5.3.7 : Méthode de Gram-Schmidt classique**Entrée :** $\{u_1, \dots, u_m\}$ famille libre dans $\mathbb{R}^n$ **Sortie :** $\{v_1, \dots, v_m\}$ famille orthonormale**pour** j de 1 à m $w_j := u_j$ **pour** i de 1 à $j - 1$ $w_j := w_j - (u_j, v_i)v_i$ $v_j := \frac{w_j}{\|w_j\|}$

L'algorithme de Gram-Schmidt est instable. Pour le rendre plus stable, on s'intéresse à l'étape de projection :

$$w_j = u_j - \sum_{i=1}^{j-1} (u_j, v_i)v_i.$$

L'idée est de réaliser cette étape de projection « petit à petit », en considérant à chaque fois le produit scalaire avec le vecteur partiellement projeté. On obtient l'[Algorithme 5.3.8](#).

Algorithme 5.3.8 : Méthode de Gram-Schmidt modifiée**Entrée :** $\{u_1, \dots, u_m\}$ famille libre dans $\mathbb{R}^n$ **Sortie :** $\{v_1, \dots, v_m\}$ famille orthonormale**pour** j de 1 à m $w_j := u_j$ **pour** i de 1 à $j - 1$ $w_j := w_j - (w_j, v_i)v_i$ $v_j := \frac{w_j}{\|w_j\|}$

On peut montrer par récurrence (sur i) que les w_j calculés par la méthode de Gram-Schmidt modifiée sont les mêmes que ceux calculés par la méthode de Gram-Schmidt classique. Cette seconde approche est cependant plus stable numériquement.

5.3.3 Méthode d'Arnoldi

Soient $v \in \mathbb{R}^n \setminus \{0\}$ et $m \in \mathbb{N}^*$. La méthode d'Arnoldi vise à construire une base orthonormale de l'espace de Krylov

$$K_m = K_m(A, v) = \text{vect}\{v, Av, \dots, A^{m-1}v\}.$$

L'idée de cette méthode est d'appliquer la méthode de Gram-Schmidt pour orthonormaliser la famille $\{v, Av, \dots, A^{m-1}v\}$, qui est un choix plutôt naturel : on sait que cette famille forme une base de K_m dès que $\deg(A, v) \geq m$. On obtient l'[Algorithme 5.3.9](#).

Algorithme 5.3.9 : Méthode d'Arnoldi**Entrée** : $v \in \mathbb{R}^n \setminus \{0\}$, $m \in \mathbb{N}^*$ **Sortie** : Une base orthonormée $(v_j)_j$ de K_m et une matrice H

```

 $v_1 := \frac{v}{\|v\|}$ 
pour  $j$  de 1 à  $m$ 
     $w_j := Av_j$ 
    pour  $i$  de 1 à  $j$ 
         $h_{ij} := (Av_j, v_i)$ 
         $w_j := w_j - h_{ij}v_i$ 
     $h_{j+1,j} = \|w_j\|$ 
    si  $h_{j+1,j} = 0$ 
        Arrêt
     $v_{j+1} := \frac{w_j}{h_{j+1,j}}$ 

```

Théorème 5.3.10. Supposons que l'Algorithme 5.3.9 ne s'arrête pas avant l'étape m . Alors la famille de vecteurs $\{v_1, \dots, v_m\}$ forme une base orthonormée de $K_m(A, v)$.

Démonstration. Commençons par montrer que tous les v_j appartiennent à K_m . Plus précisément, on va montrer que, pour tout $j \in \llbracket 1, m \rrbracket$, il existe p_{j-1} un polynôme de degré $j-1$ tel que $v_j = p_{j-1}(A)v$. On procède par récurrence. Pour $j=1$, c'est vrai en considérant $p_0 \equiv \frac{1}{\|v\|}$. Soit donc $j \in \llbracket 1, m-1 \rrbracket$ tel que $v_i = p_{i-1}(A)v$ pour tout $i \leq j$. Alors :

$$w_j = Av_j - \sum_{i=1}^j h_{ij}v_i = Ap_{j-1}(A)v - \sum_{i=1}^j h_{ij}p_{i-1}(A)v.$$

Or, $Ap_{j-1}(A)$ est un polynôme en A de degré j et $\sum_{i=1}^j h_{ij}p_{i-1}(A)$ est un polynôme en A de degré $j-1$. On a donc trouvé un polynôme $\tilde{p}_j$ de degré j tel que $w_j = \tilde{p}_j(A)v$. On en déduit $v_{j+1} = p_j(A)v$ avec $p_j := \frac{\tilde{p}_j}{\|w_j\|}$ de degré j .

Par ailleurs, la famille $\{v_1, \dots, v_m\}$ est orthonormale par construction. En particulier, elle est libre et forme donc une base de K_m . ■

Proposition 5.3.11. Soit $j \in \llbracket 1, m \rrbracket$. Alors :

$$h_{j+1,j} = 0 \iff \deg(A, v) = j.$$

Démonstration. Dans un premier temps, supposons $\deg(A, v) = j$. Si $w_j \neq 0$, alors on peut définir $v_{j+1} \neq 0$, et ainsi $\dim K_{j+1} = j+1$. Or, comme $\deg(A, v) = j$, $K_{j+1} = K_j$, donc $\dim K_{j+1} = \dim K_j = j$: on obtient une contradiction, donc $w_j = 0$, d'où $h_{j+1,j} = 0$ et l'Algorithme 5.3.9 est arrêté.

Réciproquement, supposons $h_{j+1,j} = 0$, c'est-à-dire $w_j = 0$. Or, $w_j = \tilde{p}_{j-1}(A)v$ avec $\tilde{p}_j$ un polynôme de degré j . Donc $w_j = 0$ implique $\deg(A, v) \leq j$. Or, si $\mu := \deg(A, v) < j$, par la première partie de la preuve, on aurait $h_{\mu+1,\mu} = 0$ et l'algorithme se serait arrêté à l'étape μ . On en déduit $\deg(A, v) = j$. ■

Dans la suite, on note $V_m \in M_{n \times m}(\mathbb{R})$ la matrice dont les colonnes sont les vecteurs v_j construits par la méthode d'Arnoldi :

$$\forall (i, j) \in \llbracket 1, n \rrbracket \times \llbracket 1, m \rrbracket, \quad (V_m)_{ij} = (v_j)_i.$$

On note également $\bar{H}_m \in M_{(m+1) \times m}(\mathbb{R})$ la matrice définie dans l'Algorithme 5.3.9 (les coefficients n'étant pas explicitement définis étant nuls). Il s'agit d'une matrice de Hessenberg. Enfin, on note $H_m \in M_m(\mathbb{R})$ la matrice obtenue en retirant la dernière ligne de $\bar{H}_m$.

Proposition 5.3.12. Les matrices précédentes vérifient les relations suivantes :

$$\begin{aligned} AV_m &= V_m H_m + w_m e_m^T, \\ AV_m &= V_{m+1} \bar{H}_m, \\ V_m^T AV_m &= H_m. \end{aligned}$$

Démonstration. La j -ème colonne du produit AV_m est donnée par Av_j . La définition de w_j dans l'Algorithme 5.3.9 peut s'écrire :

$$w_j = Av_j - \sum_{i=1}^j h_{ij} v_i,$$

ce qui permet d'obtenir, via la définition de v_{j+1} :

$$Av_j = \sum_{i=1}^j h_{ij} v_i + w_j = \sum_{i=1}^j h_{ij} v_i + h_{j+1,j} v_{j+1} = \sum_{i=1}^{j+1} h_{ij} v_i.$$

On reconnaît alors la j -ème colonne de $V_{m+1} \bar{H}_m$, donc $AV_m = V_{m+1} \bar{H}_m$. On écrit alors cette égalité par blocs :

$$AV_m = \begin{pmatrix} V_m & v_{m+1} \end{pmatrix} \begin{pmatrix} H_m \\ h \end{pmatrix},$$

avec $h := (0, \dots, 0, h_{m+1,m}) = h_{m+1,m} e_m^T$. Le produit par blocs donne ainsi :

$$AV_m = V_m H_m + v_{m+1} h_{m+1,m} e_m^T$$

et la première égalité voulue s'ensuit alors en reprenant la définition de v_{m+1} . Multiplions cette relation par V_m^T :

$$V_m^T AV_m = V_m^T V_m H_m + V_m^T w_m e_m^T.$$

L'orthonormalité des v_j implique $V_m^T V_m = I$ et $V_m^T w_m = 0$, d'où la troisième et dernière relation. ■

La méthode d'Arnoldi telle que décrite dans l'[Algorithme 5.3.9](#) utilise le procédé d'orthonormalisation de Gram-Schmidt classique. Celui-ci étant instable, en pratique, on utilisera plutôt la méthode de Gram-Schmidt modifiée, ce qui donne l'[Algorithme 5.3.13](#).

Algorithme 5.3.13 : Méthode d'Arnoldi modifiée

Entrée : $v \in \mathbb{R}^n \setminus \{0\}$, $m \in \mathbb{N}^*$

Sortie : Une base orthonormée $(v_j)_j$ de K_m et une matrice H

```

 $v_1 := \frac{v}{\|v\|}$ 
pour  $j$  de 1 à  $m$ 
     $w_j := Av_j$ 
    pour  $i$  de 1 à  $j$ 
         $h_{ij} := (w_j, v_i)$ 
         $w_j := w_j - h_{ij}v_i$ 
     $h_{j+1,j} = \|w_j\|$ 
    si  $h_{j+1,j} = 0$ 
        Arrêt
     $v_{j+1} := \frac{w_j}{h_{j+1,j}}$ 

```

5.3.4 Full Orthogonalization Method (FOM)

Reprenons notre système $Ax = b$ et considérons une donnée initiale $x^{(0)} \in \mathbb{R}^n$. On note le premier résidu

$$r^{(0)} := b - Ax^{(0)}.$$

Comme indiqué dans l'introduction de cette section, on peut montrer que

$$x \in x^{(0)} + K_n(A, r^{(0)}).$$

L'idée est donc de rechercher une approximation $x^{(m)}$ de x dans $x^{(0)} + K_m(A, r^{(0)})$ pour un certain $m \in \llbracket 1, n \rrbracket$. Plus précisément, on va rechercher $x^{(m)} \in x^{(0)} + K_m$ tel que le résidu $r^{(m)} := b - Ax^{(m)}$ est orthogonal à K_m . Pour cela, on va construire une base $\{v_1, \dots, v_m\}$ de K_m via la méthode d'Arnoldi, ce qui permet notamment de définir les matrices V_m et H_m . Comme $x^{(m)} - x^{(0)} \in K_m$, on trouve $y^{(m)} \in \mathbb{R}^m$ tel que :

$$x^{(m)} = x^{(0)} + V_m y^{(m)}.$$

Ainsi :

$$r^{(m)} = b - Ax^{(m)} = r^{(0)} - AV_m y^{(m)}.$$

On cherche à avoir $r^{(m)}$ orthogonal à K_m , ce qui peut se traduire par :

$$V_m^T r^{(m)} = 0.$$

Autrement dit :

$$V_m^T r^{(0)} - V_m^T A V_m y^{(m)} = 0.$$

Or, $V_m^T A V_m = H_m$, donc on obtient :

$$H_m y^{(m)} = V_m^T r^{(0)}.$$

On peut aller un peu plus loin. L'algorithme d'Arnoldi commence par définir $v_1 := \frac{r^{(0)}}{\|r^{(0)}\|}$. Ainsi $r^{(0)} = \|r^{(0)}\| v_1$ et $V_m^T r^{(0)} = \|r^{(0)}\| V_m^T v_1$. Par orthonormalité de $\{v_1, \dots, v_m\}$, on trouve $V_m^T v_1 = e_1$ le premier vecteur de la base canonique de $\mathbb{R}^n$. Ainsi :

$$H_m y^{(m)} = \|r^{(0)}\| e_1.$$

On a alors tous les ingrédients pour définir l'[Algorithme 5.3.14](#).

Algorithme 5.3.14 : Full Orthogonalization Method (FOM)

Entrée : $A \in M_n(\mathbb{R})$ inversible, $b \in \mathbb{R}^n$, $x^{(0)} \in \mathbb{R}^n$, $m \in \mathbb{N}^*$

Sortie : $x^{(m)}$ une approximation de la solution de $Ax = b$ dans $x^{(0)} + K_m(A, r^{(0)})$

$$r^{(0)} := b - Ax^{(0)}$$

$$v_1 := \frac{r^{(0)}}{\|r^{(0)}\|}$$

pour j de 1 à m

$$w_j := A v_j$$

pour i de 1 à j

$$h_{ij} := (w_j, v_i)$$

$$w_j := w_j - h_{ij} v_i$$

$$h_{j+1,j} := \|w_j\|$$

si $h_{j+1,j} = 0$

└ Sortir de la boucle

$$v_{j+1} := \frac{w_j}{h_{j+1,j}}$$

Résoudre le système $H_m y^{(m)} = \|r^{(0)}\| e_1$

$$x^{(m)} := x^{(0)} + V_m y^{(m)}$$

Proposition 5.3.15. Le résidu obtenu par FOM est donné par :

$$r^{(m)} := b - Ax^{(m)} = -h_{m+1,m} y_m^{(m)} v_{m+1},$$

dont la norme est donnée par :

$$\|r^{(m)}\| = h_{m+1,m} |y_m^{(m)}|.$$

Démonstration. Revenant à la définition de $x^{(m)}$, on a :

$$r^{(m)} = b - Ax^{(m)} = r^{(0)} - AV_m y^{(m)}.$$

Or, $AV_m = V_m H_m + w_m e_m^T$, ainsi que $r^{(0)} = \|r^{(0)}\|v_1$, d'où :

$$r^{(m)} = \|r^{(0)}\|v_1 - V_m H_m y^{(m)} - w_m e_m^T y^{(m)}.$$

Par définition de $y^{(m)}$, $H_m y^{(m)} = \|r^{(0)}\|e_1$. Comme $V_m e_1 = v_1$, on en déduit :

$$\|r^{(0)}\|v_1 - V_m H_m y^{(m)} = 0.$$

Par ailleurs, $e_m^T y^{(m)} = y_m^{(m)}$. Par définition de v_{m+1} , on a $w_m = h_{m+1,m} v_{m+1}$, ce qui donne bien

$$r^{(m)} := b - Ax^{(m)} = -h_{m+1,m} y_m^{(m)} v_{m+1}.$$

Comme $h_{m+1,m}$ et $y_m^{(m)}$ sont des scalaires, on obtient l'expression de la norme de $r^{(m)}$ en utilisant $\|v_{m+1}\| = 1$. ■

Cette estimation du résidu peut a priori sembler assez peu utile : à première vue, on n'a effectivement plus besoin de calculer $x^{(m)}$, mais on a toujours besoin de $y^{(m)}$, et c'est justement l'étape la plus coûteuse dans le calcul du nouvel itéré puisqu'elle consiste en la résolution d'un système linéaire. On peut cependant légèrement modifier l'algorithme de la FOM pour ne plus seulement calculer la matrice H mais également sa triangularisation, qui peut être mise à jour relativement facilement à mesure que de nouvelles colonnes sont ajoutées. On obtient ainsi, sans trop de peine, un système triangulaire à chaque étape, ce qui permet de calculer plus rapidement $y^{(m)}$. Mais cette estimation du résidu nous indique que tout $y^{(m)}$ n'est pas nécessaire : seule sa dernière composante est utilisée. C'est là que les choses deviennent intéressantes : le système étant triangulaire supérieur, c'est justement la dernière composante qui sera calculée avant toute autre. Autrement dit, pour estimer le résidu, on a juste besoin de mettre à jour la triangularisation du système et d'effectuer une simple division pour calculer $y_m^{(m)}$. L'idée est alors de réaliser ce calcul à la fin de chaque étape j pour savoir si on s'arrête, ou si on continue jusqu'à m .

Proposition 5.3.16. Pour tout $j \in \llbracket 1, m \rrbracket$, le calcul du j -ème vecteur orthogonal dans la FOM nécessite :

- $2n^2 + 2(2j + 1)n - j$ opérations si A est pleine,
- $2N_{\text{nz}} + 2(2j + 1)n - j$ opérations si A est creuse.

Le calcul de m vecteurs nécessite alors :

- $2mn(n + m + 2) - \frac{m(m+1)}{2}$ opérations si A est pleine,
- $2m(N_{\text{nz}} + (m + 2)n) - \frac{m(m+1)}{2}$ opérations si A est creuse.

Démonstration. Soit $j \in \llbracket 1, m \rrbracket$. Le vecteur w_j est initialisé à Av_j : ce calcul nécessitera plus ou moins d'opérations selon la structure de A , et selon sa méthode de stockage. Dans

le cas le plus général où A est pleine, on aura besoin de n^2 multiplications et $n(n-1)$ additions. Si A est creuse, on aura besoin de N_{nz} multiplications et $N_{\text{nz}} - n$ additions.

Ensuite, pour i variant de 1 à j , on calcule un produit scalaire : n multiplications et $n-1$ additions. On fait ensuite le calcul $w_j - h_{ij}v_i$, soit une multiplication et une addition/soustraction pour chacune des n composantes.

Une fois la boucle sur i terminée, on calcule la norme du vecteur w_j : c'est un produit scalaire, donc n multiplications et $n-1$ additions, suivi d'un calcul de racine carrée. Reste la définition de v_{j+1} qui nécessite une division sur chacune des n composantes.

On somme le tout pour obtenir :

- $n^2 + (2j+1)n$ ou $N_{\text{nz}} + (2j+1)n$ multiplications,
- $n^2 + 2jn - (j+1)$ ou $N_{\text{nz}} + 2jn - (j+1)$ additions,
- n divisions,
- 1 racine carrée.

On somme le tout pour obtenir le résultat, et le nombre d'opérations pour le calcul de m vecteurs s'obtient en sommant sur j variant de 1 à m . ■

Si A est creuse avec $N_{\text{nz}} = \mathcal{O}(n)$, le nombre d'opérations nécessaires à l'orthogonalisation est alors de l'ordre de $\mathcal{O}(m^2n)$ (en supposant, comme c'est le cas en pratique, que m est petit par rapport à n). En ce qui concerne le coût de stockage, si A est par exemple stockée au format CSR, on a besoin de $2N_{\text{nz}} + n + 1$ emplacements. Le stockage de V_m nécessite nm places, et on a besoin de stocker w_j , donc n emplacements. Concernant H_m , il peut être bon de remarquer qu'un stockage CSR ne serait pas avantageux en terme de mémoire : à moins de disposer d'un format bien spécifique, le stockage plein reste la meilleure option, ce qui nécessite $m(m+1)$ emplacements. En oubliant les éventuelles variables temporaires utilisées pendant les calculs, on obtient $2N_{\text{nz}} + n(m+2) + m(m+1) + 1$ places mémoire. Encore une fois, si $N_{\text{nz}} = \mathcal{O}(n)$ et m petit par rapport à n , le coût mémoire du calcul de m vecteurs orthogonaux est grossièrement en $\mathcal{O}(nm)$.

Comme $x \in x^{(0)} + K_n$, on s'attend à avoir une meilleure approximation de la solution en prenant m grand. En pratique, les coûts de calcul et de mémoire sont limitants. Une option est de « redémarrer » la méthode : une fois $x^{(m)}$ calculé, on recommence en considérant cette approximation comme nouvelle donnée initiale. On obtient l'[Algorithme 5.3.17](#).

Algorithme 5.3.17 : Restarted FOM (FOM(m))

Entrée : $A \in M_n(\mathbb{R})$ inversible, $b \in \mathbb{R}^n$, $x^{(0)} \in \mathbb{R}^n$, $m \in \mathbb{N}^*$

tant que *non convergence*

┌ Appliquer FOM avec $x^{(0)}$ pour obtenir $x^{(m)}$
└ $x^{(0)} := x^{(m)}$

5.3.5 Méthode Generalized Minimum Residual (GMRES)

L'approche de GMRES est assez proche de la FOM, dans le sens où on recherche toujours une approximation dans $x^{(0)} + K_m$. La différence est au niveau de la projection : plutôt que de projeter sur K_m , on recherche la projection sur AK_m .

On considère une donnée initiale $x^{(0)} \in \mathbb{R}^n$ dans le but de résoudre le système linéaire $Ax = b$. On note toujours le premier résidu $r^{(0)} := b - Ax^{(0)}$. Comme dans le cas de la FOM, on va rechercher $x^{(m)} \in x^{(0)} + K_m(A, r^{(0)})$. Cela revient à rechercher un vecteur $y^{(m)} \in \mathbb{R}^m$ tel que $x^{(m)} = x^{(0)} + V_m y^{(m)}$ où V_m est la matrice d'une base orthonormale de K_m . L'idée de GMRES est de minimiser la norme du résidu :

$$\|b - Ax^{(m)}\|.$$

Or :

$$\begin{aligned} b - Ax^{(m)} &= b - A(x^{(0)} + V_m y^{(m)}) \\ &= r^{(0)} - AV_m y^{(m)} \\ &= r^{(0)} - V_{m+1} \bar{H}_m y^{(m)}. \end{aligned}$$

En calculant la base de K_m par la méthode d'Arnoldi, on définit $v_1 = \frac{r^{(0)}}{\|r^{(0)}\|}$, d'où $r^{(0)} = \|r^{(0)}\|v_1$. Par ailleurs, $v_1 = V_{m+1}e_1$ avec e_1 le premier vecteur de la base canonique de $\mathbb{R}^{m+1}$. On obtient ainsi :

$$\begin{aligned} b - Ax^{(m)} &= \|r^{(0)}\|V_{m+1}e_1 - V_{m+1}\bar{H}_m y^{(m)} \\ &= V_{m+1} \left(\|r^{(0)}\|e_1 - \bar{H}_m y^{(m)} \right). \end{aligned}$$

Or, V_{m+1} est orthonormale, donc :

$$\|b - Ax^{(m)}\| = \left\| V_{m+1} \left(\|r^{(0)}\|e_1 - \bar{H}_m y^{(m)} \right) \right\| = \left\| \|r^{(0)}\|e_1 - \bar{H}_m y^{(m)} \right\|.$$

Pour obtenir l'approximation $x^{(m)}$, l'idée est donc de minimiser $y^{(m)} \mapsto \left\| \|r^{(0)}\|e_1 - \bar{H}_m y^{(m)} \right\|$ pour obtenir $y^{(m)}$ et ensuite de définir $x^{(m)} = x^{(0)} + V_m y^{(m)}$. On obtient l'[Algorithme 5.3.18](#).

On peut tout de suite remarquer que, étant donnée la définition de $y^{(m)}$, cette méthode est optimale dans le sens où le résidu sera toujours le plus petit possible (dans l'espace considéré) : $y^{(m)}$ est le vecteur qui minimise ledit résidu. Il reste bien sûr à répondre à une question importante : en pratique, comment calculer ce minimiseur ? C'est l'objet du résultat suivant.

Proposition 5.3.19. Soit $Q \in M_{m+1}(\mathbb{R})$ la matrice de rotation telle que $\bar{R} := Q\bar{H}_m \in M_{(m+1) \times m}(\mathbb{R})$ est triangulaire supérieure. Soit $\bar{g} := Q(\|r^{(0)}\|e_1) \in \mathbb{R}^{m+1}$. Soient $R \in M_m(\mathbb{R})$ la matrice obtenue en supprimant la dernière ligne de $\bar{R}$, et $g \in \mathbb{R}^m$ le vecteur des m premières composantes de $\bar{g}$. Alors :

1. R est inversible ;
2. le minimiseur de $y \mapsto \left\| \|r^{(0)}\|e_1 - \bar{H}_m y \right\|$ est l'unique solution du système triangulaire $Ry = g$;

Algorithme 5.3.18 : GMRES**Entrée :** $A \in M_n(\mathbb{R})$ inversible, $b \in \mathbb{R}^n$, $x^{(0)} \in \mathbb{R}^n$, $m \in \mathbb{N}^*$ **Sortie :** $x^{(m)}$ une approximation de la solution de $Ax = b$ dans $x^{(0)} + K_m(A, r^{(0)})$

$$r^{(0)} := b - Ax^{(0)}$$

$$v_1 := \frac{r^{(0)}}{\|r^{(0)}\|}$$

pour j de 1 à m

$$w_j := Av_j$$

pour i de 1 à j

$$h_{ij} := (w_j, v_i)$$

$$w_j := w_j - h_{ij}v_i$$

$$h_{j+1,j} := \|w_j\|$$

si $h_{j+1,j} = 0$

Sortir de la boucle

$$v_{j+1} := \frac{w_j}{h_{j+1,j}}$$

Calculer le minimiseur $y^{(m)}$ de $y \mapsto \|\|r^{(0)}\|e_1 - \bar{H}_m y\|$

$$x^{(m)} := x^{(0)} + V_m y^{(m)}$$

3. le résidu est donné par :

$$b - Ax^{(m)} = V_{m+1}Q^T(g_{m+1}e_{m+1})$$

et ainsi :

$$\|b - Ax^{(m)}\| = |g_{m+1}|.$$

Démonstration. Matrice inversible. Partons de l'égalité :

$$AV_m = V_{m+1}\bar{H}_m.$$

Comme $Q^T Q = I$, on peut écrire :

$$AV_m = V_{m+1}Q^T Q \bar{H}_m = V_{m+1}Q^T \bar{R}$$

par définition de $\bar{R}$. La matrice A étant supposée inversible, elle est en particulier de rang maximal. Le rang de AV_m est donc le même que celui de V_m , lui-même égal à m par définition. De la même façon, le rang de $V_{m+1}Q^T \bar{R}$ est égal au rang de $\bar{R}$. Or, $\bar{R}$ n'est autre que la concaténation de R avec un vecteur nul, donc le rang de $\bar{R}$ n'est autre que le rang de R . Autrement dit, on a montré que R était de rang m : elle est de rang maximal, et donc inversible.

Minimiseur explicite. On utilise les propriétés de Q :

$$\|\bar{g} - \bar{R}y\| = \|Q(\|r^0\|e_1 - \bar{H}_m y)\| = \|\|r^0\|e_1 - \bar{H}_m y\|.$$

Or, comme $\bar{R}$ est triangulaire supérieure, sa dernière ligne est uniquement composée de zéros. Ainsi :

$$\bar{g} - \bar{R}y = \begin{pmatrix} g \\ g_{m+1} \end{pmatrix} - \begin{pmatrix} R \\ 0 \end{pmatrix} y = \begin{pmatrix} g - Ry \\ g_{m+1} \end{pmatrix},$$

d'où :

$$\|\bar{g} - \bar{R}y\|^2 = \|g - Ry\|^2 + |g_{m+1}|^2.$$

C'est une somme de réels positifs : son minimiseur est donc le minimiseur des deux quantités. Or, $|g_{m+1}|$ est constante par rapport à y , donc le minimiseur de $\|\bar{g} - \bar{R}y\|$ est le minimiseur de $\|g - Ry\|$. Comme R est inversible par le premier point, il s'agit de l'unique solution du système $Ry = g$.

Résidu. On revient à l'égalité :

$$b - Ax^{(m)} = V_{m+1}(\|r^{(0)}\|e_1 - \bar{H}_m y^{(m)}).$$

Comme $Q^T Q = I$, on peut toujours écrire :

$$b - Ax^{(m)} = V_{m+1} Q^T Q (\|r^{(0)}\|e_1 - \bar{H}_m y^{(m)}).$$

En développant par rapport à Q , on obtient :

$$b - Ax^{(m)} = V_{m+1} Q^T (\bar{g} - \bar{R}y^{(m)}).$$

Or $y^{(m)}$ est l'unique solution de $Ry^{(m)} = g$, ce qui revient à avoir les m premières composantes de $\bar{g} - \bar{R}y^{(m)}$ égales à zéro. Ainsi :

$$b - Ax^{(m)} = V_{m+1} Q^T (g_{m+1} e_{m+1}).$$

L'estimation de la norme s'ensuit alors en utilisant l'orthonormalité de V_{m+1} et de Q . ■

Tout comme pour la FOM, on pourra donc écrire des variations de l'[Algorithme 5.3.18](#) exploitant ce résultat pour tester régulièrement si le résidu a une norme acceptable et ainsi décider de continuer ou non le processus, sans avoir à calculer $x^{(m)}$ à chaque fois.

5.3.6 Préconditionnement

Comme pour toutes les méthodes itératives vues jusque-là, la convergence de la FOM et de GMRES dépend du conditionnement de A . Pour espérer accélérer la convergence, on peut donc chercher à preconditionner le système de la même manière que pour le gradient conjugué :

$$Ax = b \iff C^{-1}Ax = C^{-1}b.$$

Contrairement au gradient conjugué, en revanche, on n'a pas besoin que la matrice du système soit symétrique définie positive : on peut donc s'autoriser, a priori, n'importe quelle matrice inversible. Bien sûr, en pratique, on se donne quelques contraintes. La première est évidente : il faut que le conditionnement de $C^{-1}A$ soit meilleur que celui de A pour justifier l'augmentation du coût d'une itération.

C'est justement de là que vient la seconde contrainte. En écrivant $C^{-1}A$, on obtient sur le papier une simple matrice, et il suffit de dérouler les algorithmes avec cette nouvelle matrice sans se préoccuper du reste. On pourrait le faire, mais il faudrait pour cela calculer C^{-1} , ce qu'il vaut mieux éviter en général (aussi bien pour des raisons de coût de calcul que pour des raisons de stockage). En regardant d'un peu plus près les algorithmes définis précédemment, la matrice du système est utilisée pour calculer la base de l'espace de Krylov : partant d'un vecteur v , on multiplie par la matrice pour obtenir un autre vecteur w . Avec un système préconditionné, on aura donc :

$$w := C^{-1}Av.$$

L'idée est donc de réaliser ce calcul en deux étapes principales. D'abord, on calcule $u := Av$, puis on résout le système $Cw = u$. Cela évite d'avoir à stocker C^{-1} . On en vient alors à la contrainte qu'il est bon d'imposer à C : les systèmes qu'elle définit doivent être « faciles à résoudre ».

5.3.7 Factorisation LU incomplète

Il y a bien sûr un choix aussi évident qu'absurde pour préconditionner le système : prendre $C = A$. Ce choix est évident, parce qu'il donne un système défini par l'identité, donc il n'y a rien à faire pour le résoudre. Mais il est tout aussi absurde : si A est si facile à inverser, pourquoi passer par une méthode itérative ?

Le système $Cw = u$ peut être résolu facilement si on connaît une décomposition de C sous la forme d'un produit de deux matrices triangulaires, par exemple. Restons donc encore un peu dans un cadre idyllique : si on a accès à la décomposition LU de A , le choix évident et absurde précédent devient $C = A = LU$. Donc C est bien facile à inverser... mais autant résoudre directement le système $LUx = b$!

Toutes ces réflexions mènent finalement à une question : pourquoi refuse-t-on de calculer la factorisation LU d'une matrice ? Le coût de calcul est une chose, bien sûr, mais le plus gros obstacle se situe plutôt du côté du stockage : les matrices L et U ne sont jamais aussi creuses que A . On peut améliorer un peu les choses avec une factorisation PLU, mais il n'y aura pas de miracle : on devra toujours stocker (beaucoup) plus de coefficients que pour A .

Il vient alors une idée : peut-on construire une approximation de la factorisation LU, moins précise, mais plus creuse ? C'est le principe de la factorisation LU incomplète qui consiste à construire $\tilde{L}$ et $\tilde{U}$ telles que :

$$A = \tilde{L}\tilde{U} + R,$$

où est une matrice de « reste ». On prendra alors $C = \tilde{L}\tilde{U}$, ce qui définira bien un système facile à inverser par remontée/descente.

Il existe plusieurs variantes de la factorisation LU incomplète (aussi appelée ILU, pour « Incomplete LU »). Celle que l'on se propose d'introduire ici se base sur une approche de calcul de la factorisation LU légèrement différente que celles vues jusque-là, décrite dans l'[Algorithme 5.3.20](#).

Algorithme 5.3.20 : Factorisation LU par ligne**Entrée :** $A \in M_n(\mathbb{R})$ admettant une factorisation LU**Sortie :** $L \in M_n(\mathbb{R})$ triangulaire inférieure à diagonale unité et $U \in M_n(\mathbb{R})$ triangulaire supérieure telles que $A = LU$ $U := A$ $L := I$ **pour** i de 2 à n **pour** k de 1 à $i - 1$ $\alpha := \frac{U_{ik}}{U_{kk}}$ **pour** j de k à n $U_{ij} := U_{ij} - \alpha U_{kj}$ $L_{ik} := \alpha$

L'idée derrière cette approche est d'appliquer le pivot de Gauss pour construire U , et de stocker les coefficients des matrices de transvection utilisée pour cela. Plus exactement, on stocke les opposés de ces coefficients, pour la simple et bonne raison que c'est l'inverse de ces matrices qui nous intéresse, et pas les transvections elles-mêmes. Ici, plutôt que de créer les zéros colonne par colonne comme on le faisait dans la version initiale, on les crée ligne par ligne.

Exemple 5.3.21. Déroulons l'algorithme sur un exemple simple. On considère $A \in M_3(\mathbb{R})$. Ainsi :

$$U^{(0)} = \begin{pmatrix} U_{11}^{(0)} & U_{12}^{(0)} & U_{13}^{(0)} \\ U_{21}^{(0)} & U_{22}^{(0)} & U_{23}^{(0)} \\ U_{31}^{(0)} & U_{32}^{(0)} & U_{33}^{(0)} \end{pmatrix}.$$

La première étape consiste à annuler les premiers coefficients de la seconde ligne, jusqu'au coefficient diagonal (exclus). Matriciellement, on peut écrire quelque chose comme :

$$\begin{pmatrix} 1 & 0 & 0 \\ -\frac{U_{21}^{(0)}}{U_{11}^{(0)}} & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} U_{11}^{(0)} & U_{12}^{(0)} & U_{13}^{(0)} \\ U_{21}^{(0)} & U_{22}^{(0)} & U_{23}^{(0)} \\ U_{31}^{(0)} & U_{32}^{(0)} & U_{33}^{(0)} \end{pmatrix} = \begin{pmatrix} U_{11}^{(0)} & U_{12}^{(0)} & U_{13}^{(0)} \\ 0 & U_{22}^{(1)} & U_{23}^{(1)} \\ U_{31}^{(0)} & U_{32}^{(0)} & U_{33}^{(0)} \end{pmatrix}.$$

Attaquons la troisième ligne. On a cette fois-ci deux coefficients à annuler : le premier,

et le deuxième. On obtient :

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -\frac{U_{31}^{(0)}}{U_{11}^{(0)}} & 0 & 1 \end{pmatrix} \begin{pmatrix} U_{11}^{(0)} & U_{12}^{(0)} & U_{13}^{(0)} \\ 0 & U_{22}^{(1)} & U_{23}^{(1)} \\ U_{31}^{(0)} & U_{32}^{(0)} & U_{33}^{(0)} \end{pmatrix} = \begin{pmatrix} U_{11}^{(0)} & U_{12}^{(0)} & U_{13}^{(0)} \\ 0 & U_{22}^{(1)} & U_{23}^{(1)} \\ 0 & U_{32}^{(1)} & U_{33}^{(1)} \end{pmatrix},$$

puis :

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -\frac{U_{32}^{(1)}}{U_{22}^{(1)}} & 1 \end{pmatrix} \begin{pmatrix} U_{11}^{(0)} & U_{12}^{(0)} & U_{13}^{(0)} \\ 0 & U_{22}^{(1)} & U_{23}^{(1)} \\ 0 & U_{32}^{(1)} & U_{33}^{(1)} \end{pmatrix} = \begin{pmatrix} U_{11}^{(0)} & U_{12}^{(0)} & U_{13}^{(0)} \\ 0 & U_{22}^{(1)} & U_{23}^{(1)} \\ 0 & 0 & U_{33}^{(2)} \end{pmatrix}.$$

À partir de ce simple exemple, on peut déjà remarquer le problème de remplissage. En effet, si $A_{23} = 0$, mais $A_{13} \neq 0$, alors on aura $U_{23} \neq 0$.

On va légèrement modifier cet algorithme pour imposer explicitement des zéros dans les matrices $\tilde{L}$ et $\tilde{U}$. Ici, on choisit d'imposer que ces nouvelles matrices aient la même « structure » que A : si $A_{ij} = 0$, alors $\tilde{L}_{ij} = \tilde{U}_{ij} = 0$. On obtient l'[Algorithme 5.3.22](#).

Algorithme 5.3.22 : ILU(0)

Entrée : $A \in M_n(\mathbb{R})$ admettant une factorisation LU

Sortie : $\tilde{L} \in M_n(\mathbb{R})$ triangulaire inférieure à diagonale unité et $\tilde{U} \in M_n(\mathbb{R})$ triangulaire supérieure telles que $A = \tilde{L}\tilde{U} + R$

$\tilde{U} := A$

$\tilde{L} := I$

pour i de 2 à n

pour k de 1 à $i - 1$

si $A_{ik} \neq 0$

$\alpha := \frac{\tilde{U}_{ik}}{\tilde{U}_{kk}}$

pour j de k à n

si $A_{ij} \neq 0$

$\tilde{U}_{ij} := \tilde{U}_{ij} - \alpha \tilde{U}_{kj}$

$\tilde{L}_{ik} := \alpha$

Exemple 5.3.23. Construisons la factorisation LU incomplète d'une matrice $A \in M_4(\mathbb{R})$ tridiagonale :

$$A := \begin{pmatrix} A_{11} & 0 & A_{13} & 0 \\ A_{21} & A_{22} & 0 & A_{24} \\ 0 & A_{32} & A_{33} & 0 \\ 0 & 0 & A_{43} & A_{44} \end{pmatrix}.$$

Première étape : on doit supprimer le premier élément de la deuxième ligne. Ceci est équivalent à multiplier par une certaine matrice de transvection, qui va ainsi créer un coefficient non nul en position (2, 3). Pour garder un zéro dans cette position, on le « déplace » sur une matrice de reste.

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ -\frac{U_{21}^{(0)}}{U_{11}^{(0)}} & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} U_{11}^{(0)} & 0 & U_{13}^{(0)} & 0 \\ U_{21}^{(0)} & U_{22}^{(0)} & 0 & U_{24}^{(0)} \\ 0 & U_{32}^{(0)} & U_{33}^{(0)} & 0 \\ 0 & 0 & U_{43}^{(0)} & U_{44}^{(0)} \end{pmatrix} \\ = \begin{pmatrix} U_{11}^{(0)} & 0 & U_{13}^{(0)} & 0 \\ 0 & U_{22}^{(1)} & 0 & U_{24}^{(1)} \\ 0 & U_{32}^{(0)} & U_{33}^{(0)} & 0 \\ 0 & 0 & U_{43}^{(0)} & U_{44}^{(0)} \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & R_{23} & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

En plus condensé :

$$T^{(1)}U^{(0)} = U^{(1)} + R^{(1)}.$$

Continuons. Comme le premier coefficient de la troisième ligne est déjà nul, passons au second. Là encore, comme $U_{24}^{(1)} \neq 0$, on va créer un coefficient non nul en position (3, 4), qu'il faut donc déplacer.

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -\frac{U_{32}^{(0)}}{U_{22}^{(1)}} & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} U_{11}^{(0)} & 0 & U_{13}^{(0)} & 0 \\ 0 & U_{22}^{(1)} & 0 & U_{24}^{(1)} \\ 0 & U_{32}^{(0)} & U_{33}^{(0)} & 0 \\ 0 & 0 & U_{43}^{(0)} & U_{44}^{(0)} \end{pmatrix} \\ = \begin{pmatrix} U_{11}^{(0)} & 0 & U_{13}^{(0)} & 0 \\ 0 & U_{22}^{(1)} & 0 & U_{24}^{(1)} \\ 0 & 0 & U_{33}^{(2)} & 0 \\ 0 & 0 & U_{43}^{(0)} & U_{44}^{(0)} \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & R_{34} \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

On obtient :

$$T^{(2)}U^{(1)} = U^{(2)} + R^{(2)}.$$

Troisième et dernière étape : il reste l'avant-dernier coefficient de la dernière ligne.

Cette transformation ne va pas créer de reste.

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -\frac{U_{43}^{(0)}}{U_{33}^{(2)}} & 1 \end{pmatrix} \begin{pmatrix} U_{11}^{(0)} & 0 & U_{13}^{(0)} & 0 \\ 0 & U_{22}^{(1)} & 0 & U_{24}^{(1)} \\ 0 & 0 & U_{33}^{(2)} & 0 \\ 0 & 0 & U_{43}^{(0)} & U_{44}^{(0)} \end{pmatrix} \\ = \begin{pmatrix} U_{11}^{(0)} & 0 & U_{13}^{(0)} & 0 \\ 0 & U_{22}^{(1)} & 0 & U_{24}^{(1)} \\ 0 & 0 & U_{33}^{(2)} & 0 \\ 0 & 0 & 0 & U_{44}^{(3)} \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix},$$

soit :

$$T^{(3)}U^{(2)} = U^{(3)} + R^{(3)}.$$

Inversons la machine. En notant $L^{(i)} = (T^{(i)})^{-1}$, on a :

$$U^{(2)} = L^{(3)}U^{(3)} + L^{(3)}R^{(3)}.$$

Or, $L^{(3)}$ réalise une combinaison linéaire entre des lignes, là où $R^{(3)}$ a au plus un élément non nul. Donc $L^{(3)} = R^{(3)}$. Donc :

$$U^{(2)} = L^{(3)}U^{(3)} + R^{(3)}.$$

Continuons :

$$\begin{aligned} U^{(1)} &= L^{(2)}U^{(2)} + \underbrace{L^{(2)}R^{(2)}}_{=R^{(2)}} \\ &= L^{(2)}L^{(3)}U^{(3)} + R^{(3)} + R^{(2)}, \end{aligned}$$

et finalement :

$$A = U^{(0)} = L^{(1)}L^{(2)}L^{(3)}U^{(3)} + R^{(3)} + R^{(2)} + R^{(1)}.$$

En notant $\tilde{L} := L^{(1)}L^{(2)}L^{(3)}$, $\tilde{U} := U^{(3)}$ et $R := R^{(3)} + R^{(2)} + R^{(1)}$, on a donc bien $A = \tilde{L}\tilde{U} + R$.

La méthode décrite dans l'[Algorithme 5.3.22](#) ne s'appuie que sur la structure de A . Il est possible d'autoriser plus ou moins de remplissage dans les matrices $\tilde{L}$ et $\tilde{U}$, dans l'objectif de se rapprocher de L et U . Ce sont les méthodes ILU(p), pour un certain $p \in \mathbb{N}$. On peut également utiliser d'autres critères, comme comparer la valeur d'un coefficient par rapport à un certain seuil : ce sont les méthodes ILUT.

Annexe A

Opérations élémentaires

Dans ce chapitre, x et y désignent deux flottants en base b et en précision p , normalisés et positifs. On fixe dès maintenant leur décomposition :

$$x = b^{e_x} \underbrace{\sum_{n=0}^p \frac{d_n^x}{b^n}}_{=:s_x} \quad \text{et} \quad y = b^{e_y} \underbrace{\sum_{n=0}^p \frac{d_n^y}{b^n}}_{=:s_y},$$

avec $e_x \in \mathbb{Z}$ et $e_y \in \mathbb{Z}$ les exposants respectifs de x et y (sans restriction), et $s_x \in [1, b[$ et $s_y \in [1, b[$ leurs significandes. Rappelons les définitions des suites introduites dans la preuve du [Théorème 1.1.1](#) :

$$a_n^x := \frac{\lfloor b^n s_x \rfloor}{b^n} \quad \text{et} \quad d_n^x := \lfloor b^n s_x \rfloor - b^n a_{n-1}^x,$$

représentant la troncature et la suite des digits de s_x . On rappelle la convention $a_{-1}^x := 0$. On définit de manière similaire les suites associées à s_y .

A.1 Ordre et digits

Commençons par formaliser quelques résultats intuitifs. Le résultat suivant démontre que deux flottants sont égaux si et seulement si leurs exposants et leurs digits sont égaux.

Lemme A.1.1. Soient x et y deux flottants normalisés positifs. Alors :

$$x = y \iff (e_x = e_y \text{ et } \forall n \in \llbracket 0, p \rrbracket, d_n^x = d_n^y).$$

Démonstration. Supposons $x = y$. Alors $\log_b x = \log_b y$, d'où :

$$e_x + \log_b s_x = e_y + \log_b s_y.$$

Or, $s_x \in [1, b[$, donc $\log_b s_x \in [0, 1[$, et de même pour s_y . En prenant la partie entière de l'égalité précédente, on obtient alors :

$$e_x = e_y.$$

On en déduit directement l'égalité des significandes, qui définissent donc les mêmes suites $(a_n^x)_n$ et $(a_n^y)_n$, et donc les mêmes suites $(d_n^x)_n$ et $(d_n^y)_n$. La réciproque est évidente. ■

Le résultat suivant formalise une condition qui apparaît naturellement quand on compare des flottants : si deux flottants différents ont le même exposant, alors ils partageront les mêmes digits, jusqu'à ce que l'un diffère, suivant l'ordre des flottants.

Lemme A.1.2. Soient x et y deux flottants normalisés dans $[1, b[$ (i.e. $e_x = e_y = 0$). Alors :

$$x < y \iff (\exists n_0 \in \llbracket 0, p \rrbracket / d_{n_0}^x < d_{n_0}^y \text{ et } \forall n \in \llbracket 0, n_0 - 1 \rrbracket, d_n^x = d_n^y).$$

Démonstration. Supposons $x < y$. Comme $x \neq y$, il existe $n_0 \in \llbracket 0, p \rrbracket$ tel que $d_{n_0}^x \neq d_{n_0}^y$. Considérons n_0 minimal pour cette propriété, c'est-à-dire :

$$\forall n \in \llbracket 0, n_0 - 1 \rrbracket, \quad d_n^x = d_n^y.$$

On montre, similairement à ce qui est fait dans la preuve du [Théorème 1.1.1](#) :

$$\forall n \in \llbracket 0, n_0 - 1 \rrbracket, \quad a_n^x = \sum_{m=0}^n \frac{d_m^x}{b^m} = \sum_{m=0}^n \frac{d_m^y}{b^m} = a_n^y.$$

Comme $d_{n_0}^x \neq d_{n_0}^y$, on a directement $a_{n_0}^x \neq a_{n_0}^y$. Or, en utilisant $x < y$, on peut directement montrer :

$$\forall n \in \mathbb{N}, \quad a_n^x \leq a_n^y.$$

Ainsi : $d_{n_0}^x < d_{n_0}^y$. Or :

$$a_{n_0}^x = \sum_{n=0}^{n_0} \frac{d_n^x}{b^n} = \sum_{n=0}^{n_0-1} \frac{d_n^x}{b^n} + \frac{d_{n_0}^x}{b^{n_0}}.$$

Comme $d_n^x = d_n^y$ pour $n \in \llbracket 0, n_0 - 1 \rrbracket$, on écrit la même décomposition pour $a_{n_0}^y$ et on déduit de $a_{n_0}^x < a_{n_0}^y$ que $d_{n_0}^x < d_{n_0}^y$.

Réciproquement, supposons qu'il existe $n_0 \in \llbracket 0, p \rrbracket$ tel que $d_{n_0}^x < d_{n_0}^y$ et :

$$\forall n \in \llbracket 0, n_0 - 1 \rrbracket, \quad d_n^x = d_n^y.$$

Commençons par écrire :

$$x = \underbrace{\sum_{n=0}^{n_0-1} \frac{d_n^x}{b^n}}_{=:s} + \frac{d_{n_0}^x}{b^{n_0}} + \underbrace{\sum_{n=n_0+1}^p \frac{d_n^x}{b^n}}_{=:r_x},$$

et on décompose y de manière similaire (grâce à l'égalité des premiers digits) :

$$y = s + \frac{d_{n_0}^y}{b^{n_0}} + r_y.$$

Tous les digits sont dans $\llbracket 0, b-1 \rrbracket$ donc, en particulier :

$$0 \leq \sum_{n=n_0+1}^p \frac{d_n^x}{b^n} \leq \sum_{n=n_0+1}^p \frac{b-1}{b^n} = \frac{1}{b^{n_0}} - \frac{1}{b^p},$$

d'où l'on déduit :

$$0 \leq r_x < \frac{1}{b^{n_0}}.$$

On obtient bien sûr la même inégalité pour r_y . Par ailleurs, $d_{n_0}^x$ et $d_{n_0}^y$ sont entiers donc l'inégalité $d_{n_0}^x < d_{n_0}^y$ se réécrit $d_{n_0}^x + 1 \leq d_{n_0}^y$. Ainsi :

$$\frac{d_{n_0}^x}{b^{n_0}} + r_x < \frac{d_{n_0}^x + 1}{b^{n_0}} \leq \frac{d_{n_0}^y}{b^{n_0}} \leq \frac{d_{n_0}^y}{b^{n_0}} + r_y.$$

Finalement, on a obtenu :

$$x = s + \frac{d_{n_0}^x}{b^{n_0}} + r_x < s + \frac{d_{n_0}^y}{b^{n_0}} + r_y = y,$$

soit $x < y$. ■

Le prochain résultat concerne l'ordre entre deux flottants quelconques. L'inégalité sur les significandes pourra être traduite à l'aide du lemme précédent.

Lemme A.1.3. Soient x et y deux flottants normalisés positifs. Alors :

$$x < y \iff (e_x < e_y \text{ ou } (e_x = e_y \text{ et } s_x < s_y)).$$

Démonstration. Supposons $x < y$. Autrement dit :

$$b^{e_x} s_x < b^{e_y} s_y,$$

ou encore :

$$\frac{s_x}{s_y} < b^{e_y - e_x}.$$

Or $s_x \in [1, b[$, et de même pour s_y , donc :

$$\frac{1}{b} < \frac{s_x}{s_y} < b^{e_y - e_x},$$

d'où :

$$b^{e_y - e_x + 1} > 1,$$

c'est-à-dire :

$$e_y - e_x + 1 > 0.$$

Comme e_x et e_y sont entiers, ceci implique $e_x \leq e_y$. Si $e_x = e_y$, l'inégalité $b^{e_x} s_x < b^{e_y} s_y$ implique directement $s_x < s_y$.

Réciproquement, supposons $e_x < e_y$. Alors $e_x + 1 \leq e_y$. Comme $1 \leq s_x < b$ et de même pour s_y , on a :

$$x = b^{e_x} s_x < b^{e_x+1} \leq b^{e_y} \leq b^{e_y} s_y = y.$$

Supposons maintenant $e_x = e_y$ et $s_x < s_y$. Alors :

$$x = b^{e_x} s_x < b^{e_x} s_y = b^{e_y} s_y = y.$$

■

A.2 Addition

On s'intéresse à l'addition des flottants (toujours supposés positifs) x et y . Quitte à échanger leurs rôles, on supposera dans la suite $x \leq y$. Ainsi, $e_x \leq e_y$. La première étape consiste à faire en sorte que les deux flottants soient écrits avec le même exposant, le plus grand des deux. Dans ce but, on pose $\delta := e_y - e_x \geq 0$ et on écrit :

$$x = b^{e_x} \sum_{n=0}^p \frac{d_n^x}{b^n} = b^{e_y} b^{-\delta} \sum_{n=0}^p \frac{d_n^x}{b^n} = b^{e_y} \sum_{n=0}^p \frac{d_n^x}{b^{n+\delta}} = b^{e_y} \sum_{n=\delta}^{p+\delta} \frac{d_{n-\delta}^x}{b^n}.$$

Remarque A.2.1. En faisant cela, on perd l'écriture normalisée de x . Ce n'est que temporaire, le temps de réaliser l'opération.

Comme x et y partagent maintenant le même exposant, on peut écrire une première approche de l'addition :

$$z := x + y = b^{e_y} \left(\sum_{n=0}^{\delta-1} \frac{d_n^y}{b^n} + \sum_{n=\delta}^p \frac{d_{n-\delta}^x + d_n^y}{b^n} + \sum_{n=p+1}^{p+\delta} \frac{d_{n-\delta}^x}{b^n} \right).$$

On peut alors commencer par définir une première version de la décomposition de z :

$$\tilde{d}_n^z := \begin{cases} d_n^y, & \text{si } n \in \llbracket 0, \delta - 1 \rrbracket, \\ d_{n-\delta}^x + d_n^y, & \text{si } n \in \llbracket \delta, p \rrbracket, \\ d_{n-\delta}^x, & \text{si } n \in \llbracket p+1, p+\delta \rrbracket. \end{cases}$$

Pour $n \in \llbracket p+1, p+\delta \rrbracket$, $d_{n-\delta}^x \in \llbracket 0, b-1 \rrbracket$ donc il n'y a rien à faire : la définition est « légale ». En revanche :

$$\forall n \in \llbracket \delta, p \rrbracket, \quad 0 \leq d_{n-\delta}^x + d_n^y \leq 2b - 2.$$

Il peut donc y avoir un problème si l'un des $d_{n-\delta}^x + d_n^y$ est plus grand que b . Cependant, on peut remarquer :

$$b \leq \tilde{d}_n^z \leq 2b - 2 \implies 0 \leq \tilde{d}_n^z - b \leq b - 2.$$

L'idée est donc de transformer ce digit temporaire en un « vrai » digit. On ne peut cependant pas tout simplement retrancher b : cela changerait le résultat de l'opération. Il faut trouver une astuce :

$$\frac{\tilde{d}_n^z}{b^n} = \frac{\tilde{d}_n^z - b + b}{b^n} = \frac{\tilde{d}_n^z - b}{b^n} + \frac{1}{b^{n-1}}.$$

Si on retranche b au n -ème digit, il faut donc ajouter 1 au $n - 1$ -ème digit : c'est la retenue quand on pose une addition à l'école. Comme la propagation de la retenue se fait de n à $n - 1$, il faut traiter les digits un à un, de droite à gauche. Il est par ailleurs possible qu'elle parvienne jusqu'aux digits $n \in \llbracket 0, \delta - 1 \rrbracket$. C'est pourquoi, même si à première vue ces digits étaient dans le bon intervalle (puisque issus de ceux de y seulement), il est nécessaire de vérifier s'ils ne sont pas devenus « trop grands ». On va donc trouver un cas particulier : $n = 0$. Que faire si $\tilde{d}_0^z \geq b$? Une seule réponse possible : on ajoute un digit, ce qui aura pour effet d'augmenter l'exposant du résultat. La dernière étape consiste à tronquer la décomposition du résultat : jusque-là, on a pris au mot la recommandation de la norme IEEE 754 qui indique que les opérations doivent être réalisées avec une « précision infinie », mais il ne faut pas oublier que le résultat doit lui retrouver la précision autorisée par le système. Au final, on obtient l'[Algorithme A.2.2](#).

Remarque A.2.3. Il ne s'agit bien sûr pas de l'algorithme utilisé par nos ordinateurs (en tout cas, il vaut mieux l'espérer!). En particulier, il est inutile de stocker les derniers digits de x si $\delta > 0$, puisqu'ils sont tout simplement ignorés à la fin. L'[Algorithme A.2.2](#) présente juste une manière de réaliser l'addition entre deux flottants, afin de mieux comprendre le comportement de cette opération.

Remarque A.2.4. La troncature finale est ce qui permet d'expliquer le phénomène d'absorption. Dès que $\delta > 0$, les δ derniers digits sont ignorés (ils ne peuvent même pas être à l'origine d'une retenue). Plus δ est grand, plus on va ignorer de digits. Les cas les plus problématiques arrivent lorsque $\delta > p$: si le calcul est réellement fait en précision infinie, comme c'est le cas dans l'[Algorithme A.2.2](#), il est possible que certains digits de x soient à l'origine d'une retenue, et donc qu'ils influencent le résultat, mais cela n'ira pas plus loin.

Remarque A.2.5. Tout au long de cette section, on a supposé que x et y étaient tous les deux positifs. S'ils sont tous les deux négatifs, on peut reprendre le même algorithme pour calculer $|x| + |y|$, et inverser le signe pour retrouver le résultat :

$$x + y = (-|x|) + (-|y|) = -(|x| + |y|).$$

Si x et y sont de signes opposés, il ne s'agit pas d'une addition, mais d'une soustraction.

Algorithme A.2.2 : Addition de deux flottants

Entrée : $x = b^{e_x} \sum_{n=0}^p \frac{d_n^x}{b^n}$ et $y = b^{e_y} \sum_{n=0}^p \frac{d_n^y}{b^n}$ deux flottants normalisés positifs tels que $x \leq y$.

Sortie : $z := x + y$

$\delta := e_y - e_x$

pour n de 0 à $\delta - 1$

└ $\tilde{d}_n^z := d_n^y$

pour n de δ à p

└ $\tilde{d}_n^z := d_{n-\delta}^x + d_n^y$

pour n de $p+1$ à $p+\delta$

└ $\tilde{d}_n^z := d_{n-\delta}^x$

pour n de $p+\delta$ à $p+1$

└ $d_n^z := \tilde{d}_n^z$

pour n de p à 1

└ **si** $\tilde{d}_n^z < b$

└ $d_n^z := \tilde{d}_n^z$

└ **sinon**

└ $d_n^z := \tilde{d}_n^z - b$

└ $\tilde{d}_{n-1}^z := \tilde{d}_{n-1}^z + 1$

si $\tilde{d}_0^z < b$

└ $d_0^z := \tilde{d}_0^z$

└ $e_z := e_y$

└ $z := b^{e_z} \sum_{n=0}^p \frac{d_n^z}{b^n}$

sinon

└ $d_0^z := \tilde{d}_0^z - b$

└ $e_z := e_y + 1$

└ $z := 1 + \sum_{n=1}^p \frac{d_{n-1}^z}{b^n}$

A.3 Soustraction

On s'intéresse maintenant à la soustraction des flottants (positifs) x et y . On supposera dans la suite $x < y$. Comme $e_x \leq e_y$, il faut, comme pour l'addition, faire en sorte d'avoir le même exposant. En posant $\delta := e_y - e_x \geq 0$, on écrit :

$$x = b^{e_y} \sum_{n=\delta}^{p+\delta} \frac{d_{n-\delta}^x}{b^n}.$$

On obtient ainsi :

$$z := y - x = b^{e_y} \left(\sum_{n=0}^{\delta-1} \frac{d_n^y}{b^n} + \sum_{n=\delta}^p \frac{d_n^y - d_{n-\delta}^x}{b^n} + \sum_{n=p+1}^{p+\delta} \frac{-d_{n-\delta}^x}{b^n} \right),$$

d'où l'on déduit les « digits » temporaires de z :

$$\tilde{d}_n^z := \begin{cases} d_n^y, & \text{si } n \in \llbracket 0, \delta - 1 \rrbracket, \\ d_n^y - d_{n-\delta}^x, & \text{si } n \in \llbracket \delta, p \rrbracket, \\ -d_{n-\delta}^x, & \text{si } n \in \llbracket p+1, p+\delta \rrbracket. \end{cases}$$

Contrairement à l'addition, ici cette définition pose problème dès les premiers digits (toujours en partant de la droite) : comme $-d_{n-\delta}^x \in \llbracket 1-b, 0 \rrbracket$, on ne peut pas les utiliser comme digits définitifs (sauf s'ils sont nuls). Remarquons cependant :

$$1 - b \leq \tilde{d}_n^z \leq -1 \quad \implies \quad 1 \leq \tilde{d}_n^z + b \leq b - 1.$$

Il faut donc ajouter b au n -ème digit. Faisons comme pour l'addition :

$$\frac{\tilde{d}_n^z}{b^n} = \frac{\tilde{d}_n^z + b - b}{b^n} = \frac{\tilde{d}_n^z + b}{b^n} - \frac{1}{b^{n-1}}.$$

Si on veut retrouver un digit légal, et donc ajouter b au n -ème digit, il faut retrancher b au $(n-1)$ -ème digit : on retrouve encore une fois le principe de la retenue.

Il faut donc traiter les digits de droite à gauche, jusqu'à $\tilde{d}_0^z$. La question est alors la suivante : est-il possible que ce dernier digit soit lui-même négatif ? La réponse est non. En effet, deux cas se présentent. Si $\delta > 0$, alors $\tilde{d}_0^z = d_0^y$, ou $\tilde{d}_0^z = d_0^y - 1$ si une retenue a été propagée jusque-là. Or, y est normalisé, donc $d_0^y \geq 1$. Ainsi, $\tilde{d}_0^z \geq 0$. Si $\delta = 0$, x et y ont le même exposant. Comme $x < y$, il existe $n_0 \in \llbracket 0, p \rrbracket$ tel que $d_n^x = d_n^y$ pour tout $n \in \llbracket 0, n_0 - 1 \rrbracket$ et $d_{n_0}^x < d_{n_0}^y$. Ainsi $\tilde{d}_{n_0}^z \geq 1$. Après une éventuelle retenue, on aura donc $\tilde{d}_{n_0}^z \geq 0$: la propagation de la retenue s'arrêtera donc là.

Qu'en est-il de l'écriture finale de z ? A priori, l'exposant de e_z de z devrait être e_y , mais ce n'est pas toujours le cas. En effet, il se peut que les premiers digits de z tels qu'ils ont été calculés soient nuls, ce qui n'est pas compatible avec une écriture normalisée : il faut « décaler » l'exposant de sorte que le premier digit soit non nul. Il est tentant de réutiliser le n_0 déterminé précédemment, mais $d_{n_0}^z$ pourrait très bien être lui-même nul si

une retenue y a été propagée. Le plus simple est donc de définir un nouveau n_0 comme étant l'indice du premier digit non nul de z . On peut alors écrire :

$$z = b^{e_y} \sum_{n=0}^{p+\delta} \frac{d_n^z}{b^n} = b^{e_y} \underbrace{\sum_{n=0}^{n_0-1} \frac{d_n^z}{b^n}}_{=0} + b^{e_y} \sum_{n=n_0}^{p+\delta} \frac{d_n^z}{b^n} = b^{e_y-n_0} \sum_{n=0}^{p+\delta-n_0} \frac{d_{n+n_0}^z}{b^n}.$$

On doit stocker une représentation de z avec une précision p : il faut donc tronquer ce résultat... ce qui n'est possible que si $p + \delta - n_0 \geq p$. Si ce n'est pas le cas, il faut rajouter des zéros. Or, on ne peut pas les rajouter à la fin (cela reviendrait à proposer une mauvaise représentation du nombre, puisque rien n'indique que les digits manquants soient nuls). Il faut donc les rajouter au début : dans ce cas, z sera un flottant dénormalisé. Notons Δ le nombre de zéros à rajouter :

$$\Delta := p - (p + \delta - n_0) = n_0 - \delta.$$

Ainsi :

$$\begin{aligned} z &= b^{e_y-n_0} \sum_{n=0}^{p-\Delta} \frac{d_{n+n_0}^z}{b^n} \\ &= b^{e_y-n_0} \sum_{n=\Delta}^p \frac{d_{n+n_0-\Delta}^z}{b^{n-\Delta}} \\ &= b^{e_y-n_0+\Delta} \sum_{n=\Delta}^p \frac{d_{n+n_0-\Delta}^z}{b^n} \end{aligned}$$

Les digits de 0 à $\Delta - 1$ étant nuls, on peut alors écrire :

$$z = b^{e_y-n_0+\Delta} \sum_{n=0}^p \frac{d_{n+n_0-\Delta}^z}{b^n} = b^{e_y-n_1} \sum_{n=0}^p \frac{d_{n+n_1}^z}{b^n},$$

où on a posé $n_1 := n_0 - \Delta$. On peut remarquer que l'écriture obtenue est la même que la précédente, si ce n'est que l'on utilise n_1 au lieu de n_0 . Comme $n_1 = \delta$, il suffit alors de définir $n_1 = \min(n_0, \delta)$ et les deux cas seront traités automatiquement. Avec tout ceci en tête, on obtient l'[Algorithme A.3.1](#).

Remarque A.3.2. Tout comme pour l'addition, il ne faut pas croire qu'il s'agit là de l'algorithme réellement implémenté dans les ordinateurs. L'[Algorithme A.3.1](#) propose une façon de voir la soustraction entre deux flottants, qui n'est pas optimale, mais qui permet de mieux comprendre le comportement de cette opération et les erreurs qu'elle peut engendrer.

Algorithme A.3.1 : Soustraction

Entrée : $x = b^{e_x} \sum_{n=0}^p \frac{d_n^x}{b^n}$ et $y = b^{e_y} \sum_{n=0}^p \frac{d_n^y}{b^n}$ deux flottants normalisés positifs tels que $x < y$.

Sortie : $z := y - x$

$\delta := e_y - e_x$

pour n de 0 à $\delta - 1$

└ $\tilde{d}_n^z := d_n^y$

pour n de δ à p

└ $\tilde{d}_n^z := d_n^y - d_{n-\delta}^x$

pour n de $p+1$ à $p+\delta$

└ $\tilde{d}_n^z := -d_{n-\delta}^x$

pour n de $p+\delta$ à 1

┌ **si** $\tilde{d}_n^z \geq 0$

└ $d_n^z := \tilde{d}_n^z$

sinon

└ $d_n^z := \tilde{d}_n^z + b$

└ $\tilde{d}_{n-1}^z := \tilde{d}_{n-1}^z - 1$

$d_0^z := \tilde{d}_0^z$

Définir n_0 tel que : $d_{n_0}^z \neq 0$ et $\forall n \in \llbracket 0, n_0 - 1 \rrbracket, d_n^z = 0$

$n_1 := \min(n_0, \delta)$

$e_z := e_y - n_1$

$z := b^{e_z} \sum_{n=0}^p \frac{d_{n+n_1}^z}{b^n}$

Remarque A.3.3. L'[Algorithme A.3.1](#) effectue un calcul en « précision infinie ». En particulier, on ne restreint pas le nombre de digits supplémentaires utilisés si $\delta > 0$. Ces « digits de garde » ne sont pas nécessaires dans le cas de l'addition, mais sont nécessaires dans le cas de la soustraction puisqu'ils sont sources de retenues.

Remarque A.3.4. Le passage par la précision infinie ne permet pas d'empêcher les cas d'annulation catastrophique : si beaucoup des premiers digits s'annulent, on peut perdre suffisamment de précision pour n'obtenir qu'un nombre dénormalisé. Il est à noter qu'il ne s'agit pas de la forme dénormalisée proposée par la norme IEEE 754. Pour cela, il faudrait ramener le nombre à l'exposant minimal autorisé (ce qui impliquerait une potentielle perte de précision supplémentaire).

Remarque A.3.5. L'[Algorithme A.3.1](#) n'est valable que pour calculer $z = y - x$ dans le cas $0 < x < y$. Si $0 < y < x$, on peut l'utiliser pour calculer $\tilde{z} := x - y$ et en déduire $z = -\tilde{z}$. Si x et y sont tous les deux négatifs, alors :

$$z = y - x = -(|y| - |x|)$$

donc on peut calculer $|y| - |x|$ et en déduire z . Dans le cas où $x < 0 < y$, alors

$$z = y - x = y + |x|$$

et il s'agit d'une addition. On se ramène également à l'addition dans le cas $y < 0 < x$. Enfin, le cas $x = y$ est évident.

A.4 Multiplication

Intéressons-nous à la multiplication des flottants, toujours positifs, x et y . Écrivons directement une première approche :

$$z := xy = \left(b^{e_x} \sum_{n=0}^p \frac{d_n^x}{b^n} \right) \left(b^{e_y} \sum_{n=0}^p \frac{d_n^y}{b^n} \right) = b^{e_x+e_y} \sum_{n=0}^p \sum_{m=0}^p \frac{d_n^x d_m^y}{b^{n+m}}.$$

Il ne s'agit pas encore d'une première décomposition de z : le coefficient de chaque puissance de b n'apparaît pas clairement. La décomposition d'un nombre flottant n'étant finalement rien d'autre qu'un polynôme, on peut réécrire ce produit :

$$\begin{aligned} z &= b^{e_x+e_y} \sum_{n=0}^{2p} \sum_{m=n-\min(n,p)}^{\min(n,p)} \frac{d_m^x d_{n-m}^y}{b^n} \\ &= b^{e_x+e_y} \left(\sum_{n=0}^p \sum_{m=0}^n \frac{d_m^x d_{n-m}^y}{b^n} + \sum_{n=p+1}^{2p} \sum_{m=n-p}^p \frac{d_m^x d_{n-m}^y}{b^n} \right). \end{aligned}$$

On procède alors comme pour l'addition et la soustraction en définissant des digits temporaires :

$$\tilde{d}_n^z := \begin{cases} \sum_{m=0}^n d_m^x d_{n-m}^y, & \text{si } n \in \llbracket 0, p \rrbracket, \\ \sum_{m=n-p}^p d_m^x d_{n-m}^y, & \text{si } n \in \llbracket p+1, 2p \rrbracket. \end{cases}$$

Les digits de x et y étant tous compris entre 0 et $b-1$, on va avoir :

$$\forall n \in \llbracket 0, p \rrbracket, \quad 0 \leq \tilde{d}_n^z \leq (n+1)(b-1)^2,$$

ainsi que :

$$\forall n \in \llbracket p+1, 2p \rrbracket, \quad 0 \leq \tilde{d}_n^z \leq (2p-n+1)(b-1)^2.$$

Il est donc possible d'avoir des digits temporaires supérieurs ou égaux à b . Comme dans le cas de l'addition, de telles positions vont générer des retenues. Effectuons la division euclidienne d'un tel digit par b :

$$\tilde{d}_n^z = qb + r, \quad \text{avec } q \in \mathbb{N} \text{ et } r \in [0, b-1].$$

On obtient alors :

$$\frac{\tilde{d}_n^z}{b^n} = \frac{\tilde{d}_n^z - qb + qb}{b^n} = \frac{r}{b^n} + \frac{q}{b^{n-1}}.$$

Reste la question du 0-ème digit : que faire si $\tilde{d}_0^z \geq b$? Dans ce cas, il faut créer autant de nouveaux digits que nécessaire. On commence donc par effectuer la division euclidienne de $\tilde{d}_0^z$ par b pour connaître le premier digit à créer. Si ce nouveau digit est lui encore trop grand, on effectue sa division euclidienne par b pour connaître le second digit à créer, et ainsi de suite, jusqu'à avoir un digit « légal », compris entre 0 et $b-1$. L'exposant de z devra alors être décalé autant de fois que de nouveaux digits ont été créés. En effet, notons k le nombre de digits nouvellement créés. Autrement dit :

$$\tilde{d}_0^z = \sum_{n=-k}^0 \frac{d_n^z}{b^n}.$$

Ainsi :

$$z = b^{e_x+e_y} \left(\sum_{n=-k}^0 \frac{d_n^z}{b^n} + \sum_{n=1}^p \frac{d_n^z}{b^n} \right) = b^{e_x+e_y+k} \left(\sum_{n=0}^k \frac{d_{n-k}^z}{b^n} + \sum_{n=k+1}^{p+k} \frac{d_{n-k}^z}{b^n} \right).$$

Remarque A.4.2. L'[Algorithme A.4.1](#) n'utilise pas le fait que x et y sont positifs. Dans le cas général, on peut simplement utiliser cet algorithme en utilisant les exposants et les significandes de x et y . On détermine le signe de z indépendamment, à partir de ceux de x et y .

A.5 Division

On s'intéresse maintenant à la division entre deux flottants positifs x et y , en supposant $y > 0$. Commençons par écrire :

$$z := \frac{x}{y} = \frac{b^{e_x} s_x}{b^{e_y} s_y} = \frac{\sum_{n=0}^p \frac{d_n^x}{b^n}}{s_y} = \sum_{n=0}^p \frac{d_n^x}{s_y} \frac{1}{b^n}.$$

Les digits de z seraient donc tout simplement donnés par $\tilde{d}_n^z := \frac{d_n^x}{s_y}$. Comme $d_n^x \in \llbracket 0, b-1 \rrbracket$ et $s_y \in [1, b[$, on a bien $\tilde{d}_n^z \in [0, b-1]$. Cependant, rien ne garantit qu'il s'agisse d'un entier :

Algorithme A.4.1 : Multiplication

Entrée : $x = b^{e_x} \sum_{n=0}^p \frac{d_n^x}{b^n}$ et $y = b^{e_y} \sum_{n=0}^p \frac{d_n^y}{b^n}$ deux flottants normalisés positifs

Sortie : $z := xy$

pour n de 0 à p

$$\left[\begin{array}{l} \tilde{d}_n^z := \sum_{m=0}^n d_m^x d_{n-m}^y \end{array} \right.$$

pour n de $p+1$ à $2p$

$$\left[\begin{array}{l} \tilde{d}_n^z := \sum_{m=n-p}^p d_m^x d_{n-m}^y \end{array} \right.$$

pour n de $2p$ à 1

si $\tilde{d}_n^z < b$

$$\left| \begin{array}{l} d_n^z := \tilde{d}_n^z \end{array} \right.$$

sinon

Effectuer la division euclidienne $\tilde{d}_n^z = qb + r$

$$\left| \begin{array}{l} d_n^z := r \end{array} \right.$$

$$\left| \begin{array}{l} \tilde{d}_{n-1}^z := \tilde{d}_{n-1}^z + q \end{array} \right.$$

$k := 0$

tant que $\tilde{d}_{-k}^z \geq b$

Effectuer la division euclidienne $\tilde{d}_{-k}^z = qb + r$

$$\left| \begin{array}{l} d_{-k}^z := r \end{array} \right.$$

$$\left| \begin{array}{l} \tilde{d}_{-(k+1)}^z := q \end{array} \right.$$

$$\left| \begin{array}{l} k := k + 1 \end{array} \right.$$

$$d_{-k}^z := \tilde{d}_{-k}^z$$

$$e_z := e_x + e_y + k$$

$$z := b^{e_z} \sum_{n=0}^p \frac{d_{n-k}^z}{b^n}$$

cela ne peut donc pas être un digit ! Une solution simple consiste bien sûr à prendre la partie entière de ce nombre. On peut alors écrire

$$\tilde{d}_n^z = \lfloor \tilde{d}_n^z \rfloor + f_n, \quad \text{avec } f_n := \tilde{d}_n^z - \lfloor \tilde{d}_n^z \rfloor.$$

On aimerait définir le n -ème digit de z comme étant $\lfloor \tilde{d}_n^z \rfloor$, donc l'idée est de « renvoyer » f_n sur le $(n+1)$ -ème digit. On peut faire cela assez facilement :

$$\frac{\tilde{d}_n^z}{b^n} = \frac{\lfloor \tilde{d}_n^z \rfloor + f_n}{b^n} = \frac{\lfloor \tilde{d}_n^z \rfloor}{b^n} + \frac{f_n}{b^n} = \frac{\lfloor \tilde{d}_n^z \rfloor}{b^n} + \frac{bf_n}{b^{n+1}}.$$

On souhaite obtenir une écriture normalisée du résultat. Il faut donc s'intéresser au cas où les premiers digits déterminés sont nuls. Une possibilité est de compter ces digits nuls pour ensuite décaler l'exposant du résultat. Au final, on obtient l'[Algorithme A.5.1](#).

Algorithme A.5.1 : Division

Entrée : $x = b^{e_x} \sum_{n=0}^p \frac{d_n^x}{b^n} \geq 0$ et $y = b^{e_y} s_y > 0$ deux flottants normalisés

Sortie : $z := \frac{x}{y}$

$f_{-1} := 0$

$k := 0$

pour n de 0 à p

$\tilde{d}_n^z := \frac{d_n^x + b f_{n-1}}{s_y}$	
$d_n^z := \lfloor \tilde{d}_n^z \rfloor$	
$f_n := \tilde{d}_n^z - d_n^z$	
si $d_n^z = 0$ et $k = n$	
$k := k + 1$	

$e_z := e_x - e_y - k$

$z := b^{e_z} \sum_{n=0}^p \frac{d_{n+k}^z}{b^n}$

Remarque A.5.2. De même que pour la multiplication, l'Algorithme A.5.1 peut aussi s'utiliser avec des nombres non positifs : dans ce cas, le signe se déduit indépendamment.	
--	--

Annexe B

Polynômes de Tchebychev

Les polynômes de Tchebychev sont utiles un peu partout en analyse numérique. On regroupe ici quelques résultats intéressants, certains étant utilisés dans le reste du cours.

B.1 Définition et premières propriétés

Une manière de définir les polynômes de Tchebychev est d'utiliser une formule de récurrence. On définit ainsi $(T_k)_k$ par $T_0 := 1$, $T_1 := X$ et :

$$\forall k \in \mathbb{N}^*, \quad T_{k+1} = 2XT_k - T_{k-1}.$$

Proposition B.1.1. Pour tout $k \in \mathbb{N}$, T_k est de degré k et son coefficient dominant, pour $k \geq 1$, est donné par 2^{k-1} .

Démonstration. On procède par récurrence. Pour le degré, on commence par $T_0 = 1$, qui est bien de degré 0, et $T_1 = X$, qui est bien de degré 1. Par ailleurs, le coefficient dominant de T_1 est $1 = 2^{1-1}$. On doit également calculer :

$$T_2 = 2XT_1 - T_0 = 2X^2 - 1,$$

dont le coefficient dominant est $2 = 2^{2-1}$. Supposons donc que la propriété est vraie jusqu'à un certain $k \in \mathbb{N}^*$. Alors :

$$T_{k+1} = 2XT_k - T_{k-1}$$

est de degré $\deg T_k + 1 = k + 1$ car $\deg T_{k-1} = k - 1$ donc aucune compensation n'aura lieu. De même, le coefficient dominant de T_{k+1} est donné par deux fois celui de T_k , c'est-à-dire $2 \times 2^{k-1} = 2^k$. ■

Proposition B.1.2. Pour tout $k \in \mathbb{N}$, le polynôme T_k est pair si k est pair, et impair si k est impair.

Démonstration. On vérifie que $T_0 = 1$ est pair, tandis que $T_1 = X$ est impair. Supposons donc ce résultat vrai jusqu'à un certain $k \in \mathbb{N}$. Une autre manière de l'écrire est que la parité de T_k est celle de k :

$$T_k(-X) = (-1)^k T_k(X).$$

À l'étape suivante, on a ainsi :

$$\begin{aligned} T_{k+1}(-X) &= -2XT_k(-X) - T_{k-1}(-X) \\ &= -2X(-1)^k T_k(X) - (-1)^{k-1} T_{k-1}(X) \\ &= (-1)^{k+1} (2XT_k(X) - T_{k-1}(X)) \\ &= (-1)^{k+1} T_{k+1}(X), \end{aligned}$$

où il ne faut pas oublier l'égalité tout simple $(-1)^{k-1} = (-1)^{k+1} !$ ■

B.2 Une histoire de cosinus

Les polynômes de Tchebychev sont souvent connus pour leur restriction à l'intervalle $[-1, 1]$. Et pour cause : ils peuvent y être exprimés comme des cosinus, ce qui peut paraître plutôt surprenant au premier abord pour des polynômes. Montrons cela.

Lemme B.2.1. Pour tout $k \in \mathbb{N}$, T_k vérifie la relation :

$$\forall \theta \in \mathbb{R}, \quad T_k(\cos \theta) = \cos(k\theta).$$

Démonstration. On procède encore une fois par récurrence. L'initialisation est assez facile :

$$\begin{aligned} T_0(\cos \theta) &= 1 = \cos(0 \cdot \theta), \\ T_1(\cos \theta) &= \cos \theta = \cos(1 \cdot \theta). \end{aligned}$$

Supposons donc le résultat vrai jusqu'à un certain rang $k \in \mathbb{N}$. Alors :

$$\begin{aligned} T_{k+1}(\cos \theta) &= 2 \cos \theta T_k(\cos \theta) - T_{k-1}(\cos \theta) \\ &= 2 \cos \theta \cos(k\theta) - \cos((k-1)\theta). \end{aligned}$$

Or :

$$\forall (a, b) \in \mathbb{R}^2, \quad 2 \cos a \cos b = \cos(a+b) + \cos(a-b).$$

Ici, on obtient donc :

$$2 \cos \theta \cos(k\theta) = \cos((k+1)\theta) + \cos((k-1)\theta),$$

et ainsi :

$$T_{k+1}(\cos \theta) = \cos((k+1)\theta),$$

et le résultat est démontré. ■

Proposition B.2.2. Pour tout $k \in \mathbb{N}$, on a :

$$\forall u \in [-1, 1], \quad T_k(u) = \cos(k \arccos u).$$

Démonstration. Soit $u \in [-1, 1]$. Alors il existe un unique $\theta \in [0, \pi]$ tel que $u = \cos \theta$. On applique alors le lemme précédent :

$$T_k(u) = T_k(\cos \theta) = \cos(k\theta) = \cos(k \arccos u).$$

■

De cette expression, on peut déjà tirer un résultat simple :

$$\forall k \in \mathbb{N}, \quad T_k(1) = 1 \quad \text{et} \quad T_k(-1) = (-1)^k,$$

simplement en calculant $\arccos 1 = 0$ et $\arccos(-1) = \pi$ (ou, si on ne veut le faire que pour 1, utiliser la parité de T_k pour -1).

On peut déterminer les zéros de cette expression plutôt facilement. Autrement dit, on peut calculer toutes les racines de T_k situées sur l'intervalle $[-1, 1]$. C'est l'objet du résultat suivant, où on a également la bonne idée de compter le nombre de racines obtenues.

Proposition B.2.3. Pour tout $k \in \mathbb{N}^*$, T_k admet k racines distinctes, toutes situées dans l'intervalle $[-1, 1]$ et données par :

$$\forall m \in \llbracket 0, k-1 \rrbracket, \quad u_m := \cos\left(\frac{\pi}{2k} + \frac{m\pi}{k}\right).$$

Démonstration. Soit $u \in [-1, 1]$ tel que $T_k(u) = 0$. Alors :

$$T_k(u) = \cos(k \arccos u) = 0.$$

Ainsi, il existe $m \in \mathbb{Z}$ tel que :

$$k \arccos u = \frac{\pi}{2} + m\pi,$$

ce qui donne :

$$u = \cos\left(\frac{\pi}{2k} + \frac{m\pi}{k}\right) = \cos t_m \quad \text{avec} \quad t_m := \frac{\pi}{2k} + \frac{m\pi}{k}.$$

Déterminons les t_m situés dans l'intervalle $[0, \pi]$:

$$0 \leq t_m \leq \pi \quad \Longleftrightarrow \quad \frac{1}{2} \leq m \leq k - \frac{1}{2}.$$

Comme m est entier, on obtient k points distincts :

$$\forall m \in \llbracket 0, k-1 \rrbracket, \quad t_m = \frac{\pi}{2k} + \frac{m\pi}{k},$$

et les racines de T_k correspondantes $\cos t_m$.

On a donc trouvé k racines de T_k , qui est un polynôme de degré k : il n'y en a donc pas d'autre. ■

On peut procéder de manière similaire pour déterminer les extrema de ces polynômes sur l'intervalle $[-1, 1]$.

Proposition B.2.4. Pour tout $k \in \mathbb{N}^*$, T_k atteint exactement $k+1$ fois son extremum sur l'intervalle $[-1, 1]$:

$$\forall m \in \llbracket 0, k \rrbracket, \quad T_k(u_m) = (-1)^m \quad \text{avec } u_m = \cos\left(\frac{m\pi}{k}\right).$$

Démonstration. Sur cet intervalle, on a déjà montré :

$$\forall u \in [-1, 1], \quad T_k(u) = \cos(k \arccos u),$$

donc, en particulier :

$$\forall u \in [-1, 1], \quad |T_k(u)| \leq 1.$$

Soit $u \in [-1, 1]$ un extremum de T_k , c'est-à-dire $|T_k(u)| = 1$, ou encore $T_k(u) = \pm 1$. On a donc :

$$\cos(k \arccos u) = \pm 1.$$

Ainsi, il existe $m \in \mathbb{Z}$ tel que :

$$k \arccos u = m\pi.$$

Autrement dit :

$$u = \cos\left(\frac{m\pi}{k}\right).$$

En notant

$$t_m := \frac{m\pi}{k},$$

on peut rechercher $t_m \in [0, \pi]$:

$$0 \leq t_m \leq \pi \quad \Longleftrightarrow \quad 0 \leq m \leq k.$$

On obtient ainsi $k+1$ extrema distincts, pour $m \in \llbracket 0, k \rrbracket$. ■

B.3 Et aussi de cosinus hyperbolique

Nous avons déjà montré une certaine relation entre les polynômes de Tchebychev et le cosinus. On peut en fait montrer la même relation avec le cosinus hyperbolique.

Lemme B.3.1. Pour tout $k \in \mathbb{N}$, T_k vérifie la relation :

$$\forall y \in \mathbb{R}, \quad T_k(\cosh y) = \cosh(ky).$$

Démonstration. Commençons par initialiser la récurrence :

$$\begin{aligned} T_0(\cosh y) &= 1 = \cosh(0 \cdot y), \\ T_1(\cosh y) &= \cosh y = \cosh(1 \cdot y). \end{aligned}$$

Supposons le résultat vrai jusqu'à un certain rang $k \in \mathbb{N}$. Alors :

$$\begin{aligned} T_{k+1}(\cosh y) &= 2 \cosh y T_k(\cosh y) - T_{k-1}(\cosh y) \\ &= 2 \cosh y \cosh(ky) - \cosh((k-1)y). \end{aligned}$$

Or :

$$\forall (a, b) \in \mathbb{R}^2, \quad 2 \cosh a \cosh b = \cosh(a+b) + \cosh(a-b),$$

et on peut donc conclure comme pour le cosinus. ■

Proposition B.3.2. Pour tout $k \in \mathbb{N}$, on a :

$$\forall x \in [1, +\infty[, \quad T_k(x) = \cosh(k \operatorname{arccosh} x).$$

Démonstration. Soit $x \in [1, +\infty[$. Il existe un unique $y \in \mathbb{R}^+$ tel que $\cosh y = x$. On applique alors le lemme précédent pour conclure. ■

De cette formule on tire notamment :

$$\forall x \in [1, +\infty[, \quad T_k(x) \geq 1.$$

Proposition B.3.3. Pour tout $k \in \mathbb{N}$, on a :

$$\forall x \in]-\infty, -1], \quad T_k(x) = (-1)^k \cosh(k \operatorname{arccosh}(-x)).$$

Démonstration. On utilise simplement le résultat précédent et la parité. Pour tout $x \in]-\infty, -1]$, $-x \in [1, +\infty[$ et on peut écrire :

$$T_k(x) = (-1)^k T_k(-x) = (-1)^k \cosh(k \operatorname{arccosh}(-x)).$$
■

B.4 Une expression à base de racines carrées

Continuons notre petit panorama avec une expression explicite des polynômes de Tchebychev, pas forcément la plus connue, mais parfois utile.

Lemme B.4.1. Pour tout $k \in \mathbb{N}$, on a :

$$\forall x \in [1, +\infty[, \quad T_k(x) = \frac{1}{2} \left(\left(x + \sqrt{x^2 - 1} \right)^k + \left(x - \sqrt{x^2 - 1} \right)^k \right).$$

Démonstration. Soit $x \in [1, +\infty[$. On trouve un unique $y \in \mathbb{R}^+$ tel que $\cosh y = x$. Ainsi :

$$T_k(x) = T_k(\cosh y) = \cosh(ky) = \frac{e^{ky} + e^{-ky}}{2}.$$

Or :

$$e^{ky} = (e^y)^k = (\cosh y + \sinh y)^k.$$

Comme $y \geq 0$, alors $\sinh y \geq 0$, donc on peut écrire

$$\sinh y = \sqrt{\cosh^2 y - 1}.$$

Ainsi :

$$e^{ky} = \left(\cosh y + \sqrt{\cosh^2 y - 1} \right)^k = \left(x + \sqrt{x^2 - 1} \right)^k,$$

en revenant à la définition de y . On montre de la même façon :

$$e^{-ky} = (\cosh y - \sinh y)^k = \left(x - \sqrt{x^2 - 1} \right)^k,$$

d'où le résultat. ■

Lemme B.4.2. Pour tout $k \in \mathbb{N}$, on a :

$$\forall x \in]-\infty, -1], \quad T_k(x) = \frac{1}{2} \left(\left(x + \sqrt{x^2 - 1} \right)^k + \left(x - \sqrt{x^2 - 1} \right)^k \right).$$

Démonstration. Soit $x \in]-\infty, -1]$. Alors $-x \in [1, +\infty[$ et ainsi, grâce au résultat précédent :

$$\begin{aligned} T_k(-x) &= \frac{1}{2} \left(\left(-x + \sqrt{(-x)^2 - 1} \right)^k + \left(-x - \sqrt{(-x)^2 - 1} \right)^k \right) \\ &= \frac{1}{2} \left(\left(-x + \sqrt{x^2 - 1} \right)^k + \left(-x - \sqrt{x^2 - 1} \right)^k \right) \\ &= \frac{1}{2} \left(\left(- \left(x - \sqrt{x^2 - 1} \right) \right)^k + \left(- \left(x + \sqrt{x^2 - 1} \right) \right)^k \right) \\ &= \frac{(-1)^k}{2} \left(\left(x - \sqrt{x^2 - 1} \right)^k + \left(x + \sqrt{x^2 - 1} \right)^k \right), \end{aligned}$$

et on conclut avec la parité de T_k . ■

On peut faire un raisonnement similaire à celui effectué sur $[1, +\infty[$ en travaillant sur $[-1, 1]$, à ceci près qu'il faut passer par les nombres complexes.

Lemme B.4.3. Pour tout $k \in \mathbb{N}$, on a :

$$\forall x \in [-1, 1], \quad T_k(x) = \frac{1}{2} \left(\left(x + \sqrt{x^2 - 1} \right)^k + \left(x - \sqrt{x^2 - 1} \right)^k \right).$$

Démonstration. Soit $x \in [-1, 1]$. Il existe un unique $y \in [0, \pi]$ tel que $\cos y = x$. Ainsi :

$$T_k(x) = T_k(\cos y) = \cos(ky) = \frac{e^{iky} + e^{-iky}}{2}.$$

Or :

$$e^{iky} = (e^{iy})^k = (\cos y + i \sin y)^k.$$

Comme $y \in [0, \pi]$, $\sin y \geq 0$, donc :

$$\sin y = \sqrt{1 - \cos^2 y},$$

d'où :

$$i \sin y = \sqrt{-1} \sqrt{1 - \cos^2 y} = \sqrt{\cos^2 y - 1},$$

et on peut faire de même avec e^{-iky} pour obtenir le résultat. ■

On a finalement couvert tout l'intervalle réel, avec une seule et même formule. On formalise ce résumé dans le résultat suivant.

Proposition B.4.4. Pour tout $k \in \mathbb{N}$, on a :

$$\forall x \in \mathbb{R}, \quad T_k(x) = \frac{1}{2} \left(\left(x + \sqrt{x^2 - 1} \right)^k + \left(x - \sqrt{x^2 - 1} \right)^k \right).$$

Annexe C

Autour d'un mystère de l'arithmétique flottante

On considère un système de nombres flottants en base $b = 2$, en précision $p \in \mathbb{N}^*$, et sans restriction d'exposants pour simplifier. On pose $x := 0.1$ et on note $\hat{x} := \text{fl}(x)$ son représentant flottant. On pose également $y := 0.2$, $z := 0.3$, et $\hat{y} := \text{fl}(y)$, $\hat{z} := \text{fl}(z)$. On définit $\sigma := \hat{x} + \hat{y}$, et on cherche à déterminer si $\hat{\sigma} := \text{fl}(\sigma) = \hat{z}$. Autrement dit, a-t-on $\text{fl}(\text{fl}(0.1) + \text{fl}(0.2)) = \text{fl}(0.3)$?

C.1 Observations

Avant d'aller plus loin, commençons par quelques observations. On utilise les notations du cours, c'est-à-dire que, pour $x \in \mathbb{R}$, l'exposant de x est donné par $\lfloor \log_2 x \rfloor$, son significande est obtenu avec $s := 2^{-e}x$. On calcule ensuite $\mu := 2^p s$, de sorte que $x = 2^{e-p}\mu$. On calcule $m \in \llbracket 2^p, 2^{p+1} - 1 \rrbracket$ comme étant l'arrondi de μ (qui vaut donc $\lfloor \mu \rfloor$ ou $\lfloor \mu \rfloor + 1$ selon celui qui est le plus proche). Si μ est exactement au milieu de l'intervalle ($\mu = \lfloor \mu \rfloor + \frac{1}{2}$), alors on arrondit au nombre pair le plus proche. On obtient alors $\text{fl}(x) = 2^{e-p}m$. Le [Tableau C.1](#) liste quelques valeurs de m_σ et m_z obtenues pour différentes précisions.

p	μ_x	m_x	μ_z	m_z	μ_σ	m_σ
3	12.8	13	9.6	10	9.75	10
4	25.6	26	19.2	19	19.5	20
5	51.2	51	38.4	38	38.25	38
6	102.4	102	76.8	77	76.5	76
7	204.8	205	153.6	154	153.75	154
8	409.6	410	307.2	307	307.5	308
9	819.2	819	614.4	614	614.25	614
10	1638.4	1638	1228.8	1229	1228.5	1228

TABLE C.1 – Quelques représentations flottantes de 0.3 et 0.1 + 0.2.

Si ce tableau est probablement quelque peu indigeste au premier coup d'œil, on peut

y remarquer un fait intéressant : dans certains cas $m_z = m_\sigma$. Comme les exposants de z et de σ sont les mêmes (on va le montrer juste après), on en déduit que, dans ces cas-là, $\hat{z} = \hat{\sigma}$. Des valeurs de p testées dans ce tableau, il apparaît que l'égalité intervient lorsque p est impair. Lorsque p est pair, $m_z \neq m_\sigma$. On a parfois $m_\sigma = m_z + 1$, parfois $m_\sigma = m_z - 1$. Plus précisément, on alterne entre ces deux possibilités. Nous allons montrer que tout cela est vrai, en toute généralité (et donc pas seulement jusqu'à $p = 10$!).

Au passage, c'est ce qui explique qu'un ordinateur nous dise que $0.1 + 0.2 \neq 0.3$: le format binary64 (double précision) utilise une précision $p = 52$, qui est paire. Il y a quelques années, lorsque la simple précision était le standard, et donc le format binary32, les ordinateurs nous disaient bien que $0.1 + 0.2 = 0.3$: ce format utilise une précision $p = 23$, qui est impaire !

C.2 Quelques relations préliminaires

Commençons par remarquer :

$$0.0625 = 2^{-4} \leq 0.1 < 0.125 = 2^{-3}.$$

Ainsi :

$$-4 \leq \log_2 x < -3,$$

d'où l'on déduit l'exposant de x : $e_x = \lfloor \log_2 x \rfloor = -4$. On obtient ainsi le significande de x :

$$s_x = 2^{-e_x} x = 2^4 \cdot 0.1 = \frac{2^4}{10} = 1.6.$$

On peut alors définir $\mu_x = 2^p s_x$. L'arrondi $m_x \in \{\lfloor \mu_x \rfloor, \lfloor \mu_x \rfloor + 1\}$ permet alors d'obtenir le représentant flottant de x : $\hat{x} = 2^{-4-p} m_x$.

Occupons-nous maintenant de y en notant :

$$0.125 = 2^{-3} \leq 0.2 < 0.25 = 2^{-2},$$

d'où $e_y = -3$. Remarquons maintenant que $y = 2x$. Ainsi :

$$s_y = 2^{-e_y} y = 2^3 \cdot 2x = 2^4 x = s_x.$$

On en déduit directement $\mu_y = \mu_x$ et donc $m_y = m_x$. Donc :

$$\hat{y} = 2^{-3-p} m_y = 2 \cdot 2^{-4-p} m_x = 2\hat{x}.$$

L'exposant de z s'obtient aussi facilement que les autres :

$$0.25 = 2^{-2} \leq 0.3 < 0.5 = 2^{-1},$$

donc $e_z = -2$. Remarquons que $z = 3x$. Ainsi :

$$s_z = 2^2 z = 2^2 \cdot 3x = \frac{3}{4} 2^4 x = \frac{3}{4} s_x.$$

On en déduit directement :

$$\mu_z = 2^p s_z = 2^p \cdot \frac{3}{4} s_x = \frac{3}{4} \cdot 2^p s_x = \frac{3}{4} \mu_x.$$

C.3 Expression de la somme

Remarquons maintenant que :

$$s_x = 1.6 \geq 1.5 = \frac{1+2}{2} = 2^{-1} + 1,$$

d'où :

$$\mu_x = 2^p s_x \geq 2^{p-1} + 2^p.$$

Ainsi :

$$\lfloor \mu_x \rfloor \geq \lfloor 2^{p-1} + 2^p \rfloor = 2^{p-1} + 2^p.$$

Or, $m_x \in \{\lfloor \mu_x \rfloor, \lfloor \mu_x \rfloor + 1\}$, donc on a toujours :

$$m_x \geq \lfloor \mu_x \rfloor \geq 2^{p-1} + 2^p.$$

Par construction, on a par ailleurs $m_x < 2^{p+1}$, d'où pour résumer :

$$2^{p-1} + 2^p \leq m_x < 2^{p+1}.$$

Exprimons la somme qui nous intéresse :

$$\sigma = \hat{x} + \hat{y} = \hat{x} + 2\hat{x} = 3\hat{x} = 2^{-4-p} \cdot 3m_x.$$

Or :

$$3 \cdot 2^{p-1} + 3 \cdot 2^p \leq 3m_x < 3 \cdot 2^{p+1}.$$

Tâchons de ne conserver que des puissances de deux. Commençons par le membre de droite :

$$3 \cdot 2^{p+1} = (2+1) \cdot 2^{p+1} = 2^{p+2} + 2^{p+1} = \frac{2^{p+3} + 2^{p+2}}{2} < 2^{p+3}.$$

Occupons-nous maintenant du membre de gauche :

$$\begin{aligned} 3 \cdot 2^{p-1} + 3 \cdot 2^p &= 2^p + 2^{p-1} + 2^{p+1} + 2^p \\ &= 2 \cdot 2^p + 2^{p-1} + 2^{p+1} \\ &= 2^{p+1} + 2^{p-1} + 2^{p+1} \\ &= 2^{p+2} + 2^{p-1} \\ &\geq 2^{p+2}. \end{aligned}$$

Ainsi :

$$2^{p+2} \leq 3m_x < 2^{p+3},$$

d'où :

$$2^{-2} = 2^{-4-p} \cdot 2^{p+2} \leq \sigma < 2^{-4-p} \cdot 2^{p+3} = 2^{-1},$$

d'où l'on déduit $e_\sigma = -2$.

Continuons sur notre lancée :

$$s_\sigma = 2^2 \sigma = 2^{-2-p} \cdot 3m_x.$$

Ainsi :

$$\mu_\sigma = 2^p s_\sigma = 2^{-2} \cdot 3m_x = \frac{3}{4}m_x.$$

Dans la suite, on notera m_σ l'arrondi de μ_σ .

C.4 Une histoire de parité

C.4.1 Premier cas impair

Supposons qu'il existe $q \in \mathbb{N}$ tel que $p = 4q + 1$. Alors on trouve $k \in \mathbb{N}$ tel que :

$$2^p = 2^{4q+1} = 10k + 2.$$

Ainsi :

$$2^{p+4} = 16 \cdot 2^p = 10 \cdot 16k + 32 = 10(16k + 3) + 2.$$

On obtient :

$$\mu_x = 2^{p+4} \cdot 0.1 = \frac{2^{p+4}}{10} = 16k + 3 + \frac{1}{5}.$$

On en déduit $\lfloor \mu_x \rfloor = 16k + 3$, et surtout :

$$m_x = \lfloor \mu_x \rfloor \quad \text{car } \frac{1}{5} < \frac{1}{2}.$$

Donc :

$$\mu_x = \lfloor \mu_x \rfloor + \frac{1}{5} = m_x + \frac{1}{5}.$$

On a ainsi :

$$\frac{3}{4}\mu_x = \frac{3}{4}m_x + \frac{3}{20}.$$

Or, $\frac{3}{4}\mu_x = \mu_z$ et $\frac{3}{4}m_x = \mu_\sigma$. Donc :

$$\mu_\sigma = \mu_z - \frac{3}{20}.$$

Par ailleurs :

$$2^{p+2} = 4 \cdot 2^p = 10 \cdot 4k + 8,$$

donc :

$$3 \cdot 2^{p+2} = 10 \cdot 12k + 24 = 10(12k + 2) + 4.$$

Ainsi :

$$\mu_z = \frac{3}{4}\mu_x = \frac{3 \cdot 2^{p+2}}{10} = 12k + 2 + \frac{2}{5}.$$

On en déduit $m_z = \lfloor \mu_z \rfloor$ et ainsi :

$$\mu_z = m_z + \frac{2}{5}.$$

On réunit le tout :

$$\mu_\sigma = m_z + \frac{2}{5} - \frac{3}{20} = m_z + \frac{1}{4}.$$

On en déduit $\lfloor \mu_\sigma \rfloor = m_z$, et $m_\sigma = \lfloor \mu_\sigma \rfloor$ car $\frac{1}{4} < \frac{1}{2}$. Autrement dit, $m_\sigma = m_z$ et $\hat{\sigma} = \hat{z}$.

C.4.2 Second cas impair

Deuxième possibilité pour que p soit impair : $p = 4q + 3$. Dans ce cas, on trouve $k \in \mathbb{N}$ tel que :

$$2^p = 2^{4q+3} = 10k + 8.$$

Ainsi :

$$2^{p+4} = 10 \cdot 16k + 128 = 10(16k + 12) + 8.$$

On obtient :

$$\mu_x = \frac{2^{p+4}}{10} = 16k + 12 + \frac{4}{5}.$$

On en déduit $\lfloor \mu_x \rfloor = 16k + 12$ et $m_x = \lfloor \mu_x \rfloor + 1$. Ainsi :

$$\mu_x = \lfloor \mu_x \rfloor + \frac{4}{5} = m_x - 1 + \frac{4}{5} = m_x - \frac{1}{5}.$$

On a ainsi :

$$\frac{3}{4}\mu_x = \frac{3}{4}m_x - \frac{3}{20}.$$

On a donc trouvé :

$$\mu_\sigma = \mu_z + \frac{3}{20}.$$

Par ailleurs :

$$2^{p+2} = 10 \cdot 4k + 32 = 10(4k + 3) + 2,$$

d'où :

$$3 \cdot 2^{p+2} = 10(12k + 9) + 6.$$

On en déduit :

$$\mu_z = \frac{3 \cdot 2^{p+2}}{10} = 12k + 9 + \frac{3}{5}.$$

Donc $m_z = \lfloor \mu_z \rfloor + 1$ et :

$$\mu_z = \lfloor \mu_z \rfloor + \frac{3}{5} = m_z - 1 + \frac{3}{5}.$$

Ainsi :

$$\mu_\sigma = m_z - 1 + \frac{3}{5} + \frac{3}{20} = m_z - 1 + \frac{3}{4},$$

d'où l'on déduit $\lfloor \mu_\sigma \rfloor = m_z - 1$ et $m_\sigma = \lfloor \mu_\sigma \rfloor + 1 = m_z$, d'où $\hat{\sigma} = \hat{z}$.

C.4.3 Premier cas pair

Supposons maintenant que p soit de la forme $4q + 2$. Alors il existe $k \in \mathbb{N}$ tel que :

$$2^p = 2^{4q+2} = 10k + 4.$$

Ainsi :

$$2^{p+4} = 10 \cdot 16k + 64 = 10(16k + 6) + 4.$$

Donc :

$$\mu_x = \frac{2^{p+4}}{10} = 16k + 6 + \frac{2}{5}.$$

Ainsi $m_x = \lfloor \mu_x \rfloor$ et :

$$\mu_x = m_x + \frac{2}{5}.$$

On en déduit :

$$\mu_z = \mu_\sigma + \frac{3}{10}.$$

Reprenons nos puissances de deux :

$$2^{p+2} = 10 \cdot 4k + 16 = 10(4k + 1) + 6.$$

Ainsi :

$$3 \cdot 2^{p+2} = 10(12k + 3) + 18 = 10(12k + 4) + 8.$$

Donc :

$$\mu_z = 12k + 4 + \frac{4}{5},$$

d'où l'on déduit :

$$m_z = \lfloor \mu_z \rfloor + 1 = 12k + 5,$$

dont on peut déjà remarquer qu'il est impair. Reprenons :

$$\mu_z = \lfloor \mu_z \rfloor + \frac{4}{5} = m_z - 1 + \frac{4}{5}.$$

Ainsi :

$$\mu_\sigma = \mu_z - \frac{3}{10} = m_z - 1 + \frac{4}{5} - \frac{3}{10} = m_z - 1 + \frac{1}{2},$$

donc $\lfloor \mu_\sigma \rfloor = m_z - 1$. La partie fractionnaire étant exactement $\frac{1}{2}$, on doit arbitrer et choisir l'entier pair le plus proche. Comme m_z est impair, on obtient $m_\sigma = m_z - 1$. En particulier, $m_\sigma \neq m_z$, d'où $\hat{\sigma} \neq \hat{z}$.

C.4.4 Second cas pair

Reste un dernier cas : $p = 4q + 4$. Dans ce cas, on trouve $k \in \mathbb{N}$ tel que :

$$2^p = 2^{4q+4} = 10k + 6.$$

Alors :

$$2^{p+4} = 10 \cdot 16k + 96 = 10(16k + 9) + 6,$$

d'où :

$$\mu_x = \frac{2^{p+4}}{10} = 16k + 9 + \frac{3}{5}.$$

En particulier, $m_x = \lfloor \mu_x \rfloor + 1$ et :

$$\mu_x = m_x - 1 + \frac{3}{5} = m_x - \frac{2}{5}.$$

On en déduit :

$$\mu_\sigma = \mu_z + \frac{3}{10}.$$

Par ailleurs :

$$2^{p+2} = 10 \cdot 4k + 24 = 10(4k + 2) + 4,$$

d'où :

$$3 \cdot 2^{p+2} = 10(12k + 6) + 12 = 10(12k + 7) + 2,$$

donc :

$$\mu_z = 12k + 7 + \frac{1}{5}.$$

En particulier, $m_z = \lfloor \mu_z \rfloor = 12k + 7$ qui est impair, et :

$$\mu_z = m_z + \frac{1}{5}.$$

On obtient alors :

$$\mu_\sigma = m_z + \frac{1}{5} + \frac{3}{10} = m_z + \frac{1}{2}.$$

Donc $\lfloor \mu_\sigma \rfloor = m_z$, mais on doit arbitrer et arrondir au pair, ce que m_z n'est pas, donc $m_\sigma = \lfloor \mu_\sigma \rfloor + 1 = m_z + 1$, donc $\hat{\sigma} \neq \hat{z}$.

C.5 Dernière remarque

L'utilisation des unités des puissances de deux permet aussi de montrer assez facilement que 0.1 ne peut être représenté de manière exacte en base 2 en précision finie. En effet, on peut toujours écrire :

$$\mu_x = \frac{2^{p+4}}{10}.$$

Dire que x peut être représenté de manière exacte signifie que l'on a $x = \hat{x}$. Comme $x = 2^{-4-p}\mu_x$ et $\hat{x} = 2^{-4-p}m_x$, cette égalité est équivalente à avoir $\mu_x = m_x$, ce qui arrive si et seulement si $\mu_x \in \mathbb{Z}$.

Or, quel que soit $p \in \mathbb{N}^*$, il existe $k \in \mathbb{N}$ tel que :

$$2^{p+4} = 10k + d, \quad \text{avec } d \in \{2, 4, 8, 6\}.$$

Ainsi :

$$\mu_x = k + \frac{d}{10}.$$

Pour que μ_x soit entier, il faudrait donc avoir $\frac{d}{10} = 0$, soit encore $d = 0$, ce qui n'arrive jamais.

Annexe D

Factorisation LU des matrices tridiagonales

On considère $A \in M_n(\mathbb{R})$ tridiagonale admettant une factorisation LU. On va montrer que cette structure est idéale pour la factorisation LU : les matrices L et U seront elles bidiagonales, ce qui est plutôt intéressant en terme de stockage. En bonus, on va également montrer que la surdiagonale de U n'est autre que celle de A .

On s'appuie pour cela sur l'[Algorithme 3.4.3](#), et on procède par récurrence sur $k \in \llbracket 1, n \rrbracket$: on va montrer le résultat ligne par ligne pour U , et colonne par colonne pour L .

La première ligne de U n'est autre que celle de A , et la première colonne de L est celle de A , divisée par A_{11} . Le résultat est donc vrai pour $k = 1$. Supposons qu'il le soit jusqu'à un certain $k \in \llbracket 1, n - 1 \rrbracket$. La $(k + 1)$ -ème ligne de U peut être définie par :

$$\forall j \in \llbracket k + 1, n \rrbracket, \quad U_{k+1,j} = A_{k+1,j} - \sum_{i=1}^k L_{k+1,i} U_{i,j}.$$

Pour tout $i \in \llbracket 1, k \rrbracket$, par hypothèse de récurrence, si $j \notin \{i, i + 1\}$, alors $U_{ij} = 0$. Pour $i \in \llbracket 1, k - 1 \rrbracket$, on a $i + 1 \leq k < k + 1 \leq j$, donc $U_{ij} = 0$. Il ne reste alors au plus que le cas $i = k$ dans la somme :

$$\forall j \in \llbracket k + 1, n \rrbracket, \quad U_{k+1,j} = A_{k+1,j} - L_{k+1,k} U_{k,j}.$$

Pour $j \geq k + 2 > k + 1$, $U_{k,j} = 0$ et ainsi :

$$U_{k+1,j} = A_{k+1,j},$$

ce qui montre que la $(k + 1)$ -ème ligne de U a bien la structure annoncée.

La $(k + 1)$ -ème colonne de L peut quant à elle être définie par :

$$\forall i \in \llbracket k + 2, n \rrbracket, \quad L_{i,k+1} = \frac{1}{U_{k+1,k+1}} \left(A_{i,k+1} - \sum_{j=1}^k L_{ij} U_{j,k+1} \right).$$

Comme pour U , pour $j \in \llbracket 1, k \rrbracket$, par hypothèse de récurrence, si $i \notin \{j, j+1\}$, alors $L_{ij} = 0$. Or, pour $j \in \llbracket 1, k \rrbracket$, $j+1 \leq k+1 < k+2 \leq i$, donc $L_{ij} = 0$, donc la somme est nulle et :

$$\forall i \in \llbracket k+2, n \rrbracket, \quad L_{i,k+1} = \frac{A_{i,k+1}}{U_{k+1,k+1}}.$$

Par hypothèse sur A , on obtient là encore la structure annoncée.

Les matrices L et U ayant une structure aussi creuse que celle de A , il peut être intéressant d'adapter l'algorithme de construction de ces matrices : la version générale va en effet devoir calculer inutilement plein de coefficients qui ne sont rien d'autre que zéro. Comme L et U sont bidiagonales, il n'y a que quatre diagonales à construire au total. Mieux encore, la diagonale principale de L n'est constituée que de 1, donc il n'y a rien à calculer, et il en va de même pour la surdiagonale de U qui est la même que celle de A . Il ne reste donc que deux diagonales à construire : la sous-diagonale de L , et la diagonale principale de U , pour lesquelles on va simplement réutiliser les formules connues (en prenant bien soin de ne pas écrire des sommes inutilement longues !). On obtient l'[Algorithme D.0.1](#).

Algorithme D.0.1 : Factorisation LU d'une matrice tridiagonale

```

 $U := 0$ 
 $L := I$ 
pour  $j$  de 1 à 2
   $U_{1j} := A_{1j}$ 
pour  $k$  de 2 à  $n$ 
   $L_{k,k-1} := \frac{A_{k,k-1}}{U_{k-1,k-1}}$ 
   $U_{kk} := A_{kk} - L_{k,k-1}U_{k-1,k}$ 
   $U_{k,k+1} := A_{k,k+1}$ 

```

On notera que le cas $k = n$ est en réalité un peu particulier pour la matrice U , puisque le coefficient $U_{n,n+1}$ n'existe pas. Comme il ne fait intervenir aucune opération élémentaire, dans un souci de simplicité, on choisit d'ignorer ce cas particulier.

L'initialisation de la première ligne de U ne nécessite aucune opération élémentaire. Pour chaque $k \in \llbracket 2, n \rrbracket$, on compte 1 division pour le coefficient sous-diagonal de L , ainsi que 1 multiplication et 1 addition pour le coefficient diagonal de U , et rien pour le coefficient surdiagonal de U . Au total, on obtient :

$$\sum_{k=2}^n (1 + 1 + 1) = 3(n-1) = 3n - 3 \text{ flops.}$$

Intéressons-nous maintenant aux aspects numériques : en pratique, avec les erreurs d'arrondis, on n'obtiendra pas exactement L et U . Comme on n'effectue que peu d'opérations, on devrait cependant ne pas en être trop éloignés. Vérifions cela.

À chaque étape $k \in \llbracket 2, n \rrbracket$, on calcule le coefficient sous-diagonal de L avec la formule :

$$L_{k,k-1} := \frac{A_{k,k-1}}{U_{k-1,k-1}}.$$

En appliquant le [Lemme 3.2.7](#), on trouve ainsi $\theta_1^{k,k-1}$ tel que $|\theta_1^{k,k-1}| \leq \gamma_1$ et :

$$\hat{L}_{k,k-1} \hat{U}_{k-1,k-1} (1 + \theta_1^{k,k-1}) = A_{k,k-1}.$$

Autrement dit :

$$\underbrace{\hat{L}_{k,k-1} \hat{U}_{k-1,k-1}}_{(\hat{L}\hat{U})_{k,k-1}} = A_{k,k-1} - \underbrace{\hat{L}_{k,k-1} \hat{U}_{k-1,k-1} \theta_1^{k,k-1}}_{=:(\delta_A)_{k,k-1}}.$$

Le coefficient diagonal de U est calculé par la formule :

$$U_{kk} := A_{kk} - L_{k,k-1} U_{k-1,k}.$$

Toujours par application du même lemme, on trouve $\theta_1^{k,k}$ et $\tilde{\theta}_1^{k,k}$ tels que :

$$\hat{U}_{kk} (1 + \theta_1^{k,k}) = A_{kk} - \hat{L}_{k,k-1} \hat{U}_{k,k-1} (1 + \tilde{\theta}_1^{k,k}).$$

Comme $\hat{L}_{kk} = 1$, on obtient là encore :

$$(\hat{L}\hat{U})_{kk} = A_{kk} + (\delta_A)_{kk}.$$

La définition du coefficient sur-diagonal de U n'amène à aucune erreur d'arrondi. Toujours en utilisant $\hat{L}_{kk} = 1$, on obtient :

$$(\hat{L}\hat{U})_{k,k+1} = A_{k,k+1}.$$

Annexe E

Préconditionnement du Laplacien discret

On s'intéresse à notre matrice tridiagonale préférée :

$$A := \begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{pmatrix},$$

de taille $n \times n$.

E.1 Éléments propres

La matrice A est symétrique de manière évidente. Montrons qu'elle est également définie positive. Soit pour cela $x \in \mathbb{R}^n$. On note $x = (x_1, \dots, x_n)^T$ et on pose $x_0 = x_{n+1} = 0$

pour simplifier la suite. Ainsi :

$$\begin{aligned}
(Ax, x) &= \sum_{i=1}^n (-x_{i-1} + 2x_i - x_{i+1})x_i \\
&= \sum_{i=1}^n (-x_{i-1}x_i + x_i^2) + \sum_{i=1}^n (x_i^2 - x_{i+1}x_i) \\
&= \sum_{i=1}^n (-x_{i-1}x_i + x_i^2) + \sum_{i=2}^{n+1} (x_{i-1}^2 - x_i x_{i-1}) \\
&= -\underbrace{x_0}_{=0}x_1 + x_1^2 + \sum_{i=2}^n (x_i^2 - 2x_i x_{i-1} + x_{i-1}^2) + x_n^2 - x_n \underbrace{x_{n+1}}_{=0} \\
&= (x_1 - x_0)^2 + \sum_{i=2}^n (x_i - x_{i-1})^2 + (x_{n+1} - x_n)^2 \\
&= \sum_{i=1}^{n+1} (x_i - x_{i-1})^2 \geq 0.
\end{aligned}$$

De plus, si $(Ax, x) = 0$, alors :

$$\forall i \in \llbracket 1, n+1 \rrbracket, \quad (x_i - x_{i-1})^2 = 0,$$

donc tous les x_i sont égaux. Comme $x_0 = x_{n+1} = 0$, ils sont tous nuls, donc $x = 0$.

Soit $\lambda \in \text{Sp}(A)$, et soit $x \in \mathbb{R}^n \setminus \{0\}$ un vecteur propre associé. Commençons par montrer que $\lambda \in \mathbb{R}_*^+$. En effet, par définition de λ et de x , on a :

$$Ax = \lambda x.$$

La transposée de cette égalité donne alors :

$$x^T A^T = \lambda x^T.$$

On multiplie à droite par x :

$$x^T A^T x = \lambda x^T x.$$

À droite, on reconnaît $x^T x = \|x\|^2$. À gauche, comme A est symétrique, on trouve $(Ax, x) \in \mathbb{R}_*^+$ par ce qui précède. Donc $\lambda \in \mathbb{R}_*^+$.

Par ailleurs, le théorème de Gerschgorin permet de montrer que λ est dans l'union de tous les disques de Gerschgorin, union qui n'est ici rien d'autre que le disque centré en 2 et de rayon 2. On a donc obtenu :

$$\text{Sp}(A) \subset]0, 4].$$

Nous allons chercher à expliciter x et λ . On reprend la notation $x = (x_1, \dots, x_n)^T$ ainsi que la convention $x_0 = x_{n+1} = 0$. Comme $Ax = \lambda x$, on trouve ainsi :

$$\forall k \in \llbracket 1, n \rrbracket, \quad -x_{k-1} + 2x_k - x_{k+1} = \lambda x_k.$$

En rassemblant les termes en x_k et en décalant l'indice, on obtient :

$$\forall k \in \llbracket 0, n-1 \rrbracket, \quad -x_k + (2 - \lambda)x_{k+1} - x_{k+2} = 0.$$

On définit alors le polynôme caractéristique associé :

$$P(X) = -X^2 + (2 - \lambda)X - 1.$$

Soit Δ son discriminant :

$$\Delta = (2 - \lambda)^2 - 4.$$

Or, $\lambda \in]0, 4]$, donc :

$$|2 - \lambda| \leq 2,$$

soit :

$$\Delta = (2 - \lambda)^2 - 4 \leq 0.$$

Supposons $\Delta = 0$. Alors P admet une racine réelle double r et il existe $(\alpha, \beta) \in \mathbb{R}^2$ tel que :

$$\forall k \in \llbracket 0, n+1 \rrbracket, \quad x_k = (\alpha + \beta k)r^k.$$

En considérant $k = 0$, on obtient :

$$0 = x_0 = \alpha,$$

donc :

$$\forall k \in \llbracket 0, n+1 \rrbracket, \quad x_k = \beta k r^k.$$

Considérons alors $k = n+1$:

$$0 = x_{n+1} = \beta(n+1)r^{n+1}.$$

Donc, soit $\beta = 0$, soit $r = 0$: dans tous les cas, $x = 0$, ce qui contredit sa définition. Ainsi, le cas $\Delta = 0$ n'est pas possible.

Comme $\Delta < 0$, le polynôme P , qui est à coefficients réels (puisque λ est réelle), admet ainsi deux racines complexes conjuguées, que l'on notera r et $\bar{r}$. Or :

$$P(X) = -X^2 + (2 - \lambda)X - 1 = -(X - r)(X - \bar{r}) = -X^2 + (r + \bar{r})X - |r|^2.$$

Par identification, on obtient notamment :

$$|r| = 1,$$

donc r est de module 1. Ainsi, il existe $\theta \in \mathbb{R}$ tel que :

$$r = e^{i\theta}.$$

On trouve ainsi $(\alpha, \beta) \in \mathbb{R}^2$:

$$\forall k \in \llbracket 0, n+1 \rrbracket, \quad x_k = \alpha \cos(k\theta) + \beta \sin(k\theta).$$

Considérons $k = 0$:

$$0 = x_0 = \alpha.$$

Ainsi :

$$\forall k \in \llbracket 0, n+1 \rrbracket, \quad x_k = \beta \sin(k\theta).$$

Par ailleurs :

$$0 = x_{n+1} = \beta \sin((n+1)\theta).$$

Comme $x \neq 0$, on ne peut pas avoir $\beta = 0$. Donc $\sin((n+1)\theta) = 0$. Ainsi, il existe $m \in \mathbb{Z}$ tel que :

$$(n+1)\theta = m\pi,$$

d'où :

$$\theta = \frac{m\pi}{n+1}.$$

De plus, on ne peut avoir $m \in (n+1)\mathbb{Z}$, car sinon on aurait également $x = 0$. Finalement, on a donc obtenu :

$$\forall k \in \llbracket 0, n+1 \rrbracket, \quad x_k = \beta \sin\left(k \frac{m\pi}{n+1}\right).$$

De l'identification polynômiale réalisée plus haut, on peut également déduire :

$$r + \bar{r} = 2 - \lambda.$$

Ainsi :

$$\lambda = 2 - (r + \bar{r}) = 2 - (e^{i\theta} + e^{-i\theta}) = 2 - 2 \cos \theta.$$

De l'expression de θ trouvée précédemment, on tire donc :

$$\lambda = 2 - 2 \cos\left(\frac{m\pi}{n+1}\right).$$

Notons $y := \frac{m\pi}{n+1}$, de sorte que $\lambda = 2 - 2 \cos y$. Faisons un peu de trigonométrie :

$$\cos y = \cos\left(2 \frac{y}{2}\right) = \cos^2 \frac{y}{2} - \sin^2 \frac{y}{2} = 1 - 2 \sin^2 \frac{y}{2}.$$

On en déduit :

$$\lambda = 2 - 2 \cos\left(\frac{m\pi}{n+1}\right) = 4 \sin^2\left(\frac{m\pi}{2(n+1)}\right).$$

Bien évidemment, on n'a pas obtenu une infinité de valeurs propres : on peut simplement considérer $m \in \llbracket 1, n \rrbracket$ (les cas $m = 0$ et $m = n+1$ étant exclus par ce qui précède), les autres valeurs étant les mêmes.

E.2 Un mauvais conditionnement

La matrice A étant symétrique, on peut obtenir son conditionnement en norme 2 par la formule :

$$\text{cond}_2(A) = \frac{\max_{m \in \llbracket 1, n \rrbracket} |\Lambda_m|}{\min_{m \in \llbracket 1, n \rrbracket} |\Lambda_m|}.$$

Or, on connaît les valeurs propres de A qui sont telles que :

$$0 < \Lambda_1 < \Lambda_2 < \dots < \Lambda_n.$$

Ainsi :

$$\text{cond}_2(A) = \frac{\Lambda_n}{\Lambda_1} = \frac{4 \sin^2 \left(\frac{n\pi}{2(n+1)} \right)}{4 \sin^2 \left(\frac{\pi}{2(n+1)} \right)} = \frac{\sin^2 \left(\frac{n\pi}{2(n+1)} \right)}{\sin^2 \left(\frac{\pi}{2(n+1)} \right)}.$$

D'une part, on a :

$$\lim_{n \rightarrow +\infty} \sin^2 \left(\frac{n\pi}{2(n+1)} \right) = \sin^2 \frac{\pi}{2} = 1.$$

Ainsi :

$$\forall \varepsilon > 0, \exists N \in \mathbb{N} / \forall n \geq N, -\varepsilon \leq \sin^2 \left(\frac{n\pi}{2(n+1)} \right) - 1 \leq \varepsilon.$$

En prenant par exemple $\varepsilon = \frac{1}{2}$, on a donc pour n assez grand :

$$\sin^2 \left(\frac{n\pi}{2(n+1)} \right) \geq \frac{1}{2}.$$

D'autre part, pour $x \in [0, \pi]$, on peut montrer $0 \leq \sin x \leq x$, donc $\sin^2 x \leq x^2$ et donc, dans notre cas :

$$\sin^2 \left(\frac{\pi}{2(n+1)} \right) \leq \frac{\pi^2}{4(n+1)^2}.$$

En combinant ces deux inégalités, on obtient donc, pour n assez grand :

$$\text{cond}_2(A) \geq \frac{2(n+1)^2}{\pi^2} = c(n+1)^2$$

avec $c := \frac{2}{\pi^2}$.

Pour $a < b$, on pose :

$$A_h := \frac{1}{h^2} A, \quad \text{avec } h := \frac{b-a}{n+1}.$$

Comme on sait que le conditionnement d'une matrice est invariant par multiplication par un scalaire, on obtient pour n assez grand (et donc pour h assez petit) :

$$\text{cond}_2(A_h) = \text{cond}_2(A) \geq c(n+1)^2 = c \frac{(b-a)^2}{h^2} = \frac{\tilde{c}}{h^2}$$

avec $\tilde{c} := c(b-a)^2$.

Par ailleurs, pour obtenir les valeurs propres de A_h , il suffit de multiplier celles de A par $\frac{1}{h^2}$. On obtient :

$$\forall m \in \llbracket 1, n \rrbracket, \quad \lambda_m = \frac{4}{h^2} \sin^2 \left(\frac{m\pi}{2(b-a)} h \right).$$

E.3 Préconditionnement

La matrice A_h étant symétrique définie positive, on peut utiliser le gradient conjugué pour résoudre les systèmes linéaires associés. L'analyse numérique nous apprend que la résolution de l'EDP sera plus précise en prenant h petit. Et tout le problème est là : avec l'inégalité obtenue précédemment, on constate que le conditionnement augmente proportionnellement à l'inverse du carré de h , ce qui est plutôt handicapant puisque cela signifie que le taux de convergence sera proche de 1 assez vite.

On cherche alors une matrice C symétrique définie positive permettant d'améliorer le conditionnement du système. On a déjà vu dans la section dédiée au gradient conjugué que le système qui nous intéresse est défini par la matrice $C^{-1/2} A_h C^{-1/2}$: il faut que cette matrice ait un meilleur conditionnement que celui de A_h . Décomposons A_h de la manière habituelle :

$$A_h = D - E - E^T.$$

Un choix naturel consiste à tenter le préconditionneur de Jacobi : $C = D$. On va montrer que c'est inutile. En effet, la diagonale de A est constante : elle est uniquement constituée de 2. La diagonale de A_h est donc elle aussi constante. Plus précisément, on a :

$$C = D = \frac{2}{h^2} I.$$

On en déduit directement :

$$C^{1/2} = \frac{\sqrt{2}}{h} I,$$

donc :

$$C^{-1/2} A_h C^{-1/2} = \frac{h^2}{2} A_h,$$

et on se souvient alors que le conditionnement est invariant par multiplication par un scalaire donc, quelle que soit la norme :

$$\text{cond}(C^{-1/2} A_h C^{-1/2}) = \text{cond}(A_h).$$

On aura donc peut-être un peu plus de chance avec le préconditionneur SSOR. On pose $C = TT^T$ où :

$$T := \frac{1}{\sqrt{\omega}\sqrt{2-\omega}} (D - \omega E) D^{-1/2}, \quad \text{avec } \omega \in]0, 2[.$$

Soit $x \in \mathbb{R}^n \setminus \{0\}$. Alors :

$$(Dx, x) = \left(\frac{2}{h^2} Ix, x \right) = \frac{2}{h^2} \|x\|^2.$$

Par ailleurs, on peut montrer via le quotient de Rayleigh que :

$$(A_h x, x) \geq \lambda_{\min} \|x\|^2,$$

avec $\lambda_{\min}$ la plus petite valeur propre de A_h . Ici, on obtient :

$$(A_h x, x) \geq \lambda_1 \|x\|^2 = \frac{4}{h^2} \sin^2 \left(\frac{\pi}{2(b-a)} h \right) \|x\|^2.$$

Ainsi :

$$\frac{(Dx, x)}{(A_h x, x)} \leq \frac{\frac{2}{h^2} \|x\|^2}{\frac{4}{h^2} \sin^2 \left(\frac{\pi}{2(b-a)} h \right) \|x\|^2} = \frac{1}{2 \sin^2 \left(\frac{\pi}{2(b-a)} h \right)} =: \mu.$$

Par ailleurs, toujours pour $x \in \mathbb{R}^n \setminus \{0\}$, on peut écrire :

$$(x, ED^{-1}E^T x) = (E^T x, D^{-1}E^T x).$$

Comme $D = \frac{2}{h^2} I$, on obtient ainsi :

$$(x, ED^{-1}E^T x) = \frac{h^2}{2} \|E^T x\|^2.$$

Or E est l'opposée de la partie triangulaire inférieure stricte de A_h , ce qui peut s'écrire :

$$E = \frac{1}{h^2} \begin{pmatrix} 0 & & & \\ 1 & 0 & & \\ & \ddots & \ddots & \\ & & 1 & 0 \end{pmatrix}, \quad \text{donc} \quad E^T = \frac{1}{h^2} \begin{pmatrix} 0 & 1 & & \\ & \ddots & \ddots & \\ & & 0 & 1 \\ & & & 0 \end{pmatrix}.$$

Autrement dit :

$$E^T x = \frac{1}{h^2} \begin{pmatrix} x_2 \\ \vdots \\ x_n \\ 0 \end{pmatrix},$$

et ainsi :

$$\|E^T x\|^2 = \frac{1}{h^4} \sum_{i=2}^n x_i^2 \leq \frac{1}{h^4} \|x\|^2.$$

Ainsi on a obtenu :

$$(x, ED^{-1}E^T x) \leq \frac{1}{2h^2} \|x\|^2.$$

On en déduit :

$$(x, ED^{-1}E^T x) - \frac{1}{4}(Dx, x) \leq \frac{1}{2h^2}\|x\|^2 - \frac{1}{4}(Dx, x) = \frac{1}{2h^2}\|x\|^2 - \frac{1}{2h^2}\|x\|^2 = 0.$$

Comme $(Ax, x) > 0$, on en déduit notamment :

$$\frac{(x, ED^{-1}E^T x) - \frac{1}{4}(Dx, x)}{(Ax, x)} \leq 0.$$

On peut appliquer la [Proposition 5.2.28](#) :

$$\forall x \in \mathbb{R}^n \setminus \{0\}, \quad \frac{2 - \omega}{1 + \mu \frac{(2-\omega)^2}{4\omega}} \leq \frac{(A_h x, x)}{(C x, x)} \leq 1.$$

On utilise alors la [Proposition 5.2.24](#) et la [Proposition 5.2.26](#) :

$$\text{cond}_2(C^{-1/2} A_h C^{-1/2}) \leq \frac{1}{\frac{2-\omega}{1 + \mu \frac{(2-\omega)^2}{4\omega}}} = \frac{1 + \mu \frac{(2-\omega)^2}{4\omega}}{2 - \omega}.$$

Il reste maintenant à chercher à obtenir la meilleure borne possible. On pose pour cela

$$f: \omega \mapsto \frac{1 + \mu \frac{(2-\omega)^2}{4\omega}}{2 - \omega} = \frac{1}{2 - \omega} + \mu \frac{2 - \omega}{4\omega}.$$

Cherchons les éventuels points critiques de f . On calcule pour cela :

$$f'(\omega) = \frac{1}{(2 - \omega)^2} - \frac{\mu}{2\omega^2}.$$

On peut alors résoudre l'équation $f'(\omega) = 0$, ce qui donne :

$$\mu = \frac{2\omega^2}{(2 - \omega)^2} = \left(\frac{\sqrt{2}\omega}{2 - \omega} \right)^2.$$

Ainsi :

$$\frac{\sqrt{2}\omega}{2 - \omega} = \pm \sqrt{\mu}.$$

Comme $\omega \in]0, 2[$, on trouve :

$$\frac{\sqrt{2}\omega}{2 - \omega} = \sqrt{\mu}.$$

On obtient alors :

$$\omega_{\text{opt}} = \frac{2\sqrt{\mu}}{\sqrt{2} + \sqrt{\mu}}.$$

On peut par ailleurs montrer qu'il s'agit d'un minimum puisque :

$$f''(\omega) = \frac{2}{(2-\omega)^3} + \frac{\mu}{\omega^3} > 0.$$

Explicitons un peu plus la valeur de ω_{opt} en utilisant la valeur de μ :

$$\omega_{\text{opt}} = \frac{2}{\sqrt{\frac{2}{\mu}} + 1}.$$

Or :

$$\frac{2}{\mu} = 4 \sin^2 \left(\frac{\pi}{2(b-a)} h \right).$$

Comme $\sin \left(\frac{\pi}{2(n+1)} \right) \geq 0$, on obtient :

$$\omega_{\text{opt}} = \frac{2}{2 \sin \left(\frac{\pi}{2(b-a)} h \right) + 1}.$$

Pour alléger la suite, on note :

$$s := \sin \left(\frac{\pi}{2(b-a)} h \right).$$

Ainsi :

$$\omega_{\text{opt}} = \frac{2}{2s + 1}.$$

Donc :

$$2 - \omega_{\text{opt}} = 2 - \frac{2}{2s + 1} = \frac{4s}{2s + 1},$$

d'où :

$$\frac{2 - \omega_{\text{opt}}}{4\omega_{\text{opt}}} = \frac{4s}{2s + 1} \frac{1}{4} \frac{2s + 1}{2} = \frac{s}{2}.$$

On trouve alors :

$$f(\omega_{\text{opt}}) = \frac{1}{2 - \omega_{\text{opt}}} + \mu \frac{2 - \omega_{\text{opt}}}{4\omega_{\text{opt}}} = \frac{2s + 1}{4s} + \mu \frac{s}{2}.$$

Or :

$$\mu = \frac{1}{2s^2},$$

donc :

$$f(\omega_{\text{opt}}) = \frac{2s + 1}{4s} + \frac{1}{4s} = \frac{s + 1}{2s} = \frac{1 + \sin \left(\frac{\pi}{2(b-a)} h \right)}{2 \sin \left(\frac{\pi}{2(b-a)} h \right)}.$$

Rappelons que, pour x proche de zéro, on a :

$$\sin x = x + o(x).$$

Posons, toujours pour alléger les notations :

$$p := \frac{\pi}{2(b-a)},$$

de sorte que :

$$\sin\left(\frac{\pi}{2(b-a)}h\right) = \sin(ph).$$

Pour h petit, on a alors :

$$\sin(ph) = ph + o(ph).$$

Ainsi :

$$f(\omega_{\text{opt}}) = \frac{1 + ph + o(h)}{2ph + o(h)}.$$

On en déduit :

$$2phf(\omega_{\text{opt}}) = \frac{1 + ph + o(h)}{1 + o(1)}.$$

Ainsi :

$$\lim_{h \rightarrow 0} 2phf(\omega_{\text{opt}}) = 1,$$

soit encore :

$$\lim_{h \rightarrow 0} \frac{\pi}{b-a} hf(\omega_{\text{opt}}) = 1,$$

ou même :

$$\lim_{h \rightarrow 0} \frac{f(\omega_{\text{opt}})}{\frac{1}{\frac{\pi}{b-a}h}} = 1,$$

ce qui montre que $f(\omega_{\text{opt}})$ est équivalent à $\frac{b-a}{\pi h}$ en zéro : d'un conditionnement en $\frac{1}{h^2}$, on est donc passés à du $\frac{1}{h}$.

Annexe F

Gradient biconjugué

Soit $n \in \mathbb{N}^*$. Soit $A \in M_n(\mathbb{R})$ inversible. On s'intéresse à la résolution du système

$$Ax = b$$

avec $b \in \mathbb{R}^n$ donné. Plus précisément, on va s'intéresser à la méthode du gradient biconjugué. Soit $x^{(0)} \in \mathbb{R}^n$ une donnée initiale. On va construire la suite d'itérés $(x^{(k)})_{k \in \mathbb{N}}$ par récurrence :

$$\forall k \in \mathbb{N}, \quad x^{(k+1)} := x^{(k)} + \gamma^{(k)} d^{(k)},$$

où $\gamma^{(k)} \in \mathbb{R}^*$ est le pas et $d^{(k)} \in \mathbb{R}^n$ est la direction de descente. La suite des directions de descente est elle-même construite par récurrence :

$$\forall k \in \mathbb{N}, \quad d^{(k+1)} := r^{(k+1)} + \alpha_k d^{(k)},$$

où $\alpha_k \in \mathbb{R}^*$ et $d^{(0)} := r^{(0)}$. On reprend les espaces R_k , D_k et K_k définis dans la [Sous-section 5.2.4](#).

F.1 Premières propriétés

Comme on l'a fait pour le gradient conjugué, on peut faire en sorte de ne pas faire appel à la définition du résidu à chaque étape : cette suite vérifie elle-même une relation de récurrence. En effet :

$$\begin{aligned} r^{(k+1)} &= b - Ax^{(k+1)} \\ &= b - Ax^{(k)} - \gamma^{(k)} Ad^{(k)} \\ &= r^{(k)} - \gamma^{(k)} Ad^{(k)}. \end{aligned}$$

Proposition F.1.1. La suite des directions de descente vérifie :

$$\forall k \in \mathbb{N}, \quad d^{(k)} \in r^{(k)} + R_{k-1},$$

| où $R_{-1} := \{0\}$.

Démonstration. On procède par récurrence sur $k \in \mathbb{N}$. Pour $k = 0$, comme $d^{(0)} = r^{(0)}$, le résultat est évident. Supposons qu'il soit vrai pour un certain rang $k \in \mathbb{N}$. Alors :

$$d^{(k+1)} = d^{(k+1)} + \gamma^{(k)} d^{(k)} \in r^{(k+1)} + R_k,$$

car $d^{(k)} \in R_k$ par hypothèse de récurrence. ■

Tout comme pour le gradient conjugué, on peut montrer que les trois espaces vectoriels définis sont en réalité les mêmes.

Proposition F.1.2. On a :

$$\forall k \in \mathbb{N}, \quad R_k = D_k = K_k.$$

Démonstration. On procède par récurrence sur $k \in \mathbb{N}$. Pour $k = 0$, comme $d^{(0)} = r^{(0)}$, le résultat est direct. Supposons qu'il existe $k \in \mathbb{N}$ tel que $R_k = D_k = K_k$. On a déjà montré $d^{(k+1)} \in R_{k+1}$, donc $D_{k+1} \subset R_{k+1}$. De plus :

$$r^{(k+1)} = d^{(k+1)} - \gamma^{(k)} d^{(k)} \in D_{k+1},$$

donc $R_{k+1} \subset D_{k+1}$. On a ainsi montré $R_{k+1} = D_{k+1}$.

Comme $d^{(k)} \in D_k = K_k$ par hypothèse de récurrence, on peut écrire

$$d^{(k)} = \sum_{j=0}^k \beta_j A^j r^{(0)},$$

donc :

$$Ad^{(k)} = \sum_{j=0}^k \beta_j A^{j+1} r^{(0)}.$$

On en déduit :

$$r^{(k+1)} = \underbrace{r^{(k)}}_{\in R_k = K_k} - \gamma^{(k)} \underbrace{Ad^{(k)}}_{\in K_{k+1}} \in K_{k+1},$$

donc $R_{k+1} \subset K_{k+1}$.

Réécrivons maintenant :

$$Ad^{(k)} = \beta_k A^{k+1} r^{(0)} + \sum_{j=1}^k \beta_{j-1} A^j r^{(0)},$$

de sorte que :

$$\gamma^{(k)} \beta_k A^{k+1} r^{(0)} = \underbrace{r^{(k)} - r^{(k+1)}}_{\in R_{k+1}} - \underbrace{\gamma^{(k)} \sum_{j=1}^k \beta_{j-1} A^j r^{(0)}}_{\in K_k = R_k}.$$

Par hypothèse, $\gamma^{(k)} \neq 0$. De plus, si $\beta_k = 0$, alors $d^{(k)} \in K_{k-1} = D_{k-1}$, ce qui est impossible si $d^{(k)} \neq 0$, donc $\beta_k \neq 0$, et ainsi $A^{k+1} r^{(0)} \in R_{k+1}$, d'où $K_{k+1} \subset R_{k+1}$, et donc l'égalité. ■

F.2 Système artificiel

La méthode du gradient biconjugué consiste en réalité à résoudre deux systèmes à la fois : celui qui nous intéresse, et un autre, complètement artificiel et défini par la transposée de A . On considère ainsi le système

$$A^T \tilde{x} = \tilde{b}$$

où $\tilde{b} \in \mathbb{R}^n$ est arbitraire (et n'a plus vraiment d'importance une fois l'algorithme défini). On peut ainsi définir les itérés de ce système $(\tilde{x}^{(k)})_{k \in \mathbb{N}}$ par récurrence :

$$\forall k \in \mathbb{N}, \quad \tilde{x}^{(k+1)} := x^{(k)} + \gamma^{(k)} \tilde{d}^{(k)},$$

où $\gamma^{(k)}$ est le même que celui utilisé pour la « vraie » suite des itérés, et $\tilde{d}^{(k)} \in \mathbb{R}^n$ est une nouvelle direction de descente. Cette suite est construite à partir d'un certain $\tilde{x}^{(0)} \in \mathbb{R}^n$ qui n'a pas non plus vraiment d'importance ici puisque l'on ne va pas tarder à complètement oublier ces itérés artificiels. Avant cela, on va les utiliser pour obtenir une relation de récurrence sur les résidus artificiels :

$$\begin{aligned} \tilde{r}^{(k+1)} &= \tilde{b} - A^T \tilde{x}^{(k+1)} \\ &= \tilde{r}^{(k)} - \gamma^{(k)} A^T \tilde{d}^{(k)}. \end{aligned}$$

On obtient une relation similaire à celle obtenue pour les résidus initiaux. On définit ainsi les descentes artificielles de la même manière : partant de $\tilde{d}^{(0)} := \tilde{r}^{(0)}$, on pose

$$\tilde{d}^{(k+1)} := \tilde{r}^{(k+1)} + \alpha_k \tilde{d}^{(k)},$$

où les α_k sont les mêmes que ceux utilisés pour les descentes initiales.

Pour compléter la « copie » du système original, on définit, pour tout $k \in \mathbb{N}$:

$$\begin{aligned} \tilde{R}_k &:= \text{vect}\{\tilde{r}^{(0)}, \dots, \tilde{r}^{(k)}\}, \quad \tilde{D}_k := \text{vect}\{\tilde{d}^{(0)}, \dots, \tilde{d}^{(k)}\}, \\ \tilde{K}_k &:= \text{vect}\{\tilde{r}^{(0)}, A^T \tilde{r}^{(0)}, \dots, (A^T)^k \tilde{r}^{(0)}\}. \end{aligned}$$

Tout comme pour le système original, on peut montrer que tous ces espaces sont en réalité les mêmes. La preuve est exactement la même.

Proposition F.2.1. On a :

$$\forall k \in \mathbb{N}, \quad \tilde{R}_k = \tilde{D}_k = \tilde{K}_k.$$

F.3 Bi-orthogonalité et biconjugaison

La méthode du gradient biconjugué repose sur l'idée de faire en sorte que deux résidus successifs sont bi-orthogonaux et que deux directions de descente successives sont A -biconjuguées, ce qui se traduit par :

$$\forall k \in \mathbb{N}, \quad (r^{(k)}, \tilde{r}^{(k+1)}) = (\tilde{r}^{(k)}, r^{(k+1)}) = 0 \quad \text{et} \quad (A d^{(k)}, \tilde{d}^{(k+1)}) = (A^T \tilde{d}^{(k)}, d^{(k+1)}) = 0.$$

On peut expliciter les $\gamma^{(k)}$ et α_k permettant cela. Mais avant, voyons déjà un résultat important : le fait de forcer la bi-orthogonalité et la biconjugaison entre deux étapes successives implique en réalité que ces deux propriétés sont vraies à toutes les étapes.

Proposition F.3.1. Pour tout $k \in \mathbb{N}^*$ et pour tout $i \in \llbracket 0, k-1 \rrbracket$:

$$(r^{(i)}, \tilde{r}^{(k)}) = (\tilde{r}^{(i)}, r^{(k)}) = 0 \quad \text{et} \quad (Ad^{(i)}, \tilde{d}^{(k)}) = (A^T \tilde{d}^{(i)}, d^{(k)}) = 0.$$

Démonstration. On procède par récurrence sur $k \in \mathbb{N}^*$. Pour $k = 1$, il suffit d'utiliser l'hypothèse sur $\gamma^{(0)}$ et α_0 :

$$(r^{(0)}, \tilde{r}^{(1)}) = (\tilde{r}^{(0)}, r^{(1)}) = 0 \quad \text{et} \quad (Ad^{(0)}, \tilde{d}^{(1)}) = (A^T \tilde{d}^{(0)}, d^{(1)}) = 0.$$

Supposons le résultat vrai pour un certain $k \in \mathbb{N}^*$. Dans le cas où $i = k$, il s'agit encore une fois d'utiliser l'hypothèse sur $\gamma^{(k)}$ et α_k . Considérons alors $i \in \llbracket 0, k-1 \rrbracket$. En utilisant l'expression de $\tilde{r}^{(k+1)}$, on a pour commencer :

$$(r^{(i)}, \tilde{r}^{(k+1)}) = (r^{(i)}, \tilde{r}^{(k)}) - \gamma^{(k)}(r^{(i)}, A^T \tilde{d}^{(k)}).$$

Or, $(r^{(i)}, \tilde{r}^{(k)}) = 0$ par hypothèse de récurrence car $i < k$. Par ailleurs, $r^{(i)} \in R_i = D_i$, donc on peut écrire

$$r^{(i)} = \sum_{j=0}^i \beta_j d^{(j)}.$$

Ainsi :

$$(r^{(i)}, A^T \tilde{d}^{(k)}) = \sum_{j=0}^i \beta_j (d^{(j)}, A^T \tilde{d}^{(k)}).$$

Or, pour tout $j \in \llbracket 0, i \rrbracket$, $j \leq i < k$, donc $(d^{(j)}, A^T \tilde{d}^{(k)}) = 0$ par hypothèse de récurrence, et on a ainsi trouvé :

$$(r^{(i)}, \tilde{r}^{(k+1)}) = 0.$$

On montre de la même façon que $(\tilde{r}^{(i)}, r^{(k+1)}) = 0$.

Utilisons maintenant l'expression de $\tilde{d}^{(k+1)}$:

$$(Ad^{(i)}, \tilde{d}^{(k+1)}) = (Ad^{(i)}, \tilde{r}^{(k+1)}) + \alpha_k (Ad^{(i)}, \tilde{d}^{(k)}).$$

Comme $i < k$, on applique l'hypothèse de récurrence : $(Ad^{(i)}, \tilde{d}^{(k)}) = 0$. Par ailleurs, $d^{(i)} \in D_i = K_i$, donc $Ad^{(i)} \in K_{i+1} = R_{i+1}$, d'où le fait que l'on puisse écrire :

$$Ad^{(i)} = \sum_{j=0}^{i+1} \lambda_j r^{(j)}.$$

Ainsi :

$$(Ad^{(i)}, \tilde{r}^{(k+1)}) = \sum_{j=0}^{i+1} \lambda_j (r^{(j)}, \tilde{r}^{(k+1)}).$$

Par ce qui précède, on obtient, pour tout $j \in \llbracket 0, i+1 \rrbracket$, $(r^{(j)}, \tilde{r}^{(k+1)}) = 0$ car $j \leq i+1 \leq k$. On a donc obtenu

$$(Ad^{(i)}, \tilde{d}^{(k+1)}) = 0,$$

et un calcul similaire donne $(A^T \tilde{d}^{(i)}, d^{(k+1)}) = 0$. ■

On va maintenant pouvoir expliciter les coefficients. Commençons par $\gamma^{(k)}$. On cherche à forcer la contrainte :

$$(\tilde{r}^{(k)}, r^{(k+1)}) = 0.$$

L'expression de $r^{(k+1)}$ permet alors d'écrire :

$$\gamma^{(k)}(\tilde{r}^{(k)}, Ad^{(k)}) = (\tilde{r}^{(k)}, r^{(k)}).$$

Or :

$$\tilde{d}^{(k)} \in \tilde{r}^{(k)} + \tilde{R}_{k-1} = \tilde{r}^{(k)} + \tilde{D}_{k-1},$$

ce qui se traduit par :

$$\tilde{d}^{(k)} = \tilde{r}^{(k)} + \sum_{j=0}^{k-1} \beta_j \tilde{d}^{(j)}.$$

Ainsi :

$$(\tilde{r}^{(k)}, Ad^{(k)}) = (\tilde{d}^{(k)}, Ad^{(k)}) + \sum_{j=0}^{k-1} \beta_j (\tilde{d}^{(j)}, Ad^{(k)}).$$

Pour $j \in \llbracket 0, k-1 \rrbracket$, la relation de biconjugaison implique $(\tilde{d}^{(j)}, Ad^{(k)}) = 0$. Autrement dit :

$$(\tilde{r}^{(k)}, Ad^{(k)}) = (\tilde{d}^{(k)}, Ad^{(k)}).$$

En supposant $(\tilde{d}^{(k)}, Ad^{(k)}) \neq 0$, on obtient ainsi :

$$\gamma^{(k)} = \frac{(r^{(k)}, \tilde{r}^{(k)})}{(Ad^{(k)}, \tilde{d}^{(k)})}.$$

Remarque F.3.2. On obtient bien sûr la même expression en imposant $(r^{(k)}, \tilde{r}^{(k+1)}) = 0$.

Pour le coefficient α_k , on cherche à forcer la contrainte :

$$(Ad^{(k)}, \tilde{d}^{(k+1)}) = 0.$$

L'expression de $\tilde{d}^{(k+1)}$ donne alors :

$$\alpha_k(Ad^{(k)}, \tilde{d}^{(k)}) = -(Ad^{(k)}, \tilde{r}^{(k+1)}).$$

L'expression de $\tilde{r}^{(k+1)}$ permet cependant d'écrire :

$$Ad^{(k)} = \frac{1}{\gamma^{(k)}} (r^{(k)} - r^{(k+1)}).$$

Ainsi :

$$(Ad^{(k)}, \tilde{r}^{(k+1)}) = \frac{1}{\gamma^{(k)}} (r^{(k)}, \tilde{r}^{(k+1)}) - \frac{1}{\gamma^{(k)}} (r^{(k+1)}, \tilde{r}^{(k+1)}).$$

Or, $(r^{(k)}, \tilde{r}^{(k+1)}) = 0$ par bi-orthogonalité. En utilisant l'expression de $\gamma^{(k)}$ précédemment obtenue, on trouve alors :

$$(Ad^{(k)}, \tilde{r}^{(k+1)}) = -\frac{(Ad^{(k)}, \tilde{d}^{(k)})}{(r^{(k)}, \tilde{r}^{(k)})}(r^{(k+1)}, \tilde{r}^{(k+1)}).$$

On en déduit alors :

$$\alpha_k = \frac{(r^{(k+1)}, \tilde{r}^{(k+1)})}{(r^{(k)}, \tilde{r}^{(k)})}.$$

F.4 Algorithme

On a maintenant tous les ingrédients nécessaires pour écrire l'algorithme du gradient biconjugué. Ou presque : il reste à définir $\tilde{r}^{(0)}$. Une seule condition est vraiment nécessaire pour cela : garantir $(r^{(0)}, \tilde{r}^{(0)}) \neq 0$. Un choix relativement commun consiste à prendre $\tilde{r}^{(0)} := r^{(0)}$. On peut par ailleurs noter deux petites optimisations possibles : le produit $Ad^{(k)}$ est utilisé à deux reprises, ce qui justifie l'utilité de le stocker. On remarque également que le produit scalaire $(r^{(k)}, \tilde{r}^{(k)})$ est lui aussi réutilisé, ce qui justifie qu'on le stocke. Finalement, on obtient l'[Algorithme F.4.1](#).

Algorithme F.4.1 : Gradient biconjugué

```

 $r^{(0)} := b - Ax^{(0)}$ 
 $d^{(0)} := r^{(0)}$ 
 $\tilde{r}^{(0)} := r^{(0)}$ 
 $\tilde{d}^{(0)} := \tilde{r}^{(0)}$ 
 $\rho_0 := (r^{(0)}, \tilde{r}^{(0)})$ 
pour  $k \in \mathbb{N}$ 
     $z^{(k)} := Ad^{(k)}$ 
     $\gamma^{(k)} := \frac{\rho_k}{(z^{(k)}, \tilde{d}^{(k)})}$ 
     $x^{(k+1)} := x^{(k)} + \gamma^{(k)}d^{(k)}$ 
     $r^{(k+1)} := r^{(k)} - \gamma^{(k)}z^{(k)}$ 
     $\tilde{r}^{(k+1)} := \tilde{r}^{(k)} - \gamma^{(k)}A^T\tilde{d}^{(k)}$ 
     $\rho_{k+1} := (r^{(k+1)}, \tilde{r}^{(k+1)})$ 
     $\alpha_k := \frac{\rho_{k+1}}{\rho_k}$ 
     $d^{(k+1)} := r^{(k+1)} + \alpha_k d^{(k)}$ 
     $\tilde{d}^{(k+1)} := \tilde{r}^{(k+1)} + \alpha_k \tilde{d}^{(k+1)}$ 

```

Proposition F.4.2. Soit $N \in \mathbb{N}^*$. L'exécution de N itérations de l'[Algorithme F.4.1](#) nécessite $2n^2 + 2n - 1 + (4n^2 + 12n)N$ flops si A est pleine et $2N_{\text{nz}} + 2n - 1 + (4N_{\text{nz}} + 12n)N$ flops si A est creuse et stockée au format CSR.

Démonstration. La définition de $r^{(0)}$ requiert

- 1 produit matrice-vecteur,
- n additions (une pour chaque composante des vecteurs).

La définition de ρ_0 est un produit scalaire qui nécessite donc

- n multiplications,
- $n - 1$ additions.

Au total, l'initialisation de l'algorithme nécessite donc :

- 1 produit matrice-vecteur,
- n multiplications,
- $n + (n - 1) = 2n - 1$ additions.

À chaque itération, on compte :

- 2 produits matrice-vecteur (un pour la définition de $z^{(k)}$ et un autre dans la mise à jour du résidu artificiel),
- $n + n + n + n + n + n + n = 7n$ multiplications (il faut penser qu'il y a n multiplications dans un produit du type $\gamma^{(k)}d^{(k)}$, par exemple),
- $(n - 1) + n + n + n + (n - 1) + n + n = 7n - 2$ additions,
- 2 divisions.

Si la matrice est pleine, un produit matrice-vecteur nécessite :

- n^2 multiplications,
- $n(n - 1)$ additions.

L'initialisation requiert donc un total de

$$n^2 + n(n - 1) + n + (2n - 1) = 2n^2 + 2n - 1$$

flops, et chaque itération nécessite

$$2n^2 + 2n(n - 1) + 7n + (7n - 2) + 2 = 4n^2 + 12n$$

flops.

Si la matrice est creuse et stockée au format CSR, un produit matrice-vecteur nécessite :

- N_{nz} multiplications,
- $N_{\text{nz}} - n$ additions.

Ainsi, l'initialisation nécessite

$$N_{\text{nz}} + (N_{\text{nz}} - n) + n + (2n - 1) = 2N_{\text{nz}} + 2n - 1$$

flops, et chaque itération nécessite

$$2N_{\text{nz}} + 2(N_{\text{nz}} - n) + 7n + (7n - 2) + 2 = 4N_{\text{nz}} + 12n$$

flops. ■

F.5 Convergence

Dans le cas du gradient conjugué, on a montré que la méthode convergeait en un nombre fini d'itérations. Si tout va bien (c'est-à-dire, si on ne se retrouve pas à diviser par zéro à une étape), on montre de même que la méthode du gradient biconjugué converge en autant d'itérations.

Théorème F.5.1. Supposons que, pour tout $k \in \llbracket 0, n-1 \rrbracket$, $\gamma^{(k)}$ et α_k soient bien définis. Alors $r^{(n)} = 0$.

Démonstration. Soit $(\lambda_0, \dots, \lambda_{n-1}) \in \mathbb{R}^{n-1}$ tel que

$$\sum_{k=0}^{n-1} \lambda_k \tilde{r}^{(k)} = 0.$$

Soit $i \in \llbracket 0, n-1 \rrbracket$. Alors :

$$0 = \left(\sum_{k=0}^{n-1} \lambda_k \tilde{r}^{(k)}, r^{(i)} \right) = \sum_{k=0}^{n-1} \lambda_k (\tilde{r}^{(k)}, r^{(i)}).$$

Or, si $k \neq i$, alors $(\tilde{r}^{(k)}, r^{(i)})$ par bi-orthogonalité, donc :

$$\lambda_i (\tilde{r}^{(i)}, r^{(i)}) = 0.$$

Comme α_i est bien défini, cela signifie que $(\tilde{r}^{(i)}, r^{(i)}) \neq 0$, donc $\lambda_i = 0$.

Ainsi, la famille $\{\tilde{r}^{(0)}, \dots, \tilde{r}^{(n-1)}\}$ est libre dans $\mathbb{R}^n$ et contient n vecteurs : il s'agit donc d'une base. Autrement dit, $\tilde{R}_{n-1} = \mathbb{R}^n$.

Par bi-orthogonalité, $r^{(n)}$ est orthogonal à tous les vecteurs de base de $\tilde{R}_{n-1}$: par ce qui précède, il est donc orthogonal à $\mathbb{R}^n$ tout entier. Ainsi, $r^{(n)} = 0$. ■

Remarque F.5.2. Dans cette preuve, on a montré que $\tilde{R}_{n-1} = \mathbb{R}^n$, mais un procédé similaire permet aussi de montrer que $R_{n-1} = \mathbb{R}^n$.

Annexe G

Absorption

G.1 Un peu de théorie

Quand on a défini les systèmes de nombres flottants, on a pu observer plusieurs phénomènes qui apparaissent en arithmétique flottante. Entre autres choses, on a ainsi évoqué le phénomène d'absorption : lors de l'addition de deux flottants, si l'un des deux est « trop petit » par rapport à l'autre, il est « absorbé » et n'influence tout simplement pas le résultat (ce qui ne signifie pas pour autant qu'il est arrondi à zéro). Nous allons chercher à préciser un peu tout cela. Commençons par obtenir une condition nécessaire et suffisante pour que deux réels soient arrondis au même flottant.

Théorème G.1.1. Soient deux réels x et y non nuls dans la plage de F . Alors

$$\text{fl}(x) = \text{fl}(y) \iff \left| \frac{\text{fl}(y) - x}{\text{fl}(y)} \right| < u \iff \left| \frac{\text{fl}(x) - y}{\text{fl}(x)} \right| < u.$$

Démonstration. On rappelle que

$$\text{fl}(x) = x(1 + \delta_x) \quad \text{et} \quad \text{fl}(y) = y(1 + \delta_y)$$

avec $|\delta_x| < u$ et $|\delta_y| < u$.

Dans un premier temps, supposons que $\text{fl}(x) = \text{fl}(y)$. Alors :

$$\frac{\text{fl}(y) - x}{\text{fl}(y)} = \frac{\text{fl}(x) - x}{\text{fl}(x)} = \frac{-x\delta_x}{x(1 + \delta_x)} = -\frac{\delta_x}{1 + \delta_x}.$$

Ainsi :

$$\left| \frac{\text{fl}(y) - x}{\text{fl}(y)} \right| = \left| \frac{\delta_x}{1 + \delta_x} \right| = \frac{|\delta_x|}{1 + \delta_x} \leq |\delta_x| < u$$

car $\frac{1}{1 + \delta_x} \leq 1$.

Réciproquement, supposons

$$\left| \frac{\text{fl}(y) - x}{\text{fl}(y)} \right| < u,$$

c'est-à-dire :

$$-\text{fl}(y)u < x - \text{fl}(y) < \text{fl}(y)u,$$

ou encore :

$$\text{fl}(y)(1 - u) < x < \text{fl}(y)(1 + u).$$

Ainsi, $\text{fl}(y)$ est le flottant le plus proche de x , donc $\text{fl}(x) = \text{fl}(y)$. ■

Remarque G.1.2. Dans l'énoncé de ce résultat, il faut bien noter que l'on s'intéresse à la distance entre l'un des deux réels et le représentant flottant de l'autre réel. L'inégalité

$$\left| \frac{y - x}{y} \right| < u$$

est une condition nécessaire pour avoir $\text{fl}(x) = \text{fl}(y)$, mais il ne s'agit pas d'une condition suffisante. Par exemple, x et y pourraient être très proches, mais situés d'une part et d'autre du milieu du segment formé par deux flottants : ils seront alors arrondis chacun à son extrémité de segment.

En supposant que l'on représente l'un des deux flottants de manière exacte (le plus grand), on peut alors obtenir une condition nécessaire et suffisante plutôt pratique pour voir survenir le phénomène d'absorption.

Théorème G.1.3. Soient x et y deux réels non nuls dans la plage de F . Supposons que $\text{fl}(x) = x$. Alors

$$\text{fl}(x + y) = \text{fl}(x) \iff \left| \frac{y}{x} \right| < u.$$

Démonstration. Il s'agit simplement d'utiliser la condition nécessaire et suffisante obtenue précédemment :

$$\text{fl}(x + y) = \text{fl}(x) \iff \left| \frac{\text{fl}(x) - (x + y)}{\text{fl}(x)} \right| < u.$$

On utilise alors l'hypothèse $\text{fl}(x) = x$ qui permet de simplifier un bon nombre de termes. ■

Dans la pratique, on a peu de chances de représenter un réel quelconque de manière exacte. Il s'agit cependant d'un résultat intéressant pour étudier certains calculs : il permet de montrer que, même sous des conditions optimales (c'est-à-dire que « tout va bien » jusque-là), un phénomène d'absorption peut survenir.

G.2 Une conséquence de l'absorption

Pour $\varepsilon \in]0, u[$, on définit la matrice

$$A := \begin{pmatrix} \varepsilon & -1 \\ 1 & 1 \end{pmatrix}.$$

Pour savoir comment vont se comporter les différentes actions que l'on pourrait être amenés à faire avec cette matrice, une idée pourrait être d'estimer son conditionnement. Ici, la matrice est suffisamment simple pour que l'on puisse le calculer directement, par exemple en norme 1. Commençons par calculer la norme de A :

$$\|A\|_1 = \max(1 + \varepsilon, 2) = 2$$

puisque $\varepsilon < 1$. L'inverse de la matrice peut lui aussi être calculé facilement :

$$A^{-1} = \frac{1}{1 + \varepsilon} \begin{pmatrix} 1 & 1 \\ -1 & \varepsilon \end{pmatrix},$$

et ainsi :

$$\|A^{-1}\|_1 = \max\left(\frac{2}{1 + \varepsilon}, 1\right) = \frac{2}{1 + \varepsilon}.$$

On obtient alors :

$$\text{cond}_1(A) = \|A\|_1 \cdot \|A^{-1}\|_1 = \frac{4}{1 + \varepsilon},$$

ce qui semble tout à fait raisonnable.

Calculons la décomposition LU de A . On cherche alors

$$L = \begin{pmatrix} 1 & 0 \\ L_{21} & 1 \end{pmatrix} \quad \text{et} \quad U = \begin{pmatrix} U_{11} & U_{12} \\ 0 & U_{22} \end{pmatrix}$$

telles que

$$A = \begin{pmatrix} \varepsilon & -1 \\ 1 & 1 \end{pmatrix} = LU = \begin{pmatrix} U_{11} & U_{12} \\ U_{11}L_{21} & L_{21}U_{12} + U_{22} \end{pmatrix}.$$

On en déduit :

$$U_{11} = \varepsilon, \quad U_{12} = -1,$$

puis :

$$L_{21} = \varepsilon^{-1}$$

et enfin :

$$U_{22} = 1 + \varepsilon^{-1}.$$

Ainsi :

$$L = \begin{pmatrix} 1 & 0 \\ \varepsilon^{-1} & 1 \end{pmatrix} \quad \text{et} \quad U = \begin{pmatrix} \varepsilon & -1 \\ 0 & 1 + \varepsilon^{-1} \end{pmatrix}.$$

On cherche alors à écrire les représentants numériques de ces matrices. Pour simplifier, on va supposer que ε et son inverse sont représentés de manière exacte : $\text{fl}(\varepsilon) = \varepsilon$ et $\text{fl}(\varepsilon^{-1}) = \varepsilon^{-1}$. Comme 1 et -1 sont des entiers, ils sont eux aussi représentés de manière exacte. Il n'y a donc pas vraiment de problème pour représenter L . En revanche, l'un des coefficients de U fait intervenir la somme de 1 et de ε^{-1} . On peut utiliser le [Théorème G.1.3](#) :

$$\frac{1}{\varepsilon^{-1}} = \varepsilon < u,$$

donc on aura :

$$\text{fl}(1 + \varepsilon^{-1}) = \text{fl}(\varepsilon^{-1}).$$

On obtient ainsi :

$$\hat{L} = \begin{pmatrix} 1 & 0 \\ \varepsilon^{-1} & 1 \end{pmatrix} \quad \text{et} \quad \hat{U} = \begin{pmatrix} \varepsilon & -1 \\ 0 & \varepsilon^{-1} \end{pmatrix},$$

d'où l'on tire rapidement :

$$A - \hat{L}\hat{U} = \begin{pmatrix} \varepsilon & -1 \\ 1 & 1 \end{pmatrix} - \begin{pmatrix} \varepsilon & -1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix},$$

ce qui représente une erreur plutôt importante.

Voyons comment cela se passe si on permute les lignes de A . On considère

$$P := \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

et on s'intéresse à la matrice PA . À nouveau, on recherche les matrices L et U telles que

$$PA = \begin{pmatrix} 1 & 1 \\ \varepsilon & -1 \end{pmatrix} = LU = \begin{pmatrix} 1 & 0 \\ L_{21} & 1 \end{pmatrix} \begin{pmatrix} U_{11} & U_{12} \\ 0 & U_{22} \end{pmatrix} = \begin{pmatrix} U_{11} & U_{12} \\ L_{21}U_{11} & L_{21}U_{12} + U_{22} \end{pmatrix}.$$

On obtient ici :

$$U_{11} = 1, \quad U_{12} = 1, \quad L_{21} = \varepsilon, \quad U_{22} = -1 - \varepsilon.$$

Ainsi :

$$L = \begin{pmatrix} 1 & 0 \\ \varepsilon & 1 \end{pmatrix} \quad \text{et} \quad U = \begin{pmatrix} 1 & 1 \\ 0 & -1 - \varepsilon \end{pmatrix}.$$

Numériquement, on a cette fois-ci :

$$\frac{\varepsilon}{1} = \varepsilon < u,$$

donc il va y avoir absorption :

$$\text{fl}(-1 - \varepsilon) = \text{fl}(-1) = -1.$$

Ainsi :

$$\hat{L} = \begin{pmatrix} 1 & 0 \\ \varepsilon & 1 \end{pmatrix} \quad \text{et} \quad \hat{U} = \begin{pmatrix} 1 & 1 \\ 0 & -1 \end{pmatrix}.$$

On obtient alors :

$$PA - \hat{L}\hat{U} = \begin{pmatrix} 1 & 1 \\ \varepsilon & -1 \end{pmatrix} - \begin{pmatrix} 1 & 1 \\ \varepsilon & \varepsilon - 1 \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 0 & -\varepsilon \end{pmatrix},$$

ce qui est déjà bien meilleur.

Annexe H

Convergence de la série harmonique ?

Pour $n \in \mathbb{N}^*$, on note

$$S_n := \sum_{k=1}^n \frac{1}{k}$$

qui n'est rien d'autre que la somme partielle de la bien connue série harmonique. On s'intéresse ici à deux manières de calculer cette somme numériquement.

H.1 Méthode classique

L'approche qui semble la plus naturelle pour calculer numériquement S_n consiste simplement à ajouter, les uns après les autres, les différents termes à sommer, en commençant par $k = 1$, $k = 2$, et ainsi de suite. On obtient l'[Algorithme H.1.1](#).

Algorithme H.1.1 : Un calcul numérique de la somme partielle de la série harmonique

```
S := 0
pour k de 1 à n
└ S := S + 1/k
Sf,n := S
```

On note $\hat{S}_{f,n}$ la valeur numérique obtenue par l'[Algorithme H.1.1](#). On note

$$\hat{S}^{(0)} = 0,$$

puis

$$\hat{S}^{(1)} = \text{fl}(\hat{S}^{(0)} + 1) = 1,$$

puis

$$\hat{S}^{(2)} = \text{fl}\left(\hat{S}^{(1)} + \frac{1}{2}\right) = \left(1 + \frac{1}{2}(1 + \varepsilon_2)\right)(1 + \delta_2),$$

avec $|\varepsilon_2| < u$ représentant l'erreur lors de la division, et $|\delta_2| < u$ représentant l'erreur lors de l'addition. En continuant, on obtient l'expression suivante qui peut être justifiée par récurrence :

$$\hat{S}_{f,n} = \sum_{k=1}^n \frac{1}{k} (1 + \varepsilon_k) \prod_{\ell=k}^n (1 + \delta_\ell),$$

où on a posé $\varepsilon_1 = \delta_1 = 0$ pour alléger l'écriture.

Pour chaque $k \in \llbracket 1, n \rrbracket$, on peut écrire :

$$(1 + \varepsilon_k) \prod_{\ell=k}^n (1 + \delta_\ell) = 1 + \theta_{n-k+2}^{(k)} \quad \text{avec } |\theta_{n-k+2}^{(k)}| \leq \gamma_{n-k+2}.$$

On en déduit :

$$\hat{S}_{f,n} = \sum_{k=1}^n \frac{1}{k} (1 + \theta_{n-k+2}^{(k)}) = \underbrace{\sum_{k=1}^n \frac{1}{k}}_{=S_n} + \sum_{k=1}^n \frac{1}{k} \theta_{n-k+2}^{(k)}.$$

Ainsi :

$$|\hat{S}_{f,n} - S_n| \leq \sum_{k=1}^n \frac{1}{k} |\theta_{n-k+2}^{(k)}| \leq \sum_{k=1}^n \frac{1}{k} \gamma_{n-k+2}.$$

On sait que la suite $(\gamma_n)_n$ est croissante, donc la suite $(\gamma_{n-k+2})_k$ est décroissante. La suite $(\frac{1}{k})_k$ est elle aussi décroissante. Comme il s'agit de deux suites positives, on en déduit que la suite $(\frac{1}{k} \gamma_{n-k+2})_k$ est décroissante : tous ses termes sont donc majorés par le premier. Ainsi :

$$|\hat{S}_{f,n} - S_n| \leq \sum_{k=1}^n \frac{1}{1} \gamma_{n-1+2} = \sum_{k=1}^n \gamma_{n+1} = \frac{n(n+1)u}{1 - (n+1)u}.$$

H.2 Une autre approche

Avec un changement d'indice, on obtient :

$$S_n = \sum_{k=1}^n \frac{1}{k} = \sum_{k=0}^{n-1} \frac{1}{n-k},$$

ce qui est la formalisation d'une idée simple : plutôt que de sommer les termes de gauche à droite, on les somme de droite à gauche. On obtient l'[Algorithme H.2.1](#).

En procédant comme pour l'algorithme précédent, on obtient que la valeur numérique $\hat{S}_{b,n}$ obtenue par l'[Algorithme H.2.1](#) est donnée par :

$$\hat{S}_{b,n} = \sum_{k=0}^{n-1} \frac{1}{n-k} (1 + \sigma_k) (1 + \varepsilon_k) \prod_{\ell=k}^{n-1} (1 + \delta_\ell),$$

Algorithme H.2.1 : Un autre calcul numérique de la somme partielle de la série harmonique

```

 $S := 0$ 
pour  $k$  de 0 à  $n - 1$ 
   $S := S + \frac{1}{n - k}$ 
 $S_{b,n} := S$ 

```

les σ_k représentant les erreurs dans les soustractions $n - k$, ε_k représentant les erreurs dans les divisions et les δ_ℓ les erreurs dans les additions. Pour tout $k \in \llbracket 0, n - 1 \rrbracket$, on peut écrire :

$$(1 + \sigma_k)(1 + \varepsilon_k) \prod_{\ell=k}^{n-1} (1 + \delta_\ell) = 1 + \theta_{n-k+2}^{(k)}, \quad \text{avec } |\theta_{n-k+2}^{(k)}| \leq \gamma_{n-k+2}.$$

On en déduit :

$$\hat{S}_{b,n} = S_n + \sum_{k=0}^{n-1} \frac{1}{n-k} \theta_{n-k+2}^{(k)},$$

et ainsi :

$$|\hat{S}_{b,n} - S_n| \leq \sum_{k=0}^{n-1} \frac{1}{n-k} \gamma_{n-k+2}.$$

On peut montrer, par exemple en étudiant la fonction $x \mapsto \frac{1}{n-x} \gamma_{n-x+2}$, que la suite $\left(\frac{1}{n-k} \gamma_{n-k+2} \right)_k$ est croissante, donc tous ses termes sont majorés par le dernier :

$$|\hat{S}_{b,n} - S_n| \leq \sum_{k=0}^{n-1} \frac{1}{n - (n-1)} \gamma_{n-(n-1)+2} = \sum_{k=0}^{n-1} \gamma_3 = \frac{3nu}{1-3u}.$$

H.3 Stabilité

En observant les bornes obtenues sur les erreurs, on peut déjà remarquer que les comportements sont différents. Essayons de formaliser un peu cela en considérant $n \in \mathbb{N}^*$ grand, mais raisonnable numériquement : on suppose qu'il existe $\varepsilon > 0$ tel que

$$(n+1)u \leq 1 - \varepsilon.$$

Ainsi :

$$\frac{1}{1 - (n+1)u} \leq \frac{1}{\varepsilon}.$$

On obtient :

$$|\hat{S}_{f,n} - S_n| \leq \frac{n(1-\varepsilon)}{\varepsilon}.$$

Comme $(n+1)u \leq 1 - \varepsilon$, on a également $nu \leq 1 - \varepsilon$, et aussi $3u \leq 1 - \varepsilon$, d'où :

$$|\hat{S}_{b,n} - S_n| \leq \frac{3(1 - \varepsilon)}{\varepsilon}.$$

Ainsi, l'erreur commise par $\hat{S}_{b,n}$ est bornée uniformément, ce qui n'est pas le cas de celle commise par $\hat{S}_{f,n}$ qui finit par ne plus être contrôlée.

H.4 Phénomènes numériques

Commençons par un petit résultat classique.

Lemme H.4.1. Pour tout $n \in \mathbb{N}^*$:

$$S_n \geq \ln(n+1).$$

Démonstration. Pour tout $k \in \mathbb{N}^*$, on peut appliquer le théorème des accroissements finis à la fonction $\ln$ sur l'intervalle $[k, k+1]$ pour obtenir l'existence d'un $c_k \in [k, k+1]$ tel que :

$$\ln(k+1) - \ln(k) = \ln'(c_k)(k+1 - k) = \frac{1}{c_k}.$$

Comme $k \leq c_k \leq k+1$, on obtient en particulier :

$$\frac{1}{k} \geq \frac{1}{c_k} = \ln(k+1) - \ln(k).$$

Pour $n \in \mathbb{N}^*$, on obtient alors :

$$S_n = \sum_{k=1}^n \frac{1}{k} \geq \sum_{k=1}^n (\ln(k+1) - \ln(k)) = \ln(n+1).$$

■

De ce résultat, on peut notamment déduire que

$$\lim_{n \rightarrow +\infty} S_n = +\infty,$$

ce qui présage déjà quelques problèmes numériques à l'horizon.

H.4.1 Absorption

Le problème de stabilité de l'[Algorithme H.1.1](#) vient principalement du fait que l'on somme des nombres de plus en plus petits. Au bout d'un moment, le terme $\frac{1}{k}$ finira par devenir trop petit par rapport au reste de la somme et sera tout simplement absorbé. Ainsi, on peut trouver $N_f \in \mathbb{N}^*$ tel que :

$$\forall n \geq N_f, \quad \hat{S}_{f,n} = \hat{S}_{f,N_f} < +\infty.$$

Voyons si on peut trouver une borne sur N_f . Pour cela, on cherche une condition pour avoir :

$$\hat{S}_{f,N_f+1} = \hat{S}_{f,N_f},$$

ou, dit autrement :

$$\text{fl} \left(\hat{S}_{f,N_f} + \frac{1}{N_f + 1} \right) = \hat{S}_{f,N_f}.$$

Simplifions le problème en supposant que tout est parfait jusque-là : $\hat{S}_{f,N_f} = S_{N_f}$. On peut alors appliquer le [Théorème G.1.3](#) : pour qu'il y ait absorption, il faut et il suffit que

$$\frac{1}{\frac{N_f+1}{S_{f,N_f}}} < u,$$

ce qui revient à :

$$(N_f + 1)S_{f,N_f} > \frac{1}{u}.$$

Or, par le [Lemme H.4.1](#), on sait que $S_{f,N_f} \geq \ln(N_f + 1)$, donc il suffit que N_f vérifie

$$(N_f + 1) \ln(N_f + 1) > \frac{1}{u}$$

pour avoir absorption. Rappelons par ailleurs que :

$$\forall x \in \mathbb{R}_*^+, \quad \ln(x) \geq 1 - \frac{1}{x},$$

donc il suffit que N_f vérifie :

$$\frac{1}{u} < (N_f + 1) \left(1 - \frac{1}{N_f + 1} \right) = N_f.$$

H.4.2 Arrondi à zéro

Comme les nombres sommés par l'[Algorithme H.2.1](#) sont de plus en plus grands, on peut raisonnablement penser que le phénomène d'absorption ne surviendra pas. D'autres problèmes peuvent cependant arriver. Dans un premier temps, on peut penser à l'arrondi à zéro : au bout d'un moment, le terme $\frac{1}{k}$ devient vraiment trop petit et est considéré comme zéro numériquement. On peut ainsi trouver $N_b \in \mathbb{N}^*$ tel que :

$$\forall n \geq N_b, \quad \hat{S}_{b,n} = \hat{S}_{b,N_b} < +\infty.$$

Cela arrive lorsque le terme qui doit être ajouté à $\hat{S}_{b,N_b}$ est arrondi à zéro. Ce terme n'est autre que

$$\frac{1}{N_b + 1}.$$

Il est arrondi à zéro lorsqu'il est plus petit que le plus petit flottant représentable dans le système F . Autrement dit :

$$\frac{1}{N_b + 1} < F_\nu = b^{e_{\min} - p}.$$

Ainsi :

$$N_b + 1 > b^{p-e_{\min}},$$

ou encore :

$$N_b \geq b^{p-e_{\min}}.$$

H.4.3 Arrondi à l'infini

Supposons un instant que le phénomène précédent n'arrive pas. Un autre problème finira alors par arriver : la somme finira par devenir suffisamment grosse pour être arrondie à l'infini. On trouve ainsi $N_\infty \in \mathbb{N}^*$ tel que :

$$\forall n \geq N_\infty, \quad \hat{S}_{b,n} = \hat{S}_{b,N_\infty} = +\infty.$$

Cela arrive lorsque la somme devient plus grande que le plus grand flottant représentable dans le système F . En supposant, pour simplifier, que tout allait bien jusque-là, on doit avoir $\hat{S}_{b,N_\infty} = \text{fl}(S_{N_\infty})$. On cherche donc N_∞ tel que :

$$S_{N_\infty} > F_{\max}.$$

Utilisons à nouveau le [Lemme H.4.1](#) : une condition suffisante sur N_∞ devient ainsi :

$$\ln(N_\infty + 1) > F_{\max},$$

ce qui revient à :

$$N_\infty + 1 > e^{F_{\max}},$$

et donc :

$$N_\infty \geq e^{F_{\max}}.$$

H.4.4 Discussion sur les bornes

Résumons ce qui a été obtenu. Dès que $n \geq b^{p-e_{\min}}$, alors les derniers $\frac{1}{k}$ sont arrondis à zéro et $\hat{S}_{b,n}$ stagne. Cela pourrait également arriver avec $\hat{S}_{f,n}$, mais celle-ci stagne avant : dès que $n > \frac{b^p}{2}$, l'absorption fait son effet.

Reste la stagnation de $\hat{S}_{b,n}$ à l'infini : celle-ci n'arrive en réalité jamais en pratique, puisque les arrondis à zéro arrive bien avant.

Références

- [Cia82] Philippe G. CIARLET. *Introduction à l'analyse numérique matricielle et à l'optimisation*. Masson, 1982.
- [Hig02] Nicholas HIGHAM. *Accuracy and Stability of Numerical Algorithms*. SIAM, 2002.
- [Ins85] INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS. « IEEE Standard for Binary Floating-Point Arithmetic ». In : *ANSI/IEEE Std 754-1985* (1985), p. 1-20.
- [Saa03] Y. SAAD. *Iterative Methods for Sparse Linear Systems*. SIAM, 2003.
- [Wik23] WIKIPÉDIA. *Terre*. 2023. URL : <https://fr.wikipedia.org/wiki/Terre>.