Étudiants ingénieurs en aérospatial

Mémoire de 3e année

Optimisation des méthodes itératives pour la résolution de systèmes linéaires

Auteurs :
M. Audet Yoann
M. Chandon Clément
M. De Claverie Chris
M. Huynh Julien

Encadrant :
Pr. Bletzacker Laurent

Version 0.0 du
6 avril 2019

Remerciements

Table des matières

Chapitre 1

Introduction

Chapitre 2

Présentation des méthodes itératives classiques

2.1 Présentation générale des méthodes

2.2 Méthodes classiques

2.2.1 Méthode de Jacobi

2.2.2 Méthode de Gauss-Seidel

2.3 Une nouvelle méthode : Richardson

2.3.1 Présentation de la méthode

Ci-dessus, nous avons exposé les deux principales méthodes que l'on a utilisé lors des cours et TP. Cependant, il est aussi possible pour nous de trouver d'autres méthodes de résolution. Pour cela, il nous faut juste réécrire le problème sous une autre forme que celles précédemment définies. Ainsi, nous pouvons utiliser la décomposition de la forme :

$$Ax = b \tag{2.1}$$
$$Px = (P - A)x + b \tag{2.2}$$

On remarque que peut importe la valeur de la matrice P dans l'équation ci-dessus, les deux équations sont équivalentes. Ainsi, résoudre le premier système revient donc à résoudre le second. La méthode Richardson se base sur cette décomposition. L'idée est de poser :

$$P = \beta I \text{ avec } I \text{ la matrice identité et } \beta \in \mathbb{R}^* \tag{2.3}$$

Ainsi, nous avons notre système qui s'écrit de la manière suivante :

$$\begin{aligned} \beta I x &= (\beta I - A)x + b \quad (2.4) \\ x &= (I - \frac{1}{\beta}A)x + \frac{1}{\beta}b \quad (2.5) \end{aligned}$$

Pour un soucis d'écriture, nous allons écrire la formule précédente sous la forme :

$$x = (I - \gamma A)x + \gamma b \text{ avec } \gamma = \frac{1}{\beta} \quad (2.6)$$

Ainsi l'idée est de construire une suite $x^{(k)}$ qui va converger vers la solution exacte du système que l'on notre ici x^*. Cette suite est définie de la manière suivante :

$$x^{(k+1)} = (I - \gamma A)x^k + \gamma b \quad (2.7)$$

Par définition de la suite, la matrice d'itération, notée ici R est :

$$R = I - \gamma A \quad (2.8)$$

Nous réécrivons la suite sous la forme :

$$x^{(k+1)} = Rx^k + K \text{ avec } K = \gamma b \quad (2.9)$$

Si cette suite converge, alors nous sommes en mesure de trouver une solution x^* approchant la vraie solution du système. Ainsi, l'étude se porte donc sur la convergence de cette suite. Comme pour les autres méthodes itératives, la condition de convergence est la même que précédemment : le rayon spectrale de la matrice d'itération doit être strictement inférieur à 1. L'avantage de cette méthode est que la matrice d'itération dépends de γ. Ainsi, en jouant sur cette valeur de γ, il est possible de faire converger la suite en prenant une valeur qui fait que le rayon spectral est inférieur à 1. On peut même produire une étude qui fait que l'on va minimiser cette valeur du rayon spectral pour obtenir une meilleur convergence. Cette démarche sera expliqué dans la suite de l'exposé.

2.3.2 Étude de convergence sur un exemple

Pour illustrer cette exemple, nous allons prendre un système linéaire quelconque. Dans un premier temps, nous allons trouver sa solution théorique puis appliquer la méthode de Richardson. Cela nous permettra d'étudier la convergence de la suite et la condition d'arrêt de notre algorithme. Pour cela, nous allons prendre le système 2×2 suivant :

$$\begin{cases} -3x + 2y = 1 \\ x + -4y = -7 \end{cases} \quad (2.10)$$

Ce système de base est peut être résolu assez trivialement et on obtient le couple de solution suivant :

$$(x, y) = (1, 2) \tag{2.11}$$

Notre but est maintenant de retrouver ces résultats grâce à la méthode de Richardson. Pour cela nous écrivons le système (2.10) sous sa forme matricielle :

$$\underbrace{\begin{pmatrix} -3 & 2 \\ 1 & -4 \end{pmatrix}}_{A} \times \underbrace{\begin{pmatrix} x \\ y \end{pmatrix}}_{x} = \underbrace{\begin{pmatrix} 1 \\ -7 \end{pmatrix}}_{b} \tag{2.12}$$

On pose, d'après la définition de la méthode, la matrice P :

$$P = \gamma I = \begin{pmatrix} \gamma & 0 \\ 0 & \gamma \end{pmatrix} \tag{2.13}$$

et on rappelle que l'on a :

$$x^{(k+1)} = (I - \gamma A)x^k + \gamma b \text{ avec } R = (I - \gamma A) \tag{2.14}$$

Dans notre cas, la matrice d'itération est la suivante :

$$R = \begin{pmatrix} 1 + 3\gamma & -2\gamma \\ -\gamma & 1 + 4\gamma \end{pmatrix} \tag{2.15}$$

On cherche les valeurs propres de celle-ci grâce son polynôme caractéristique :

$$\begin{aligned} det(R - \lambda I) &= \begin{bmatrix} 1 + 3\gamma - \lambda & -2\gamma \\ -\gamma & 1 + 4\gamma - \lambda \end{bmatrix} && (2.16) \\ &= ((1 + 3\gamma) - \lambda)((1 + 4\gamma) - \lambda) - 2\gamma^2 && (2.17) \\ &= \lambda^2 - (2 + 7\gamma)\lambda + 1 + 7\gamma + 10\gamma^2 && (2.18) \\ &= \lambda^2 - (2 + 7\gamma)\lambda + (1 + 2\gamma)(1 + 5\gamma) && (2.19) \\ &= (\lambda - (1 + 2\gamma))(\lambda - (1 + 5\gamma)) && (2.20) \end{aligned}$$

Ainsi, les deux valeurs propres sont :

$$\lambda_1 = 1 + 2\gamma \text{ ou } \lambda_2 = 1 + 5\gamma \tag{2.21}$$

Il nous faut donc maintenant étudier le rayon spectral :

$$\rho(R) = max(|1 + 2\gamma|, |1 + 5\gamma|) < 1 \tag{2.22}$$

Pour trouver le maximum, on cherche quand les quantités sont égales :

$$\begin{cases} 1+2\gamma = 1+5\gamma \Leftrightarrow \gamma = 0 \\ 1+2\gamma = -1-5\gamma \Leftrightarrow \gamma = -\frac{2}{7} \end{cases} \tag{2.23}$$

Il vient de cette étude :

$$\begin{cases} \gamma \in [-\frac{2}{7}, 0] \Rightarrow \rho(R) = |1+2\gamma| \\ \text{Sinon } \rho(R) = |1+5\gamma| \end{cases} \tag{2.24}$$

Nous cherchons ensuite les valeurs pour lesquelles le rayon spectral est égal à 1. Comme les deux fonctions sont croissantes, il suffit de trouver les valeurs pour lesquels nous avons $\rho(R) = 1$ ou -1.

$$\left\{ \gamma = 0 \Leftrightarrow \rho(R) = 1 \gamma = -0.4 \Leftrightarrow \rho(R) = -1 \right. \tag{2.25}$$

Ainsi, pour que la méthode converge sur cet exemple, il faut que :

$$\gamma \in]-0.4, 0[\tag{2.26}$$

Ensuite, il est possible d'optimiser ce résultat. Pour cela, il nus faut trouver la valeur de γ telle que le rayon spectral soit minimal. Pour cela, on cherche sur chacun des intervalles le minimum du rayon spectral. Cette valeur est la valeur à la jonction des deux intervalles donc pour $\gamma = \frac{-2}{7}$. Cela se voit simplement en regardant le graph de rho sur l'intervalle ci-dessus. Pour cette valeur de γ particulière la méthode possède la meilleur convergence. Si on revient au problème de base, nous avons alors un méthode qui converge de la meilleur façon possible pour :

$$\beta = \frac{1}{\gamma} = -\frac{7}{2} \tag{2.27}$$

2.3.3 Un peu plus de théorie ...

Maintenant que nous avons montrer la démarche sur un exemple, nous allons essayer de généraliser aux matrices quelconques que l'on veut étudier grâce à cette méthode. Dans un premier temps, nous allons étudier les valeurs propres de la matrice d'itération R (cf. équation 2.8). En notant λ_i les valeurs propres de la matrices A et μ_i les valeurs propres de la matrice R, nous avons :

$$\mu_i = 1 - \gamma\lambda_i \tag{2.28}$$

En appliquant la condition de convergence de la suite, nous obtenons les égalités sui-

vantes :

$$\begin{aligned} -1 &\leq 1-\gamma\lambda_i \leq 1 && (2.29)\\ 0 &\leq \gamma\lambda_i \leq 2 && (2.30)\\ 0 &\leq \gamma \leq \frac{2}{\lambda_i} && (2.31) \end{aligned}$$

On remarque que sur notre exemple cela est vrai. En effet, les valeurs propres de la matrice A choisie sont -5 et -2. Or $\frac{2}{-5} = -0.4$, cela confirme l'intervalle trouvé. La deuxième remarque porte sur le fait qu'il ne faut pas prendre une matrice A avec 0 en valeur propre.

Toujours dans le même esprit, nous allons chercher le meilleur γ théorique pour avoir la meilleur convergence. Ce problème est équivalent à minimiser le rayon spectral de la matrice d'itération qui dépends de γ. Or d'après les valeurs propres de cette matrice R, nous avons :

$$\rho(R) = \underset{i}{max}(|1-\gamma\lambda_i|) = max(|1-\gamma\lambda_1|, |1-\gamma\lambda_n|) \tag{2.32}$$

où λ_1, λ_n sont respectivement la plus grande et la plus petite valeur propre. Maintenant, il nous reste à résoudre :

$$|1-\gamma\lambda_1| = |1-\gamma\lambda_n| \Rightarrow \begin{cases} 1-\gamma\lambda_1 = 1-\gamma\lambda_n \Leftrightarrow \gamma = 0 \\ ou \\ 1-\gamma\lambda_1 = -1+\gamma\lambda_n \Leftrightarrow \gamma = \frac{2}{\lambda_1+\lambda_n} \end{cases} \tag{2.33}$$

Une fois que nous avons les valeurs de l'égalité, une simple étude des deux valeurs propres extrêmes nous donne que le meilleur choix de γ est :

$$\gamma = \frac{2}{\lambda_1+\lambda_n} \tag{2.34}$$

Chapitre 3

Optimisation du choix de la matrice d'itération

Nous avons vu dans la partie précédente qu'il existe différentes méthodes pour permettre de résoudre un système linéaire grâce à des méthodes itératives. Ainsi, toujours dans cette idée d'optimisation que nous avons exposé, nous nous sommes posé la question suivante : « Quelle est la matrice d'itération la plus optimisé pour résoudre un problème ». Une méthode est ressortie dans plusieurs ouvrage : Successive Over Relaxation.

3.1 Méthode SOR

C'est dans cette optique que nous nous sommes penchés sur la méthode dite "SOR".

3.1.1 Présentation de la méthode SOR

La méthode SOR (Successive Over Relaxation) est une méthode itérative dérivée de Gauss-Siedel. En effet, le processus de décoposition de la matrice A en deux matrices M et N telles que $A = M - N$ est similaire à l'algorithme de Gauss-Seidel dans la forme des matrices M et N.

Si la méthode de Gauss-Seidel, vue précédemment, définie la matrice M par $M = D - E$ avec D une matrice diagonale et E une matrice triangulaire inférieure à diagonale nulle et $N = F$, F étant une matrice triangulaire supérieure à diagonale nulle, la méthode SOR définit ses matrices de la manière suivante, en introduisant un paramètre $\omega \in \mathbb{R}^*$ dit de relaxation.

$$M = \frac{1}{\omega}D - E \tag{3.1}$$

$$N = \left(\frac{1}{\omega} - 1\right)D + F \tag{3.2}$$

Par la suite, le procédé est le identique à celui de Gauss-Seidel ou Jacobi et on introduit donc sa matrice d'itération notée B.

$$B = M^{-1}N = \left[\frac{1}{\omega}D - E\right]^{-1}\left[\left(\frac{1}{\omega} - 1\right)D + F\right] \tag{3.3}$$

On remarquera que si $\omega = 1$, on retrouve la méthode de Gauss-Seidel. De plus, si $\omega < 1$, on parle de sous-relaxation et de sur-relaxation dans le cas où $\omega > 1$.

3.1.2 Intérêt de la méthode

Cette méthode a été développée peu après la Seconde Guerre mondiale afin de proposer une manière de résoudre des systèmes d'équations linéaires, spécifique aux ordinateurs. Si à l'époque, d'autres méthodes avaient été proposées, elles étaient principalement destinées aux êtres humains qui, par des processus non applicables par des ordinateurs, pouvaient assurer la convergence des méthodes. La méthode SOR est donc une méthode qui a fait progresser ce problème en ayant une meilleure vitesse de convergence que les méthodes numériques itératives alors utilisées.

L'avantage de la méthode SOR au niveau de la convergence est mathématiquement facilitée par les deux théorèmes suivant :

1. **Théorème de Kahan (1958)** : Le rayon spectral de la matrice de relaxation, donnée par :
$$T_\omega = T(\omega) = (I - \omega L)^{-1}\omega U + (1 - \omega)I$$
vérifie que $\forall \omega \neq 0$,
$$\rho(T_\omega) \geq |\omega - 1|$$
2. **Théorème d'Ostrowski-Reich (1949-1954)** : Si la matrice A est définie positive et que $\omega \in]0; 2[$,la méthode SOR converge pour tout choix de vecteur $x^{(0)}$ initial.

Afin qu'une méthode itérative converge, il est nécessaire que le rayon spectral de la matrice d'itération soit strictement inférieur à 1. Donc, pour que la méthode ne converge pas, il faut que le rayon spectral soit supérieur ou égal à 1. Avec le théorème de Kahan, on a :

$$|\omega - 1| \geq 1 \Leftrightarrow \omega \geq 2 \text{ ou } \omega \leq 0$$

Ainsi, nous pouvons déduire du premier théorème, une condition nécessaire non suffisante de la convergence de la méthode SOR qui est :

$$0 < \omega < 2 \tag{3.4}$$

Le deuxième théorème (Ostrowski-Reich), permet quant à lui de conclure par rapport à la convergence de la méthode pour ω dans l'intervalle $]0; 2[$. La combinaison de ces deux

théorèmes nous montre que la condition donnée à l'équation (3.4) est nécessaire et, est suffisante dans le cas où A est définie positive.

De plus, dans le cas où la matrice A est tridiagonale (les coefficients qui ne sont ni sur la diagonale principale, ni celle au dessus, ni celle au dessous, sont nuls), le théorème suivant nous donne la forme du coefficient de relaxation optimal :

Si A est définie positive et est tridiagonale, alors $\rho(T_g) = [\rho(T_j)]^2 < 1$ et, le choix optimal pour le coefficient de relaxation ω est donné par :

$$\omega_{optimal} = \frac{2}{1 + \sqrt{1 - [\rho(T_j)]^2}}$$

Avec ce choix de coefficient de relaxation, on a : $\rho(T_\omega) = \omega - 1$
La preuve de ce théorème est dans : Ortega, J. M., Numerical Analysis ; A Second Course, Academic Press, New York, 1972, 201 pp.
Preuve du théorème de Kahan : On a,

$$\prod_i \lambda_i(T(\omega)) = \det(T(\omega)) = \frac{\det(\omega U + (1-\omega)I)}{\det(I - \omega L)} = (1-\omega)^n$$

Or,

$$|\prod_i \lambda_i(T(\omega))| \leq \rho(T(\omega))^n \rightarrow |\omega - 1|^n \leq \rho(T(\omega))^n$$

Ainsi,

$$\rho(T(\omega)) \geq |\omega - 1|$$

Preuve du théorème d'Ostrowski-Reich : En utilisant le théorème de Kahan,on sait qu'il est nécessaire que $0 < \omega < 2$ est un critère nécessaire et non suffisant de convergence. De plus, pour une méthode SOR, on a

$$M_{SOR}(\omega) + M^*_{SOR}(\omega) - A = \left(\frac{2}{\omega - 1}\right) D \text{ puisque } L = U^*$$

qui est symétrique définie positive si on est dans l'intervalle donné par le théorème de Kahan. Le théorème de Householder-John nous dit que pour une matrice A hermitienne définie positive, avec $A = M - N$ avec M inversible, la méthode itérative converge pour toute donnée initiale si $M + N^*$ est définie positive. (N^* étant la matrice adjointe ou trans-conjugée à N soit $N^* = {}^t\overline{N} = \overline{{}^tN}$).

Afin d'optimiser l'algorithme, on utilise souvent la méthode SSOR (Symetric Successive Over-Relaxation) afin de préconditionner la matrice avant de la traîter, cette méthode sera traîtée ultérieurement dans la partie optimisation.

3.1.3 Implémentation numérique

3.2 Les espaces de Krylov

3.2.1 Présentation théorique

De Jacobi à Krylov

On rappelle le résultat de la partie précédente sur la méthode de jacobi qui s'écrit :

$$x^{k+1} = -D^{-1}(L+U)x^k + D^{-1}b = (I - D^{-1}A)x^k + D^{-1}b \tag{3.5}$$

avec la matrice A du système qui se décompose comme : $A = D + L + U$, L une matrice triangulaire inférieur, U un matrice triangulaire supérieur et D diagonale. La matrice A est inversible car le système est supposé avoir une unique solution. On définit ensuite le résidu du système qui est par définition :

$$r^k \triangleq b - Ax^k = -A(-A^{-1}b + x^k) = -A(-x^* + x^k) \tag{3.6}$$

Où x^* est la solution réel du système. En normalisant le système ci-dessus de tel sorte que $D = I$. Alors, nous pouvons écrire la solution au rang k+1, comme celle au rang k plus le résidu :

$$\begin{aligned} x^{k+1} &= x^k + r^k && (3.7)\\ \Leftrightarrow x^{k+1} - x^* &= x^k - x^* + r^k && (3.8)\\ \Leftrightarrow -A(x^{k+1} - x^*) &= -A(x^k - x^*) - Ar^k && (3.9)\\ \Leftrightarrow r^{k+1} &= r^k - Ar^k && (3.10) \end{aligned}$$

Dans cette dernière équation récursive, nous pouvons voir que r^{k+1} est une combinaison linéaire des vecteurs r^k précédents. Ainsi :

$$r^k \in Vect\{r^0, Ar^0, ..., A^k r^0\} \tag{3.11}$$

Cela implique directement :

$$x^k - x^0 = \sum_{i=0}^{k-1} r^i \tag{3.12}$$

Donc il vient que :

$$x^k \in x^0 + Vect\{r^0, Ar^0, ..., A^k r^0\} \tag{3.13}$$

Où $Vect\{r^0, Ar^0, ..., A^k r^0\}$ est le k-ème espace de Krylov généré par A à partir de r^0 noté $\mathscr{K}_k(A, r^0)$

De nouvelles propriétés

Maintenant que nous avons une définition de ces espaces, nous allons montrer plusieurs propriétés pour ensuite construire l'algorithme afin de l'implémenter. Le premier fait remarquable des espaces de krylov est que par construction, nous avons :

$$\mathscr{K}_{k-1}(A, r^0) \in \mathscr{K}_k(A, r^0) \tag{3.14}$$

Ensuite, il est important de montrer que la solution que l'on cherche appartient à cet espace. Nous reprenons donc notre système linéaire possédant une unique solution :

$$Ax = b \tag{3.15}$$

D'après la définition du problème, la matrice A est inversible. Nous supposons que l'on a le polynôme caractéristique de A :

$$P(\lambda) = \sum_{j=0}^{n} \alpha_j t^j \Rightarrow P(0) = \alpha_0 = det(A) \neq 0 \tag{3.16}$$

Par le théorème de Cayley-Hamilton, nous pouvons obtenir la valeur de A^{-1} :

$$\begin{aligned} P(A) = \alpha_0 I + \alpha_1 A + ... + \alpha_n A^n &= 0 && (3.17)\\ \alpha_0 A^{-1} A + \alpha_1 A + ... + \alpha_n A^n &= 0 && (3.18)\\ (\alpha_0 A^{-1} + \alpha_1 + ... + \alpha_n A^{n-1}) A &= 0 && (3.19)\\ \alpha_0 A^{-1} + \alpha_1 + ... + \alpha_n A^{n-1} &= 0 && (3.20) \end{aligned}$$

Ce qui donne finalement :

$$A^{-1} = -\frac{1}{\alpha_0} \times \sum_{j=0}^{n-1} \alpha_{j+1} A^j \tag{3.21}$$

Or la solution du problème 3.15 est : $x^* = A^{-1}b$. Ce qui peut s'écrire de la façon suivante :

$$x^* = -\frac{1}{\alpha_0} \sum_{j=0}^{n-1} \alpha_{j+1} A^j b \tag{3.22}$$

Ce vecteur appartient clairement à l'espace de Krylov défini par :

$$x^* \in \mathscr{K}(A, b) \tag{3.23}$$

Ainsi, la solution de notre problème appartient à l'espace de krylov défini par les deux données du problème que sont A et b.

3.2.2 L'algorithme GMRES : Generalized Minimal RESidual method

Un des algorithmes utilisant les espaces de Krylov est l'algorithme GMRES qui se trouve ci-dessous en pseudo-code (cf. figure 3.1) avec l'implémentation de celui-ci en python. Cette méthode repose sur les espaces de krylov. En effet, dans cet algorithme nous choisissons $x_k \in \mathscr{K}_k(A, b)$ tel que l'on a $||b - Ax_k||_2$ qui est minimal. Nous nous reposons sur une méthode de projection. L'algorithme utilise aussi sur le procédé d'Arnoldi qui nous permet de créer une base orthonormée sur le sous-espace de krylov. Dans la suite de ce chapitre, nous allons étudier ces composantes permettant d'arriver à l'écriture de notre algorithme.

Généralité

Nous avons le système linéaire suivant qui possède une unique solution x^* :

$$Ax = b \tag{3.24}$$

Le résidu initial est :

$$r_0 = b - Ax^{(0)} \tag{3.25}$$

L'algorithme GMRES repose sur une méthode de projection. On écrit donc le système sous sa forme résiduelle pour faire apparaître une projection :

$$< b - Ax^*, v >= 0, \forall v \in \mathbb{R}^n \Leftrightarrow b - Ax^* \perp \mathbb{R}^n \tag{3.26}$$

Le but est maintenant de construire des solutions par itérations. On se place dans un sous-espace $\mathbf{K}$ et on essaie de trouver dans ce sous-espace x tel que :

$$b - Ax \perp \mathbf{K} \Rightarrow < Au, v >=< b, v >, v \in \mathbb{K} \tag{3.27}$$

On généralise cette méthode grâce à deux sous-espaces : $\mathbf{K}$ et $\mathbb{L}$. On cherche $x \in u_0 + \mathbf{K}$ avec :

$$b - Ax \perp \mathbb{L} \Rightarrow < Au, v >=< b, v >, v \in \mathbb{L} \tag{3.28}$$

C'est la condition de Petrov-Galerkin.

Sous la forme matricielle, nous allons créer deux matrices V et W, de taille $n \times m$, définies par les vecteurs des bases de $\mathbf{K}$ et $\mathbb{L}$:

$$V = (V_1\ V_2\ ...\ V_m) \tag{3.29}$$

$$W = (W_1\ W_2\ ...\ W_m) \tag{3.30}$$

Soit $(y, z) \in (\mathbb{R}^m)^2$. Nous pouvons définir $v \in \mathbf{K}$ et $w \in \mathbb{L}$:

$$v = Vy \tag{3.31}$$
$$w = Wz \tag{3.32}$$

Il nous faut maintenant trouver la solution de $x = x_0 + Vy \in \mathbf{K}$. Alors, grâce à 3.26 et 3.28, nous écrivons :

$$< Av, w > = < r0, w > \forall w \in L \tag{3.33}$$
$$< AVy, Wz > = < r0, Wz > \forall z \in \mathbb{R}^m \tag{3.34}$$
$$Avy = r0 \tag{3.35}$$
$$y = (W^tAV)^{-1}W^tr^0 \tag{3.36}$$

Nous obtenons donc la solution :

$$x = x_0 + V(W^tAV)^{-1}W^tr^0 \tag{3.37}$$

Nous pouvons donc maintenant expliqué ce que notre méthode de projection va faire. Il faut savoir qu'une méthode de projection est itérative et va à chaque itération faire deux choses :

— construire les sous-espaces K et L

— chercher la nouvelle itération dans $u + K$ par le procédé exposé ci-dessus.

Voici donc le principe général des méthodes de projection :

1. Choisir $\mathbf{K}^m$ et $\mathbb{L}^m$: deux sous-espaces de $\mathbb{R}^n$ de même dimension m
2. Construire V_m et W_m
3. Calculer le résidu : $r = b - Ax^{(k)}$
4. Résoudre le système $y = (W_m^tAV_m)^{-1}W_m^tr$
5. Créer la nouvelle solution : $x = x + V_my$

Pour créer le sous-espace L, il y a deux méthodes :

— $L = K$ donne lieu à la méthode du gradient conjugué

— $L = AK$ donne lieu à la méthode GMRES.

La méthode du gradient conjugué ayant été vue en cours, nous avons choisi la deuxième méthode de génération de L. Dans les deux cas cela revient à résoudre le problème des moindres carrés :

$$\begin{cases} \text{On cherche } x^(k) \in x_0 + \mathbf{K}_m \text{ telle que :} \\ min_{v \in \mathbf{K}_m} \frac{1}{2}||A(x_0 + v) - b||_2^2 \end{cases} \tag{3.38}$$

La démonstration liant les deux problèmes ne sera pas réaliser ici. [1]

1. Elle se trouve ici : `https://www2.mat.ulaval.ca/fileadmin/Cours/MAT-17992/notes_gmres.pdf`

Explication de l'algorithme GMRES

Cet algorithme se base dans un premier temps sur les itérations d'Arnoldi. Ici nous ne présenterons que la version qui est moins sensible aux erreurs d'arrondi car notre but est de justifier l'implémentation numérique de l'algorithme. Ce procédé d'Arnoldi a pour but de créer une base orthonormée du sous-espace de Krylov. Le procédé se résume par :

1. $v_1 = \frac{r_0}{||r_0||}$
2. Pour j = 1, ..., m ; on calcule : $w = Avj$
3. Pour i = 1, ..., j ; on calcule : $w = w - (w, v_i)v_i$
4. $\bar{v}_{j+1} = \frac{w}{||w||}$

Nous introduisons alors un objet : $\overline{H}_m = (h_{ij})$ la matrice de Heissenberg :

$$h_{ij} = \begin{cases} < Av_j, v_i >, \ i \leq j \\ ||v_{j+1}||, \ i = j+1 \\ 0, \ sinon \end{cases} \tag{3.39}$$

Cet objet permet d'écrire de façon matricielle notre procédé car :

$$AV_m = V_{m+1}\overline{H}_m \tag{3.40}$$

et :

$$\bar{v}_{j+1} = Av_j - \sum_{i=1}^{j} h_{i,j}v_i \tag{3.41}$$

$$\Leftrightarrow Av_j = \bar{v}_{j+1} + \sum_{i=1}^{j} h_{i,j}v_i \tag{3.42}$$

$$\Leftrightarrow Av_j = h_{j+1,j}v_{j+1} + \sum_{i=1}^{j} h_{i,j}v_i \tag{3.43}$$

Par ce procédé, nous arrivons à construire une base orthonormée de l'espace de Krylov.

Maintenant que nous avons cette base, nous pouvons résoudre le problème des moindres carrés exprimés ci-dessus (cf. équation 3.38). Nous obtenons les propriétés suivantes (qui ne seront pas démontrées) :

— $||r_{m+1}|| \leq ||r_m||$

— L'algorithme converge en n itérations au maximum car $\mathscr{K}_m(A, r0) = \mathbb{R}^n$ [2]

Voici le code et le pseudo-code de l'algorithme :

2. La démonstration se trouve ici : `https://www.math.kth.se/na/SF2524/matber15/gmres.pdf`

Generalized Minimum Residual Method (GMRES)

$x_0 =$ initial guess
$r = b - Ax_0$
$q_1 = r/||r||_2$
for $k = 1, 2, \ldots, m$
  $y = Aq_k$
  for $j = 1, 2, \ldots, k$
    $h_{jk} = q_j^T y$
    $y = y - h_{jk} q_j$
  end
  $h_{k+1,k} = ||y||_2$ (If $h_{k+1,k} = 0$, skip next line and terminate at bottom.)
  $q_{k+1} = y/h_{k+1,k}$
  Minimize $||Hc_k - [||r||_2\ 0\ 0 \ldots\ 0]^T||_2$ for c_k
  $x_k = Q_k c_k + x_0$
end

FIGURE 3.1 – Algorithme GMRES

```python
def GMRES(A, b, espilon):
        max_iter = A.shape[0] #Number of maxiter
        mat_q = np.zeros((max_iter, max_iter + 1))
        mat_h = np.zeros((max_iter + 1, max_iter))
        norm_b = np.linalg.norm(b)
        be1 = np.zeros(max_iter + 1)
        be1[0] = norm_b
        mat_q[:, 0] = 1 / np.linalg.norm(b) * b.T # On définit ici que
        ↪ l'on a forcément x0 = 0 (r0 = b)
        for j in range(max_iter):
                mat_q[:, j+1] = A @ mat_q[:, j]
                for i in range(j+1):
                        mat_h[i, j] = mat_q[:, i] @ mat_q[:, j + 1]
                        mat_q[:, j+1] -= mat_h[i, j] * mat_q[:, i]
        mat_h[j + 1, j] = np.linalg.norm(mat_q[:, j + 1])
        mat_q[:, j + 1] /= mat_h[j + 1, j]
        y = np.linalg.lstsq(mat_h, be1, rcond=None)[0]
        residue = np.linalg.norm(y) / norm_b
        if residue < espilon:
        return mat_q[:max_iter, :max_iter] @ y, residue
        return mat_q[:max_iter, :max_iter] @ y
```

Résultat de l'algorithmes

Après implémentation de l'algorithme sur python, nous obtenons des résultats assez satisfaisant. Nous avons voulu tester la robustesse de l'algorithme que nous avons implémenté. Pour cela nous avons généré plusieurs systèmes à résoudre. Afin de compliquer la tâche de l'algorithme, nous générons des problèmes, dont la taille de la matrice dépasse les 1000 lignes, sans aucun conditionnement. Pour ce faire nous générons aléatoirement les matrices en prenant des nombres entre 100 et 1000. Ensuite nous calculons l'erreur sur ce résultat par :

$$\epsilon = |Ax^* - b| \ x^* \text{ étant la solution donnée par l'algorithme} \tag{3.44}$$

Voici le graphique que nous obtenons :

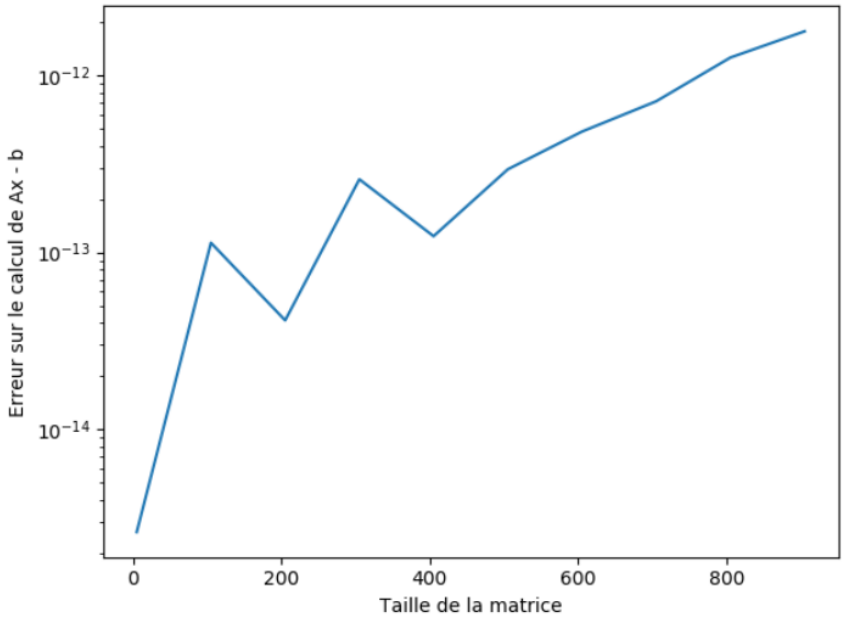

FIGURE 3.2 – Erreur sur la résolution des systèmes avec l'algorithme GMRES

Nous pouvons conclure de celui-ci que l'algorithme est robuste. En effet, l'erreur dans le calcul fait ci-dessus est négligeable quelque soit la taille de la matrice.

Chapitre 4

Des algorithmes complexes...

4.1 Optimisation des méthodes

4.1.1 Optimisation mathématique : Préconditionnement

On peut illustrer les problèmes des méthodes itératives à travers de la méthode du gradient conjugué. En effet, les conditionnements des matrices étudiées sont parfois très grands et il est ainsi nécessaire d'effectuer une opération dite de préconditionnement afin de minimiser les résidus des itérations.

Afin de préconditionner une matrice, on introduit une matrice de préconditionnement que l'on note C. Cette matrice interviendra dans l'algorithme du gradient conjugué afin de l'optimiser. La matrice C diffèrera selon la méthode de préconditionnement choisie. Dans ce mémoire, nous nous focaliserons sur la méthode SSOR.

L'objectif mathématique de l'introduction de la matrice C est de mieux répartir les valeurs propres du système linéaire étudié, ce qui va permettre d'accélérer la méthode du gradient conjugué.

En effet, une meilleure répartition des valeurs propres de la matrice liée au système étudié permet de faire rapprocher les lignes de niveau de la fonction associée au système vers des cercles. Cela permettra ensuite d'avoir moins d'itérations pour la méthode de descente du gradient conjugué.

4.1.2 Optimisation numérique

4.2 Étude de la complexité

Chapitre 5

Conclusion & ouverture

Énormément de problèmes peuvent aujourd'hui se caractériser où s'approximer par un système linéaire qu'il faut résoudre. C'est dans cette optique que nous avons étudié de nouveaux moyens de résoudre, à travers les méthodes itératives, ces systèmes. En partant de la base de ces méthodes, et en complexifiant à chaque étape le procédé de résolution nous parvenu à améliorer nos algorithme.

La première étape fut de partir du simple algorithme de Jacobi et de Gauss-Seidel. En introduisant une décomposition particulière sous la forme M - N, nous avons pu paramétrer la décomposition par un nombre qui nous a permis de redécouvrir la méthode de Richardson qui permet de choisir pour chaque matrice un nombre α optimal qui accroître la vitesse de convergence. Cette nouvelle méthode sera ensuite améliorer par le procédé de relaxation et l'algorithme SOR qui donne d'encore meilleurs résultats. L'amélioration de ce dernier, par la méthode SSOR a aussi été présentée.

Cependant, cela n'était pas suffisant car un problème n'avait pas encore de solution. En effet, il nous fallait s'assurer que nos algorithmes soient robustes mais surtout que l'on puisse assurer la convergence en un nombre fini d'itération relativement petit et connu à l'avance si possible. C'est pour cela que l'on a introduit les espaces de Krylov. Ils nous ont permis d'obtenir un cadre théorique et des méthodes mettant à l'amande tous les autres procédés itératifs par sa précision, sa rapidité, sa robustesse et son application à n'importe quel cas. Ce cadre théorique a été ensuite implémenté par l'algorithme GMRES que nous avons codé.

La dernière partie de notre travail a porté sur l'optimisation numérique de ces méthodes afin de faire converger les algorithmes sur un ordinateur. Notre travail était principalement de faire quelques tests de benchmark sur les différents procédé.

Néanmoins, le travail effectué dans ce rapport n'est qu'un premier pas. En effet, nous sommes capable de résoudre très convenablement tous les problèmes linéaires qui peuvent

se présenter à nous. Cependant, il ne faut pas oublier qu'en générale les problèmes ne sont pas linéaire. Ainsi il faut maintenant trouver un moyen de linéariser les problèmes. De plus, nos méthodes s'appliquent lorsque le système linéaire possède une unique solution mais certains systèmes n'ont pas de solutions ou une infinité. Il faudra donc construire de nouvelles méthodes pour ces systèmes.

Liste des sigles et acronymes

Table des figures

Liste des tableaux