Projet de Méthodes Numériques

Résolution d'un système linéaire par la méthode du gradient conjugué

AGUIRRE Paula, CHAPEAU Paul, HOSSEIN KHAN Rémy, WILLIAMSON Måns

Table des matières

1 Introduction **3**

2 Méthode du gradient conjugué **3**
2.1 Idée de la méthode 3
2.2 Recherche d'optimalité : choix du pas et de la direction 4
2.2.1 Détermination du pas 5
2.2.2 Détermination de la direction 5
2.3 Algorithme itératif de la méthode du gradient conjugué 8
2.4 Explication du nom de la méthode du gradient conjugué 9
2.5 Étude de la méthode en arithmétique finie 9
2.5.1 Intérêt de l'étude en précision finie 9
2.5.2 Le rôle du conditionnement 10
2.5.3 Erreurs d'arrondis 10
2.5.4 Une condition de stabilité 11
2.6 Convergence de la méthode 13

3 Méthode du gradient conjugué pré conditionnée **13**
3.1 Intérêt d'un pré conditionnement 13
3.2 Le pré conditionnement 14
3.3 Algorithme de la méthode du gradient conjugué pré conditionnée 14

4 Mise en oeuvre sur MATLAB **16**
4.1 Un test d'arrêt basé sur le résidu 16
4.2 Implémentation sur MATLAB 17
4.3 Une première application dans un cas favorable 17
4.4 Différents résultats selon la valeur du second membre k 18
4.5 Une troisième application mal conditionnée 19

5 Liens avec l'optimisation et l'approximation polynomiale **19**
5.1 Optimisation 19
5.1.1 Un cas particulier d'optimisation : la méthode du point fixe . . . 20
5.2 Approximation polynomiale 20
5.2.1 La formule de Newton 20
5.2.2 Polynômes orthogonaux 20
5.2.3 Correspondance avec les espaces Lp à poids 20
5.2.4 Extension de la méthode du gradient conjugé pour les polynômes 21

6 Bibliographie **22**

7 Signatures **22**

1 Introduction

La méthode du gradient conjugué, introduite en 1950 par Magnus R. HESTENES, un des auteurs de l'[article], et Cornelius LANCZOS, apparait comme une amélioration de la méthode du gradient, connue depuis un siècle.

Initialement, la méthode du gradient conjuguée se calcule directement, mais elle requiert plus d'espace mémoire qu'une méthode itérative. Ceci, et le fait qu'elle génère de nombreux erreurs d'arrondi, explique son moindre succès.

Cependant, la version itérative, que nous allons étudier, assure une convergence très satisfaisante dès les premières étapes, lui apportant une forte popularité.

2 Méthode du gradient conjugué

2.1 Idée de la méthode

L'objectif est de résoudre le système linéaire suivant : $Ax = k$, où A est une matrice symétrique définie positive de $M_n(\mathbb{R})$, et x et k des vecteurs de $\mathbb{R}^n$.

On note $err = h - x$ l'erreur relative à x, où h est la solution du système.

On définit la fonction g par [1] :

$$g(x) = (h - x, A(h - x))$$

Résoudre le système revient à minimiser cette fonction. En effet si $g(x) = 0$, comme A est symétrique définie positive, on a alors que $err = 0$ et c'est ainsi qu'on aura résolu le problème.

Remarque 1. *Calculons le gradient de g, soit* $l \in \mathbb{R}^n$,

$$\begin{aligned} g(x+l) &= (h - x - l, Ah - Ax - Al) \\ &= (h - x, A(h - x)) + (h - x, Al) + (-l, A(h - x)) + (-l, -Al) \\ &= g(x) - 2(l, A(h - x)) + (l, Al) \end{aligned}$$

Or $(l, Al) = o(l)$ *par l'inégalité de Cauchy-Schwarz, ainsi*

$$g(x + l) = g(x) + dg_x(l) + o(l)$$

Or $dg_x(l) = (l, \nabla g(x))$, *on identifie alors*

$$\begin{aligned} \nabla g(x) &= -2A(h - x) \\ &= -2(k - Ax) \\ &= -2r \end{aligned}$$

où $r = k - Ax$ *est le résidu.*

1. $(.,.)$ est le produit scalaire canonique et $\| \, . \, \|$ est la norme associée

Remarque 2. *Néanmoins, nous n'allons pas étudier la fonction g mais f qui se définit par :*

$$f(x) = \frac{1}{2}(x, Ax) - (x, k)$$

Le choix de cette fonction se justifie non seulement car $\nabla f(x) = -r$ *et non* $-2r$ *mais surtout car f(x) se calcule sans connaitre h.*

Propriété 1. *Minimiser la fonction g revient à minimiser la fonction f. Et c'est pour cela qu'on se permet d'étudier la fonction f.*

Démonstration.

$$\begin{aligned}
f(x) = & \quad f(h + (x - h)) \\
= & \quad \frac{1}{2}(h + (x - h), A(h + (x - h)) - (h + (x - h), k) \\
= & \quad \frac{1}{2}(h, Ah) - (h, k) + \frac{1}{2}[((x - h, Ax) + (h, A(x - h))] - (x - h, Ah) \\
= & \quad f(h) + \frac{1}{2}[(x - h, Ax) - (x - h, Ah)] \\
= & \quad f(h) + \frac{1}{2}g(x)
\end{aligned}$$

Comme A est symétrique définie positive, $\forall x \neq h, g(x) > 0$, donc $f(x) - f(h) > 0$ soit $f(x) > f(h)$ et donc h est bien le minimum de la fonction g et de f. □

L'idée de la méthode est la suivante : en partant d'un point $x^{(0)}$ de $\mathbb{R}^n$, on va chercher à déterminer des directions de déplacements qui permettent de se rapprocher le plus possible de la solution h. On va alors effectuer un pas à partir de $x^{(0)}$ le long d'une direction $p^{(0)}$, puis fixer le long de celle-ci un nouveau point $x^{(1)}$ à partir duquel on itère le procédé jusqu'à convergence. A l'étape i, $x^{(i+1)}$ est déterminé par :

$$x^{(i+1)} = x^{(i)} + a^{(i)}p^{(i)}$$

où $a^{(i)}$ est la valeur qui fixe la longueur du pas le long de $p^{(i)}$. Il faut donc choisir une direction de descente et un pas tel que f atteigne un minimum local le long de cette direction.

Remarque 3. *Sachant que la pente de plus grande descente est le long du gradient, celui-ci représente un premier choix de direction. Soit :* $p^{(i)} = \nabla f(x^{(i)}) = -r^{(i)}$. *Cela correspond à la méthode du gradient. Cependant, nous allons voir par la suite qu'il existe un choix de direction plus pertinent.*

2.2 Recherche d'optimalité : choix du pas et de la direction [2]

Plaçons-nous à l'étape i .

2. Cette sous-partie a été rédigée à l'aide du [livre1]

Définition 1. *Un vecteur $x^{(i)}$ est dit optimal par rapport à une direction $p \neq 0$ si*

$$f(x^{(i)}) \leq f(x^{(i)} + \lambda p), \forall \lambda \in \mathbb{R} \tag{1}$$

Si $x^{(i)}$ est optimal par rapport à n'importe quelle direction d'un espace vectoriel V, on dit que $x^{(i)}$ est un minimum global de V.

2.2.1 Détermination du pas

Intéressons-nous au pas $a^{(i)}$ tel que $f(x^{(i+1)}) = min\{f(x^{(i)} + \lambda p^{(i)}) | \lambda \in \mathbb{R}\}$

$$\begin{aligned}
f(x^{(i+1)}) &= f(x^{(i)} + a^{(i)}p^{(i)}) \\
&= \frac{1}{2}(x^{(i)} + a^{(i)}p^{(i)}, A(x^{(i)} + a^{(i)}Ap^{(i)}) - (x^{(i)} + a^{(i)}p^{(i)}, k) \\
&= \frac{1}{2}(x^{(i)}, Ax^{(i)}) - (x^{(i)}, k) + a^{(i)}((x^{(i)}, Ap^{(i)}) - (p^{(i)}, k)) + (a^{(i)})^2(p^{(i)}, Ap^{(i)}) \\
&= \frac{1}{2}(x^{(i)}, Ax^{(i)}) - (x^{(i)}, k) + a^{(i)}((p^{(i)}, k) - (Ax^{(i)}, pi)) + (a^{(i)})^2(p^{(i)}, Ap^{(i)})
\end{aligned}$$

On obtient un polynôme du second degré en $a^{(i)}$, comme le coefficient dominant est $(p^{(i)}, Ap^{(i)}) > 0$, cette fonction admet un minimum en :

$$a^{(i)} = \frac{(p^{(i)}, k - Ax^{(i)})}{(p^{(i)}, Ap^{(i)})} = \frac{(p^{(i)}, r^{(i)})}{(p^{(i)}, Ap^{(i)})} \tag{2}$$

2.2.2 Détermination de la direction

Avant de construire une suite de directions $(p^{(i)})_{i \in \{0,\dots,n\}}$, on va déterminer les propriétés nécessaires pour l'optimisation. Pour cela, on prendra d'abord une direction p fixée.

Propriété 2. *$x^{(i)}$ est optimal par rapport à une direction p si et seuleument si*

$$(p, r^{(i)}) = 0$$

Démonstration. A partir de la définition (1), f admet un minimum local le long de p pour $\lambda = 0$. La dérivée partielle de f par rapport λ s'annule donc pour $\lambda = 0$. Or :

$$\frac{\partial f}{\partial \lambda}(x^{(i)} + \lambda p) = (p, Ax^{(i)} - b) + \lambda(p, Ap) = (p, r^{(i)}) + \lambda(p, Ap)$$

Donc $\frac{\partial f}{\partial \lambda}(x^{(i)} + 0) = 0$ si et seulement si $(p, r^{(i)}) = 0$ □

Remarque 4. *Cette propriété assure l'optimalité du vecteur $x^{(i)}$ par rapport à p, mais rien ne l'assure pour le vecteur $x^{(i+1)}$. On voudrait donc que les directions de descente conservent l'optimalité des vecteurs à chaque itération.*
La propriété suivante va nous donner les conditions sur la nouvelle direction q à prendre afin de conserver l'optimalité par rapport à la direction p déjà optimisée.

Propriété 3. *S'il existe un vecteur q de $\mathbb{R}^n$ tel que $x^{(i+1)} = x^{(i)} + q$, si $x^{(i)}$ est optimal par rapport à une direction p et si $(p, Aq) = 0$, alors $x^{(i+1)}$ est optimal par rapport à p.*

Démonstration. Pour montrer cette propriété, on va montrer l'équivalence.
Supposons qu'il existe un vecteur q de $\mathbb{R}^n$ tel que $x^{(i+1)} = x^{(i)} + q$.
Supposons de plus que $x^{(i)}$ soit optimal par rapport à une direction p et $x^{(i+1)}$ aussi. Par la Propriété 1 :

$$0 = (p, r^{(i+1)}) = (p, r^{(i)} - Aq) = -(p, Aq)$$

si et seuleument si

$$0 = (p, Aq) \tag{3}$$

□

Remarque 5. *Pour préserver l'optimalité entre les itérées successives, les directions de descente doivent donc être mutuellement A-conjuguées. On peut maintenant établir notre suite.*

Propriété 4. *On prendra $p^{(0)} = r^{(0)}$ car on cherche à descendre selon une direction de plus grande pente qui est celle du gradient de f, or $\nabla f(x) = -r$.*
Puis, afin de conserver l'optimalité entre les itérées successives, on prendra $\forall i \in \{1, ..., n-1\}$,

$$p^{(i+1)} = r^{(i+1)} - b^{(i)} p^{(i)}, \tag{4}$$

où

$$b^{(i)} = \frac{(Ap^{(i)}, r^{(i+1)})}{(Ap^{(i)}, p^{(i)})} \tag{5}$$

Afin de démontrer la propriété, nous allons démontrer le lemme suivant :

Lemme [3] 1. *On a l'égalité entre les sous-espaces suivants :*

$$Vect(r^{(0)}, Ar^{(0)}, ..., A^i r^{(0)}) = Vect(r^{(0)}, ..., r^i) = Vect(p^{(0)}, ..., p^{(i)}) \tag{6}$$

et, pour tout j tel que $0 \leq j \leq i$, on a les relations d'orthogonalité suivantes :

$$(p^{(i+1)}, Ap^{(j)}) = 0 \tag{7}$$

$$(r^{(i+1)}, p^{(j)}) = 0 \tag{8}$$

$$(r^{(i+1)}, r^{(j)}) = 0 \tag{9}$$

Démonstration.

Montrons (6) par récurrence sur i :
Initialisation : $p^{(0)} = r^{(0)}$ par définition.
Hérédité : Supposons que $Vect(r^{(0)}, Ar^{(0)}, ..., A^i r^{(0)}) = Vect(r^{(0)}, ..., r^{(i)}) = Vect(p^{(0)}, ..., p^{(i)})$.
Il suffit de montrer que $p^{(i+1)} \in Vect(r^{(0)}, ..., r^{(i+1)})$ et que $r^{(i+1)} \in Vect(p^{(0)}, ..., p^{(i+1)})$.
Or $p^{(i+1)} = r^{(i+1)} - b^{(i)} p^{(i)}$, il vient directement que, par combinaison linéaire, $p^{(i+1)} \in$

3. Ce lemme et sa preuve sont inspirés du [cours1]

$Vect(r^{(0)}, ..., r^{(i+1)})$.
De même, $r^{(i+1)} = p^{(i+1)} + b^{(i)}p^{(i)}$, d'où $r^{(i+1)} \in Vect(p^{(0)}, ..., p^{(i+1)})$.
Il reste à démontrer que $A^{i+1}r^{(0)} \in Vect(r^{(0)}, ..., r^{(i+1)})$ et que $r^{(i+1)} \in Vect(r^{(0)}, Ar^{(0)}, ..., A^{i+1}r^{(0)})$.
Or $r^{(i+1)} = k - Ax^{(i+1)} = k - Ax^{(i)} - a^{(i)}Ap^{(i)} = r^{(i)} - a^{(i)}Ap^{(i)}$ d'où le résultat.
Pour l'inclusion réciproque, on écrit par hypothèse de récurrence que $A^i r^{(0)} = \sum_{l=0}^{i} \lambda_i p^{(i)}$.
Donc $A^{i+1}r^{(0)} = \sum_{l=0}^{i} \lambda_i Ap^{(i)} = \sum_{l=0}^{i} \frac{\lambda_i}{a^{(i)}}(r^{(i)} - r^{(i+1)})$, d'où le résultat.
On peut alors conclure la récurrence.

Montrons (7) et (8) par récurrence sur i :
Initialisation :

$$\begin{aligned}(p^{(1)}, Ap^{(0)}) &= (r^{(1)}, Ap^{(0)}) - b^{(0)}(p^{(0)}, Ap^{(0)}) \\ &= (r^{(1)}, Ap^{(0)}) - \frac{(Ar^0), r^{(1)})}{(Ar^{(0)}, r^{(0)})}(r^{(0)}, Ar^{(0)}) \\ &= 0\end{aligned}$$

Et

$$\begin{aligned}(r^{(1)}, p^{(0)}) &= (r^{(0)}, r^{(0)}) - a^{(0)}(Ar^{(0)}, r^{(0)}) \\ &= (r^{(0)}, r^{(0)}) - \frac{(r^0), r^{(0)})}{(r^{(0)}, Ar^{(0)})}(Ar^{(0)}, r^{(0)}) = 0\end{aligned}$$

Hérédité : Supposons les relations vraies au rang i-1.
Pour tout $j \leq i - 1$,

$$(r^{(i+1)}, p^{(j)}) = (r^{(i)}, p^{(j)}) - a^{(i)}(Ap^{(i)}, r^{(j)})$$

Or, par hypothèse de récurrence, $(r^{(i)}, p^{(j)}) = 0$ et $(Ap^{(i)}, r^{(j)}) = 0$, donc $(r^{(i+1)}, p^{(j)}) = 0$.
Et

$$\begin{aligned}(p^{(i+1)}, Ap^{(j)}) &= (r^{(i+1)}, Ap^{(j)}) - b^{(i)}(p^{(i)}, Ap^{(j)}) \\ &= (r^{(i+1)}, Ap^{(j)})\end{aligned}$$

Or par (6), $p^{(j)} \in Vect(r^{(0)}, Ar^{(0)}, ..., A^j r^{(0)})$, donc $Ap^{(j)} \in Vect(Ar^{(0)}, ..., A^{j+1}r^{(0)})$, or toujours par (6), $Vect(Ar^{(0)}, ..., A^{j+1}r^{(0)}) \subset Vect(p^{(0)}, ..., p^{(j+1)})$. Or $(r^{(i+1)}, p^{(j)}) = 0$ pour $j \leq i$, donc $(r^{(i+1)}, Ap^{(j)}) = 0$ et finalement $(p^{(i+1)}, Ap^{(j)}) = 0$.
Ceci conclut la récurrence.

Montrons (9) : Pour tout $0 \leq j \leq i$, par (6), $r^{(j)} \in Vect(p^{(0)}, ..., p^{(j)})$. Donc, par (8), $(r^{(i+1)}, r^{(j)}) = 0$. □

On peut enfin démontrer que la suite des $(p^{(i)})_{i \in \{0,...,n\}}$ conserve l'optimalité à chaque itération.

Démonstration. Le coefficient $b^{(i)} \in \mathbb{R}$ est tel que (3) soit vérifié pour tout $j \in \{0, ..., i\}$ avec $q = p^{(i+1)}$ et $p = p^{(j)}$, soit

$$(Ap^{(j)}, p^{(i+1)}) = 0 \tag{10}$$

Puis on écrit cette dernière équation pour $j = i$ et on utilise (4) pour trouver (5).
De plus, cette égalité est vraie pour tout $j \in \{0, ..., i - 1\}$ où $i \geq 1$,vu en (7) dans le lemme. □

2.3 Algorithme itératif de la méthode du gradient conjugué

— *Initialisation :* choisir $x^{(0)}$, $r^{(0)} = k - Ax^{(0)}$, $p^{(0)} = r^{(0)}$
— *Itérations :* Pour i allant de 0 à $n-1$,
$a^{(i)} = \frac{(p^{(i)}, r^{(i)})}{(p^{(i)}, Ap^{(i)})}$
$x^{(i+1)} = x^{(i)} + a^{(i)} p^{(i)}$
$r^{(i+1)} = r^{(i)} - a^{(i)} Ap^{(i)}$
$b^{(i)} = \frac{(Ap^{(i)}, r^{(i+1)})}{(Ap^{(i)}, p^{(i)})}$
$p^{(i+1)} = r^{(i+1)} - b^{(i)} p^{(i)}$

Propriété 5. *Les $a^{(i)}$ et $b^{(i)}$ s'écrivent aussi*

$$a^{(i)} = \frac{\| r^{(i)} \|^2}{(p^{(i)}, Ap^{(i)})}, b^{(i)} = -\frac{\| r^{(i+1)} \|^2}{\| r^{(i)} \|^2} \tag{11}$$

Démonstration. Pour tout $1 \leq i \leq n$,

$$\begin{aligned} (p^{(i)}, r^{(i)}) = & \quad (r^{(i)}, r^{(i)}) - b^{(i-1)}(p^{(i-1)}, r^{(i)}) \\ = & \quad \| r^{(i)} \|^2 \end{aligned}$$

En effet, $(p^{(i-1)}, r^{(i)}) = 0$ par (8).
Et comme $p^{(0)} = r^{(0)}$, on a aussi que $(p^{(0)}, r^{(0)}) = \| r^{(0)} \|^2$. Ceci nous permet de trouver la valeur de $a^{(i)}$ recherchée.

Pour tout $i \in \{0, ..., n-1\}$, montrons que $b^{(i)}$ s'écrit sous cette forme (11).

$$\begin{aligned} (Ap^{(i)}, r^{(i+1)}) = & \quad (\frac{-1}{a^{(i)}}(r^{(i+1)} - r^{(i)}), r^{(i+1)}) \\ = & \quad \frac{-1}{a^{(i)}}(\| r^{(i+1)} \|^2 - (r^{(i)}, r^{(i+1)})) \end{aligned}$$

Or, par (9), $(r^{(i)}, r^{(i+1)}) = 0$. Donc $(Ap^{(i)}, r^{(i+1)}) = \frac{-1}{a^{(i)}} \| r^{(i+1)} \|^2$. Et

$$(Ap^{(i)}, p^{(i)}) = \quad (\frac{-1}{a^{(i)}}(r^{(i+1)} - r^{(i)}), p^{(i)})$$

Or par (8), $(r^{(i+1)}, p^{(i)}) = 0$, et on vient de démontrer que $r^{(i)}, p^{(i)}) = \| r^{(i)} \|^2$. Donc $(Ap^{(i)}, p^{(i)}) = \frac{1}{a^{(i)}} \| r^{(i)} \|^2$. Ce qui nous permet de conclure.

□

Remarque 6. *L'intérêt d'utiliser les expressions (11) est que dans la boucle d'itération, on calcule une fois $\| r^{(i)} \|^2$ et on le garde en mémoire. Ceci permet d'effectuer moins de calcul lors de la mise en oeuvre. Il se trouve que l'[article] trouve des résultats plus précis avec les formules initiales mais que la différence avec (11) est négligeable sauf dans des cas mal conditionnés.*

2.4 Explication du nom de la méthode du gradient conjugué

A la première étape, on "relâche" $x^{(0)}$ selon la direction de $p^{(0)}$, donc celle du gradient de f, puis on cherche λ un réel tel que λ minimise $f(x^{(0)} + \lambda p^{(0)})$.

On pose $x^{(1)} = x^{(0)} + \lambda p^{(0)}$, et on se restreint à $\pi^{(1)} = \{(x, Ap^{(0)}) = 0\}$ pour minimiser $f(x^{(1)})$, puisqu'on a déjà minimisé sur $Vect(p^{(0)})$. A l'étape suivante, on procède de la même façon mais en "relâchant" sur $p^{(1)}$, qui est en fait le gradient de f en $x^{(1)}$ projeté sur $\pi^{(1)}$. Ainsi de suite, on aura construit une suite $(x^{(n)})_{n\in\mathbb{N}}$ telle que $f(x^{(n)}) \to 0$. La méthode tient donc son nom du fait qu'elle utilise une approximation du gradient par les conjugués.

2.5 Étude de la méthode en arithmétique finie

2.5.1 Intérêt de l'étude en précision finie

Les méthodes que nous étudions sont numériques, elles sont destinées à être appliquées à l'ordinateur or l'architecture d'une machine ne permet pas d'atteindre une précision ultime.

En effet, les nombres y sont représentés en binaire, c'est-à-dire que les nombres entiers sont écrits en base 2 et stockés sur 8, 16, 32, 64 bits où un bit représente la valeur 0 ou 1 présente dans la représentation binaire du nombre.

En guise d'exemple, imaginons un vieil ordinateur ne pouvant stocker des entiers naturels que sur 16 bits, alors il ne pourra traiter que des nombres compris entre 0 et ($2^0 + ... + 2^{15} = 2^{16} - 1 =$) 65535.

Quant aux nombres qui ne sont pas entiers, ils sont représentés en virgule flottante, ce qui signifie que l'ordinateur associe la position d'un bit mis à 1 à une certaine puissance de $\frac{1}{2}$, les chiffres significatifs d'un nombre décimal sont donc calculés en faisant la somme des puissances de $\frac{1}{2}$ associées à la représentation en virgule flottante de ce nombre. Par exemple, la valeur 1/10 ne peut être obtenue en sommant des puissances de $\frac{1}{2}$ et ne pourra donc jamais être représentée de manière exacte en virgule flottante bien qu'elle soit approchée de manière très précise.

C'est en tentant de comprendre l'architecture de l'ordinateur que nous avons pu avoir une intuition quant à la définition de précision finie qu'on oppose à exacte. On constate en effet que les capacités de stockage d'une machine sont limitées, que la précision d'un nombre peut dépendre du nombre de bits utilisés dans leur représentation, ce nombre de bits étant fini.

Les méthodes numériques sont appliquées a des problèmes de grande taille où ces imprécisions, c'est-à-dire des erreurs d'arrondi peuvent mener à des résultats aberrants.

Il s'agit alors d'essayer de contrôler, prévoir et anticiper ces erreurs en étudiant le comportement des erreurs selon la matrice A du système linéaire.

C'est cela qui nous conduit à l'étude de la stabilité de la matrice,que nous interprétons ici comme sa résistance aux erreurs d'arrondi.

2.5.2 Le rôle du conditionnement

Définition 2. *On appelle conditionnement de la matrice A :* $Cond(A) =\| A \|\| A^{-1} \|$.

Pour la suite de l'étude, on utilisera la norme 2, on rappelle que : $\| A \|_2= \sqrt{\rho(A^t A)}$.

Or ici, A est symétrique donc, en notant λmax (respectivement λmin) la valeur propre de plus grand module (respectivement de plus petit module),
$\| A \|= \sqrt{\rho(A^2)} = \sqrt{\lambda max^2} = \lambda max$ et $\| A^{-1} \|= \frac{1}{\lambda min}$.

Ainsi $Cond(A) = \frac{\lambda max}{\lambda min}$.

Propriété 6. $Cond(A) \geq 1$

Démonstration. $1 =\| I_n \|=\| AA^{-1} \|\leq\| A \|\| A^{-1} \|\leq Cond(A)$. □

Remarque 7. *On remarque que toute matrice de la forme* $A = cI_n$*, avec* $c \in \mathbb{R}_+$*, a pour conditionnement* $Cond(A) = 1$.

Plus le conditionnement est grand, plus le système est sensible aux perturbations dues aux erreurs d'arrondi. En effet, le pré-conditionnement est un ensemble d'opérations sur la matrice permettant de réduire le nombre d'opérations à effectuer dans la méthode. Essayer de transformer la matrice d'itération A en matrice creuse va sensiblement réduire le taux d'erreurs d'arrondi puisque nous allons éliminer certaines opérations pouvant causer des erreurs d'arrondi.

On verra en effet, que le cond(A) est lié à la convergence de la méthode.

La grande taille des matrices que l'on veut traiter ne permet pas une résolution exacte car plus la taille augmente plus les erreurs d'arrondi vont avoir de chances de modifier considérablement le résultat.

L'[article] de Hestenes et Stiefel aura été une de nos sources principales pour l'étude des erreurs d'arrondi car nous avons éprouvé une certaine difficulté à recueillir des études précises de ces erreurs : l'amélioration des performances informatiques a probablement réduit les travaux de recherche sur ce sujet.

2.5.3 Erreurs d'arrondis

Une chose importante à comprendre dans l'étude des erreurs d'arrondis est que, comme l'environnement numérique nous contraint à travailler en précision "finie", certains résultats de calculs vont être biaisés.

Par conséquent les produits scalaires $(r^{(i-1)}, r^{(i)})$ et $(Ap^{(i-1)}, p^{(i)})$ qui assurent la bonne conjugaison et orthogonalité des vecteurs peuvent être non nuls, bien que très proches de 0.

L'[article] introduit des calculs qui vont nous permettre de voir comment les erreurs d'arrondis vont se propager, de la première à la dernière itération.

C'est pourquoi nous parlons de "propagation" et cherchons des formules de récurrence.

En procédant par insertion des formules $p^{(i+1)} = r^{(i+1)}+b^{(i)}p^{(i)}$; $r^{(i+1)} = r^{(i)}-a^{(i)}Ap^{(i)}$; $a^{(i)} = \frac{\|r^{(i)}\|^2}{(p^{(i)},Ap^{(i)})}$ dans le produit scalaire $(r^{(i)}, r^{(i+1)})$, nous obtenons

$$(r^{(i)}, r^{(i+1)}) = b^{(i-1)}a^{(i)}(Ap^{(i-1)}, p^{(i)}) \tag{12}$$

De plus, en utilisant $p^{(i+1)} = r^{(i+1)} + b^{(i)}p^{(i)}$; $r^{(i+1)} = r^{(i)} - a^{(i)}Ap^{(i)}$; $a^{(i)} = \frac{\|r^{(i)}\|^2}{(p^{(i)},Ap^{(i)})}$; $b^{(i)} = \frac{\|r^{(i+1)}\|^2}{\|r^{(i)}\|^2}$ dans $(Ap^{(i)}, p^{(i+1)})$, nous avons

$$(Ap^{(i)}, p^{(i+1)}) = \frac{1}{a^{(i)}}(r^{(i)}, r^{(i+1)}) \tag{13}$$

Celles-ci nous permettent d'obtenir, à nos yeux la formule la plus importante car, c'est une suite récurrente linéaire d'ordre 1, on a ainsi l'erreur à la prochaine étape en fonction d'un coefficient directeur et de l'erreur à l'étape précédente.

$$(r^{(i)}, r^{(i+1)}) = \frac{b^{(i+1)}a^{(i)}}{a^{(i-1)}}(r^{(i-1)}, r^{(i)}) \tag{14}$$

$$(Ap^{(i)}, p^{(i+1)}) = b^{(i-1)}(Ap^{(i-1)}, p^{(i)}) \tag{15}$$

L'article introduit les formules (14) et (15) sous une forme alternative :

$$\frac{(r^{(i)}, r^{(i+1)})}{\| r^{(i)} \|^2} = \frac{a^{(i)}}{a^{(i-1)}} \frac{(r^{(i-1)}, r^{(i)})}{\| r^{(i-1)} \|^2} \tag{16}$$

$$\frac{(Ap^{(i)}, p^{(i+1)})}{(Ap^{(i)}, p^{(i)})} = \frac{a^{(i)}}{a^{(i-1)}} \frac{(Ap^{(i-1)}, p^{(i)})}{(Ap^{(i-1)}, p^{(i-1)})} \tag{17}$$

Selon Hestenes et Stiefel, cette formule de récurrence permet de contrôler les résultats des calculs, c'est-à-dire qu'elle met en valeur, à chaque étape, les erreurs d'arrondi effectués par la machine. Par conséquent, il est possible de comprendre si le résultat a été altéré par les erreurs d'arrondi ou bien par des erreurs extérieures. En effet, on imagine que dans les années 50, le matériel informatique était sujet à de nombreux bugs, bien davantage qu'aujourd'hui.

La formule (17) est utile dans le sens où, si nous bâtissons la matrice $\begin{pmatrix} (Ap^{(1)}, p^{(1)}) & (Ap^{(1)}, p^{(2)}) & \cdots & (Ap^{(1)}, p^{(n)}) \\ (Ap^{(2)}, p^{(1)}) & (Ap^{(2)}, p^{(2)}) & & \vdots \\ \vdots & & \ddots & \vdots \\ (Ap^{(n)}, p^{(1)}) & (Ap^{(n)}, p^{(2)}) & \cdots & (Ap^{(n)}, p^{(n)}) \end{pmatrix}$, on remarque que le rapport $\frac{(Ap^{(i)},p^{(i+1)})}{(Ap^{(i)},p^{(i)})}$ est le quotient d'un élément de la diagonale associé à un éléments qui est juste à droite de celui-ci. La formule (17) nous explique l'évolution de ce quotient lorsque l'on va descendre le long de la diagonale. Cette matrice peut être un très bon outil de contrôle d'erreurs et les chercheurs ont sans doute de nombreuses fois analysé l'évolution des quotients de propagation, de manière globale ou selon une certaine étape i.

2.5.4 Une condition de stabilité

L'étude de l'article nous conduit à penser que, dû aux erreurs d'arrondi, les $a^{(i)}$ que nous avons calculés à l'aide de la formule sont entachés d'erreurs d'arrondi et conduisent à un résultat approximatif de la solution h.

Il s'agit alors d'introduire des coefficients $a^{(i)'}$ qui conduiraient eux à la solution exacte h, en ayant

$$h = x^{(0)} + a^{(0)'}p^{(0)} + a^{(1)'}p^{(1)} + ... + a^{(n-1)'}p^{(n-1)} \tag{18}$$

Il pourrait être intéressant d'étudier en quoi les coefficients a' qui mèneraient à une solution exacte, diffèrent des coefficients $a^{(i)}$ que nous avons obtenus dans la construction de la méthode.

En prenant le produit scalaire $(h, Ap^{(i)})$ et en développant h nous obtenons

$$\sum_k a^{(k)'}(Ap^{(i)}, p^{(k)}) = (r^{(0)}, p^{(i)}) \tag{19}$$

On constate alors que résoudre $Ax = k$ revient à résoudre le système linéaire (19), c'est-à-dire trouver les a'. Cependant il y aura toujours des erreurs d'arrondi, mais ce système (19) devrait avoir une diagonale strictement dominante car les vecteurs $p^{(i)}$ sont A-conjugués or en vertu du cours de cette année, de telles matrices ont de bonnes caractéristiques pour la résolution de systèmes linéaires.

De plus, on constate que les formules (16) et (17) mettent en valeur le fait que, si le quotient $\frac{a^{(i)}}{a^{(i-1)}}$ augmente alors les produits scalaires $(r^{(i+1)}, r^{(i)})$ et $(Ap^{(i+1)}, p^{(i)})$ vont aussi augmenter, les paramètres accélération (les $a^{(i)}$) sont donc un bon indicateur de la propagation des erreurs.

On veut ainsi que les quotients $\frac{(Ap^{(i)}, p^{(k)})}{(Ap^{(i)}, p^{(i)})}$, avec $i \neq k$, soit très petit et notamment pour $k = i + 1$. C'est là qu'intervient (17) et nous en déduisons donc que plus les quotients $\frac{a^{(i)}}{a^{(i-1)}}$ augmentent plus on s'éloigne d'un système à diagonale dominante et on obtient donc un système de moins en moins simple à résoudre, ce qui engendre donc des erreurs d'arrondi.

On a donc mis en valeur le fait que l'on s'éloigne de la solution a' quand la propagation d'erreur est forte.

Nous avons vu précédemment que le nombre de conditionnement correspond au quotient $\frac{\lambda max}{\lambda min}$. Il est important de noter que lorsque ce quotient tend vers un, c'est-à-dire que le spectre de la matrice est très serré dans le sens où les valeurs propres sont proches l'une de l'autre, la matrice va se rapprocher de la matrice constante $c * I$ avec c une constante.

Par conséquent, ces matrices diagonales sont très faciles à résoudre (cf. rôle du conditionnement) et ce, de manière très rapide car les algorithmes utilisés par les logiciels de calcul numérique tels que Matlab traitent les matrices creuses et a fortiori diagonales de manière spécifiques (stockage en tableau linéaire par exemple, le logiciel s'organise pour que son utilisation de la mémoire vive soit la plus efficace), c'est-à-dire qu'en analysant le format et la composition de la matrice, l'ordinateur va pouvoir adapter son calcul et omettre de nombreuses opérations !

En outre, un encadrement du quotient $\frac{a^{(i)}}{a^{(i-1)}}$ est obtenu à l'aide de l'encadrement $\frac{1}{\lambda max} < a^{(i)} < \frac{1}{\lambda min}$ que l'on peut obtenir par le calcul du Théorème (20) [4].

4. Ce théorème est vu dans la partie suivante et une preuve est donnée dans l'[article]

Ainsi nous obtenons une relation entre le rayon spectral de la matrice et la propagation d'erreurs d'arrondi : resserrer le spectre, c'est-à-dire réduire l'écart entre λmax et λmin implique une réduction de la propagation des erreurs d'arrondi.

2.6 Convergence de la méthode [5]

Théorème 1. *Soit A une matrice définie positive d'ordre n. Toute méthode qui utilise des directions conjuguées pour résoudre $Ax = k$ conduit à la solution exacte en au plus n itérations .*

Démonstration. Les directions $p^{(0)}, p^{(1)}, ..., p^{(n-1)}$ forment une base A-orthogonale de $\mathbb{R}^n$. De plus, par (8), le vecteur $r^{(i)}$ est orthogonale à l'espace $P_{i-1} = Vect\{p^{(0)}, p^{(1)}, ..., p^{(i-1)}\}$. Donc $r^{(n)}$ est orthogonal à P_{n-1}, or $P_{n-1} = \mathbb{R}^n$, donc $r^{(n)} = 0$, d'où $x^{(n)} = h$. □

Théorème 2. *Soit A une matrice définie positive d'ordre n, et soit λmin (respectivement λmax) la valeur propre de plus petit module (respectivement de plus grand module). La méthode du gradient conjugué pour la résolution de $Ax = k$ converge après au plus n étapes. De plus, $\forall i \in \{0, ..., i-1\}$,*

$$\| err^{(i)} \|_A \leq 2\left(\frac{\sqrt{\lambda max} - \sqrt{\lambda min}}{\sqrt{\lambda max} + \sqrt{\lambda min}}\right)^i \| err^{(0)} \|_A \tag{20}$$

où $\| err^{(i)} \|_A = (Aerr, err)^{\frac{1}{2}}$.

Démonstration. La convergence de la méthode en au plus n étapes correspond au théorème 1. Et on admettra l'inégalité car la preuve fait appel à des connaissances qui ne sont pas de notre niveau.

□

Remarque 8. *Le taux de convergence est $\frac{\sqrt{\lambda max} - \sqrt{\lambda min}}{\sqrt{\lambda max} + \sqrt{\lambda min}}$,d'après (20), qui s'écrit aussi $1 - \frac{2}{\sqrt{\frac{\lambda max}{\lambda min}} + 1}$. Ainsi, la méthode converge plus vite lorsque le taux de convergence s'approche de zéro, soit quand le rapport $\frac{\lambda max}{\lambda min}$ est aussi proche de un que possible. Il est donc intéressant de connaitre le spectre de la matrice afin de prévoir la vitesse de convergence.*

3 Méthode du gradient conjugué pré conditionnée

3.1 Intérêt d'un pré conditionnement

En dimension n élevée, on ne désire pas effectuer les n itérations qui seraient trop couteuses. Pour cela, un autre critère d'arrêt s'impose, lorsque le résidu est inférieur à une certaine tolérance, on considère que l'approximation est suffisamment bonne.

5. Le Théorème 1 est dans le [livre1] et le Théorème 2 et sa preuve se trouvent dans le [cours1]

Dans ce cas là, on a vu dans la Remarque précédente que la convergence de la méthode dépend du rapport $\frac{\lambda max}{\lambda min}$. Or il arrive que ce rapport soit élevé, n'assurant pas ainsi une convergence rapide. De plus, l'étude en arithmétique finie nous montre qu'il est intéressant de réduire ce rapport afin de minimiser les erreurs d'arrondis. Pour remédier à ce problème, il est utile de pré conditionner la matrice A pour resserrer le spectre autour d'une seule valeur propre.

3.2 Le pré conditionnement

Si P est une matrice de pré conditionnement symétrique définie positive, la méthode du gradient conjugué pré conditionnée consiste à appliquer la méthode du gradient conjugué au système pré conditionné [6] :

$$\overline{A}y = \overline{k} \tag{21}$$

où $\overline{A} = P^{-\frac{1}{2}}AP^{-\frac{1}{2}}$; $y = P^{\frac{1}{2}}x$; $\overline{k} = P^{-\frac{1}{2}}k$.

Propriété 7. *P doit satisfaire quelques propriétés :*
- *P est symétrique définie positive ;*
- *si A est une matrice creuse, on souhaite que $\overline{A}$ le soit aussi, pour cela, P aussi ;*
- *P doit être facile à construire pour ne pas alourdir la méthode ;*
- *$Pz = r$ doit être facile à résoudre (on verra plus tard le rôle de z) ;*
- *enfin, le plus important : $Cond(\overline{A}) \simeq 1 << Cond(A)$.*

3.3 Algorithme de la méthode du gradient conjugué pré conditionnée

Remarque 9. *Le calcul de $P^{-1/2}$ et les multiplications lors du passage au système pré conditionné sont bien trop couteux. Pour cela, on peut exprimer les formules de la méthode du gradient conjugué en se ramenant à P^{-1}.*

Voici l'algorithme de la méthode du gradient conjugué pré conditionné :
- *Initialisation :* choisir $x^{(0)}$, $r^{(0)} = k - Ax^{(0)}$, $z^{(0)} = P^{-1}r^{(0)}$, $p^{(0)} = z^{(0)}$
- *Itérations :* Pour i allant de 0 à $n-1$,
  $a^{(i)} = \frac{(r^{(i)}, z^{(i)})}{(p^{(i)}, Ap^{(i)})}$
  $x^{(i+1)} = x^{(i)} + a^{(i)}p^{(i)}$
  $r^{(i+1)} = r^{(i)} - a^{(i)}Ap^{(i)}$
  $Pz^{(i+1)} = r^{(i+1)}$
  $b^{(i)} = -\frac{(r^{(i+1)}, z^{(i+1)})}{(r^{(i)}, z^{(i)})}$
  $p^{(i+1)} = z^{(i+1)} - b^{(i)}p^{(i)}$

Propriété 8. *Cet algorithme donne la même solution approchée x que celui de la méthode du gradient conjugué appliqué au système (21).*

6. cette définition a été donnée dans le [livre1]

Notons $\overline{a}$, $\overline{x}$, $\overline{r}$, $\overline{b}$ et $\overline{p}$ les quantités correspondantes au système (21). Alors, on a les égalités : $\overline{x^{(0)}} = P^{\frac{1}{2}}x^{(0)}$, $\overline{r^{(0)}} = P^{-\frac{1}{2}}r^{(0)}$, $\overline{p^{(0)}} = P^{\frac{1}{2}}p^{(0)}$ et pour tout i allant de 0 à $n-1$,
$\overline{a^{(i)}} = a^{(i)}$
$\overline{x^{(i+1)}} = P^{\frac{1}{2}}x^{(i+1)}$
$\overline{r^{(i+1)}} = P^{-\frac{1}{2}}r^{(i+1)}$
$\overline{b^{(i)}} = b^{(i)}$
$\overline{p^{(i+1)}} = P^{\frac{1}{2}}p^{(i+1)}$.

Démonstration. Procédons par récurrence :

— Initialisation :
En (21), $\overline{x}^{(0)} = y^{(0)} = P^{\frac{1}{2}}x^{(0)}$.
D'après l'algorithme de la méthode du gradient conjugué appliqué en (21), on a $\overline{r^{(0)}} = P^{-\frac{1}{2}}k - P^{-\frac{1}{2}}AP^{-\frac{1}{2}}\overline{x^{(0)}} = P^{-\frac{1}{2}}(k - AP^{-\frac{1}{2}}P^{\frac{1}{2}}x^{(0)}) = P^{-\frac{1}{2}}r^{(0)}$.
$\overline{p^{(0)}} = \overline{r^{(0)}} = P^{-\frac{1}{2}}r^{(0)} = P^{\frac{1}{2}}z^{(0)} = P^{\frac{1}{2}}p^{(0)}$.

— Hérédité : Supposons les propriétés vraies jusqu'au rang $i-1$ pour a et b, et jusqu'au rang i pour le reste,
Commençons par a : $\overline{a^{(i)}} = \frac{\|\overline{r^{(i)}}\|^2}{(\overline{p^{(i)}}, P^{-\frac{1}{2}}AP^{-\frac{1}{2}}\overline{p^{(i)}})} = \frac{(P^{-\frac{1}{2}}r^{(i)}, P^{-\frac{1}{2}}r^{(i)})}{(P^{-\frac{1}{2}}\overline{p^{(i)}}, AP^{-\frac{1}{2}}\overline{p^{(i)}})}$, sachant que $(P^{\frac{1}{2}})^t = P^{\frac{1}{2}}$ car P est symétrique définie positive. Et comme, par hypothèse de récurrence, $P^{-\frac{1}{2}}\overline{p^{(i)}} = p^{(i)}$, on a que $\overline{a^{(i)}} = \frac{(r^{(i)}, P^{-1}r^{(i)})}{(p^{(i)}, Ap^{(i)})}$. De plus, $P^{-1}r^{(i)} = z^{(i)}$, d'où $\overline{a^{(i)}} = a^{(i)}$.
Procédons pour x : $\overline{x^{(i+1)}} = \overline{x^{(i)}} + a^{(i)}\overline{p^{(i)}} = P^{\frac{1}{2}}x^{(i)} + a^{(i)}P^{\frac{1}{2}}p^{(i)}$, par hypothèse de récurrence sur x et p. Ainsi $\overline{x^{(i+1)}} = P^{\frac{1}{2}}(x^{(i)} + a^{(i)}p^{(i)}) = P^{\frac{1}{2}}x^{(i+1)}$.
Procédons pour r : $\overline{r^{(i+1)}} = \overline{r^{(i)}} - a^{(i)}P^{-\frac{1}{2}}AP^{-\frac{1}{2}}\overline{p^{(i)}} = P^{-\frac{1}{2}}(r^{(i)} - a^{(i)}AP^{-\frac{1}{2}}P^{\frac{1}{2}}p^{(i)}) = P^{-\frac{1}{2}}r^{(i+1)}$.
Procédons pour b : $\overline{b^{(i)}} = -\frac{\|\overline{r^{(i+1)}}\|^2}{\|\overline{r^{(i)}}\|^2} = -\frac{(r^{(i+1)}, z^{(i+1)})}{(r^{(i)}, z^{(i)})}$, par le même calcul que pour a. D'où $\overline{b^{(i)}} = b^{(i)}$.
Finalement, pour p : $\overline{p^{(i+1)}} = \overline{r^{(i+1)}} - b^{(i)}\overline{p^{(i)}} = P^{-\frac{1}{2}}r^{(i+1)} - b^{(i)}P^{\frac{1}{2}}p^{(i)} = P^{\frac{1}{2}}(P^{-1}r^{(i+1)} - b^{(i)}p^{(i)}) = P^{\frac{1}{2}}(z^{(i+1)} - b^{(i)}p^{(i)}) = P^{\frac{1}{2}}p^{(i+1)}$.
Ceci conclut la récurrence.

Ces égalités nous montrent qu'une fois le x trouvé par l'égalité $y = P^{\frac{1}{2}}x$, avec y solution approchée du système (21) par la méthode du gradient conjugué ; celui-ci est le même que celui donné par le nouvel algorithme.

□

4 Mise en oeuvre sur MATLAB

4.1 Un test d'arrêt basé sur le résidu [7]

Pour arrêter l'algorithme, vu qu'on ne sait pas calculer directement l'erreur, il se trouve qu'on peut la majorer par le résidu grâce à :

$$\| h - x^{(i)} \| = \| A^{-1}k - x^{(i)} \| = \| A^{-1}r^{(i)} \| \leq \| A^{-1} \| \| r^{(i)} \|$$

En général, l'erreur relative est une valeur plus pertinente comme critère d'arrêt, or vu précédemment, on la majore par le biais du résidu normalisé $\frac{\|r^{(i)}\|}{\|k\|}$. Effectivement :

$$\frac{\| h - x^{(i)} \|}{\| h \|} \leq \frac{\| A^{-1} \| \| r^{(i)} \|}{\| A^{-1}Ah \|}$$

or $Ah = k$, ainsi

$$\frac{\| h - x^{(i)} \|}{\| h \|} \leq \frac{\| A^{-1} \| \| r^{(i)} \|}{\| A^{-1}k \|}$$

De plus, par le théorème spectral, $\lambda min \| k \| \leq \| Ak \|$. En appliquant ceci à A^{-1}, on trouve que $\frac{1}{\lambda max} \| k \| \leq \| A^{-1}k \|$. Et donc

$$\frac{\| h - x^{(i)} \|}{\| h \|} \leq \frac{\| A^{-1} \| \| r^{(i)} \|}{\| A^{-1}k \|} \leq \frac{\rho(A^{-1}) \| r^{(i)} \|}{\frac{1}{\lambda max} \| k \|} = \frac{\frac{1}{\lambda min} \| r^{(i)} \|}{\frac{1}{\lambda max} \| k \|} = \frac{\lambda max}{\lambda min} \frac{\| r^{(i)} \|}{\| k \|}$$

Lors de la mise en oeuvre, on prendra comme critère d'arrêt : $error = \frac{\|r^{(i)}\|}{\|k\|} \leq tol$, où tol est la tolérance choisie.

7. Cette sous-partie a été inspirée du [livre1]

4.2 Implémentation sur MATLAB [8]

```
function[x,error,iter,flag]= cgp(A,x,k,P,tol)
n= size(A,1);
flag = 0;
iter = 0;
knorm = norm(k);
if (knorm == 0) %Pour éviter les divisions par zéro, si k est nul,
    knorm = 1; %on considère l'erreur absolue au lieu de l'erreur relative.
end
r = k - A*x;
error = norm(r)/knorm;
if (error < tol ) return
end
for iter = 1:n
    z = P \ r; %on sous-traite le calcul de P*z=r, avec une fonction de Matlab
    rho = (r'*z);
    if iter > 1
        b = - rho /rho1;
        p = z - b*p;
    else
        p= z;
    end
    q = A*p;
    a = rho/ (p'*q);
    x = x + a*p;
    r = r - a*q;
    error=norm(r)/knorm;
    if (error <= tol)
        break
    end
    rho1 = rho; %rho1 permet de stocker la valeur de rho à l'étape antérieure
end
if (error <= tol)
        flag = 1;
end
end
```

Remarque 10. *Ce code retourne :*

- *x, l'approximation de la solution du système ;*
- *error, expliqué précédemment ;*
- *iter, le nombre d'itérations effectuées ;*
- *flag, qui vaut 1 si error $\leq$ tol en fin d'algorithme et 0 sinon.*

Remarque 11. *Lors de l'application de ce code, nous allons prendre $P = I_n$ car, en pratique, il n'est pas évident de trouver une matrice de grande taille pour laquelle on connait un pré conditionnement adapté.*

4.3 Une première application dans un cas favorable

L'[article] nous propose un exemple dans lequel la solution trouvée est exacte : tous les vecteurs calculés lors des itérations ont des coordonnées entières, ce qui fait qu'il n'y a pas d'erreur d'arrondi.

8. Le code, même si légérement modifié, provient du [livre1]

Nous prenons :

$$A = \begin{pmatrix} 1 & 2 & -1 & 1 \\ 2 & 5 & 0 & 2 \\ -1 & 0 & 6 & 0 \\ 1 & 2 & 0 & 3 \end{pmatrix} ; k = \begin{pmatrix} 0 \\ 2 \\ -1 \\ 1 \end{pmatrix} ; x^{(0)} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} ; P = I_4 ; tol = 0.0001.$$

Le code nous retourne :

$$x = \begin{pmatrix} -65 \\ 24 \\ -11 \\ 6 \end{pmatrix} ; error = 0 ; iter = 4 ; flag = 1.$$

On remarquera que le pré conditionnement n'est pas nécessaire. En effet, il suffit de quatre étapes pour obtenir la solution exacte, même si le rapport $cond(A) = \frac{\lambda max}{\lambda min} = 553.0780$.

Ceci est un cas particulier car on s'attend généralement à ce que l'algorithme fasse appel à des vecteurs utilisant des coordonnées à virgule flottante.

4.4 Différents résultats selon la valeur du second membre k

On souhaite appliquer la méthode à des matrices de taille plus importante, pour mettre à l'épreuve notre algorithme. Dans ce cas on va étudier l'influence du vecteur k. On prendra des matrices de taille $n = 20$. Les paramètres d'entrée sont :

$$A = \begin{pmatrix} 2 & -1 & 0 & \cdots & \cdots & \cdots & 0 \\ -1 & 2 & -1 & 0 & & & \vdots \\ 0 & -1 & 2 & -1 & 0 & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & 0 & -1 & 2 & -1 & 0 \\ \vdots & & & 0 & -1 & 2 & -1 \\ 0 & \cdots & \cdots & \cdots & 0 & -1 & 2 \end{pmatrix} ; x^{(0)} = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix} ; P = I_{20} ; tol = 10^{-12}.$$

Dans un premier cas :

$k_1 = \begin{pmatrix} 0 \\ 1 \\ \vdots \\ 1 \\ 0 \end{pmatrix}$ ce qui donne en sortie : $error1 = 6.0459 * 10^{-17}$, $iter1 = 9$, $flag1 = 1$ et les composantes de $Ax - k_1$ sont de l'ordre de $10^{-14} << 1$.

Dans un deuxième cas : $k2 = rand^{9}(20, 1)$ ce qui donne en sortie : $error2 = 1.7443 * 10^{-16}$, $iter2 = 20$, $flag2 = 1$. Les composantes de $Ax - k_2$ sont de l'ordre de $10^{-14} << 1$.

9. rand est une fonction MATLAB qui donne une matrice à composantes entre 0 et 1 tirées de façon aléatoire

Remarque 12. *Notons que* $cond(A) = 178.0643$*, ce qui n'est pas énorme ni minuscule pour une matrice de taille* 20*. Les résultats sont précis, ce qui affirme la justesse de la méthode dans des cas ni bien, ni mal conditionnés.*

Remarque 13. *Pour l'influence de* k*, on observe que lorsque celui-ci est régulier, la convergence est très satisfaisante car une excellente précision est atteinte en moins de la moitié du nombre d'étapes prévues.*

Par ailleurs, lorsque k *ne présente pas de régularités (ici généré aléatoirement), l'algorithme atteint la même précision après les* 20 *itérations.*

4.5 Une troisième application mal conditionnée

Pour cette troisième application, on prend :

$$B = hilb^{10}(20)\,;\, x^{(0)} = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix}\,;\, P = I_{20}\,;\, tol = 10^{-12}\,;\, k^{(0)} = \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}.$$

Ce qui donne :

$error3 = 29.7848\,;\, iter3 = 20\,;\, flag3 = 0$. Les composantes de $Bx - k_3$ sont de l'ordre de 10 ce qui n'est pas négligeable devant les composantes de k_3.

Remarque 14. *Notons que* $cond(B) = 3.2437 * 10^{18}$*, ce qui illustre la relation entre le conditionnement et les erreurs d'arrondi. Cette matrice aurait besoin d'un pré conditionnement afin de réduire l'erreur.*

5 Liens avec l'optimisation et l'approximation polynomiale

5.1 Optimisation

Les problèmes d'optimisation cherchent à trouver des extrema pour des fonctions assez régulières.

Pour une fonction $q : \mathbb{R}^n \longrightarrow \mathbb{R}$, de classe C^1, la méthode du gradient permet de trouver les minimums de q. Si, de plus, q est quadratique et définie positive alors, en notant ϕ sa forme polaire, il est possible de mettre en place la méthode du gradient conjuguée, en s'assurant que les directions sont ϕ-orthogonales. Dans ce contexte, la résolution d'un système linéaire revient à minimiser une certaine fonction quadratique, celle qu'on a introduite f.

10. hilb est une fonction MATLAB qui donne directement une matrice de Hilbert

5.1.1 Un cas particulier d'optimisation : la méthode du point fixe

Soit $A \in \mathbb{R}^{nxn}$ une matrice symétrique définie positive. A est aussi la matrice d'un endomorphisme $u_A : \mathbb{R}^n \to \mathbb{R}^n$ où $\mathbb{R}^n$ est muni d'une norme $\| \ \|$, on assimilera u_A à A. A peut donc être vue comme une application linéaire continue sur l'espace complet $\mathbb{R}^n$.

L'inégalité $\| Ax \| \leq \| A \| \| x \|$ donne que pour tout (x, y) de $\mathbb{R}^n$,$\| Ax - Ay \| = \| A(x - y) \| \leq \| A \| \| x - y \|$. Si $\| A \| < 1$, l'application A est donc contractante, et le théorème du point fixe s'applique.

Un problème se pose lorsqu'on travaille avec les méthodes itératives numériquement, le corps $\mathbb{R}$ est approché par les nombres à virgule flottante et l'espace vectoriel n'est donc pas complet.

La méthode du gradient conjugué utilise la suite itérative $x_{k+1} = x_k + \alpha_k d_k$ pour trouver le minimum de la fonction $f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^t A\boldsymbol{x} - \boldsymbol{b}^t\boldsymbol{x}$.

Si on pose $g(x) = x + \alpha d$ et $x_{k+1} = g(x_k)$ on voit que la suite $(x_k)_{k\in\mathbb{N}}$ définie ci-dessus ressemble à une suite contractante (en supposant que le corps associé est $\mathbb{R}$).

5.2 Approximation polynomiale

5.2.1 La formule de Newton

On veut mettre à jour l'approximation $\Pi_{n-1}(x)$ de f étant donné un point $(x_n, y_n) \in \mathbb{R}^2$. On pose $\Pi_n(x) = \Pi_{n-1}(x) + q_n(x)$ ou $q_n(x)$ est un polynôme de degré n. En écrivant $q_n(x) = \Pi_n(x) - \Pi_{n-1}(x)$ on voit que $q_n(x)$ s'annule en $(x_i)_{i=0}^{n-1}$, ainsi écrire $q_n(x) = a_n w_n(x)$ où $w_n(x) = \prod_{i=0}^{n-1}(x - x_i)$.

La formule de newton correspond à la version itérative des polynômes de Lagrange, ce qui est un parallèle avec la méthode itérative du gradient conjugué qui est aussi liée à une méthode directe.

5.2.2 Polynômes orthogonaux

Si $(f_n(x))_{n\in\mathbb{N}} \in (\mathcal{C}^0([a, b])$ est une suite convergente pour la norme $\| \ \|_\infty$ alors elle converge aussi pour la norme $\| \ \|_2$ $(= (\int_a^b |f|^2 dx)^{\frac{1}{2}})$.

En outre, on a que $\| f-g \|_{L^2([a,b])} = (\int_a^b |f-g|^2 dx)^{\frac{1}{2}} \leq \int_a^b |f-g| dx \leq sup_{f,g\in\mathcal{C}_0([a,b])}|f-g||b-a| = \| f - g \|_\infty |b-a|$.

L'application $\mathcal{T} : (\mathcal{C}^0([a, b]), \| \ \|_\infty) \to \mathcal{L}^2([a, b])$ est donc une injection.

Soit $\omega(x) \in L^1([a, b])$ à valeurs dans $\mathbb{R}^+$.

On peut alors définir un produit scalaire sur $\mathcal{C}^0([a, b])$ par $(f, g)_\omega = \int_a^b f(x)\overline{g(x)}\omega(x)dx$, et ensuite une norme $\| f \|_2 = \| f \|_{L^2([a,b])} := (f, f)_\omega^{\frac{1}{2}}$.

On voit maintenant que $\| f \|_2 = (\int_a^b f(x)\overline{f(x)}\omega(x)dx)^{-\frac{1}{2}} \leq \int_a^b \omega(x)dx \| f \|_\infty$

5.2.3 Correspondance avec les espaces Lp à poids

Par la formule de quadrature de Gauss on peut créer une correspondance isométrique entre une base pour $\mathbb{R}^n$ A orthogonale et une base de polynômes orthogonaux pour $\mathbb{P}^n$.

Soit A une matrice symétrique définie positive. On veut résoudre le systéme $Ax = k$. Étant donné un vecteur d'initiation x_0 on pose $k - Ax_0 = r_0 = \alpha_1 e_1 + ...\alpha_n e_n$ ou $e_1...e_n$ sont les vecteurs propres normalisés de A. On remarque que $A^k r_0 = {\lambda_1}^k e_1 + ... + {\lambda_n}^k e_n$ ou $\lambda_i, i = 1...n$ sont les valeurs propres de A.

Soit ${p_k}_{k=1}^n$ une base de polynômes orthogonaux pour le produit scalaire $\int_0^l p_k p_j \omega(x)dx$. La formule de quadrature de Gauss dit que pour un polynôme quelconque $R(x)$ (de degré inférieur ou égal à $2n-1$) il existe des constantes $m_i, i = 1...n$ telles que $\int_0^l R(x)\omega(x)dx = m_1 R(x_1) + ...m_n R(x_n)$ où $(x_k)_{k=1}^n$ sont les zéros de $p_n(x)$ le n^{ime} polynôme de la base $(p_k)_{k=1}^n$.

Si on pose $m_i = {\alpha_i}^2 i = 1...n$ et $R(x) = x^{j+k}$ avec $j + k \leq 2n - 1$ on voit que $\int_0^l x^{j+k}\omega(x)dx = m_1 {\lambda_1}^{j+k} + ... + m_n {\lambda_n}^{j+k} = {\alpha_1}^2 A^{j+k} + ... + {\alpha_n}^2 A^{j+k} = \left(A^j r_0, A^k r_0\right)$, ce qui donne par correspondance isométrique, $\int_0^l R(x)R'(x)\omega(x)dx = (r, r')$. où R et R' sont de degré inférieur ou égal à n-1.

De la même manière on a que $\int_0^l R(x)R'(x)x\omega(x)dx = (r, Ar')$. On peut donc parler de polynômes "A-conjugés".

Remarque : il est important que la matrice A ci-dessus a comme valeurs propres les zéros de $p_n(x)$.

5.2.4 Extension de la méthode du gradient conjugé pour les polynômes

On peut montrer qu'une suite de polynômes orthogonaux satisfait la relation de récurrence : $R_{i+1} = (d_i + a_i x)R_i(x) + c_i R_{i-1}(x)$.

Si on normalise la suite en posant $R_i(x) = 1, i = 0, ...n$, on peut ré-écrire cette formule sous la forme : $P_i(x) = R_i(x) + b_i P_{i-1}(x)$ ou $P_i(x) = -\frac{R_{i+1} - R_i}{a_i x}$ et $b_i = \frac{\int_0^l R_i(x)^2 \omega(x)dx}{\int_0^l R_{i-1}(x)^2 \omega(x)dx} = \frac{\|r_i\|^2}{\|r_{i-1}\|}$.

Si on pose $a_i = \frac{\|r_i\|^2}{\|p_i\|_A{}^2}$ on obtient de la même manière la relation : $R_{i+1}(x) = R_i(x) - a_i x P_i(x)$.

Les deux formules récursives pour R_{i+1} et P_{i+1} correspondent isométriquement aux relations récursives qu'on trouve dans la méthode du gradient conjugé ; $p_i = r_i + b_{i-1} p_{i-1}$ et $r_{i+1} = r_i - a_i A p_i$.

6 Bibliographie

[article] Magnus R. HESTENES et Eduard STIEFEL, *Methods of Conjugate Gradients for Solving Linear Systems*, Journal of Research of the National Bureau of Standards, 1952.

[livre1] A.QUARTERONI, R.SACCO et F.SALERI, *Méthodes numériques pour le calcul scientifique : Programmes en MATLAB*, Springer, Collection IRIS, 2000.

[livre2] A.QUARTERONI, R.SACCO et F.SALERI, *Méthodes numériques : Algorithmes, analyse et applications*, Springer, 2007.

[livre3] Kermit SIGMON, *MATLAB - Aide mémoire*, Springer, 1999.

[livre4] J. STOER et R.BULIRSCH, *Introduction to Numerical Analisis*, Springer, 1993.

[livre5] G. SZEGO, *Orthogonal Polynomials*, Colloquium Publications, American Mathematical Society, 1939.

[cours1] Amic FROUVELLE, *Méthodes numériques : optimisation L3 2015-2016 - 2e semestre*, 18 avril 2016. Lien :*https* : *//www.ceremade.dauphine.fr/ amic/enseignement/MNO*2016/*MNO*2016.*pdf*.

[cours2] Thomas CLUZEAU, *Analyse Numérique. Lien* : *http* : *//www.unilim.fr/pages*$_p$*erso/thomas.cluzeau/Enseignement/PolyAnalyseNum.pdf*.

[cours3] Jonathan R. SHEWCHUK, *An Introduction to the Conjugate Gradient Method Without the Agonizing Pain*, 1994. Lien : *https* : *//www.cs.cmu.edu/ quake* − *papers/painless* − *conjugate* − *gradient.pdf*.

[cours4] Daniel KAUTH, *Optimisation numérique-Méthodes du gradient conjugué linéaire*, 2009 Lien : *http* : *//perso.unifr.ch/ales.janka/numeroptim/*07$_c$*onjgrad.pdf*.

7 Signatures