USTL

LABORATOIRE D'ANALYSE NUMERIQUE ET D'OPTIMISATION

n° d'ordre : 1876

THESE

Nouveau régime

présentée à
l'Université des Sciences et Technologies de Lille

pour obtenir le titre de

DOCTEUR en MATHEMATIQUES
par

Mohammed HEYOUNI

METHODE DE HESSENBERG GENERALISEE ET APPLICATIONS

soutenue le 19 décembre 1996 devant la commission d'examen

Membres du jury

Président :	C. BREZINSKI, Professeur, Université de Lille 1
Rapporteurs :	M. REDIVO-ZAGLIA, Professeur, Université de Padoue, Italie H. A. VAN DER VORST, Professeur,Université d'Utrecht, Hollande
Membres :	B. GERMAIN-BONNE, Professeur, Université de Lille 1 H. SADOK, Professeur, Université du Littoral, Calais et Directeur de Thèse

Laboratoire d'Analyse Numérique et d'Optimisation, UFR IEEA - M3, USTL ,
59655 Villeneuve d'Ascq - Cedex - FRANCE.

Cette thèse a été effectuée au sein du laboratoire d'Analyse Numérique et d'Optimisation à l'Université des Sciences et Technologies de Lille, sous la direction du professeur Hassane SADOK de l'université du Littoral. Qu'il trouve ici l'expression de ma reconnaissance, et de mes vifs remerciements, pour avoir accepté de diriger ma recherche, pour ses conseils, pour ses idées et surtout pour le temps qu'il a bien voulu me consacrer ainsi que pour son incessant soutien.

Mes remerciements vont aussi à Claude BREZINSKI, Professeur à l'Université des Sciences et Technologies de Lille et directeur du laboratoire d'Analyse Numérique et d'Optimisation, qui m'a fait l'honneur de présider le jury.

Je suis sincèrement flatté et honoré de la présence au jury du Professeur Michela REDIVO-ZAGLIA de l'Univeristé de Padoue (Italie), ainsi que du Professeur Henk A. Van DER VORST de l'Université d'Utrecht (Hollande). Je tiens à les remercier vivement d'avoir bien voulu être rapporteurs de ce travail.

Monsieur Bernard GERMAIN-BONNE, Professeur à l'Universitè des Sciences et Technologies de Lille, a accepté d'examiner ce travail, je lui en suis reconnaissant. Je tiens à le remercier d'avoir lu si soigneusement le manuscrit.

Je remercie également mes collègues ainsi que les membres du Laboratoire d'Analyse Numérique et d'Optimisation de Lille pour l'aide apporté à ce travail. Je remercie particulieurement A. MESSAOUDI, A. ABKOWICZ, E. H. AYACHOUR ainsi que madame Françoise Tailly et le souriant Azeddine ESSAI, qui j'espère gardera toujours le sourire.

Je tiens aussi à remercier les membres du Laboratoire de Mathématiques Appliquées de l'Université du Littoral qui m'ont permis d'effectuer une partie des tests numériques présentés dans ce travail. Que K. JBILOU, M. PREVOST, et A. SALAM soient assurés de ma reconnaissance.

Un grand merci a mes amis Abderrahim Yachou, Abdelkader Bensaid, Charaf Elhami, Fouad Driouch, Jamal El-Khattabi, Mimoun Mezroui, Yasser Karzazi, Yassine Ruichek, ... pour leur aide à la réalisation de l'avant et l'après thèse.

Que mes amis Abdou, Abdellah, Bouchra, Fatima, Jean Bernard, M'hammed, Mostafa[2], Noureddine, Omar, Rachid, Said, Zavier, Zineb, ..., soient assurés de ma plus vive reconnaissance pour leur présence, pour leur soutien et pour les matches qu'on a fait ensemble !

Enfin, je voudrais remercier les membres de la famille Bekkoucha qui pour une certaine part ont contribué à mon arrivée à Lille. Je remercie également les membres de la famille Soussi pour les conseils et pour l'aide apportée à la réalisation de l'après thèse.

Finalement, je voudrais remercier tous les membres de ma famille, ma grande mère, mon père, ma mère, ma tante, ainsi que mes frères et soeurs qui n'ont pas cessé de m'encourager materiellement et moralement et qui ont certainement contribué à leur façon à mon succès. Pour cela, je voudrais aussi leur dédier ce travail. Je remercie également mon oncle Ahmed ainsi que sa femme Claire pour les journées passées chez eux, pour leur encouragement ainsi que pour leur aide et conseils.

Mes remerciements vont aussi à mes voisins Oujdis (qui sont en fait Feguiguis et Berkanis), particulierement à Hassan, Aziz et Hafid. Que mes amis : Basso, Mansouri, Bekhti soient assurés de ma considération et de mon estime.

Résumé

Cette thèse contient une généralisation de certaines méthodes de sous espaces de Krylov pour la résolution des systèmes linéaires de grande taille. Cette généralisation est basée sur l'utilisation du processus de Hessenberg Généralisé, et est appelée méthode de Hessenberg Généralisée. Elle se subdivise en deux importantes classes de méthodes.

La première classe est celle des méthodes de Galerkin, elle contient comme cas particuliers les méthodes FOM, BCG, Hessenberg, ... La seconde classe est celle des méthodes de semi-minimisation du résidu, elle contient comme cas particuliers les méthodes GMRES, QMR, CMRH, ... Une partie de cette thèse est consacrée à la comparaison numérique de ces trois dernières méthodes.

Un résultat important établi dans cette thèse concerne le lien existant entre les méthodes de Galerkin et les méthodes de semi-minimisation du résidu. Ce lien n'est autre qu'une procédure de "lissage variable". Différentes relations entre les itérés des deux classes de méthodes étudiées permettent d'expliquer la corrélation existante entre elles.

Le dernier aspect de ce travail concerne la résolution des systèmes non linéaires par la méthode de Newton Hessenberg Généralisée. Une comparaison entre les méthodes Newton-GMRES et Newton-CMRH illustre l'application de la méthode Hessenberg Généralisée aux systèmes non linéaires de grande taille.

Mots clefs

Méthodes de sous espace de Krylov, systèmes linéaires, processus de Hessenberg Généralisé, méthodes de Galerkin, méthodes de semi-minimisation du résidu, lissage variable, systèmes non linéaires, méthode de Newton.

Abstract

In this thesis, a generalization of some Krylov subspace methods for solving large systems of linear equations is given. This generalization is based on the use of the Generalized Hessenberg process and is called the Generalized Hessenberg method. This generalization contains two important classes of methods.

The first class is the so called Galerkin methods, it contains as particular cases FOM, BCG, Hessenberg, ..., . The second class is the Minimizing Residual Seminorm methods. GMRES, QMR, CMRH, ... are particular cases of this class. A numerical comparison of the last three methods is performed.

An important result given in this thesis concerns the relationship between the Galerkin and the Minimal Residual Seminorm methods. In fact, we show that the Minimal Residual seminorm methods are obtained from the Galerkin methods by a variable smoothing procedure. Different relations between the iterates of the methods allows us to explain the correlation between the two considered classes of methods.

Another aspect of this work concerns the problem of solving nonlinear systems of equations. This is done by the Newton-Generalized Hessenberg method. A comparison between the Newton-GMRES and Newton-CMRH methods illustrates the use of the Generalized Hessenberg method for solving nonlinear system of equations.

Key words

Krylov subspace methods, linear systems, Generalized Hessenberg process, Galerkin methods, Minimizing Residual Seminorm methods, variable smoothing, nonlinear systems, Newton method.

Table des matières

Introduction générale **1**

1 Le processus de Hessenberg Généralisé **5**

1.1 Description générale 6

1.2 Choix particuliers 11

1.2.1 Processus d'Arnoldi 11

1.2.2 Processus de Hessenberg 15

1.2.3 Processus de Lanczos 19

1.3 Processus de Hessenberg Généralisé Incomplet 24

1.3.1 Description 24

1.3.2 Cas particuliers 26

2 La méthode de Hessenberg Généralisée pour les systèmes linéaires **31**

2.1 La méthode de Galerkin 33

2.1.1 Dérivation de l'algorithme 33

2.1.2 Cas particuliers 37

2.2 La méthode MRSe 38
2.2.1 Dérivation de l'algorithme 38
2.2.2 Décomposition $\mathcal{QR}$ 41
2.2.3 Critère d'arrêt 44
2.3 Implémentation pratique de la méthode MRSe 46
2.3.1 La méthode RMRSe(m) 47
2.3.2 La méthode TMRSe(l) 48
2.4 Cas particuliers 49
2.4.1 GMRES 49
2.4.2 QMR 52
2.4.3 CMRH 54
2.4.4 DQGMRES(l) 56
2.4.5 DTCMRH(l) 57
2.5 Comparaison des méthodes MRSe 58
2.5.1 Algorithmes "complets" 59
2.5.2 Algorithmes redémarrés 67
2.5.3 Algorithmes tronqués 75
2.5.4 Analyse et discussion des résultats 79

3 Sur une procédure de "smoothing" variable **83**

3.1 La procédure de "smoothing" variable 85
3.1.1 Définition 85

3.1.2 Propriétés de la procédure de "smoothing" variable 86

3.2 Liens entre les méthodes GAL et MRSe 89

3.3 Quelques propriétés de la Z_k-semi-norme 94

3.4 Conclusion 101

3.4.1 Le cas Arnoldi/GMRES 101

3.4.2 Le cas BCG/QMR 101

3.4.3 Le cas Hessenberg/CMRH 102

3.4.4 Autres cas 102

4 La méthode de Newton Hessenberg Généralisée 103

4.1 Introduction 103

4.2 La méthode de Newton Hessenberg Généralisée 106

4.2.1 Version différence finie de la méthode de Hessenberg Généralisée 107

4.2.2 Analyse de la propagation de l'erreur dans la méthode de Hessenberg Généralisée 109

4.2.3 Propriétés de la méthode version différence finies de la méthode de Hessenberg Généralisée 113

4.2.4 Convergence locale de la version différence finie de la méthode NHG 115

4.3 Résultats numériques 121

4.3.1 La version différence finie de la méthode NG 122

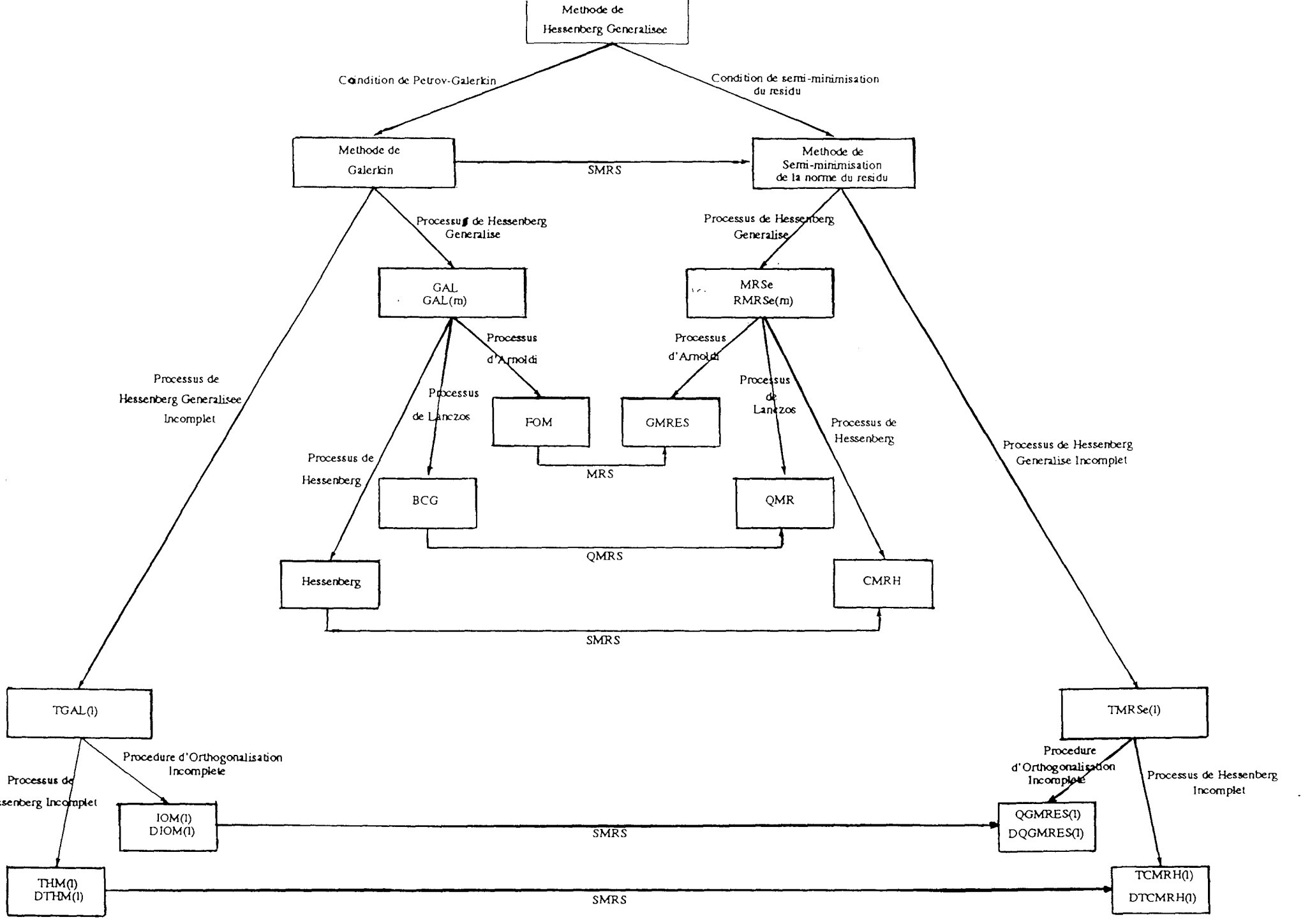
4.3.2 La version différence finie de la méthode NC 123

4.3.3 Implémentation des algorithmes et choix des paramètres 124

4.3.4 Description des exemples et résultats 125

4.3.5 Analyse et discussion des résultats 140

Références **141**


Methode de
Hessenberg Generalisee
Condition de Petrov-Galerkin
Condition de semi-minimisation
du residu
Methode de
Galerkin
SMRS
Methode de
Semi-minimisation
de la norme du residu
Processus de Hessenberg
Generalise
Processus de Hessenberg
Generalise
GAL
GAL(m)
MRSe
RMRSe(m)
Processus
d'Arnoldi
Processus
d'Arnoldi
Processus de
Hessenberg Generalisee
Incomplet
Processus
de Lanczos
Processus
de Lanczos
FOM
GMRES
Processus de
Hessenberg
Processus de
Hessenberg
MRS
BCG
QMR
QMRS
Hessenberg
CMRH
SMRS
Processus de Hessenberg
Generalise Incomplet
TGAL(l)
TMRSe(l)
Procedure d'Orthogonalisation
Incomplete
Procedure
d'Orthogonalisation
Incomplete
Processus de
Hessenberg Incomplet
Processus de Hessenberg
Incomplet
IOM(l)
DIOM(l)
SMRS
QGMRES(l)
DQGMRES(l)
THM(l)
DTHM(l)
SMRS
TCMRH(l)
DTCMRH(l)

Introduction générale

Parmi les problèmes fondamentaux en analyse numérique, la résolution des systèmes d'équations linéaires et non linéaires constitue une part importante sur laquelle beaucoup de travaux et d'articles ont été publiés ces dernières années.

Les systèmes d'équations linéaires surviennent fréquemment dans le calcul scientifique. Ils interviennent lorsqu'on discrétise par la méthode des différences finies ou par la méthode des éléments finis un problème d'équations aux dérivées partielles. On peut aussi les rencontrer comme étapes intermédiaires dans le calcul d'une solution d'un problème non linéaire.

Soit à résoudre le système $Ax = f$, où A est une matrice réelle inversible de taille n. Une première approche pour la résolution de ce système est l'utilisation des méthodes directes. Il s'agit dans ces méthodes de factoriser A en produit de facteurs (en général ces facteurs sont plus faciles et plus simples à manipuler que la matrice A) que l'on utilise pour le calcul de $x_* = A^{-1}f$. Les factorisations les plus souvent utilisées dans la pratique sont la factorisation $\mathcal{LU}$ ou la factorisation $\mathcal{QR}$. Ces méthodes ont largement été utilisées, cependant elles nécessitent un stockage de l'ordre de $\mathcal{O}(n^2)$ et un coût algorithmique de l'ordre de $\mathcal{O}(n^3)$. On voit alors qu'on ne peut les appliquer à des problèmes de grande taille même si les matrices associées à ces problèmes sont souvent creuses. Il en est ainsi dans la discrétisation des équations aux dérivées partielles .

Une alternative aux méthodes directes est donnée par les méthodes itératives . L'idée générale et commune à ces méthodes est de construire par un algorithme donné une suite $\{x_k\}_{k=1}^n$ de solutions approchées de la solution exacte $x_* = A^{-1}f$ du système linéaire à résoudre. Les itérés $\{x_k\}_{k=1}^n$ que nous esperons voir converger vers x_* sont solutions de systèmes dont la taille est plus petite que celle du système initial. Parmi ces méthodes, citons la méthode du Gradient Conjugué (CG : Conjugate Gradient method) due à Hestenes et Stiefel [29]. Cette méthode est extrêmement efficace lorsque la matrice du système est symétrique définie positive, puisqu'elle ne requiert que le stockage d'un nombre limité de vecteurs.

Pour les systèmes symétriques indéfinis, les méthodes MINRES (MINimum RESidual method) et SYMMLQ (SYMMetric LQ) dues à Paige et Saunders [36] sont une alternative à la méthode du Gradient Conjugué. Lorsque les matrices ne sont pas symétriques, nous pouvons appliquer la méthode du Gradient Conjugué à l'une des deux formes du système d'équations normales associé au système initial $Ax = f$. Nous obtenons alors les deux méthodes CGNE (Conjugate Gradient on Normal equations to minimize the Error) due à Craig [14] qui résout le système $AA^Ty = f$ et détermine la solution $x = A^Ty$, ainsi que CGNR (Conjugate Gradient on Normal equations to minimize the Residual) due à Hestnes et Stiefel [29] qui résout le système $A^TAx = \tilde{f}$ avec $\tilde{f} = A^Tf$. Notons cependant que la convergence de ces deux méthodes peut être très lente car le spectre des matrices associées aux équations normales est moins favorable que celui de A.

La méthode du Gradient Bi-Conjugué (BCG: BiConjugate Gradient method) due à Lanczos [33] et à Fletcher [19] génère par utilisation de l'algorithme du Gradient Conjugué deux familles de vecteurs biorthogonales. La première famille est basée sur un système de matrice associée A, alors que la seconde est basée sur un système dont la matrice est A^T. Comme la méthode du Gradient Conjugué, l'algorithme BCG utilise un stockage limité de vecteurs, cependant la convergence peut être irrégulière et l'algorithme peut être sujet à des "breakdown" (i.e. division par un nombre nul).

Plus récemment, Saad a utilisé le processus d'Arnoldi [2, 39, 40] et une condition d'orthogonalité (ou de Petrov-Galerkin) imposée sur le résidu afin de définir la méthode FOM (Full Orthogonalization Method) dite encore la méthode d'Arnoldi [40]. La méthode GMRES (Generalized Minimal RESidual method) due à Saad et Schultz [42] combine le processus d'Arnoldi et une condition de "minimisation" du résidu afin de remédier au problème du "breakdown" qui peut survenir dans la méthode FOM. La méthode QMR (Quasi Minimal Residual method) (due à Freund dans le cas symétrique [20] et à Freund et Nachtigal dans le cas non symétrique [21]) utilise le processus de Lanczos [32, 61] et une condition de "quasi-minimisation" du résidu afin d'éviter un possible "breakdown" dans l'algorithme BCG (cette méthode étant elle-même basée sur le processus de Lanczos et sur la condition de Petrov-Galerkin). En utilisant la même technique de "quasi-minimisation" du résidu ainsi que le processus de Hessenberg [61], Sadok dans [46] a proposé la méthode CMRH (Changing Minimal Residual method based on the Hessenberg process). En remplaçant dans la méthode GMRES le processus d'Arnoldi par la procédure d'Orthogonalisation Incomplète [41, 39], Wu et Saad ont défini la méthode DQGMRES [43], afin de diminuer le coût algorithmique nécessité par la méthode GMRES.

Le thème essentiel de ce travail est de présenter sous une forme générale les différentes méthodes que sont les méthodes FOM, BCG, GMRES, QMR, CMRH et DQGMRES. En combinant le processus de Hessenberg Généralisé [61] (dont les processus: Arnoldi,

Lanczos et Hessenberg sont des cas particuliers) avec une certaine condition d'orthogonalité sur les résidus, nous définissons la méthode de Hessenberg Généralisée.

La méthode de Hessenberg Généralisée se subdivise en deux classes de méthodes. La première classe est celle des méthodes de Galerkin, elle est obtenue par imposition de la condition de Petrov-Galerkin, elle contient comme cas particuliers les méthodes : FOM, DIOM [41], BCG et Hessenberg. La seconde classe de méthodes est obtenue par imposition d'une condition de semi-minimisation du résidu. Cette classe de méthodes est désignée par MRSe (Minimal Residual Seminorm method) [27]. Elle contient comme cas particuliers les méthodes GMRES, DQGMRES, QMR, CMRH et DTCMRH.

Les deux classes de méthodes de Galerkin et MRSe sont étroitement liées entre elles puisque nous montrons que la méthode MRSe peut être déduite par un "smoothing variable" de la méthode de Galerkin [27]. Ce résultat généralise les résultats de Brown et Weiss qui stipulent que les itérés des méthodes GMRES et QMR peuvent être construits respectivement à partir des itérés des méthodes FOM et BCG [8], [56, 57].

La méthode de Hessenberg Généralisée sera ensuite utilisée dans le cadre de la méthode de Newton dans le but de résoudre des systèmes d'équations non linéaires.

Détaillons maintenant le plan de la thèse.

Dans le **chapitre 1**, nous rappelons le processus de Hessenberg Généralisé et donnons quelques propriétés de ce processus. Nous rappelons aussi comment retrouver les différents cas particuliers que sont les processus d'Arnoldi, de Lanczos et de Hessenberg. Ensuite, nous étudions quelques propriétés du processus de Hessenberg Généralisé Incomplet qui admet comme cas particuliers la procédure d'Orthogonalisation Incomplète ainsi que le processus de Hessenberg Incomplet.

Au **chapitre 2**, après avoir décrit les conditions de Petrov-Gelerkin et de semi-minimisation du résidu, nous serons en mesure d'utiliser le processus de Hessenberg Généralisé afin de définir la méthode de Hessenberg Généralisée ainsi que les deux classes de méthodes qui sont les méthodes de Galerkin (GAL) et la méthode MRSe. Comme cas particulier nous retrouvons les méthodes FOM, DIOM, BCG, Hessenberg (de la classe des méthodes de Galerkin), ainsi que les méthodes GMRES, DQGMRES, QMR, CMRH et DTCMRH (de la classe des méthodes MRSe). Nous rappelons aussi quelques techniques utilisées dans l'implémentation pratique des méthodes citées précédemment. Nous terminons ce chapitre par une comparaison numérique des différentes méthodes.

Dans le **chapitre 3** nous définissons une procédure de "smoothing variable" et nous en donnons quelques propriétés. Nous établissons ensuite une relation entre les itérés de la méthode MRSe et ceux de la méthode de Galerkin. Ceci nous permet de montrer

que la méthode MRSe peut être déduite à partir de la méthode de Galerkin en utilisant l'algorithme SMRS (Semi Minimal Residual Smoothing algorithm). Nous retrouvons ainsi quelques résultats établis par Zhou et Walker [62] pour les méthodes QMR et BCG, ou par Weiss [56, 57] pour les méthodes GMRES et FOM. Le cas des méthodes DQGMRES et DIOM ainsi que des méthodes DTCMRH et Hessenberg est aussi traité.

Finalement, dans le **chapitre 4**, nous verrons comment combiner la version différence finie de la méthode de Hessenberg Généralisée et la méthode de Newton Inexacte afin de résoudre des systèmes d'équations non linéaires. La méthode ainsi obtenue est appelée version différence finie de la méthode de Newton Hessenberg Généralisée. Nous étudierons la convergence locale de cette méthode. Nous terminons ce chapitre en donnant les algorithmes des versions différences finie des méthodes Newton-GMRES et Newton-CMRH et en comparant numériquement l'efficacité de ces deux méthodes.

Chapitre 1

Le processus de Hessenberg Généralisé

De nombreux problèmes de résolution de systèmes d'équations linéaires peuvent être traités efficacement en utilisant des méthodes itératives. Une classe importante de ces techniques est connue sous le nom de méthodes de sous espace de Krylov. Il s'agit dans ces méthodes de construire une suite de solutions approchées de la solution exacte du système en partant d'une approximation initiale quelconque.

Parmi ces méthodes les plus récentes, citons les méthodes FOM (Full Orthogonalization Method) [40], BCG (Bi-Conjugate Gradient method) [19, 33] et différents algorithmes qui leur sont mathématiquement équivalents dans des cas particuliers [31, 36, 26, 28]. Il existe d'autres méthodes qui sont en général plus compétitives que les précédentes. Parmi ces méthodes, nous citons GMRES (Generalized Minimal RESidual method) [42], QMR (Quasi Minimal Residual method) [20, 21] et CMRH (Changing Minimal Residual method based on the Hessenberg reduction process) [46].

Toutes les méthodes citées précédemment sont basées sur un processus propre à chaque méthode. Ces processus sont utilisés pour la réduction d'une matrice de Krylov [61, p.32] afin de construire une base de Krylov, ainsi qu'une matrice de Hessenberg supérieure. Les différents processus utilisés sont : Arnoldi, Lanczos et Hessenberg. Ils peuvent tous être obtenus comme cas particuliers d'un processus plus général qui est le processus de Hessenberg Généralisé [30, 61].

Nous commencerons dans ce premier chapitre par donner une description générale du processus de Hessenberg Généralisé au paragraphe 1.1.

Au paragraphe 1.2, nous verrons comment nous pouvons retrouver différents processus connus à partir du processus de Hessenberg Généralisé. En 1.2.1 nous étudierons et donnerons quelques propriétés du processus d'Arnoldi. En 1.2.2 nous nous intéresserons à l'étude du processus de Hessenberg, nous verrons qu'il faut adopter une stratégie de pivotage afin d'augmenter l'efficacité de l'algorithme. Nous terminerons ce second paragraphe par l'étude du processus de Lanczos en 1.2.3.

Afin de réduire le coût algorithmique et la place mémoire nécessités par le processus de Hessenberg Généralisé, nous proposerons au paragraphe 1.3 une version tronquée du processus de Hessenberg Généralisé. Nous verrons entre autres que les principaux résultats obtenus pour les différents processus particuliers Arnoldi et Hessenberg restent valables pour leur version tronquée.

1.1 Description générale

Dans cette section nous allons décrire un algorithme d'une méthode qui initialement a été étudiée pour le calcul explicite du polynôme caractéristique d'une matrice donnée. Cette méthode est due à Hessenberg [61] et à Householder et Bauer [30].

Avant de commencer la description du processus de Hessenberg Généralisé, nous avons besoin de rappeler les définitions suivantes :

Définition 1

Soit $A \in \mathbb{R}^{n\times n}$, $v \in \mathbb{R}^n$. *Le sous espace vectoriel engendré par les vecteurs* $v, Av, \ldots, A^{m-1}v$ *est appelé sous espace de Krylov d'ordre* m *associé à* A *et* v.
On note $K_m(v, A) = span\{v, Av, \ldots, A^{m-1}v\}$.

Définition 2

Le polynôme minimal p *de la matrice* A *pour le vecteur* v *est défini comme étant le polynôme de degré minimum vérifiant* $p(A)v = 0$.

Définition 3

Soit V_m *la matrice* $n \times m$ *formée des colonnes* $v, Av, \ldots, A^{m-1}v$. *La matrice* V_m *est dite matrice de Krylov associée au sous espace de Krylov* $K_m(v, A)$.

Le résultat suivant est immédiat d'après la définition même du sous espace $K_m(v, A)$:

Théorème 1

Le sous espace de Krylov $K_m(v, A)$ est le sous espace formé de tous les vecteurs y qui s'écrivent $y = p(A)v$, où $p \in \mathcal{P}_{m-1}$ (espace des polynômes de degré inférieur ou égal à $m-1$).

Théorème 2

Le sous espace de Krylov $K_m(v, A)$ est de dimension m si et seulement si le degré du polynôme minimal de A pour v est supérieur ou égal à m.

Considérons un vecteur $v \in \mathbb{R}^n$ et supposons que le degré du polynôme minimal de la matrice A pour le vecteur v soit au moins égal à m. Le processus de Hessenberg Généralisé consiste à construire une base $\{b_1, b_2, \ldots, b_m\}$ du sous espace de Krylov $K_m(v, A) = span\{v, Av, \ldots, A^{m-1}v\}$.

Le vecteur initial b_1 est choisi colinéaire à v, i.e. $b_1 = \beta_1 v$, à l'étape k, le vecteur b_{k+1} est calculé par la formule suivante :

$$\beta_{k+1} b_{k+1} = Ab_k - \sum_{i=1}^{k} h_{i,k} b_i. \tag{1.1}$$

Le paramètre β_{k+1} est choisi par des considérations numériques et est communément calculé comme facteur normalisant le vecteur b_{k+1}.

Les paramètres $\{h_{i,k}\}_{i=1}^{k}$ sont déterminés en imposant une condition d'orthogonalité sur le vecteur b_{k+1}. Pour cela, nous considérons $\{y_1, y_2, \ldots, y_m\}$ une famille de m vecteurs linéairement indépendants de $\mathbb{R}^n$, et nous imposons la condition d'orthogonalité suivante :

$$b_{k+1} \perp y_1, \ldots, y_k. \tag{1.2}$$

Notons $B_k = (b_1, b_2, \ldots, b_k)$ (respectivement $Y_k = (y_1, y_2, \ldots, y_k)$) la matrice de $\mathbb{R}^{n \times k}$ formée des colonnes $b_1, b_2, \ldots, b_k$ (respectivement $y_1, y_2, \ldots, y_k$). De même désignons par $e_k^{(n)} = (0, \ldots, 0, 1, 0, \ldots, 0)^T$ le $k^{\text{ème}}$ vecteur de la base canonique de $\mathbb{R}^n$; alors les deux relations précédentes peuvent être écrites sous la forme matricielle suivante :

- La relation (1.1) s'écrit :

$$AB_k = B_{k+1}\widetilde{H}_k, \tag{1.3}$$

ou encore

$$AB_k = B_k H_k + \beta_{k+1} b_{k+1} e_k^{(k)^T}. \tag{1.4}$$

La matrice $H_k \in \mathbb{R}^{k\times k}$ est une matrice de Hessenberg

$$H_k = \begin{pmatrix} h_{1,1} & h_{1,2} & h_{1,3} & \cdots & \cdots & h_{1,k} \\ \beta_2 & h_{2,2} & h_{2,3} & \cdots & \cdots & h_{2,k} \\ & \beta_3 & h_{3,3} & \cdots & \cdots & h_{3,k} \\ & & \beta_4 & \cdots & \cdots & h_{4,k} \\ & & & \ddots & \ddots & \vdots \\ & & & & \beta_k & h_{k,k} \end{pmatrix}, \tag{1.5}$$

et $\widetilde{H}_k \in \mathbb{R}^{(k+1)\times k}$ est la matrice de Hessenberg supérieure partitionnée sous la forme:

$$\widetilde{H}_k = \begin{pmatrix} H_k \\ (0,\ldots,0,\beta_{k+1}) \end{pmatrix} = \begin{pmatrix} H_k \\ \beta_{k+1} e_k^{(k)T} \end{pmatrix}. \tag{1.6}$$

Avant de développer la relation (1.2), nous avons besoin de la définition suivante où $I_m^{(m)}$ désigne la matrice Identité de $\mathbb{R}^{m\times m}$.

Définition 4

Soit $V_m \in \mathbb{R}^{n\times m}$, s'il existe $U_m \in \mathbb{R}^{n\times m}$ telle que $U_m^T V_m$ est inversible alors on définit V_m^L une matrice inverse généralisée à gauche de V_m par:

$$V_m^L = (U_m^T V_m)^{-1} U_m^T.$$

Cette matrice inverse généralisée vérifie $V_m^L V_m = I_m^{(m)}$.

Il est important de remarquer que cette matrice inverse dépend du choix de la matrice U_m, en particulier si $U_m = V_m$, nous obtenons V_m^+ la matrice pseudo-inverse de V_m définie par $V_m^+ = (V_m^T V_m)^{-1} V_m^T$.

- La condition d'orthogonalité (1.2) s'écrit:

$$Y_k^T B_k = L_k \tag{1.7}$$

où L_k est la matrice triangulaire inférieure de taille k donnée par:

$$L_k = \begin{pmatrix} (y_1,b_1) & 0 & \cdots & 0 \\ (y_2,b_1) & (y_2,b_2) & \ddots & \vdots \\ \vdots & \vdots & \ddots & 0 \\ (y_k,b_1) & (y_k,b_2) & \cdots & (y_k,b_k) \end{pmatrix}.$$

De la relation (1.7) nous déduisons qu'il existe B_k^L une matrice inverse généralisée à gauche de B_k telle que:

$$B_k^L = L_k^{-1} Y_k^T = (Y_k^T B_k)^{-1} Y_k^T, \tag{1.8}$$

et par les relations (1.3), (1.4) et (1.7) nous avons:

$$H_k = B_k^L A B_k = L_k^{-1} Y_k^T A B_k \tag{1.9}$$

$$\widetilde{H}_k = B_{k+1}^L A B_k = L_{k+1}^{-1} Y_{k+1}^T A B_k. \tag{1.10}$$

Ainsi la matrice $\widetilde{H}_k$ existe si et seulement si la matrice L_{k+1} est inversible.

Nous verrons après avoir décrit le processus que cette dernière égalité nous permettra d'expliquer l'existence d'un possible "breakdown" dans l'algorithme du processus de Hessenberg Généralisé.

Remarque:

Dans la suite, par souci d'homogéneité et de simplicité dans l'écriture des matrices H_k et $\widetilde{H}_k$, le scalaire β_{k+1} utilisé dans la relation (1.1) sera noté par $h_{k+1,k}$.

Dans la pratique, la mise en œuvre de l'algorithme est comme suit: supposons que nous ayons déjà construit $\{b_1, b_2, \ldots, b_k\}$ ainsi que la matrice correspondante $\widetilde{H}_{k-1}$. On veut calculer b_{k+1}, pour cela nous commençons par former le produit $u = Ab_k$, ensuite nous soustrayons à u des multiples de b_j de façon que u soit orthogonal à y_j. Ceci nous permet de déterminer les éléments $h_{j,k}$ pour $j = 1, \ldots, k$.

En effet la relation (1.1) donne:

$$h_{k+1,k} b_{k+1} = Ab_k - \sum_{i=1}^{j-1} h_{i,k} b_i - h_{j,k} b_j - \sum_{i=j+1}^{k} h_{i,k} b_i,$$

en utilisant la condition d'orthogonalité (1.2), nous obtenons:

$$h_{j,k}(y_j, b_j) = (y_j, Ab_k - \sum_{i=1}^{j-1} h_{i,k} b_i) \qquad j = 1, \ldots, k.$$

Ainsi nous avons:

$$h_{j,k} = \frac{1}{(y_j, b_j)} (y_j, Ab_k - \sum_{i=1}^{j-1} h_{i,k} b_i) \qquad j = 1, \ldots, k. \tag{1.11}$$

En ce qui concerne le choix de $h_{k+1,k}$, il suffit de le choisir non nul pour que les divers processus décrits ci dessous soient valables.

En utilisant ces formules, nous obtenons l'algorithme suivant :

Algorithme 1 : *Processus de Hessenberg Généralisé.*

- *Initialisation :* $b_1 = v$; $\eta_1 = (y_1, b_1)$;

- *Itération* : *pour* $k = 1, \ldots, m$
  $u = Ab_k$;
  pour $j = 1, \ldots, k$
  $h_{j,k} = (y_j, u)/\eta_j$; $u = u - h_{j,k} b_j$;
  fin du pour ;
  choisir $h_{k+1,k} \neq 0$, *quelconque* ;
  $b_{k+1} = u/h_{k+1,k}$; $\eta_{k+1} = (y_{k+1}, b_{k+1})$;
  fin du pour.

Naturellement, nous considérons dans cet algorithme m ($m < n$) comme nombre maximal d'itérations. De plus cet algorithme doit être arrêté dans le cas où le vecteur b_{k+1} est nul ou orthogonal à y_{k+1} (i.e. $\eta_{k+1} = 0$). Par contre si l'algorithme 1 réalise les m itérations, alors nous avons le résultat suivant :

Théorème 3

Supposons que m étapes soient réalisables dans le processus de Hessenberg Généralisé. Alors, les vecteurs $b_1, b_2, \ldots, b_m$ calculés par l'algorithme 1 forment une base du sous espace de Krylov $K_m(v, A)$.

Preuve :

Il suffit de montrer que les vecteurs $b_1, b_2, \ldots, b_m$ engendrent le sous espace vectoriel $\{p(A)v, p \in \mathcal{P}_{m-1}\}$. Le théorème 1 nous permet alors de conclure que ces mêmes vecteurs engendrent $K_m(v, A)$.

Or par récurrence, il est facile de montrer que le vecteur b_k $k = 1, \ldots, m$ peut s'écrire $b_k = p_{k-1}(A)v$ où p_{k-1} est un polynôme de degré inférieur ou égal a $k - 1$.

■

Nous avons supposé dans le théorème précédent que les différents produits scalaires $\eta_k = (y_k, b_k)$ pour $k = 1, \ldots, m$ étaient non nuls. Dans le cas où cette hypothèse n'est plus vérifiée, il peut survenir un "breakdown" à une étape k de l'algorithme si $\eta_{k+1} = (y_{k+1}, b_{k+1}) = 0$. Dans ce cas la matrice L_{k+1} n'est pas inversible.

Or par l'équation (1.10) nous avons $\widetilde{H}_k = B_{k+1}^L A B_k = L_{k+1}^{-1} Y_{k+1}^T A B_k$, et donc nous ne pouvons pas continuer le calcul des éléments de la matrice H_m.

Nous ne pouvons pas éviter ce problème si les vecteurs $\{y_1, y_2, \ldots, y_m\}$ sont supposés fixés dès le début du processus. Dans ce cas, la seule possibilité pour la construction d'une base B_m d'un sous espace de Krylov associé à A est de redémarrer le processus en choisissant b_1 égal à un nouveau vecteur de départ.

Par contre, si la famille des vecteurs $\{y_1, y_2, \ldots, y_m\}$ peut être construite ou choisie au fur et à mesure du déroulement de l'algorithme, nous pouvons remédier à ce problème de "breakdown" en choisissant un nouveau y_j vérifiant $(y_j, b_j) \neq 0$.

Nous traiterons plus en détail les différentes solutions proposées pour remédier à ce problème pour chaque cas particulier du processus de Hessenberg Généralisé.

1.2 Choix particuliers

Pour la réalisation de l'algorithme 1, il faut que nous ayons à notre disposition une famille $y_1, y_2, \ldots, y_m$. La question que nous pouvons nous poser est alors: Pouvons nous choisir une "bonne" famille $\{y_k\}_{k=1}^m$ ou la construire par un quelconque procédé afin qu'elle nous permette de calculer efficacement et/ou à moindre côut la base $\{b_1, b_2, \ldots, b_m\}$ du sous espace de Krylov $K_m(v, A)$?

La réponse à cette question est positive; en effet en adoptant des choix simples et naturels de la suite $\{y_k\}_{k=1}^m$ dans le processus de Hessenberg Généralisé nous retrouvons différents processus connus dont l'efficacité et la mise en œuvre vont être discutés dans cette section.

1.2.1 Processus d'Arnoldi

Comme il a été souligné précédemment, la mise en œuvre du processus de Hessenberg Généralisé nécessite le choix d'une famille $\{y_k\}_{k=1}^m$. Un choix naturel et efficace consiste à prendre la famille $\{y_k\}_{k=1}^m$ égale à la famille $\{b_k\}_{k=1}^m$ elle-même.

Dans ce cas la condition d'orthogonalité (1.7) s'écrit:

$$Y_k^T B_k = B_k^T B_k = L_k.$$

Or L_k est une matrice triangulaire et $L_k = B_k^T B_k$ est une matrice symétrique, donc l'égalité précédente devient :

$$B_k^T B_k = D_k, \tag{1.12}$$

où D_k est la matrice diagonale $diag(b_1^T b_1, \ldots, b_k^T b_k)$. De même par la relation (1.10) nous avons :

$$\widetilde{H}_k = D_{k+1}^{-1} B_{k+1}^T A B_k. \tag{1.13}$$

Le processus de Hessenberg Généralisé se réduit donc au processus d'Arnoldi [2, 8, 40, 42] pour la construction d'une base du sous espace de Krylov $K_m(A, v)$.

Avec le choix $\{y_k\}_{k=1}^m = \{b_k\}_{k=1}^m$, l'algorithme 1 se réduit à l'algorithme donné ci dessous :

Algorithme 2 : *Processus d'Arnoldi.*

- *Initialisation :* $b_1 = v$; $\eta_1 = (b_1, b_1)$;
- *Itération* : *pour* $k = 1, \ldots, m$
  $u = Ab_k$;
  pour $j = 1, \ldots, k$
  $h_{j,k} = (b_j, u)/\eta_j$; $u = u - h_{j,k} b_j$;
  fin du pour ;
  choisir $h_{k+1,k} \neq 0$, *quelconque ;*
  $b_{k+1} = u/h_{k+1,k}$; $\eta_{k+1} = (b_{k+1}, b_{k+1})$;
  fin du pour.

L'algorithme précédent doit être arrêté à l'étape k si $b_{k+1} = 0$, nous avons alors le résultat suivant :

Théorème 4

L'algorithme d'Arnoldi s'arrête à l'étape k si et seulement si le degré du polynôme minimal de A pour b_1 *est égal à k. Dans ce cas le sous espace de Krylov* $K_k(b_1, A)$ *est invariant par A, i.e.* $AK_k(b_1, A) = K_k(b_1, A)$.

Preuve :

Supposons que l'algorithme s'arrête à l'itération k; nous avons alors $b_i \neq 0$ pour $i = 1, \ldots, k$ et $b_{k+1} = 0$. Ainsi le degré μ du polynôme minimal associé à A et b_1 est tel que $\mu \leq k$. Or le cas $\mu < k$ n'est pas possible car sinon $K_k(b_1, A)$ serait de dimension $\mu - 1$ (cf théorème 2) et donc $b_\mu = 0$, d'où le polynôme minimal est de degré k.

Réciproquement, si le degré du polynôme minimal est égal à k alors $b_{k+1} = 0$, sinon $K_{k+1}(b_1, A)$ serait de dimension $k+1$ et donc d'après le théorème 2 nous aurons $\mu > k$.

L'invariance de $K_k(b_1, A)$ découle du fait que le polynôme minimal associé de A pour b_1 est de degré k.

■

Dans le cas où le degré du polynôme minimal associé à A et v est supérieur à m, alors m itérations sont réalisables dans le processus d'Arnoldi. Donc d'après le théorème 3 et puisque la matrice B_k est orthogonale (cf équation (1.12)), il vient que:

Théorème 5

Supposons que m étapes soient réalisables dans le processus d'Arnoldi. Alors les vecteurs $b_1, b_2, \ldots, b_m$ calculés par l'algorithme 2 forment une base orthogonale du sous espace de Krylov $K_m(v, A)$.

Dans le processus d'Arnoldi décrit par l'algorithme précédent, nous n'avons pas encore spécifié comment choisir $h_{k+1,k}$. Nous avons rappelé dans la description générale du processus de Hessenberg Généralisé que $h_{k+1,k}$ est souvent calculé comme facteur normalisant le vecteur b_{k+1}. Il suffit donc de choisir $h_{k+1,k} = \|u\|_2$ dans l'algorithme afin d'avoir $\|b_{k+1}\|_2 = 1$ à chaque étape du processus et construire ainsi une base B_k orthonormale.

L'algorithme 2.1 ci dessous décrit le processus d'Arnoldi avec "orthonormalisation".

Algorithme 2.1 : *Processus d'Arnoldi avec "orthonormalisation".*

- *Initialisation :* $b_1 = v/\|v\|_2$;
- *Itération* : *pour* $k = 1, \ldots, m$
  $u = Ab_k$;
  pour $j = 1, \ldots, k$
  $h_{j,k} = (b_j, u)$; $u = u - h_{j,k} b_j$;
  fin du pour;
  $h_{k+1,k} = \|u\|_2$; $b_{k+1} = u/h_{k+1,k}$;
  fin du pour.

Le processus d'Arnoldi avec "orthonormalisation" ainsi obtenu a été utilisé dans la méthode FOM [8, 40], GMRES [42] pour la résolution de systèmes linéaires, ainsi que dans la méthode d'Arnoldi pour le calcul des valeurs propres [39].

Remarques :

1. *Le processus d'Arnoldi décrit par l'algorithme 2.1 n'est autre que la méthode de Gram-Schmidt modifiée pour la construction de B_m, base orthonormale du sous espace de Krylov $K_m(v, A)$, (cf théorème 6 ci dessous).*

2. *Le processus d'Arnoldi nécessite à la $k^{\text{ème}}$ étape $2nk$ multiplications ainsi qu'un produit matrice vecteur dont le côut est de $n\nu$ multiplications. Pour une matrice dense $\nu = n$, alors que pour une matrice creuse $\nu = N_{nz}/n$ où N_{nz} est le nombre d'éléments non nuls de la matrice. Au total, pour m itérations le processus d'Arnoldi nécessite un coût algorithmique de l'ordre de $nm^2 + mn\nu$ multiplications.*

3. *En ce qui concerne la place mémoire, le processus d'Arnoldi nécessite un stockage des nm éléments de la matrice B_m ainsi que des $(m+3)m/2$ éléments de la matrice $\widetilde{H}_m$.*

Nous terminons ce sous-paragraphe en faisant le lien entre le processus d'Arnoldi et la décomposition $\mathcal{QR}$ de la matrice de Krylov V_m associée à $K_m(v, A)$.

Théorème 6

Soit $V_m = (v, Av \ldots, A^{m-1}v)$ la matrice de Krylov associée à $K_m(v, A)$.
Si le degré du polynôme minimal associé à A et v est supérieur ou égal à m alors la décomposition $\mathcal{Q}_m\mathcal{R}_m$ de V_m existe. De plus, nous avons $\mathcal{Q}_m = B_m$.

Preuve :

Il est clair que si le degré du polynôme minimal de A pour v est supérieur ou égal à m, alors les vecteurs $v, Av \ldots, A^{m-1}v$ sont linéairement indépendants. Il est alors possible de factoriser V_m sous la forme $V_m = \mathcal{Q}_m\mathcal{R}_m$, avec $\mathcal{Q}_m \in \mathbb{R}^{n\times m}$, matrice orthogonale, et $\mathcal{R}_m \in \mathbb{R}^{m\times m}$ matrice triangulaire supérieure à diagonale unité. Il nous reste alors à montrer que $\mathcal{Q}_m = B_m$.

Il n'est pas restrictif de supposer que $h_{k+1,k} = 1$ pour $k = 1 \ldots, m$. Dans ce cas, nous avons $b_k = p_{k-1}(A)v$ où p_{k-1} est un polynôme unitaire de $\mathcal{P}_{k-1}$. Notons alors $\alpha_0^{k-1}, \alpha_1^{k-1}, \ldots, \alpha_{k-1}^{k-1}$ les cœfficients de p_{k-1} et soit $u_k = (\alpha_0^{k-1}, \alpha_1^{k-1}, \ldots, \alpha_{k-2}^{k-1}, 1, 0, \ldots, 0)^T$ un vecteur de $\mathbb{R}^m$. Le polynôme p_{k-1} étant unitaire (i.e. $\alpha_{k-1}^{k-1} = 1$), le vecteur b_k peut être alors exprimé par :

$$b_k = V_m\, u_k,$$

où $V_m = (v, Av, \ldots, A^{k-1}v, \ldots, A^{m-1}v)$ est la matrice de Krylov associée au sous espace de Krylov $K_m(v, A)$. Soit U_m la matrice $m \times m$ suivante :

$$U_m = \begin{pmatrix} 1 & \alpha_0^1 & \cdots & \cdots & \alpha_0^{m-1} \\ 0 & 1 & \ddots & & \alpha_1^{m-1} \\ \vdots & 0 & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & \alpha_{m-2}^{m-1} \\ 0 & \cdots & \cdots & 0 & 1 \end{pmatrix},$$

nous avons alors $B_m = V_m U_m$ Comme U_m est une matrice triangulaire supérieure à diagonale unité, alors $\mathcal{R}_m = U_m^{-1}$ existe, nous avons alors :

$$V_m = B_m \mathcal{R}_m.$$

Par unicité de la décomposition $\mathcal{QR}$, nous obtenons $\mathcal{Q}_m = B_m$.

■

1.2.2 Processus de Hessenberg

Un autre choix évident dans le processus de Hessenberg Généralisé consiste à prendre $\{y_k\}_{k=1}^m$ égal à la famille $\{e_k^{(n)}\}_{k=1}^m$ où $e_k^{(n)}$ est le $k^{\text{ème}}$ vecteur de la base canonique de $\mathbb{R}^n$. Ce choix particulier nous permet de retrouver le processus de Hessenberg [61].

Essayons de voir comment se transforment les différentes relations données par le processus de Hessenberg Généralisé.

Ecrivons la matrice B_k sous la forme :

$$B_k = \begin{pmatrix} B_k^1 \\ B_k^2 \end{pmatrix}, \tag{1.14}$$

où B_k^1 est la matrice $k \times k$ formée des k premières lignes de B_k. En partitionnant de même Y_k^T sous la forme :

$$Y_k^T = \begin{pmatrix} I_k^{(k)} , & 0 \end{pmatrix}, \tag{1.15}$$

la condition d'orthogonalité (1.2) devient :

$$Y_k^T B_k = B_k^1 = L_k, \tag{1.16}$$

donc B_k^1 est une matrice triangulaire inférieure et B_k est une matrice trapézoïdale inférieure.

Avec le choix $\{y_k\}_{k=1}^m = \{e_k^{(n)}\}_{k=1}^m$ l'algorithme 1 se réduit à un algorithme plus simple puisque nous n'avons plus à calculer le produit scalaire (y_j, u) qui n'est autre maintenant que la $j^{\text{ème}}$ composante du vecteur u.

Ainsi nous avons l'algorithme suivant :

Algorithme 3 : *Processus de Hessenberg.*

- *Initialisation :* $b_1 = v$; $\eta_1 = (b_1)_1$;

- *Itération* : *pour* $k = 1, \ldots, m$
  $u = Ab_k$;
  pour $j = 1, \ldots, k$
  $h_{j,k} = (u)_j/\eta_j$; $u = u - h_{j,k}b_j$;
  fin du pour ;
  choisir $h_{k+1,k} \neq 0$, *quelconque ;*
  $b_{k+1} = u/h_{k+1,k}$; $\eta_{k+1} = (b_{k+1})_{k+1}$;
  fin du pour.

Dans le cas où il n y a pas de "breakdown", nous avons le résultat suivant qui est l'équivalent du théorème 5 obtenu pour le processus d'Arnoldi.

Théorème 7

Supposons que m itérations soient réalisables dans le processus de Hessenberg. Alors, les vecteurs $b_1, b_2, \ldots, b_m$ *calculés par l'algorithme 3 forment une base du sous espace de Krylov* $K_m(v, A)$. *De plus la matrice associée à cette base est trapézoïdale.*

Un "breakdown" peut survenir dans l'algorithme 3 s'il existe k tel que $(b_{k+1})_{k+1} = 0$. Cela n'implique pas que le degré du polynôme minimal associé à A et b_1 soit égal à k car b_{k+1} peut être non nul.

Nous pouvons éviter le problème du "breakdown" et assurer une bonne stabilité numérique en adoptant une stratégie de pivotage comme lors de la factorisation $\mathcal{LU}$.

Soit $p \in S_n$ (l'ensemble des permutations d'ordre n). La stratégie de pivotage consiste à faire le choix $\{y_k\}_{k=1}^m = \{e_{p(k)}^{(n)}\}_{k=1}^m$ au lieu de $\{y_k\}_{k=1}^m = \{e_k^{(n)}\}_{k=1}^m$. Ce choix revient à prendre pour $\{y_k\}_{k=1}^m$ la famille $\{e_k^{(n)}\}_{k=1}^m$, mais pas nécessairement dans son ordre naturel. Afin de déterminer $p(k)$, nous allons considérer la procédure pratique décrite dans [61].

Supposons que $p(1), p(2), \ldots, p(k)$ aient déjà été déterminés et que nous voulions calculer $p(k+1)$. Nous commençons par former le produit $u = Ab_k$. Nous soustrayons alors à u un multiple de b_1 afin d'annuler la composante $p(1)$ de u, un multiple de b_2 afin d'annuler la composante $p(2)$ de u, ..., et finalement un multiple de b_k afin d'annuler la composante $p(k)$ de u. Le vecteur ω ainsi obtenu vérifie : $(\omega)_{p(i)} = 0$ pour $i = 1, 2, \ldots, k$. Nous déterminous alors i_0 tel que $|(\omega)_{i_0}| = \|\omega\|_\infty$ et nous posons $p(k+1) = i_0$. Finalement afin d'éviter d'éventuels dépassements de capacité, nous prenons $b_{k+1} = \omega/(\omega)_{i_0}$.

Par cette procédure le vecteur b_{k+1} admet k composantes nulles, et est normalisé, i.e. $\|b_{k+1}\|_\infty = 1$. Nous n'obtenons plus une matrice B_k trapézoïdale inférieure. Mais il existe une matrice de permutation $P = (e^{(n)}_{p(1)}, e^{(n)}_{p(2)}, \ldots, e^{(n)}_{p(n)})$ telle que : $P^T B_k$ est une matrice trapézoïdale inférieure à diagonale unité.

En tenant compte des différentes remarques et observations faites sur la stratégie de pivotage, l'algorithme 3 peut être reécrit de la façon suivante, où la notation "$\alpha \longleftrightarrow \beta$" est équivalente à "$\gamma = \alpha$; $\alpha = \beta$; $\beta = \gamma$".

Algorithme 3.1 : *Processus de Hessenberg avec pivotage.*

- *Initialisation :* *poser* $p = (1, 2, \ldots, n)^T$;
  déterminer i_0 *tel que* $|(v)_{i_0}| = \|v\|_\infty$;
  $b_1 = v/(v)_{i_0}$; $p(1) \longleftrightarrow p(i_0)$;

- *Itération* : *pour* $k = 1, \ldots, m$
  $u = Ab_k$;
  pour $j = 1, \ldots, k$
  $h_{j,k} = (u)_{p(j)}$; $(u)_{p(j)} = 0$;
  pour $i = j+1, \ldots, n$
  $(u)_{p(i)} = (u)_{p(i)} - h_{j,k}(b_j)_{p(i)}$;
  fin du pour;
  fin du pour;
  si $k < n$
  déterminer i_0 *tel que* $|(u)_{i0}| = \|u\|_\infty$;
  $p(k+1) \longleftrightarrow p(i_0)$;
  fin du si ;
  $h_{k+1,k} = (u)_{i_0}$; $b_{k+1} = u/h_{k+1,k}$;
  fin du pour.

Si à l'étape k nous avons $\|u\|_\infty = 0$, alors l'algorithme précédent doit être arrêté. Nous avons alors le résultat suivant :

Théorème 8

Le processus de Hessenberg avec pivotage s'arrête à l'étape k si et seulement si le degré du polynôme minimal associé à A et v est égal à k. Dans ce cas le sous espace de Krylov $K_k(v,A)$ est invariant par A, i.e. $AK_k(v,A) = K_k(v,A)$.

Remarques :

1. *L'itération k du processus précédent nécessite $nk - k(k+1)/2$ multiplications et un produit matrice vecteur dont le côut est de $n\nu_k$ multiplications. Pour une matrice dense $\nu_k = n-k$, alors que pour une matrice creuse $n\nu_k \leq N_{nz}$ où N_{nz} est le nombre d'éléments non nuls de la matrice. Au total, pour m itérations le processus de Hessenberg nécessite un coût algorithmique de l'ordre de $nm^2/2 - m^3/6 + \sum_{k=1}^{m} n\nu_k$ multiplications.*

2. *Les $m(m+1)/2$ éléments supérieurs (respectivement les m éléments de la sous-diagonale) de $\widetilde{H}_m$ peuvent être stockés à la place des éléments nuls (respectivement égaux à l'unité) de la matrice B_{k+1}. L'algorithme de Hessenberg nécessite donc un stockage de nm éléments.*

3. *Comme pour le processus d'Arnoldi, nous pouvons choisir $h_{k+1,k} = \|u\|_2$ (i.e. b_{k+1} unitaire). Dans ce cas nous obtenons une matrice $P^T B_k$ trapézoïdale inférieure, mais non à diagonale unité.*

4. *Lors du calcul du vecteur $u = Ab_k$, les colonnes $p(1), \cdots, p(k-1)$ de la matrice A n'interviennent pas dans le produit matrice vecteur. Donc, si nous acceptons de détruire la matrice A, les éléments non nuls des vecteurs $b_1, \cdots, b_{k-1}$ ainsi que les éléments non nuls de la partie supérieure de la matrice $\widetilde{H}_m$ peuvent être stockés à la place des colonnes $p(1), \cdots, p(k-1)$ de A. Ainsi, le processus de Hessenberg ne nécessite que le stockage des éléments $h_{k+1,k}$ pour $k = 1, \ldots, m$ dans un vecteur auxiliaire de taille m.*

Avant de passer à l'étude du processus de Lanczos, donnons un résultat analogue à celui du théorème 6.

Supposons que la matrice de Krylov V_k associée à $K_k(v,A)$ soit partitionnée sous la forme :

$$V_k = \begin{pmatrix} V_k^1 \\ V_k^2 \end{pmatrix}.$$

Pour plus de simplicité, nous considérerons le processus de Hessenberg sans stratégie de pivotage.

Théorème 9

Soit $V_k = (v, Av \ldots, A^{m-1}v)$ *la matrice de Krylov associée à* $K_m(v, A)$. *Supposons que* $\forall k \in \{1, \ldots, m\}, det(V_k^1) \neq 0$, *alors la décomposition* $\mathcal{L}_m\mathcal{U}_m$ *(*$\mathcal{L}_m \in \mathbb{R}^{n\times m}$ *matrice triangulaire inférieure et* $\mathcal{U}_m \in \mathbb{R}^{m\times m}$ *matrice triangulaire supérieure à diagonale unité) de* V_m *existe. De plus, nous avons* $\mathcal{L}_m = B_m$.

Preuve :

Comme dans la démonstration du théorème 6, nous avons $V_m = B_m U_m^{-1}$, où U_m est une matrice triangulaire supérieure à diagonale unité.

Puisque V_k^1 est régulière pour tout $k \in \{1, \ldots, m\}$, il existe $\mathcal{L}_m \in \mathbb{R}^{n\times m}$ matrice triangulaire inférieure et $\mathcal{U}_m \in \mathbb{R}^{m\times m}$ matrice triangulaire supérieure à diagonale unité telles que $V_m = \mathcal{L}_m\mathcal{U}_m$.
Nous avons donc deux expressions pour la décomposition $\mathcal{LU}$ de V_m :

$$V_m = B_m U_m^{-1} = \mathcal{L}_m\mathcal{U}_m.$$

Finalement, par unicité de la factorisation $\mathcal{LU}$, nous obtenons $B_m = \mathcal{L}_m$ et $U_m^{-1} = \mathcal{U}_m$.

■

1.2.3 Processus de Lanczos

Par commodité, nous avons supposé dans la description du processus de Hessenberg Généralisé que les vecteurs $y_1, y_2, \ldots, y_m$ sont fixés ou connus dès le début du processus. En réalité, cette description n'exclut pas la possibilité que ces vecteurs soient calculés ou construits en même temps que les vecteurs $b_1, b_2, \ldots, b_m$ par un autre procédé plus efficace.

La famille de vecteurs $\{y_k\}_{k=1}^m$ peut être déterminée simultanément avec la famille $\{b_k\}_{k=1}^m$. Pour cela $\{y_k\}_{k=1}^m$ sera construite comme base d'un sous espace de Krylov associé à la matrice A^T. Ce procédé nous permet de retrouver le processus de Lanczos [32, 61].

Considérons $y \in \mathbb{R}^n$ vérifiant $y^T v \neq 0$, et calculons les vecteurs $\{y_k\}_{k=1}^m$ (qu'on notera dans la suite par $\{\widehat{b}_k\}_{k=1}^m$) par une relation basée sur A^T et de même nature que la relation (1.1).

Le processus de Lanczos construit donc une base $b_1, b_2, \ldots, b_m$ du sous espace de Krylov $K_m(v, A)$ et une base $\widehat{b}_1, \widehat{b}_2, \ldots, \widehat{b}_m$ du sous espace de Krylov $K_m(y, A^T)$ par les

relations :

$$h_{k+1,k}b_{k+1} = Ab_k - \sum_{i=1}^{k} h_{i,k}b_i \tag{1.17}$$

$$\hat{h}_{k+1,k}\hat{b}_{k+1} = A^T\hat{b}_k - \sum_{i=1}^{k} \hat{h}_{i,k}\hat{b}_i, \tag{1.18}$$

où les $h_{i,k}$, $\hat{h}_{i,k}$, $(i = 1,\ldots,k)$ sont déterminés de façon que b_{k+1} soit orthogonal à $\{\hat{b}_1,\ldots,\hat{b}_k\}$ et $\hat{b}_{k+1}$ orthogonal à $\{b_1,\ldots,b_k\}$. C'est à dire, nous imposons :

$$b_{k+1} \perp \hat{B}_k = (\hat{b}_1,\ldots,\hat{b}_k) \tag{1.19}$$

$$\hat{b}_{k+1} \perp B_k = (b_1,\ldots,b_k). \tag{1.20}$$

Matriciellement, les relations (1.17)–(1.20) s'écrivent respectivement :

$$AB_k = B_{k+1}\widetilde{H}_k \tag{1.21}$$

$$A^T\hat{B}_k = \hat{B}_{k+1}\widetilde{\hat{H}}_k \tag{1.22}$$

$$\hat{B}_k^T B_k = L_k \tag{1.23}$$

$$B_k^T\hat{B}_k = \hat{L}_k. \tag{1.24}$$

Les matrices $\widetilde{H}_k$, $\widetilde{\hat{H}}_k$ sont les matrices $(k+1)\times k$ Hessenberg supérieures formées respectivement par les éléments $h_{i,j}$ et $\hat{h}_{i,j}$, $j = 1,\ldots,k$ et $i = 1,\ldots,j+1$. Les matrices L_k et $\hat{L}_k$ étant des matrices triangulaires inférieures.

En fait, en comparant les équations (1.23) et (1.24), on voit que

$$L_k = \hat{L}_k = D_k = diag(\eta_1,\ldots,\eta_k), \tag{1.25}$$

avec $\eta_i = (\hat{b}_i, b_i)$. Les relations (1.20)–(1.24) donnent :

$$\hat{B}_k^T AB_k = D_k H_k \tag{1.26}$$

$$B_k^T A^T\hat{B}_k = D_k\widehat{H}_k. \tag{1.27}$$

Par ces deux dernières relations nous avons :

$$(D_k\widehat{H}_k)^T = D_k H_k. \tag{1.28}$$

Finalement nous obtenons :

$$H_k = {D_k}^{-1}\widehat{H}_k^T D_k = (D_k\widehat{H}_k{D_k}^{-1})^T. \tag{1.29}$$

Le premier membre de l'égalité (1.29) est une matrice de Hessenberg supérieure, alors que le second est une matrice Hessenberg inférieure. Les deux matrices H_k et $\widehat{H}_k$ sont donc tridiagonales, et les relations (1.17), (1.18) s'écrivent simplement :

$$h_{k+1,k}b_{k+1} = Ab_k - h_{k,k}b_k - h_{k-1,k}b_{k-1} \tag{1.30}$$

$$\hat{h}_{k+1,k}\hat{b}_{k+1} = A^T\hat{b}_k - \hat{h}_{k,k}\hat{b}_k - \hat{h}_{k-1,k}\hat{b}_{k-1}. \tag{1.31}$$

La relation (1.28) montre que nous avons :

$$h_{k,k} = \widehat{h}_{k,k} \quad \text{et} \quad h_{k+1,k}.h_{k,k+1} = \widehat{h}_{k+1,k}.\widehat{h}_{k,k+1}, \tag{1.32}$$

et en utilisant les conditions de biorthogonalité des familles de vecteurs $\{b_k\}_{k=1}^m$ et $\{\widehat{b}_k\}_{k=1}^m$ nous obtenons :

$$h_{k,k} = (\widehat{b}_k, Ab_k)/\eta_k = (A^T\widehat{b}_k, b_k)/\eta_k = \widehat{h}_{k,k}, \tag{1.33}$$

et

$$\begin{aligned} h_{k-1,k} &= (\widehat{b}_{k-1}, Ab_k)/\eta_{k-1} \\ &= (A^T\widehat{b}_{k-1}, b_k)/\eta_{k-1} \\ &= \widehat{h}_{k,k-1}(\widehat{b}_k, b_k)/\eta_{k-1}, \end{aligned} \tag{1.34}$$

la dernière égalité est obtenue en écrivant

$$A^T\widehat{b}_{k-1} = \widehat{h}_{k,k-1}\widehat{b}_k + \widehat{h}_{k-1,k-1}\widehat{b}_{k-1} + \widehat{h}_{k-2,k-1}\widehat{b}_{k-2}, b_k.$$

De même :

$$\begin{aligned} \widehat{h}_{k-1,k} &= (A^T\widehat{b}_k, b_k)/\eta_{k-1} \\ &= h_{k,k-1}(\widehat{b}_k, b_k)/\eta_{k-1}. \end{aligned} \tag{1.35}$$

Dans la suite nous choisirons les facteurs "normalisants" des vecteurs b_{k+1} et $\widehat{b}_{k+1}$ égaux (i.e. $\widehat{h}_{k+1,k} = h_{k+1,k}$). Dans ce cas la relation (1.32) ou les relations (1.34) et (1.35) montrent que $h_{k,k-1} = \widehat{h}_{k-1,k}$. C'est à dire que par ce choix, les matrices $\widetilde{H}_k$ et $\widetilde{\widehat{H}}_k$ obtenues par le processus de Lanczos sont identiques.

Finalement, en regroupant les formules précédentes nous obtenons le processus de Lanczos décrit par l'algorithme ci dessous :

Algorithme 4 : *Processus de Lanczos.*

- *Initialisation :* $b_1 = v$; $\widehat{b}_1 = y$; $\eta_1 = (\widehat{b}_1, b_1)$;
- *Itération* : *pour* $k = 1, \ldots, m$
  $u = Ab_k$; $\widehat{u} = A^T\widehat{b}_k$;
  pour $j = max(1, k-1), \ldots, k$
  $h_{j,k} = (\widehat{b}_j, u)/\eta_j$;
  $u = u - h_{j,k}b_j$; $\widehat{u} = \widehat{u} - h_{j,k}\widehat{b}_j$;
  fin du pour ;
  choisir $h_{k+1,k} \neq 0$, *quelconque ;*
  $b_{k+1} = u/h_{k+1,k}$; $\widehat{b}_{k+1} = \widehat{u}/h_{k+1,k}$; $\eta_{k+1} = (\widehat{b}_{k+1}, b_{k+1})$;
  fin du pour.

L'algorithme précédent doit être arrêté à l'étape k si $\eta_{k+1} = (\widehat{b}_{k+1}, b_{k+1}) = 0$. Cette situation est possible dans deux cas. Le premier survient lorsque $b_{k+1} = 0$ ou $\widehat{b}_{k+1} = 0$. Le second cas est lorsque $\widehat{b}_{k+1} \perp b_{k+1}$. Plusieurs approches ont été proposées pour remédier à cette situation. Nous reviendrons brièvement sur l'étude des deux cas où survient le "breakdown" dans le processus de Lanczos après avoir donné le résultat suivant :

Théorème 10

Supposons que m étapes soient réalisables dans le processus de Lanczos décrit par l'algorithme précédent. Alors les vecteurs $b_1, b_2, \ldots, b_m$ (respectivement $\widehat{b}_1, \widehat{b}_2, \ldots, \widehat{b}_m$) forment une base du sous espace de Krylov $K_m(v, A)$ (respectivement $K_m(y, A^T)$).
De plus, $\{b_k\}_{k=1}^m$ et $\{\widehat{b}_k\}_{k=1}^m$ sont deux familles biorthogonales, i.e.,

$$(\widehat{b}_j,\ b_i) = \eta_i \delta_{ij} \qquad 1 \leq i, j \leq m.$$

Comme il a été précédemment signalé, il peut survenir un "breakdown" dans le processus de Lanczos lorsque le vecteur b_{k+1} ou $\widehat{b}_{k+1}$ est nul. Cela peut être évité dans le cas où le vecteur y est bien choisi et lorsque le degré du polynôme minimal de A pour v, ainsi que celui de A^T pour y, est au moins égal à m. En fait nous avons le résultat suivant :

Théorème 11

Supposons que $\forall w \in K_m(v, A)$, $\forall \widehat{w} \in K_m(y, A^T)$ $(\widehat{w},\ w) \neq 0$, alors le processus de Lanczos s'arrête à l'étape k si et seulement si le degré du polynôme minimal associé à A et v (ou à A^T et y) est égal à k. Dans ce cas le sous espace de Krylov $K_k(v, A)$ (ou $K_k(y, A^T)$) est invariant par A (ou A^T).

Le second cas pour lequel nous avons $\eta_{k+1} = 0$, et donc un "breakdown" dans le processus de Lanczos, est lorsque le vecteur b_{k+1} est orthogonal au vecteur $\widehat{b}_{k+1}$, i.e. $\widehat{b}_{k+1} \perp b_{k+1}$, avec b_{k+1} et $\widehat{b}_{k+1}$ non nuls. Wilkinson [61, p.379] a qualifié cette situation de cas sérieux de "breakdown".

Plusieurs approches ont été proposées pour remédier aux problèmes de "breakdown" [6, 23, 35, 38]. L'idée essentielle de ces approches est que même si on ne peut pas définir les vecteurs b_{k+1} et $\widehat{b}_{k+1}$, il est possible qu'on puisse définir deux nouveaux vecteurs b_{k+i} et $\widehat{b}_{k+i}$ ($i \geq 2$) qui satisfont les différentes conditions du processus de Lanczos. L'algorithme peut alors être poursuivi normalement jusqu'à la fin, ou jusqu'à ce que l'on rencontre deux nouveaux vecteurs orthogonaux b_{k+j} et $\widehat{b}_{k+j}$ ($j \geq i$). Les différentes approches proposées ont été regroupées sous la dénomination "Look Ahead Lanczos algorithms" [23, 38] ou "Recursive Zoom process" [7]. Ces algorithmes sont tous basés sur

le lien existant entre le processus de Lanczos et les polynômes orthogonaux. Cependant ils diffèrent dans la stratégie adoptée pour le calcul des vecteurs b_{k+i} et $\widehat{b}_{k+i}$.

Remarques :

1. *Le processus de Lanczos nécessite à la $k^{\text{ème}}$ itération $6n$ multiplications, ainsi que deux produits matrice vecteur dont le côut est de $n\nu$ multiplications ($\nu = n$ pour une matrice dense, et $\nu = N_{nz}/n$ pour une matrice creuse). Au total pour m itérations, le processus de Lanczos nécessite un côut algorithmique de $6nm + 2nm\nu$ multiplications.*

2. *En ce qui concerne la place mémoire, le processus de Lanczos nécessite un stockage des nm éléments de la matrice B_m ainsi que des $3m - 1$ éléments des trois diagonales de la matrice $\widetilde{H}_m$.*

Nous terminons ce paragraphe consacré à la description du processus de Hessenberg Généralisé en faisant le lien entre le processus de Lanczos et le processus d'Arnoldi.

Théorème 12

Si la matrice A est symétrique et si les vecteurs de départ v et y sont égaux, alors les processus de Lanczos et d'Arnoldi appliqués à A et v sont équivalents.

Preuve :

Supposons que A soit symétrique et que $v = y$. Dans ce cas, il est clair que les sous espaces de Krylov $K_m(v, A)$ et $K_m(y, A^T)$ sont égaux. De même, par récurrence il est facile de montrer que les deux familles de vecteurs $\{b_k\}_{k=1}^m$ et $\{\widehat{b}_k\}_{k=1}^m$ calculées par le processus de Lanczos sont identiques.

De plus, la condition de biorthogonalité des deux familles montre que les vecteurs $b_1, \ldots, b_k$ ainsi obtenus sont orthogonaux.

Comme les vecteurs $b_2, \ldots, b_m$ sont calculés par la relation (1.1) qui est la même pour les processus d'Arnoldi et de Lanczos, nous pouvons affirmer que ces deux processus sont équivalents.

■

1.3 Processus de Hessenberg Généralisé Incomplet

Nous verrons dans le chapitre suivant quelques méthodes pour la résolution d'un système linéaire $Ax = f$, où A est une matrice a n lignes et n colonnes supposée inversible, et f, x deux vecteurs de $\mathbb{R}^n$. Dans ces méthodes le système initial est "approché" par un système de dimension inférieure à n plus facile à résoudre.

Ces méthodes utilisent un des algorithmes particuliers du processus de Hessenberg Généralisé essentiellement pour construire une base B_m du sous espace de Krylov $K_m(r_0, A) = span\{r_0, Ar_0, \ldots, A^{m-1}r_0\}$ où $r_0 = f - Ax_0$ et x_0 une solution initale du système à résoudre. Il faut alors remarquer que le coût algorithmique total nécessité par chaque algorithme pour le calcul de B_m ainsi que la place mémoire requise deviennent importants lorsque le nombre d'itérations m devient grand. Afin de contourner ce problème, nous avons le choix entre choisir m petit et redémarrer l'algorithme, ou bien tronquer le processus de Hessenberg Généralisé en fixant $l \leq k$ et en imposant que le vecteur b_{k+1} ne soit orthogonal qu'aux l vecteurs $y_{k-l+1}, \ldots, y_k$. Le processus de Hessenberg Généralisé ainsi obtenu est appelé processus de Hessenberg Généralisé Incomplet. Le but de cette section est de donner quelques propriétés de ce processus.

1.3.1 Description

En partant d'un vecteur initial $b_1 = v$, le processus de Hessenberg Généralisé Incomplet permet (lorsque cela est possible) de construire une base B_m de $K_m(v, A)$. A l'étape k le vecteur b_{k+1} est calculé par :

$$h_{k+1,k}b_{k+1} = Ab_k - \sum_{i=k-l+1}^{k} h_{i,k}b_i. \tag{1.36}$$

Les coefficients $\{h_{i,k}\}_{i=k-l+1}^{k}$ sont déterminés en imposant la condition d'orthogonalité :

$$b_{k+1} \perp \{y_{l-k+1}, \ldots, y_k\}. \tag{1.37}$$

Avec cette nouvelle condition d'orthogonalité imposée sur le vecteur b_{k+1}, les relations (1.3) et (1.4) sont encore valables, c'est à dire que nous avons :

$$AB_k = B_{k+1}\widetilde{H}_k = B_kH_k + h_{k+1,k}b_{k+1}e_k^{(k)T},$$

mais la matrice $\widetilde{H}_k$ n'est plus une matrice de Hessenberg supérieure. En effet la matrice $\widetilde{H}_k$ est une matrice bande de Hessenberg supérieure ayant ses surdiagonales $l+1, \ldots, k$ nulles. Par exemple, si $k = 6$, $l = 3$ la matrice $\widetilde{H}_k$ est telle que :

$$\widetilde{H}_k = \begin{pmatrix} h_{1,1} & h_{1,2} & h_{1,3} & 0 & 0 & 0 \\ h_{2,1} & h_{2,2} & h_{2,3} & h_{2,4} & 0 & 0 \\ & h_{3,2} & h_{3,3} & h_{3,4} & h_{3,5} & 0 \\ & & h_{4,3} & h_{4,4} & h_{4,5} & h_{4,6} \\ & & & h_{5,4} & h_{5,5} & h_{5,6} \\ & & & & h_{6,5} & h_{6,6} \\ & & & & & h_{7,6} \end{pmatrix},$$

de même la matrice $L_k = Y_k^T B_k$ n'est plus une matrice triangulaire inférieure mais une matrice dont les surdiagonales $2, \ldots, l+1$ sont nulles. Exemple, pour $k = 6$ et $l = 3$:

$$L_k = \begin{pmatrix} l_{1,1} & 0 & 0 & 0 & l_{1,5} & l_{1,6} \\ l_{2,1} & l_{2,2} & 0 & 0 & 0 & l_{2,6} \\ l_{3,1} & l_{3,2} & l_{3,3} & 0 & 0 & 0 \\ l_{4,1} & l_{4,2} & l_{4,3} & l_{4,4} & 0 & 0 \\ l_{5,1} & l_{5,2} & l_{5,3} & l_{5,4} & l_{5,5} & 0 \\ l_{6,1} & l_{6,2} & l_{6,3} & l_{6,4} & l_{6,5} & l_{6,6} \end{pmatrix}.$$

De la même façon que pour le processus de Hessenberg Généralisé, si la matrice L_k est inversible alors $B_k^L = L_k^{-1} Y_k^T$ définit une matrice inverse à gauche de B_k, et la matrice $\widetilde{H}_k = B_{k+1}^L A B_k = L_{k+1}^{-1} Y_{k+1}^T A B_k$.

En ne calculant que les coefficients $\{h_{i,k}\}_{i=k-l+1}^{k}$ dans l'algorithme 1, nous obtenons le processus de Hessenberg Généralisé Incomplet décrit par l'algorithme suivant :

Algorithme 5 : *Processus de Hessenberg Généralisé Incomplet.*

- *Initialisation :* $b_1 = v$; $\eta_1 = (y_1, b_1)$;
- *Itération* : *pour* $k = 1, \ldots, m, \ldots$
  $u = Ab_k$;
  pour $j = max\{1, k-l+1\}, \ldots, k$
  $h_{j,k} = (y_j, u)/\eta_j$; $u = u - h_{j,k} b_j$;
  fin du pour;
  choisir $h_{k+1,k} \neq 0$, *quelconque ;*
  $b_{k+1} = u/h_{k+1,k}$; $\eta_{k+1} = (y_{k+1}, b_{k+1})$;
  fin du pour.

Les différentes remarques et résultats concernant le problème du "breakdown" du

processus de Hessenberg Généralisé restent applicables au processus de Hessenberg Généralisé Incomplet. En particulier le théorème 3 reste valable et nous avons donc :

Théorème 13

Supposons que m étapes soient réalisables dans le processus de Hessenberg Généralisé Incomplet. Alors, les vecteurs $b_1, b_2, \ldots, b_m$ *calculés par l'algorithme 5 forment une base du sous espace de Krylov* $K_m(b_1, A)$.

1.3.2 Cas particuliers

Parmi les choix particuliers des vecteurs $\{y_i\}_{i=1}^k$ étudiés dans le processus de Hess-enberg Généralisé, seuls restent à faire les choix $y_i = b_i$ ou $y_i = e_i^{(n)}$. Le processus obtenu en faisant le choix $y_i = (A^T)^{i-1}y$ où y est un vecteur arbitraire n'est autre que le processus de Lanczos. Les deux autres choix restants sont étudiés ci dessous.

Procédure d'Orthogonalisation Incomplète : Nous avons vu précédemment que le choix $\{y_k\}_{k=1}^m = \{b_k\}_{k=1}^m$ dans le processus de Hessenberg Généralisé permet de retrouver le processus d'Arnoldi. En faisant ce même choix dans le processus de Hessenberg Généralisé Incomplet, nous retrouvons la procédure d'Orthogonalisation Incomplète due à Saad et étudiée pour la recherche des valeurs et vecteurs de Ritz associés à une valeur propre de la matrice A [39]. Cette procédure à été aussi utilisée pour la résolution des systèmes linéaires dans les méthodes IOM, DIOM [40] et DQGMRES [43].

L'algorithme ci dessous décrit la procédure d'Orthogonalisation Incomplète.

Algorithme 6 : *Procédure d'Orthogonalisation Incomplète.*

- *Initialisation :* $b_1 = v/\|v\|_2$;
- *Itération* : *pour* $k = 1, \ldots, m, \ldots$
  $u = Ab_k$;
  pour $j = max\{k - l + 1, 1\}, \ldots, k$
  $h_{j,k} = (b_j, u)$; $u = u - h_{j,k}b_j$;
  fin du pour;
  $h_{k+1,k} = \|u\|_2$; $b_{k+1} = u/h_{k+1,k}$;
  fin du pour.

La seule différence avec le processus d'Arnoldi est que à chaque étape le vecteur b_{k+1} est orthonormalisé par rapport aux l vecteurs précédents (au lieu de tous les vecteurs).

La suite de vecteurs ainsi construite est dite localement orthogonale, c'est à dire qu'elle vérifie :

$$(b_i, b_j) = \delta_{i,j}, \qquad \text{pour} \quad |i-j| < l$$

La démonstration du théorème 4 n'utilise pas le fait que la base B_k du sous espace de Krylov $K_k(b_1, A)$ construite par le processus d'Arnoldi est orthogonale. Donc le résultat de ce théorème reste valable pour la procédure d'Orthogonalisation Incomplète.

Théorème 14

L'algorithme de la procédure d'Orthogonalisation Incomplète s'arrête à l'étape k si et seulement si le degré du polynôme minimal associé à A et b_1 est égal à k. Dans ce cas le sous espace de Krylov $K_k(b_1, A)$ est invariant par A, i.e. $AK_k(b_1, A) = K_k(b_1, A)$.

Remarques :

1. *La procédure d'Orthogonalisation Incomplète nécessite à la $k^{\text{ème}}$ étape $2nl$ multiplications ainsi qu'un produit matrice vecteur dont le côut est de $n\nu$ multiplications. Pour une matrice dense $\nu = n$, alors que pour une matrice creuse $\nu = N_{nz}/n$ où N_{nz} est le nombre d'éléments non nuls de la matrice. Au total, pour m itérations, cette procédure nécessite un coût algorithmique de l'ordre de $2lnm + mn\nu$ multiplications.*

2. *En ce qui concerne la place mémoire, le processus d'Arnoldi nécessite un stockage des nm éléments de la matrice B_m ainsi que des $(m+3)m/2-(m-l)(m-l+1)/2$ éléments de la matrice $\widetilde{H}_m$.*

Processus de Hessenberg Incomplet : Le deuxième cas à étudier est le choix $\{y_k\}_{k=1}^m = \{e_k^{(n)}\}_{k=1}^m$. Dans ce cas nous obtenons le processus de Hessenberg Incomplet. Comme pour le processus de Hessenberg classique vu précédemment, il faut utiliser une stratégie de pivotage pour éviter d'éventuels "breakdown" afin d'assurer une bonne stabilité numérique.

L'algorithme suivant décrit le processus de Hessenberg Incomplet avec stratégie de pivotage :

Algorithme 7 : *Processus de Hessenberg Incomplet avec pivotage.*

- *Initialisation : poser* $p = (1, 2, \ldots, n)^T$;
  déterminer i_0 *tel que* $|(v)_{i_0}| = \|v\|_\infty$;
  $b_1 = v/(v)_{i_0}$; $p(1) \longleftrightarrow p(i_0)$;

- *Itération* : *pour* $k = max\{1, k-l+1\}, \ldots, m$
  $u = Ab_k$;
  pour $j = 1, \ldots, k$
  $h_{j,k} = (u)_{p(j)}$; $(u)_{p(j)} = 0$;
  pour $i = j+1, \ldots, n$
  $(u)_{p(i)} = (u)_{p(i)} - h_{j,k}(b_j)_{p(i)}$;
  fin du pour;
  pour $i = 1, \ldots, j-l-1$
  $(u)_{p(i)} = (u)_{p(i)} - h_{j,k}(b_j)_{p(i)}$;
  fin du pour;
  fin du pour;
  si $k < n$
  déterminer $i_0 \in \{k+1, \ldots, n\}$ *tel que* $|(u)_{i0}| = \|u\|_\infty$;
  $p(k+1) \longleftrightarrow p(i_0)$;
  fin du si ;
  $h_{k+1,k} = (u)_{i_0}$; $b_{k+1} = u/h_{k+1,k}$;
  fin du pour.

Chaque vecteur b_{k+1} construit par le processus de Hessenberg Incomplet avec stratégie de pivotage est tel que $(b_{k+1})_{p(i)} = 0$ pour $i = k-l+1, \ldots, k$ et $\|b_{k+1}\|_\infty = (b_{k+1})_{p(k+1)} = 1$.

Soit $P = (e^{(n)}_{p(1)}, e^{(n)}_{p(2)}, \ldots, e^{(n)}_{p(n)})$, il est alors evident que la matrice PB_k n'est plus trapézoïdale inférieure comme c'etait le cas pour la matrice PB_k construite par le processus de Hessenberg avec pivotage.

Remarquons que le résultat du théorème 8 établi pour le processus de Hessenberg avec pivotage reste valable pour sa version incomplète, c'est à dire que l'algorithme précédent ne doit être arrêté à une étape k si et seulement si $\|u\|_\infty = 0$; de plus nous avons le résultat suivant :

Théorème 15

Le processus de Hessenberg Incomplet avec pivotage s'arrête à l'étape k si et seulement si le degré du polynôme minimal de A pour v est égal à k. Dans ce cas le sous

espace de Krylov $K_k(v, A)$ est invariant par A, i.e. $AK_k(v, A) = K_k(v, A)$.

Remarques :

1. *L'itération k du processus précédent nécessite $(n - l - 1)l$ multiplications et un produit matrice vecteur dont le côut est de $n\nu_k$ multiplications. Pour une matrice dense $\nu_k = n - inf\{l, k\}$, alors que pour une matrice creuse $n\nu_k \leq N_{nz}$ où N_{nz} est le nombre d'éléments non nuls de la matrice. Au total, pour m itérations, le processus de Hessenberg Incomplet nécessite un coût algorithmique de l'ordre de $nlm - l^2m + \sum_{k=1}^{m} n\nu_k$ multiplications.*

2. *Les $(m-l)(2m-l+1)$ éléments non nuls (respectivement les m éléments de la sous-diagonale) de $\widetilde{H}_m$ peuvent être stockés à la place des éléments nuls (respectivement égaux à l'unité) de la matrice B_{k+1}. L'algorithme de Hessenberg incomplet nécessite donc un stockage de nm éléments.*

Conclusion

Dans ce chapitre, nous avons décrit le processus de Hessenberg Généralisé initialement étudié pour la réduction d'une matrice A à sa forme Hessenberg [30, 61] et pour le calcul des valeurs propres. Différentes procédures telles que : la procédure d'orthogonalisation d'Arnoldi, de triangularisation de Hessenberg et de tridiagonalisation de Lanczos peuvent être déduites comme cas particuliers du processus de Hessenberg Généralisé en adoptant les choix vus dans la seconde section de ce chapitre. Deux autres procédures ont également été étudiées dans la troisième section, il s'agit des versions incomplètes des processus d'Arnoldi et de Hessenberg.

L'utilisation dans le cadre de méthodes itératives pour la résolution d'un système linéaire des différents processus que nous venons d'énumérer a été largement dévelopée ces derniéres années. Ainsi, nous verrons dans le prochain chapitre la méthode de Hessenberg Généralisée qui est une généralisation de ces méthodes basée sur l'utilisation du processus de Hessenberg Généralisé.

Chapitre 2

La méthode de Hessenberg Généralisée pour les systèmes linéaires

Il a été rappelé au début du chapitre précédent que plusieurs méthodes de sous espaces de Krylov sont largement utilisées pour la résolution des systèmes d'équations linéaires algébriques. Les principales méthodes que nous avions déjà citées utilisent soit les processus d'Arnoldi, Lanczos ou Hessenberg, soit leurs versions tronquées obtenues comme cas particuliers du processus de Hessenberg Généralisé.

Le but de ce chapitre est de voir comment nous pouvons appliquer le processus de Hessenberg Généralisé comme méthode itérative pour la résolution des systèmes linéaires. La méthode obtenue sera appelée méthode de Hessenberg Généralisée. Cette méthode est constituée de deux principales classes de méthodes de résolution de systèmes linéaires. Nous verrons, en particulier, que les méthodes FOM, BCG et Hessenberg peuvent être déduites comme cas particuliers de la classe des méthodes de Galerkin. De même, nous verrons que les méthodes GMRES, QMR et CMRH appartiennent à la classe des méthodes MRSe [27].

Nous débuterons ce chapitre en posant le problème que nous nous proposons de résoudre. Nous donnons les conditions qui vont permettre de définir les deux classes de méthodes citées précédemment.

Le paragraphe 2.1 est consacré à la méthode de Galerkin ; nous verrons en 2.1.1 comment obtenir les itérés de cette méthode ainsi que son implémentation et quelques résultats théoriques vérifiés par cette méthode. Les cas particuliers de la méthode de

Galerkin sont brièvement rappelés en 2.1.2.

Une grande partie de ce chapitre est consacrée à la méthode MRSe qui sera définie au paragraphe 2.2. La dérivation, les propriétés et l'implémentation de cette méthode sont étudiées en 2.2.1. Les sous paragraphes 2.2.2 et 2.2.3 sont consacrés aux détails techniques utilisés pour l'implémentation pratique de la méthode, ces techniques sont respectivement la décomposition $\mathcal{QR}$ d'une matrice Hessenberg à l'aide des rotations de Givens et l'obtention d'un critère d'arrêt indispensable à toute méthode itérative. En 2.3, nous rappelons deux manières d'implémenter la méthode MRSe, la première est la méthode MRSe redémarrée notée RMRSe(m) et la seconde est la méthode TMRSe(l) basée sur l'utilisation du processus de Hessenberg Généralisé Incomplet.

L'avant dernier paragraphe 2.4 est consacré aux cas particuliers de la méthode MRSe. Ce sont des méthodes qui ont déjà été étudiées et sont largement utilisées dans la pratique. Ces méthodes sont rappelées dans l'ordre suivant : GMRES en 2.4.2, QMR en 2.4.3, CMRH en 2.4.3 et finalement les méthodes DQGMRES et DTCMRH en 2.4.4 et 2.4.5.

Finalement, dans le dernier paragraphe 2.5, nous donnons quelques résultats qui nous permetteront de comparer les performances des méthodes faisant partie de la méthode MRSe.

Préliminaires :

Afin de donner une présentation unifiée des différentes méthodes que nous aurons à traiter, les paramètres $h_{k+1,k}$ seront pris comme facteurs "normalisant" le vecteur b_{k+1} (i.e. $h_{k+1,k} = \|u\|$ où $\|\ .\ \|$ désigne une norme quelconque de $\mathbb{R}^n$).

Notre objectif est de résoudre le système $(\mathcal{S})$ d'équations linéaires suivant :

$$Ax = f, \qquad (\mathcal{S})$$

où A est une matrice réelle $n \times n$ inversible, f et x étant deux vecteurs de $\mathbb{R}^n$. La solution exacte du système $(\mathcal{S})$ est $x_* = A^{-1}f$.

Soient $y_1, \ldots, y_m$ des vecteurs linéairement indépendants de $\mathbb{R}^n$, et soit x_0 une approximation de x_*. Notons $r_0 = f - Ax_0$ le résidu associé à x_0.

La méthode de Hessenberg Généralisée consiste à construire une suite $\{x_m\}_{m\geq 1}$ vérifiant :

$$x_m = x_0 + \omega_m, \qquad (2.1)$$

avec $\omega_m \in K_m(r_0, A)$. Désignons par $r_m = f - Ax_m$ le résidu associé à chaque itéré x_m.

Celui-ci est complétement déterminé en imposant :

- soit la condition d'orthogonalité :

$$r_m \perp y_1, \ldots, y_m. \tag{2.2}$$

connue aussi sous le nom de condition de Petrov-Galerkin.

- soit la condition :

$$|r_m|_{Z_m} = \min_{x \in x_0 + K_m(r_0, A)} |f - Ax|_{Z_m}, \tag{2.3}$$

avec Z_m matrice symétrique non nécessairement "définie positive" qui sera précisée ultérieurement. Cette condition est appelée condition de moindres carrés ou encore condition de "semi-minimisation" du résidu.

Selon que l'on utilise l'une des deux conditions précédentes, nous obtenons deux classes de méthodes pour la résolution des systèmes linéaires. En imposant la condition de Petrov-Galerkin (2.2) nous définissons la méthode de Galerkin associée à la méthode Hessenberg Généralisée. La condition de moindres carrés nous permet de définir la méthode MRSe (Minimal Residual Seminorm method).

Dans la suite nous allons voir comment sont calculés les itérés x_m. Nous serons alors en mesure de décrire les algorithmes des deux classes de méthodes définies dans ce paragraphe.

2.1 La méthode de Galerkin

2.1.1 Dérivation de l'algorithme

La méthode de Hessenberg Généralisée permet (à l'aide du processus du même nom) de construire une base B_m du sous espace de Krylov $K_m(r_0, A)$. Or par l'équation (2.1) nous avons $\omega_m = x_m - x_0 \in K_m(r_0, A)$, il existe donc $d_m \in \mathbb{R}^m$ tel que $\omega_m = B_m d_m$. Ainsi la relation (2.1) devient :

$$x_m = x_0 + B_m d_m, \quad d_m \in \mathbb{R}^m,$$

et le résidu associé à la solution x_m peut être exprimé par :

$$\begin{aligned} r_m &= f - Ax_m \\ &= f - Ax_0 - AB_m d_m \\ &= r_0 - B_{m+1} \widetilde{H}_m d_m. \end{aligned}$$

Par la relation d'orthogonalité (2.2) nous obtenons :

$$\begin{aligned} 0 &= Y_m^T r_m \\ &= Y_m^T r_0 - (Y_m^T B_m, Y_m^T b_{m+1}) \widetilde{H}_m d_m \\ &= Y_m^T r_0 - (L_m, 0) \begin{pmatrix} H_m \\ h_{m+1,m} e_m^{(m)T} \end{pmatrix} d_m \\ &= Y_m^T r_0 - L_m H_m d_m, \end{aligned}$$

d'où

$$L_m H_m d_m = Y_m^T r_0, \tag{2.4}$$

or $Y_m^T r_0 = \beta Y_m^T b_1 = \beta l_1$ où l_1 est la première colonne de la matrice L_m et $\beta = \|r_0\|$. Cette dernière équation (2.4) nous montre que d_m est solution du système $m \times m$ suivant :

$$H_m d_m = \beta e_1^{(m)}. \tag{$\mathcal{S}_m$}$$

Finalement l'itéré x_m est défini par :

$$x_m = x_0 + \beta B_m H_m^{-1} e_1^{(m)}, \tag{2.5}$$

et l'algorithme de la méthode de Galerkin peut être formulé comme suit :

Algorithme 8 : *GAL.*

1- Initialisation :
calculer : $r_0 = f - Ax_0$; $\beta = \|r_0\|$; $b_1 = r_0/\beta$; $\eta_1 = (y_1, b_1)$;

2- Processus de Hessenberg Généralisé :
pour $k = 1, \ldots, m$
$u = Ab_k$;
pour $j = 1, \ldots, k$
$h_{j,k} = (y_j, u)/\eta_j$; $u = u - h_{j,k} b_j$;
fin du pour ;
$h_{k+1,k} = \|u\|$;
$b_{k+1} = u/h_{k+1,k}$; $\eta_{k+1} = (y_{k+1}, b_{k+1})$;
fin du pour .

3- Former l'approximation x_m :
chercher d_m *solution de :* $H_m d_m = \beta e_1^{(m)}$;
calculer $x_m = x_0 + B_m d_m$.

Dans la pratique nous devons atteindre une certaine précision sur le calcul de l'itéré x_m. Notons qu'il n'est pas difficile de savoir à quelle itération m avons nous atteint la

précision sur le calcul de x_m, en effet grâce à la relation (1.4) nous avons :

$$\begin{aligned} r_m &= r_0 - AB_m d_m \\ &= r_0 - B_m H_m d_m - h_{m+1,m} b_{m+1} e_m^{(m)T} d_m \\ &= r_0 - B_m e_1^{(m)} - h_{m+1,m} b_{m+1} e_m^{(m)T} d_m \\ &= r_0 - \beta b_1 - h_{m+1,m} b_{m+1} e_m^{(m)T} d_m \\ &= -h_{m+1,m} b_{m+1} e_m^{(m)T} d_m \end{aligned} \tag{2.6}$$

et donc $\rho_m \equiv \|r_m\|_2 = |h_{m+1,m}||e_m^{(m)T} d_m|\|b_{m+1}\|_2 = |h_{m+1,m}(d_m)_m|\|b_{m+1}\|_2$.

Cette relation est très importante puisqu'elle nous permet de calculer la norme du résidu r_m sans avoir a effectuer le produit matrice vecteur existant dans la relation classique $r_m = f - Ax_m$. De plus la relation (2.6) permet d'obtenir d'importantes propriétés.

Propriété 1

A une constante multiplicative près, le vecteur résidu r_m associé à la solution x_m est égal au vecteur b_{m+1}.

Propriété 2

En calcul exact et en l'absence de "breakdown" dans l'algorithme 8, la solution exacte x_* est obtenue en au plus m itérations où m est le degré du polynôme minimal de A pour r_0.

Preuve :

Supposons que $\forall\, k < m,\ x_k \neq x_*$ et soit r_m le résidu associé à la solution x_m. Par la relation (2.6) nous avons :

$$r_m = f - Ax_m = -h_{m+1,m} b_{m+1} e_m^{(m)T} d_m$$

or $b_{m+1} = 0$, sinon le degré du polynôme minimal de A pour r_0 serait supérieur à m .

Ainsi $r_m = 0$, c'est à dire que $x_m = A^{-1} f = x_*$.

■

A cause des limitations en place mémoire, le nombre m d'itérations dans l'algorithme GAL est nécessairement limité. Si après m étapes la solution obtenue n'est pas une "bonne" estimation de la solution $x_* = A^{-1} f$ nous pouvons redémarrer l'algorithme

en prenant x_m comme nouvelle approximation. La méthode ainsi obtenue sera notée GAL(m), son algorithme peut être formulé comme suit :

Algorithme 8.1 : *GAL(m).*

1- Initialisation :

calculer : $r_0 = f - Ax_0$; $\beta = \|r_0\|$; $b_1 = r_0/\beta$; $\eta_1 = (y_1, b_1)$;
choisir la précision ϵ

2- Processus de Hessenberg Généralisé :

pour $k = 1, \ldots, m$
$u = Ab_k$;
pour $j = 1, \ldots, k$
$h_{j,k} = (y_j, u)/\eta_j$; $u = u - h_{j,k}b_j$;
fin du pour ;
$h_{k+1,k} = \|u\|$;
$b_{k+1} = u/h_{k+1,k}$; $\eta_{k+1} = (y_{k+1}, b_{k+1})$;
fin du pour .

3- Former l'approximation x_m *:*

chercher d_m *solution de :* $H_m d_m = e_1^{(m)}$;
calculer $x_m = x_0 + B_m d_m$.

4- Redémarrer :

calculer une estimation de $\|r_m\|_2 = \|f - Ax_m\|_2$;
si $\|r_m\|_2 \leq \epsilon$ *stop ; sinon*
$x_0 = x_m$; $\beta = \|r_m\|_2$; $b_1 = r_m/\beta$; $\eta_1 = (y_1, b_1)$;
aller à l'étape 2 ;
fin du si ;

Il existe deux sources de "breakdown" dans l'algorithme de la méthode GAL. Le premier cas de "breakdown" est lié au processus de Hessenberg Généralisé. Le second survient lorsque la matrice H_m est singulière, ce qui ne permet pas de calculer l'approximation x_m. Dans ce cas il faut espérer qu'il existe $k < m$ tel que H_k soit non singulière, nous devons alors calculer x_k, la solution approchée correspondante. Cependant comme cette approximation n'est pas obtenue avec la précision voulue, nous devons redémarrer l'algorithme en prenant x_k comme nouvelle approximation.

Avant de finir la description de la méthode de Galerkin, nous devons remarquer qu'il se peut que la place mémoire ainsi que le côut algorithmique soient optimisés en utilisant le processus de Hessenberg Généralisé Incomplet. Nous obtenons ainsi une

nouvelle méthode appelée TGAL (méthode de Galerkin tronquée).

2.1.2 Cas particuliers

La méthode Arnoldi (FOM) : La méthode d'Arnoldi ou FOM (Full Ortogonalization Method) est due à Saad [40]. Cette méthode est obtenue en utilisant le processus d'Arnoldi avec "orthonormalisation" à l'étape *2* de l'algorithme 8. Notons que dans ce cas, nous avons à chaque itération k de la méthode FOM $h_{k+1,k} = \|u\|_2$, $\|b_{k+1}\|_2 = 1$ et $\rho_m = \|r_m^{\text{FOM}}\|_2 = h_{m+1,m}|e_m^{(m)T} d_m^{\text{FOM}}|$, où $x_m^{\text{FOM}}, \ldots$ désignent les itérés de la méthode d'Arnoldi/FOM. Saad a montré que l'on pouvait calculer le résidu r_m^{FOM} à chaque itération sans avoir à calculer le vecteur correspondant d_m^{FOM}. En utilisant l'élimination de Gauss avec pivotage pour adapter le décomposition $\mathcal{LU}$ de H_m, il a établi la relation suivante :

$$\rho_m = \|r_m^{\text{FOM}}\|_2 = h_{m+1,m}\|r_0\|_2 |\prod_{i \in I} l_i / u_{m,m}|$$

où I est l'ensemble des indices des itérations où il n'y a pas eu de pivotage; l_i, $(i \in I)$ sont les pivots successifs et $u_{m,m}$ le dernier élément de la matrice $\mathcal{U}$.

La décomposition $\mathcal{LU}$ donne des résultats satisfaisants en général, mais moins bien que la décomposition $\mathcal{LQ}$ proposée par Parlett [37]. Brown [8] a suggéré d'utiliser la décomposition $\mathcal{QR}$. Nous verrons une technique similaire basée sur cette décomposition lors de l'étude de la classe des méthodes MRSe.

Les méthodes IOM et DIOM : Afin d'optimiser le coût algorithmique requis par la méthode FOM, Saad a proposé d'utiliser la procédure d'Orthogonalisation Incomplète au lieu du processus d'Arnoldi [40, 41]. La méthode ainsi obtenue est désignée par IOM(l) (Incomplete Orthogonalisatiom Method). Le paramètre l étant le nombre de vecteurs par rapport auxquels le vecteur b_{k+1} est orthogonalisé. La méthode DIOM(l) est mathématiquement équivalente à la méthode IOM, cependant elle offre l'avantage d'utiliser moins de place mémoire du fait que la solution x_m^{DIOM} est calculée à partir de l'approximation précédente par un schéma de type gradient conjugué $x_m^{\text{DIOM}} = x_{m-1}^{\text{DIOM}} + z_m$. Ce schéma nécessite de garder en mémoire uniquement les vecteurs $z_{m-l+1}, \ldots, z_m$ au lieu des m vecteurs de la matrice B_m nécessaires au calcul de x_m^{DIOM} par la relation (2.5).

La méthode Hessenberg : En remplaçant dans la méthode FOM le processus d'Arnoldi par le processus de Hessenberg, nous obtenons la méthode Hessenberg pour la résolution des systèmes linéaires. Lorsque nous prenons comme choix de la famille $\{y_k\}_{k=1}^m$ les vecteurs de la base canonique dans leur ordre naturel, la méthode Hessenberg est mathématiquement équivalente à l'algorithme MMPE (Modified Minimal Polynomial Extrapolation algorithm) de Sidi, Ford et Smith [49].

Les méthodes THM et DTHM : En remplaçant respectivement dans les algorithmes des méthodes IOM(l) et DIOM(l) la procédure d'orthogonalisation incomplète par le processus de Hessenberg tronqué avec stratégie de pivotage, nous obtenons les méthodes THM(l) (Truncated Hessenberg Method) et DTHM(l) (Direct Truncated Hessenberg Method).

La méthode BCG : En utilisant le processus de Lanczos à l'étape *2* de l'algorithme de la méthode GAL, nous obtenons l'algorithme de la méthode du Bi-Gradient Conjugué. Cette méthode initialement découverte par Lanczos [32] a été donnée sous une autre forme qui lui est mathématiquement équivalente par Fletcher [19]. Etant donnée la forme tridiagonale de la matrice H_m construite par le processus de Lanczos, nous n'avons pas besoin de calculer explicitement sa décomposition $\mathcal{LU}$.

La convergence du BCG peut être très irrégulière, de plus la méthode est sujette au "breakdown" qui peut être évité par des stratégies de "look-ahead" [23, 38] ou "recursive zoom" [6, 7] ou bien par un redémarrage de l'algorithme à l'itération qui précède. Un autre inconvénient de la méthode BCG est l'utilisation du produit $A^T u$, ce dernier peut être impossible à effectuer dans certains cas où la matrice A ne peut être formée et que seulement son action sur un vecteur soit possible (exemple la matrice Jacobienne dans la résolution d'un système non linéaire par la méthode de Newton).

2.2 La méthode MRSe

2.2.1 Dérivation de l'algorithme

Avant d'expliquer comment sont obtenus les itérés, nous devons définir et donner quelques propriétés de la matrice Z_m utilisée dans la condition des moindres carrés qui définit la méthode MRSe.

La matrice Z_m est définie par :

$$Z_m = (B_{m+1}^L)^T B_{m+1}^L,$$

où $B_{m+1}^L = L_{m+1}^{-1} Y_{m+1}^T$ est une matrice inverse généralisée à gauche de B_{m+1} la matrice construite par le processus de Hessenberg Généralisé. La matrice Z_m est symétrique, mais non définie positive en général sur $\mathbb{R}^n$. Cependant nous avons le résultat suivant :

Théorème 16

La matrice Z_m est symétrique définie positive sur $K_{m+1}(r_0, A)$.

Preuve :

Soit $u \in K_{m+1}(r_0, A)$, nous avons alors $u = B_{m+1}s$ avec $s \in \mathbb{R}^{m+1}$. Ainsi :

$$u^T Z_m u = u^T (B_{m+1}^L)^T B_{m+1}^L u = s^T B_{m+1}^T (B_{m+1}^L)^T B_{m+1}^L B_{m+1} s$$

or $B_{m+1}^L B_{m+1} = I_{m+1}^{(m+1)}$, et donc pour tout vecteur u non nul dans $K_{m+1}(r_0, A)$ nous avons $s \neq 0$, ainsi

$$u^T Z_m u = s^T s = \|s\|_2^2 > 0.$$

■

Puisque la matrice Z_m est symétrique définie positive sur $K_{m+1}(r_0, A)$ nous pouvons définir le produit scalaire et la norme induite par Z_m.

Définition 5

Le Z_m-produit scalaire et la Z_m-norme correspondante sont respectivement définis par :

$$\forall x, y \in K_{m+1}(r_0, A) \quad (x,\ y)_{Z_m} = (x,\ Z_m y) \quad et \quad |x|_{Z_m} = \sqrt{(x,\ x)_{Z_m}}.$$

On notera $x \perp_{Z_m} y$ lorsque $(x,\ y)_{Z_m} = 0$.

Comme pour la méthode de Galerkin, écrivons $\omega_m = x_m - x_0$ dans la base B_m construite par le processus de Hessenberg Généralisé :

$$x_m = x_0 + B_m d_m, \quad d_m \in \mathbb{R}^m.$$

La solution approchée x_m est déterminée comme solution du problème de minimisation suivant :

$$\min_{x \in x_0 + K_m(r_0, A)} |f - Ax|_{Z_m}. \qquad (\mathcal{M}_0)$$

En utilisant les relations (1.3) et (1.7) nous avons :

$$\begin{aligned} |f - Ax|_{Z_m} &= \|B_{m+1}^L (f - Ax)\|_2 \\ &= \|B_{m+1}^L (B_{m+1} \beta e_1^{(m+1)} - AB_m d)\|_2 \\ &= \|B_{m+1}^L B_{m+1} (\beta e_1^{(m+1)} - \widetilde{H}_m d)\|_2 \\ &= \|\beta e_1^{(m+1)} - \widetilde{H}_m d\|_2, \end{aligned}$$

ainsi la résolution du problème $(\mathcal{M}_0)$ est équivalente à celle du problème de minimisation suivant :

$$\min_{d \in \mathbb{R}^m} \|\beta e_1^{(m+1)} - \widetilde{H}_m d\|_2. \qquad (\mathcal{M}_f)$$

Nous noterons $\mathcal{J}_m(d) = \|\beta e_1^{(m+1)} - \widetilde{H}_m d\|_2$. La solution de ce problème aux moindres carrés est donnée par :

$$d_m = \beta \widetilde{H}_m^+ e_1^{(m+1)}, \qquad (2.7)$$

où $\widetilde{H}_m^+ = (\widetilde{H}_m^T \widetilde{H}_m)^{-1} \widetilde{H}_m^T$ désigne la matrice pseudo-inverse de la matrice de Hessenberg supérieure $\widetilde{H}_m$.

Finalement la solution approchée x_m construite par la méthode MRSe est donnée par :

$$x_m = x_0 + \beta B_m \widetilde{H}_m^+ e_1^{(m+1)}. \qquad (2.8)$$

En modifiant l'étape *3* de l'algorithme de la méthode de Galerkin, nous obtenons celui de la méthode MRSe :

Algorithme 9 : *Méthode MRSe.*

1- Initialisation :
calculer : $r_0 = f - Ax_0$; $\beta = \|r_0\|$; $b_1 = r_0/\beta$; $\eta_1 = (y_1, b_1)$;

2- Processus de Hessenberg Généralisé :
pour $k = 1, \ldots, m$
$u = Ab_k$;
pour $j = 1, \ldots, k$
$h_{j,k} = (y_j, u)/\eta_j$; $u = u - h_{j,k} b_j$;
fin du pour ;
$h_{k+1,k} = \|u\|$;
$b_{k+1} = u/h_{k+1,k}$; $\eta_{k+1} = (y_{k+1}, b_{k+1})$;
fin du pour .

3- Former l'approximation x_m :
chercher d_m *solution de :* $\min_{d \in \mathbb{R}^m} \|\beta e_1^{(m+1)} - \widetilde{H}_m d\|_2$;
calculer $x_m = x_0 + B_m d_m$.

De façon similaire à l'algorithme de la méthode de Galerkin, l'étape *3* de l'algorithme 9 nécessite le calcul du vecteur de direction d_m solution du problème $(\mathcal{M}_f)$. La décomposition $\mathcal{QR}$ de la matrice $\widetilde{H}_m$ à l'aide de rotations de Givens offre une technique particulièrement appropriée pour trouver une telle solution. De plus elle nous permettra

de calculer la norme du résidu ou une estimation de la norme à chaque itération de l'algorithme sans avoir à construire la solution correspondante. Nous reviendrons sur l'étude de la décomposition $\mathcal{QR}$ après avoir montré le résultat suivant :

Propriété 3

Soit m le degré du polynôme minimal de A pour r_0. En calcul exact et en l'absence de "breakdown" dans le processus de Hessenberg Généralisé, les itérés x_j $j = 1, \ldots, m-1$ sont bien définis, de plus x_m est la solution exacte.

Preuve :

Le degré du polynôme minimal étant égal à m, les matrices B_j, $\widetilde{H}_j$, $j = 1, \ldots, m$ sont de rang maximal. Ainsi, chaque itéré $x_j = x_0 + \beta B_j \widetilde{H}_j^+ e_1^{(j+1)}$ pour $j = 1, \ldots, m$ est bien défini.

Comme $dim(K_m(r_0, A)) = dim(K_{m+1}(r_0, A)) = m$ alors il existe $d \in \mathbb{R}^m$ tel que $Ab_m = B_m d$, d'où :

$$b_{m+1} = Ab_m - B_m B_m^L Ab_m = B_m d - B_m B_m^L B_m d = B_m d - B_m d = 0.$$

Par conséquent $h_{m+1,m}$ est nul et la matrice $\widetilde{H}_m$ s'écrit $\widetilde{H}_m = \begin{pmatrix} H_m \\ 0 \end{pmatrix}$, et dans ce cas, nous avons $\widetilde{H}_m^+ = (H_m^{-1}, 0)$, $AB_m = B_m H_m$ et $\widetilde{H}_m^+ e_1^{(m+1)} = H_m^{-1} e_1^{(m)}$.

Finalement :

$$\begin{aligned} r_m &= f - Ax_m. \\ &= r_0 - \beta AB_m \widetilde{H}_m^+ e_1^{(m+1)} \\ &= r_0 - \beta B_m H_m H_m^{-1} e_1^{(m)} \\ &= r_0 - \beta b_1 \\ &= r_0 - r_0 \\ &= 0. \end{aligned}$$

■

2.2.2 Décomposition $\mathcal{QR}$

Les rotations de Givens sont un outil très efficace qui nous permettra d'effectuer la factorisation $\mathcal{QR}$ à chaque calcul d'une nouvelle colonne de la matrice $\widetilde{H}_m$.

Nous devons calculer "à chaque itération k" de l'algorithme de la méthode MRSe une matrice $\mathcal{Q}_k$ de taille $(k+1)\times(k+1)$ orthogonale et une matrice $\bar{\mathcal{R}}_k$ de taille $(k+1)\times k$ triangulaire supérieure telle que:

$$\widetilde{H}_k = \mathcal{Q}_k^T \bar{\mathcal{R}}_k. \tag{2.9}$$

En injectant cette dernière égalité dans l'expression de $\mathcal{J}_k(d)$ nous obtenons:

$$\begin{aligned}
\mathcal{J}_k(d) &= \|\beta e_1^{(k+1)} - \mathcal{Q}_k^T \bar{\mathcal{R}}_k d\|_2 \\
&= \|\mathcal{Q}_k^T(\mathcal{Q}_k(\beta e_1^{(k+1)}) - \bar{\mathcal{R}}_k d)\|_2 \\
&= \|\mathcal{Q}_k(\beta e_1^{(k+1)}) - \bar{\mathcal{R}}_k d\|_2 \\
&= \|\bar{g}_k - \bar{\mathcal{R}}_k d\|_2
\end{aligned}$$

avec $\bar{g}_k = \mathcal{Q}_k(\beta e_1^{(k+1)}) = (\mu_1^{(k)}, \ldots, \mu_{k+1}^{(k)}) \in \mathbb{R}^{k+1}$.

La matrice $\bar{\mathcal{R}}_k$ étant triangulaire supérieure de taille $(k+1)\times k$, il existe une matrice triangulaire supérieure $\mathcal{R}_k$ de taille $k \times k$ tel que:

$$\bar{\mathcal{R}}_k = \begin{pmatrix} \mathcal{R}_k \\ O \end{pmatrix},$$

de même, il existe $g_k \in \mathbb{R}^k$ tel que;

$$\bar{g}_k = \begin{pmatrix} g_k \\ \mu_{k+1}^{(k)} \end{pmatrix},$$

il vient alors que:

$$\mathcal{J}_k(d) = \left\| \begin{pmatrix} g_k - \mathcal{R}_k d \\ \mu_{k+1}^{(k)} \end{pmatrix} \right\|_2.$$

Ainsi la solution d_k est obtenue en résolvant le système triangulaire supérieur

$$\mathcal{R}_k d = g_k. \tag{2.10}$$

En l'absence de "breakdown" dans le processus de Hessenberg Généralisé, les éléments sous diagonaux de $\widetilde{H}_k$ sont non nuls, il vient que $\widetilde{H}_k$ est de rang maximal. La relation (2.9) montre alors que $\bar{\mathcal{R}}_k$ est elle même de rang maximal. Par conséquent $\mathcal{R}_k$ est inversible et la solution $d_k = \mathcal{R}_k^{-1} g_k$ existe toujours et est unique.

Finalement, et d'après ce qui précède, la solution approchée x_k du système $(\mathcal{S})$ existe, est unique et est donnée par:

$$x_k = x_0 + B_k \mathcal{R}_k^{-1} g_k. \tag{2.11}$$

En résumé la structure de l'algorithme de la méthode MRSe peut être décrite comme suit :

- *Calculer B_{k+1} et $\widetilde{H}_k$ par le processus de Hessenberg Généralisé.*
- *Effectuer la décomposition $\mathcal{QR}$ de $\widetilde{H}_k$ à l'aide des rotations de Givens.*
- *Calculer d_k solution du système (2.10).*
- *Calculer x_k par la relation (2.11).*

Afin de savoir si une solution x_k approche suffisamment bien la solution exacte x_*, nous devons être en mesure de calculer de façon simple et peu côuteuse le résidu r_k associé à x_k. Notons qu'il suffit aussi de calculer une estimation de la norme du résidu à chaque itération k. Nous reviendrons sur ce point après avoir vu en détail comment la factorisation $\mathcal{QR}$ peut être obtenue.

Dans la suite à chaque itération k nous désignerons par G_j (respectivement $\bar{G}_j^k$) la matrice de rotation de taille $(j+1)\times(j+1)$ (respectivement $(k+1)\times(k+1)$) qui agit sur le plan engendré par $e_j^{(n)}$ et $e_{j+1}^{(n)}$:

$$G_j = \begin{pmatrix} I_{j-1}^{(j-1)} & 0 & 0 \\ 0 & c_j & s_j \\ 0 & -s_j & c_j \end{pmatrix},$$

$$\bar{G}_j^k = \begin{pmatrix} G_j & 0 \\ 0 & I_{k-j}^{(k-j)} \end{pmatrix},$$

avec $c_j,\ s_j \in \mathbb{R}$ vérifiant $c_j^2 + s_j^2 = 1$.

Supposons que les rotations $\bar{G}_j^{k-1}\quad j = 1\ldots, k-1$ aient déjà été appliquées à $\widetilde{H}_{k-1}$ et que nous ayons obtenu la matrice $\bar{R}_{k-1}$ (qu'on stocke dans $\widetilde{H}_{k-1}$) de taille $k\times(k-1)$ triangulaire inférieure

$$\bar{\mathcal{R}}_{k-1} = \begin{pmatrix} h_{1,1}^{(k-1)} & h_{1,2}^{(k-1)} & \cdots & h_{1,k-1}^{(k-1)} \\ & h_{2,2}^{(k-1)} & \cdots & h_{2,k-1}^{(k-1)} \\ & & \ddots & \vdots \\ & & & h_{k-1,k-1}^{(k-1)} \\ 0 & 0 & \cdots & 0 \end{pmatrix}.$$

A l'étape courante k, la dernière colonne et ligne de $\widetilde{H}_k$ sont calculées. Nous commençons par prémultiplier la nouvelle colonne de $\widetilde{H}_k$ par les précédentes rotations $\bar{G}_j^k$,

nous obtenons alors la matrice $(k+1) \times k$ suivante :

$$\begin{pmatrix} & & h_{1,k}^{(k)} \\ & \mathcal{R}_{k-1} & \vdots \\ & & h_{k-1,k}^{(k)} \\ 0 & \cdots\ 0 & h_{k,k}^{(k-1)} \\ 0 & \cdots\ 0 & h_{k+1,k} \end{pmatrix}.$$

La prochaine rotation utilisée est $\bar{G}_k^k$, celle qui élimine l'élément $h_{k+1,k}$ qui n'a pas été affecté par les précédentes rotations. La rotation $\bar{G}_k^k$ est donc définie par :

$$c_k = h_{k,k}^{(k-1)} / (h_{k,k}^{(k-1)^2} + h_{k+1,k}^2)^{1/2}$$

$$s_k = h_{k+1,k} / (h_{k,k}^{(k-1)^2} + h_{k+1,k}^2)^{1/2}$$

et la matrice $\bar{R}_k$ est telle que :

$$\bar{R}_k = \begin{pmatrix} & & h_{1,k}^{(k)} \\ & \mathcal{R}_{k-1} & \vdots \\ & & h_{k-1,k}^{(k)} \\ 0 & \cdots\ 0 & h_{k,k}^{(k)} \\ 0 & \cdots\ 0 & 0 \end{pmatrix}.$$

2.2.3 Critère d'arrêt

Comme nous l'avons déjà signalé, nous devons être en mesure de calculer une estimation de la norme du résidu ou le résidu lui-même afin de savoir à chaque itération k si la solution approchée x_k est une "bonne" approximation de la solution exacte x_*. Cela est rendu possible par la technique des rotations de Givens lors de la factorisation $\mathcal{QR}$ de $\widetilde{H}_k$ utilisée pour résoudre le problème des moindres carrés $(\mathcal{M}_f)$. En effet nous avons :

$$\begin{aligned} r_k &= f - Ax_k \\ &= f - Ax_0 - AB_k \mathcal{R}_k^{-1} g_k \\ &= r_0 - B_{k+1} \widetilde{H}_k \mathcal{R}_k^{-1} g_k \\ &= B_{k+1} (\beta e_1^{(k+1)} - \widetilde{H}_k \mathcal{R}_k^{-1} g_k) \\ &= B_{k+1} \mathcal{Q}_k^T (\mathcal{Q}_k (\beta e_1^{(k+1)}) - \bar{\mathcal{R}}_k \mathcal{R}_k^{-1} g_k) \\ &= B_{k+1} \mathcal{Q}_k^T \left[\bar{g}_k - \begin{pmatrix} g_k \\ 0 \end{pmatrix} \right] \end{aligned}$$

$$= B_{k+1}\mathcal{Q}_k^T\left[\begin{pmatrix} g_k \\ \mu_{k+1}^{(k)} \end{pmatrix} - \begin{pmatrix} g_k \\ 0 \end{pmatrix}\right].$$

Finalement :

$$r_k = B_{k+1}\mathcal{Q}_k^T \begin{pmatrix} 0 \\ \vdots \\ 0 \\ \mu_{k+1}^{(k)} \end{pmatrix}. \tag{2.12}$$

En passant aux normes et par majoration nous obtenons :

$$\begin{aligned} \|r_k\|_2 &\leq \|B_{k+1}\|_2 \|\mathcal{Q}_k^T\|_2 |\mu_{k+1}^{(k)}| \\ &= |\mu_{k+1}^{(k)}| \ \|B_{k+1}\|_2. \end{aligned} \tag{2.13}$$

Connaissant une borne supérieure de $\|B_{k+1}\|_2$, nous obtenons une borne supérieure de la norme du résidu r_k, et cela pour un côut d'une multiplication à chaque itération.

Une autre possibilité consiste à calculer le résidu r_k à partir du précédent résidu r_{k-1}. D'après (2.12) r_k s'écrit :

$$r_k = \mu_{k+1}^{(k)} T_{k+1} e_{k+1}^{(k+1)}, \tag{2.14}$$

avec $T_{k+1} = \left(t_1, \ldots, t_k, \tilde{t}_{k+1}\right) = B_{k+1}\mathcal{Q}_k^T \in \mathbb{R}^{n\times(k+1)}$.
La matrice $\mathcal{Q}_k$ étant le produit des matrices de rotations $\bar{G}_j^k$, pour j variant de 1 à k, sa matrice transposée s'écrit :

$$\mathcal{Q}_k^T = \begin{pmatrix} c_1 & -s_1 & & & \\ s_1 & c_1 & & & \\ & & 1 & & \\ & & & \ddots & \\ & & & & 1 \end{pmatrix} \cdots \begin{pmatrix} 1 & & & & \\ & \ddots & & & \\ & & 1 & & \\ & & & c_k & -s_k \\ & & & s_k & c_k \end{pmatrix}$$

et par récurrence nous obtenons :

$$\begin{aligned} t_k &= c_k\tilde{t}_k + s_k b_{k+1} \\ \tilde{t}_{k+1} &= -s_k\tilde{t}_k + c_k b_{k+1} \end{aligned}$$

avec $\tilde{t}_1 = b_1$.

Avant de donner r_k en fonction de r_{k-1} nous avons besoin de donner l'expression de $\mu_{k+1}^{(k)}$ en fonction de $\mu_k^{(k-1)}$. En effet :

$$\bar{g}_k = \mathcal{Q}_k(\beta e_1^{(k+1)})$$

$$= \bar{G}_k^k \begin{pmatrix} \mathcal{Q}_{k-1} & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \beta e_1^{(k)} \\ 0 \end{pmatrix}$$
$$= \bar{G}_k^k \begin{pmatrix} \bar{g}_{k-1} \\ 0 \end{pmatrix}.$$

Finalement

$$\bar{g}_k = \begin{pmatrix} \bar{g}_{k-1} \\ \mu_k^{(k)} \\ \mu_{k+1}^{(k)} \end{pmatrix}, \tag{2.15}$$

avec

$$\mu_{k+1}^{(k)} = -s_k \mu_k^{(k-1)} \quad \text{et} \quad \mu_k^{(k)} = c_k \mu_k^{(k-1)}. \tag{2.16}$$

Ainsi de l'équation (2.14), de l'expression de T_{k+1} et de celle de $\mu_{k+1}^{(k)}$ nous avons :

$$\begin{aligned} r_k &= \mu_{k+1}^{(k)} \widetilde{t}_{k+1} \\ &= s_k^2 \mu_k^{(k-1)} \widetilde{t}_k + c_k b_{k+1}. \end{aligned}$$

Finalement l'expression recherchée de r_k en fonction de r_{k-1} est :

$$r_k = s_k^2 r_{k-1} + c_k b_{k+1}. \tag{2.17}$$

2.3 Implémentation pratique de la méthode MRSe

En supposant un calcul exact, la méthode MRSe converge en moins de n étapes. Cependant il est clair que cette propriété n'a plus d'importance lorsque la taille du système est grande. Le principal inconvénient de la méthode MRSe est l'importance du côut algorithmique ainsi que celui de la place mémoire requise par l'algorithme. Si nous n'obtenons pas rapidement une solution suffisamment proche de la solution exacte, le côut de l'algorithme deviendra rapidement prohibitif.

Deux stratégies ont été proposées pour éviter le problème du côut et de la place mémoire. La première stratégie consiste à redémarrer l'algorithme après un nombre fixé d'itérations de la méthode MRSe. La seconde solution consiste à tronquer le processus de Hessenberg Généralisé. Ces deux stratégies sont l'objet des deux sections suivantes :

2.3.1 La méthode RMRSe(m)

Une première solution classique pour contourner le problème du coût dans la méthode MRSe est de relancer l'algorithme. Après avoir fixé un nombre m d'itérations, si la solution x_m ne représente pas une "bonne" estimation de la solution exacte x_*, l'algorithme est redémarré en prenant x_m comme nouvelle solution initiale. Ce processus est répété jusqu'a obtention d'une meilleure précision sur la solution cherchée. Cette méthode sera désignée par RMRSe(m) et son algorithme peut être donné comme suit :

Algorithme 9.1 :*Méthode RMRSe(m).*

1- Initialisation :
calculer : $r_0 = f - Ax_0$; $b_1 = r_0$; $\eta_1 = (y_1, r_0)$;
choisir la précision ϵ.

2- Processus de Hessenberg Généralisé :
pour $k = 1, \ldots, m$
$u = Ab_k$;
pour $j = 1, \ldots, k$
$h_{j,k} = (y_j, u)/\eta_j$; $u = u - h_{j,k}b_j$;
fin du pour ;
$h_{k+1,k} = \|u\|$;
$b_{k+1} = u/h_{k+1,k}$; $\eta_{k+1} = (y_{k+1}, b_{k+1})$;
fin du pour .

3- Former l'approximation x_m :
chercher d_m *solution de :* $\min_{d \in \mathbb{R}^m} \|e_1^{(m+1)} - \widetilde{H}_m d\|_2$;
calculer $x_m = x_0 + B_m d_m$.

4- Redémarrer
calculer une estimation de $\|r_m\|_2 = \|f - Ax_m\|_2$.
si $\|r_m\|_2 \leq \epsilon$ *stop ; sinon*
$x_0 = x_m$; $b_1 = r_m = f - Ax_0$; $\eta_1 = (y_1, b_1)$;
aller à l'étape 2 ;
fin du si ;

Cette technique de redémarrage de la méthode MRSe n'est pas toujours efficace. En effet, il existe des exemples de systèmes linéaires pour lesquels cette stratégie ne permet pas de trouver une solution approchée de la solution exacte [8]. Dans la partie consacrée aux exemples numériques, nous donnons quelques exemples illustrant ce fait.

2.3.2 La méthode TMRSe(l)

En remplaçant dans l'algorithme de la méthode MRSe le processus de Hessenberg Généralisé par sa version incomplète, nous obtenons l'algorithme de la méthode TMRSe(l). Cette méthode est décrite par l'algorithme suivant :

Algorithme 9.2 : *Méthode TMRSe(l).*

1- Initialisation :

calculer : $r_0 = f - Ax_0$; $b_1 = r_0$; $\eta_1 = (y_1, r_0)$;

2- Processus de Hessenberg Généralisé Incomplet:

pour $k = 1, \ldots, m$

$u = Ab_k$;

pour $j = max\{1, k-l+1\}, \ldots, k$

$h_{j,k} = (y_j, u)/\eta_j$; $u = u - h_{j,k}b_j$;

fin du pour ;

$h_{k+1,k} = \|u\|$;

$b_{k+1} = u/h_{k+1,k}$; $\eta_{k+1} = (y_{k+1}, b_{k+1})$;

fin du pour .

3- Former la solution x_m *:*

chercher d_m *solution de :* $\min_{d \in \mathbb{R}^m} \|e_1^{(m+1)} - \widetilde{H}_m d\|_2$;

calculer $x_m = x_0 + B_m d_m$.

Pour calculer b_{k+1} à chaque itération k du processus de Hessenberg Généralisé Incomplet, nous n'avons besoin de garder en mémoire que les l vecteurs précédents $b_{k-l+1}, \ldots, b_k$. Cependant le calcul de $x_k = x_0 + B_k d_k$ nécessite la memorisation des k vecteurs précédents. Donc l'algorithme ci dessus tel qu'il a été formulé permet seulement de diminuer le coût algorithmique, mais ne permet pas de gagner en place mémoire. Il est possible de calculer x_k à partir de x_{k-1} par une relation de la forme $x_k = x_{k-1} + \mu_k w_k$ (comme pour la méthode DIOM(l) [40]) w_k étant des vecteurs auxiliaires calculés par des relations de récurrence "courtes".

Avant de voir comment obtenir la nouvelle approximation x_k à partir de l'ancienne approximation x_{k-1}, rappelons que chaque nouvelle colonne de la matrice $\widetilde{H}_k$ est de la forme $(0, \ldots, 0, h_{k-l+1,k}, \ldots, h_{k,k}, h_{k+1,k})^T$. Ainsi par application des rotations de Givens, $\widetilde{H}_k$ se transforme en une matrice triangulaire supérieure $\bar{R}_k$ (qui est stockée dans $\widetilde{H}_k$) ayant $(0, \ldots, 0, h_{k-l,k}^{(k)}, \ldots, h_{k,k}^{(k)}, 0)^T$ pour $k^{\text{ème}}$ colonne.

Il est clair que la relation (2.11) reste valable pour le calcul de x_k. Considérons alors

$$W_k = B_k R_k^{-1}, \tag{2.18}$$

ce qui permet d'écrire $x_k = x_0 + W_k g_k$. Or $g_k = \begin{pmatrix} g_{k-1} \\ \mu_k^{(k)} \end{pmatrix}$ (cf 2.2.2 (2.15) et (2.16)), donc

$$\begin{aligned} x_k &= x_0 + (W_{k-1}, w_k) \begin{pmatrix} g_{k-1} \\ \mu_k^{(k)} \end{pmatrix} \\ &= x_0 + W_{k-1} g_{k-1} + \mu_k^{(k)} w_k. \end{aligned}$$

Cette dernière relation nécessite le calcul récursif du vecteur w_k. Par la relation (2.18) nous avons $B_k = W_k R_k$, il vient alors que :

$$\begin{aligned} b_k &= W_k(0, \ldots, 0, h_{k-l,k}^{(k)}, \ldots, h_{k,k}^{(k)})^T \\ &= \sum_{i=k-l}^{k-1} h_{i,k}^{(k)} w_i + h_{k,k}^{(k)} w_k, \end{aligned}$$

et finalement nous obtenons :

$$w_k = (b_k - \sum_{i=k-l}^{k-1} h_{i,k}^{(k)} w_i)/h_{k,k}^{(k)}. \tag{2.19}$$

2.4 Cas particuliers

2.4.1 GMRES

Considérons le cas particulier de la méthode MRSe où le processus d'Arnoldi "avec orthonormalisation" est utilisé pour construire la base B_{m+1}. Comment se transforme alors la matrice Z_m ainsi que la semi-norme $|.|_{Z_m}$ définie sur le sous espace de Krylov $K_{m+1}(r_0, A)$? Rappelons que $Z_m = (B_{m+1}^L)^T B_{m+1}^L$ et que la matrice B_{m+1} est orthonormale.

Lemme 1

Si la base B_m est construite par le processus d'Arnoldi avec orthonormalisation alors :

1. *La matrice Z_m qui définit la semi-norme $|.|_{Z_m}$ est telle que :*

$$Z_m = B_{m+1} B_{m+1}^T, \tag{2.20}$$

2. $\forall\ x\ ,\ y\ \in K_{m+1}(r_0, A) \quad (x,y)_{Z_m} = (x,y)_2,$
en particulier $\forall\ x\ \in K_{m+1}(r_0, A) \quad |x|_{Z_m} = \|x\|_2.$

Preuve :

1. La matrice B_{m+1} étant orthonormale, nous avons $B_{m+1}^T B_{m+1} = I_{m+1}$, et donc $B_{m+1}^L = B_{m+1}^T$.
2. Ecrivons y dans la base B_{m+1} du sous espace $K_{m+1}(r_0, A)$; il existe alors $t \in \mathbb{R}^{m+1}$ tel que $y = B_{m+1}t$.
   Ainsi le Z_m-produit scalaire de x et y est :

$$(x,y)_{Z_m} = x^T Z_m y = x^T Z_m B_{m+1} t$$

   or $B_{m+1}^L = B_{m+1}^T$, ce qui implique que :

$$(x,y)_{Z_m} = x^T B_{m+1} B_{m+1}^T B_{m+1} t = x^T B_{m+1} t = x^T y = (x,y)_2.$$

■

Ainsi ce résultat montre que la condition de semi-minimisation (2.3) n'est autre que la condition de minimisation du résidu qui définit la méthode GMRES à savoir :

$$r_m^{\text{GMRES}} = \min_{x \in K_m(r_0,A)} \|f - Ax\|_2.$$

Il découle de ce qui précède que la méthode GMRES, due à Saad et Schultz [42], est un cas particulier de la méthode MRSe Cette méthode minimise de façon optimale le résidu sur $K_{m+1}(r_0, A)$, elle est mathématiquement équivalente aux méthodes GCR (Generalized Conjugate residual) [18, 55] ainsi qu'a la méthode ORTHODIR [31]

Pour plus de résultats concernant la convergence et l'implémentation de la méthode GMRES, on pourra consulter [42, 52, 53, 50, 59]. Une comparaison des méthodes FOM et GMRES est faite en [8, 15]. Notamment, la relation entre le breakdown de la méthode FOM (dû à la singularité de la matrice de Hessenberg H_k) et la stagnation du GMRES est clairement expliquée dans [8]. De plus, on montre que si la méthode GMRES converge rapidement, alors il en est de même pour FOM et dans ce cas les deux méthodes ont le même comportement. Par contre, lorsque la méthode GMRES stagne ou converge très lentement, alors les normes des résidus da la méthode FOM sont relativement grandes [8, 15]. Nous retrouverons ces résultats dans un cadre plus générale au cours du chapitre 3. Pour une analyse de la convergence des deux méthodes, on pourra aussi consulter le récent travail de Sadok [47].

Nous avons vu que lorsque le nombre d'itérations augmente dans la méthode MRSe, le coût algorithmique devient rapidement important. C'est le cas notamment de la méthode GMRES, où le coût opératoire augmente linéairement avec le nombre d'itérations. On préfére dans ce cas redémarrer l'algorithme aprés un nombre maximal m d'itérations. Cependant, cette stratégie n'est pas toujours efficace (voir les exemples : 4.3, 5.2, 6.1 et 7.3 du paragraphe 2.5.2). Une solution à ce problème est d'associer à la méthode une technique de préconditionnement (voir section 2.5). Beaucoup de travaux ont été fait dans ce sens. Notamment, deux récentes méthodes se caractérisent par le fait qu'elles permettent d'utiliser des préconditionneurs qui peuvent varier à chaque itération de la méthode. Ces deux méthodes sont la méthode FGMRES due à Saad [45] et la famille des méthodes GMRESR due à Van Der Vorst et à Vuik [54]. Dans FGMRES, les vecteurs de direction sont préconditionnés, alors que dans GMRESR, les résidus sont préconditionnés. Notons aussi que la méthode FGMRES peut être sujette au problème du breakdown, alors que ce n'est pas le cas de la méthode GMRESR. Pour une étude plus détaillée de ces deux méthodes on pourra consulter le travail de Vuik [51].

Rappelons que les matrices B_{k+1} et $\mathcal{Q}_k$ sont orthogonales, ainsi par utilisation de la relation (2.12) nous obtenons le résultat suivant :

Lemme 2

A chaque itération k de la méthode GMRES, *nous avons :*

$$\|r_k^{\text{GMRES}}\|_2 = |\mu_{k+1}^{(k)}|, \tag{2.21}$$

avec $\mu_{k+1}^{(k)} = \left(\mathcal{Q}_k(\beta e_1^{(k+1)})\right)_{k+1}$ *et* $\beta = \|r_0\|_2$.

Le résultat suivant donne une idée sur la "distance" qui sépare le résidu de la méthode MRSe de celui de la méthode GMRES.

Théorème 17

$$\|r_k^{\text{MRSe}}\|_2 \leq \kappa(B_{k+1})\|r_k^{\text{GMRES}}\|_2,$$

avec $\kappa(B_{k+1}) = \|B_{k+1}\|_2\|B_{k+1}^+\|_2$.

Preuve :

Chaque résidu r_k est dans $K_{k+1}(r_0, A)$, d'où $r_k = B_{k+1}s_k$ avec $s_k \in \mathbb{R}^{k+1}$. Ainsi

$$\|r_k^{\text{MRSe}}\|_2 \leq \|B_{k+1}\|_2\|s_k^{\text{MRSe}}\|_2, \tag{2.22}$$

or par définition du résidu r_k^{MRSe} et de la semi-norme associée à Z_k nous avons :

$$|r_k^{\text{MRSe}}|_{Z_k} = \min_{r \in K_{k+1}(r_0,A)} |r|_{Z_k} = \min_{r \in K_{k+1}(r_0,A)} \|B_{k+1}^L r\|_2,$$

et comme $s_k = B_{k+1}^L r_k$, il est facile de voir que :

$$\|s_k^{\text{MRSe}}\|_2 = |r_k^{\text{MRSe}}|_{Z_k} = \min_{s \in \mathbb{R}^{k+1}} \|s\|_2$$

et en particulier

$$\begin{aligned} \|s_k^{\text{MRSe}}\|_2 &\leq \|s_k^{\text{GMRES}}\|_2 \\ &= \|B_{k+1}^+ B_{k+1} s_k^{\text{GMRES}}\|_2 \\ &\leq \|B_{k+1}^+\|_2 \|B_{k+1} s_k^{\text{GMRES}}\|_2 \\ &= \|B_{k+1}^+\|_2 \|r_k^{\text{GMRES}}\|_2, \end{aligned}$$

en utilisant (2.22), nous obtenons

$$\|r_k^{\text{MRSe}}\|_2 \leq \|B_{k+1}\|_2 \|B_{k+1}^+\|_2 \|r_k^{\text{GMRES}}\|_2 = \kappa(B_{k+1}) \|r_k^{\text{GMRES}}\|_2.$$

■

2.4.2 QMR

Souvent la convergence de le méthode BCG est irrégulière. De plus la décomposition $\mathcal{LU}$ du système tridiagonal qui lui est associé peut ne pas exister ; cela a pour conséquence un possible "breakdown" de la méthode (autre que celui du processus de Lanczos). La méthode QMR due à Freund et Nachtigal [21] tente de surmonter ces problèmes. L'idée principale de la méthode est de résoudre le système aux moindres carrés $(\mathcal{M}_f)$ où $\widetilde{H}_m$ est la matrice tridiagonale construite par le processus de Lanczos. De plus une stratégie de "look-ahead" [23, 35, 38] peut être utilisée en vue d'éviter les "breakdown" associés au processus de Lanczos. Il existe des relations entre les deux méthodes BCG et QMR, ces relations ont été établies dans [35, 62, 15]. Notamment, dans [15], Cullum et Greenbaum expliquent l'interdépendance qui existe entre les résidus de la méthode BCG et les pseudo-résidus de la méthode QMR. Nous expliciterons cette interdépendance par une corrélation entre la Z_k-semi-norme des résidus des deux méthodes dans le prochain chapitre. Notons que pour des raisons de simplicité, nous considérons dans la suite la méthode QMR sans stratégie de "look-ahead".

Il est clair que la méthode QMR est un cas particulier de la méthode MRSe, elle peut être aussi considérée comme une méthode qui minimise le résidu dans une norme

qui depend de l'itération. Le lemme 3 ci dessous nous permet d'obtenir l'expression de cette norme. Soit B_{m+1} et $\widehat{B}_{m+1}$ les matrices construites par le processus de Lanczos (sans stratégie de "look-ahead"). En utilisant les relations (1.23) et (1.25) nous avons le résultat suivant :

Lemme 3

La matrice Z_m qui définit la semi-norme $|.|_{Z_m}$ est telle que :

$$Z_m = \widehat{B}_{m+1} D_{m+1}^{-T} D_{m+1}^{-1} \widehat{B}_{m+1}^T, \tag{2.23}$$

avec $D_{m+1} = diag(\eta_1, \ldots, \eta_{m+1})$ et $\eta_j = (\widehat{b}_j, b_j)$.

En ce qui concerne le critère d'arrêt, si nous ne voulons pas utiliser la relation (2.17) qui donne le résidu exact, et que à la place nous voulions utiliser la relation (2.13), alors nous avons le résultat suivant qui peut être utilisé comme critère d'arrêt dans l'implémentation de la méthode QMR.

Lemme 4

Si à chaque itération du processus de Lanczos, nous avons $\|b_{k+1}\|_2 = 1$ (i.e. nous prenons $h_{k+1,k} = \|u\|_2$ dans l'algorithme 4), alors à chaque itération k de la méthode QMR, *nous avons :*

$$\|r_k^{\text{QMR}}\|_2 \leq \sqrt{k+1}|\mu_{k+1}^{(k)}|, \tag{2.24}$$

avec $\mu_{k+1}^{(k)} = \left(\mathcal{Q}_k(\beta e_1^{(k+1)})\right)_{k+1}$ et $\beta = \|r_0\|_2$.

Preuve :

D'après la relation (2.12) nous avons $r_k^{\text{QMR}} = \mu_{k+1}^{(k)} B_{k+1} d$ où $d = \mathcal{Q}_k^T e_{k+1}^{(k+1)}$, et donc

$$\begin{aligned}
\|r_k^{\text{QMR}}\|_2 &= |\mu_{k+1}^{(k)}| \, \|B_{k+1} d\|_2 \\
&= |\mu_{k+1}^{(k)}| \, \left\| \sum_{i=1}^{k+1} (d)_i b_i \right\|_2 \\
&\leq |\mu_{k+1}^{(k)}| \sum_{i=1}^{k+1} \|b_i\|_2 |(d)_i| \\
&= |\mu_{k+1}^{(k)}| \|d\|_1.
\end{aligned}$$

Or $\|d\|_1 \leq \sqrt{k+1}\|d\|_2$ et comme $\|d\|_2 = 1$, nous avons : $\|r_k^{\text{QMR}}\|_2 \leq |\mu_{k+1}^{(k)}|\sqrt{k+1}$.

■

Nous terminons ce paragraphe par le résultat suivant qui donne une borne supérieure de la distance qui sépare le résidu de la méthode QMR de celui de la méthode GMRES. Ce résultat est un cas particulier du théorème 17.

Théorème 18

$$\|r_k^{\mathrm{QMR}}\|_2 \leq \kappa(B_{k+1})\|r_k^{\mathrm{GMRES}}\|_2,$$

avec $\kappa(B_{k+1}) = \|B_{k+1}\|_2\|B_{k+1}^+\|_2$.

2.4.3 CMRH

La méthode CMRH due à Sadok [46] est l'une des plus récentes méthodes qui utilise l'idée de "quasi-minimisation" du résidu. Elle est obtenue comme cas particulier de la méthode MRSe par utilisation du processus de Hessenberg avec stratégie de pivotage. Comme pour le QMR, la méthode CMRH peut être considérée comme une méthode qui minimise le résidu dans une norme qui change à chaque itération.

Soit B_{m+1} la matrice construite par le processus de Hessenberg (sans stratégie de pivotage pour plus de simplicité); considérons alors le partitionnement suivant :

$$B_{m+1} = \begin{pmatrix} B_{m+1}^1 \\ B_{m+1}^2 \end{pmatrix}$$

où B_{m+1}^1 est la matrice formée des $m+1$ premières lignes de B_{m+1}. Nous pouvons alors donner le résultat suivant :

Lemme 5

La matrice Z_m *qui définit la semi-norme* $|.|_{Z_m}$ *est telle que :*

$$Z_m = \begin{pmatrix} (B_{m+1}^1)^{-T}(B_{m+1}^1)^{-1} & 0 \\ 0 & 0 \end{pmatrix}. \tag{2.25}$$

Preuve :

En partitionnant la matrice Y_{m+1}^T sous la forme :

$$Y_{m+1}^T = (I_{m+1}, 0),$$

nous avons :

$$B_{m+1}^L = (Y_{m+1}^T B_{m+1})^{-1} Y_{m+1}^T = \left((B_{m+1}^1)^{-1}, 0\right), \tag{2.26}$$

et en effectuant le produit $\left((B^1_{m+1})^{-1},0\right)^T \left((B^1_{m+1})^{-1},0\right)$, nous obtenons l'expression de la matrice Z_m.

■

Cette expression de la matrice Z_m nous permet de donner le résultat suivant :

Lemme 6

Soit $x = (x_1,\ldots,x_n)^T$, $y = (y_1,\ldots,y_n)^T$ *deux vecteurs de* $\mathbb{R}^n$, *soit* $i \leq l$. *Désignons par* $(x)_{i:l}$, *respectivement* $(y)_{i:l}$ *le vecteur* $(x_i, x_{i+1},\ldots,x_l)^T$, *respectivement* $(y_i, y_{i+1},\ldots,y_l)^T$, *alors*

$$(x,y)_{Z_m} = ((x)_{1:m+1},(y)_{1:m+1})_{Z^1_m},$$

avec $Z^1_m = (B^1_{m+1})^{-T}(B^1_{m+1})^{-1}$.

Preuve :

Ecrivons x et y sous la forme :

$$x = \begin{pmatrix} (x)_{1:m+1} \\ (x)_{m+2:n} \end{pmatrix} \quad \text{et} \quad y = \begin{pmatrix} (y)_{1:m+1} \\ (y)_{m+2:n} \end{pmatrix}.$$

D'après l'expression de Z_m, nous avons :

$$(x,y)_{Z_m} = x^T Z_m y = ((x)_{1:m+1},(x)_{m+2:n})^T \begin{pmatrix} (B^1_{m+1})^{-T}(B^1_{m+1})^{-1} & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} (y)_{1:m+1} \\ (y)_{m+2:n} \end{pmatrix}.$$

En notant $Z^1_m = (B^1_{m+1})^{-T}(B^1_{m+1})^{-1}$, nous obtenons :

$$(x,y)_{Z_m} = (x)_{1:m+1}(B^1_{m+1})^{-T}(B^1_{m+1})^{-1}(y)_{1:m+1} = ((x)_{1:m+1},(y)_{1:m+1})_{Z^1_m}.$$

■

De même que pour la méthode QMR, nous avons les deux résultats suivants (lemme 7 et théorème 19). Le premier peut être utilisé comme critère d'arrêt dans l'implémentation de l'algorithme du CMRH. Le second donne une borne supérieure pour le rapport des résidus des méthodes CMRH et GMRES.

Lemme 7

A chaque itération k *de la méthode* CMRH, *nous avons :*

$$\|r_k^{\text{CMRH}}\|_2 \leq \sqrt{(n-k/2)(k+1)}\,|\mu^{(k)}_{k+1}|, \tag{2.27}$$

avec $\mu^{(k)}_{k+1} = \left(\mathcal{Q}_k(\beta e_1^{(k+1)})\right)_{k+1}$ *et* $\beta = \|r_0\|_\infty$.

Preuve :

Il suffit de montrer que $\|B_{k+1}\|_2 \leq \sqrt{(n-k/2)(k+1)}$, et en utilisant la relation (2.13) nous obtenons le résultat voulu.

Soit $d \in \mathbb{R}^{k+1}$, en calculant $B_{k+1}d$ nous avons :

$$\|B_{k+1}d\|_2 \leq \sum_{i=1}^{k+1} \|b_i\|_2 |(d)_i|.$$

Or chaque vecteur b_i construit par le processus de Hessenberg admet $i-1$ composantes nulles et $\|b_i\|_\infty = 1$, donc $\|b_i\|_2 \leq \sqrt{n-i+1}\|b_i\|_\infty = \sqrt{n-i+1}$. D'où

$$\|B_{k+1}d\|_2 \leq \sum_{i=1}^{k+1} \sqrt{n-i+1}|(d)_i|$$

et par l'inégalité de Cauchy-Schwarz :

$$\|B_{k+1}d\|_2 \leq \|u\|_2 \left\| \begin{pmatrix} \sqrt{n} \\ \sqrt{n-1} \\ \vdots \\ \sqrt{n-k} \end{pmatrix} \right\|_2 .$$

Finalement $\|B_{k+1}\|_2 = \max_{d \in \mathbb{R}^{k+1}} \dfrac{\|B_{k+1}d\|_2}{\|d\|_2} \leq \sqrt{\sum_{i=0}^{k} n-i} = \sqrt{(n-k/2)(k+1)}.$

■

Le résultat suivant est un cas particulier du théorème 17.

Théorème 19

$$\|r_k^{\text{CMRH}}\|_2 \leq \kappa(B_{k+1})\|r_k^{\text{GMRES}}\|_2,$$

avec $\kappa(B_{k+1}) = \|B_{k+1}\|_2\|B_{k+1}^+\|_2$.

2.4.4 DQGMRES(l)

La méthode DQGMRES a été décrite par Saad et Wu [43] et est basée sur la procédure d'Orthogonalisation Incomplète décrite en §1.3.2. Les itérés de la méthode DQGMRES son déterminés en résolvant le problème aux moindres carrés ($\mathcal{M}_f$) et sont obtenus par un schéma de type gradient conjugué afin d'optimiser le coût en place

mémoire. Il est donc clair que la méthode DQGMRES(l) est un cas particulier de la méthode DTMRSe(l).

Dans les précédents cas particuliers, nous avons à chaque fois donné une expression simple de la matrice Z_m ainsi que des résultats sur la semi-norme qui lui est associée en fonction de B_{m+1}^L, (inverse à gauche de la matrice B_{m+1} comme il a été défini par (1.8)). Malheureusement, il n'est pas possible de faire de même pour la méthode DQGMRES(l) puisque nous n'obtenons pas une expression simple de la matrice B_{m+1}^L. Cependant, il est à noter que nous avons les résultats suivants :

Lemme 8

A chaque itération k de la méthode DQGMRES, *nous avons :*

$$\|r_k^{\text{DQGMRES}}\|_2 \leq \sqrt{k+1}|\mu_{k+1}^{(k)}|, \tag{2.28}$$

avec $\mu_{k+1}^{(k)} = \left(\mathcal{Q}_k(\beta e_1^{(k+1)})\right)_{k+1}$ *et* $\beta = \|r_0\|_2$.

Preuve :

La démonstration de ce lemme est semblable à celle du lemme 4.

■

Théorème 20

$$\|r_k^{\text{DQGMRES}}\|_2 \leq \kappa(B_{k+1})\|r_k^{\text{GMRES}}\|_2,$$

avec $\kappa(B_{k+1}) = \|B_{k+1}\|_2\|B_{k+1}^+\|_2$.

2.4.5 DTCMRH(l)

En remplaçant dans l'algorithme de la méthode DQGMRES(l) [43] la procédure d'Orthogonalisation Incomplète par le processus de Hessenberg tronqué avec stratégie de pivotage (décrit par l'algorithme 7), nous obtenons une nouvelle méthode que nous désignons par DTCMRH(l) (Direct truncated CMRH method). Comme pour la précédente méthode DQGMRES(l), nous ne pouvons exprimer de façon simple la matrice B_{m+1}^L qui caractérise la semi-norme associée à cette méthode.

En ce qui concerne le critère d'arrêt nous avons le résultat suivant :

Lemme 9

A chaque itération k de la méthode DTCMRH(l), *nous avons :*

$$\|r_k^{\text{DTCMRH(l)}}\|_2 \leq \sqrt{(n-k/2)(k+1)}|\mu_{k+1}^{(k)}|, \quad pour \quad k \leq l, \tag{2.29}$$

$$\|r_k^{\text{DTCMRH(l)}}\|_2 \leq \sqrt{(n-l)(k+1)}|\mu_{k+1}^{(k)}|, \quad pour \quad k > l, \tag{2.30}$$

avec $\mu_{k+1}^{(k)} = \left(\mathcal{Q}_k(\beta e_1^{(k+1)})\right)_{k+1}$ *et* $\beta = \|r_0\|_\infty$.

Preuve :

La démonstration de ce lemme est semblable à celle du lemme 7.

■

Le résultat du théorème 17 s'applique aussi à la méthode DTCMRH(l).

Théorème 21

$$\|r_k^{\text{DTCMRH}}\|_2 \leq \kappa(B_{k+1})\|r_k^{\text{GMRES}}\|_2,$$

avec $\kappa(B_{k+1}) = \|B_{k+1}\|_2\|B_{k+1}^+\|_2$.

2.5 Comparaison des méthodes MRSe

Dans ce paragraphe, nous donnons quelques résultats numériques qui nous permettront de comparer les performances des méthodes faisant partie de la méthode MRSe.

La première partie sera consacrée à la comparaison des méthodes GMRES, QMR et CMRH (non redémarrés et non tronqués). La seconde partie est consacrée à la comparaison des versions redémarrées, i.e. des algorithmes GMRES(m) et CMRH(m) avec et sans preconditionnement. Finalement la comparaison des méthodes DQGMRES et DTCMRH sera faite dans la troisième partie.

Avant de comparer les différentes méthodes, nous avons besoin de rappeler brièvement la technique de préconditionnement utilisée dans les paragraphes 2.5.2 et 2.5.3.

Souvent, lorsque nous avons à résoudre des systèmes linéaires de grande taille, nous devons combiner la méthode itérative utilisée avec une technique de préconditionnement. Cela dans le but d'accélerer la convergence de cette méthode.

Le système à résoudre étant $Ax = f$, soit $M = M_g M_d$ une matrice $n \times n$ inversible qui approche, dans un certain sens, la matrice A du système. Au lieu de résoudre le système original, nous appliquons la méthode itérative au système linéaire équivalent.

$$A'x' = f'$$

avec $A' = M_g^{-1} A M_d^{-1}$, $f' = M_g^{-1} f$ et $x' = M_d x$.

Le préconditionnement ainsi décrit dépend du choix des matrices M_g et M_d. Si nous prenons $M_g = I_n^{(n)}$ respectivement $M_d = I_n^{(n)}$, le préconditionnement est dit à droite, respectivement à gauche de A. Rappelons que Van der Vorst et Vuik [54, 51], ainsi que Saad [45] one développé des méthodes qui permettent de varier les préconditionneurs à chaque itération de la méthode GMRES. Cependant dans les exemples à venir, nous n'avons considéré que le préconditionnement à droite par les factorisations *ILU0* et *MILU0*. Ces préconditionneurs sont basés sur la décomposition de A par la méthode $\mathcal{LU}$.

En général lorsqu'on applique la décomposition exacte $\mathcal{LU}$ à une matrice creuse, des éléments non nuls peuvent apparaitre dans des positions précédemment occupées par des zéros. Soit S un certain ensemble de positions (i, j), $1 \leq i, j \leq n$. L'idée de la factorisation incomplète est de laisser tous les éléments, qui sont en dehors de S, nuls durant la décomposition de A. En général, l'ensemble S contient toutes les positions (i, j) vérifiant $a_{i,j} \neq 0$ plus quelques autres positions particulières. En variant S, nous obtenons différentes sortes de factorisations $\mathcal{ILU}$ de A. Pour plus de renseignements concernant le préconditionnement par la factorisation incomplète $\mathcal{LU}$, on peut consulter [25, 34] ainsi que les références se trouvant dans [4, 12].

2.5.1 Algorithmes "complets"

Afin de comparer les trois méthodes GMRES, QMR et CMRH, nous avons utilisé l'implémentation de l'algorithme GMRES comme il a été décrit en [42], l'algorithme 7.1 (QMR basé sur la double récurrence et sans stratégie de "look-ahead") donné en [22] et l'implémentation de l'algorithme CMRH comme décrit en [46].

Tous les tests du présent paragraphe ont été réalisés à l'aide du logiciel MATLAB sur un "Sun SparcStation10" dont la précision machine est égale à $2.22\ 10^{-6}$.

Le système à résoudre est $Ax = f$, le second membre f est généré aléatoirement à l'aide de la fonction MATLAB *rand*. Le point de départ est $x_0 = (0, \cdots, 0)^T$. Notons que pour l'algorithme QMR, le vecteur y est lui aussi généré aléatoirement. Comme critère d'arrêt, nous utilisons l'inégalité $\|r_k\|_2 / \|f\|_2 \leq 10^{-9}$.

Les courbes représentent la norme du résidu $\|f - Ax_k\|_2$ dans une échelle logarithmique en fonction du nombre d'itérations k. La courbe de la méthode GMRES est représentée par des tirets, celle du QMR par des pointillés et celle du CMRH par la ligne continue. Après chaque courbe, un tableau résume les résultats obtenus pour chaque méthode. Ce tableau donne l'itération finale, la norme du résidu final ainsi que le côut opératoire nécessité par chaque algorithme. Le côut opératoire est donné par la fonction MATLAB *flops*.

Exemple 1.

Cet exemple a été donné dans [53], où on considère la matrice $n \times n$ $A = SBS^{-1}$, avec

$$S = \begin{pmatrix} 1 & \beta & & & \\ & 1 & \beta & 0 & \\ & & \ddots & \ddots & \\ & 0 & & \ddots & \beta \\ & & & & 1 \end{pmatrix} \quad \text{et} \quad B = \begin{pmatrix} 1 & & & \\ & 1+\alpha & & 0 \\ & & 3 & \\ & 0 & & \ddots \\ & & & & n \end{pmatrix}$$

La variable β permet de contrôler le conditionnement de la matrice S et permet donc de varier la normalité de A (voir [53]). Nous prenons $n = 150$ et considérons les trois choix suivants :

• **Exemple 1.1 :** $\beta = 0.9$, $\alpha = 1$. Pour ce choix la matrice a pour conditionnement $\kappa(A) \approx 227$. Les courbes sont celles données par la figure 1.1. Les résultats sont donnés par le tableau 1.1.

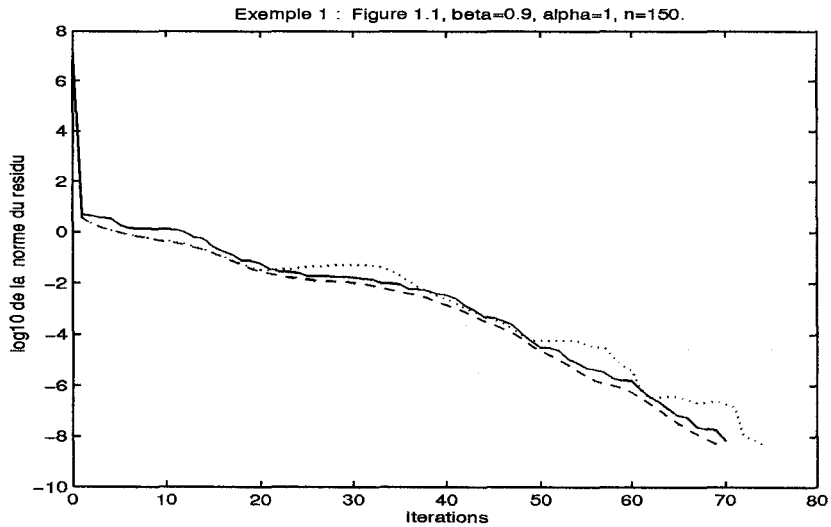


Tableau 1.1 : Résultats de l'exemple 1.1.

	GMRES	QMR	CMRH
itération	69	74	70
norme	$4.88\ 10^{-9}$	$4.79\ 10^{-9}$	$6.76\ 10^{-9}$
coût	$8.6\ 10^{6}$	$1.\ 10^{7}$	$7.9\ 10^{6}$

Les courbes des trois méthodes sont assez voisines et les résultats sont similaires. Les méthodes GMRES et CMRH ont une légère supériorité sur la méthode QMR.

• **Exemple 1.2 :** $\beta = 1.1$, $\alpha = 1$. Nous obtenons $\kappa(A) \approx 1.58\ 10^{13}$, la figure 1.2 ainsi que les résultats donnés par le tableau 1.2.

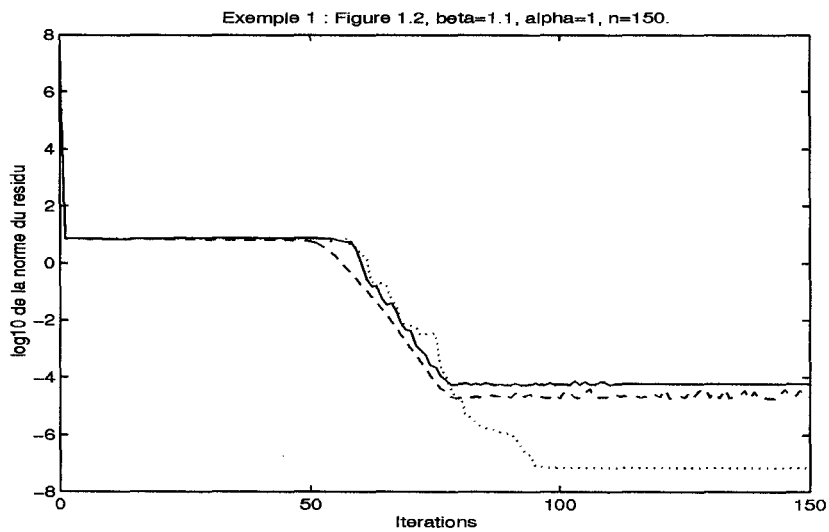


Tableau 1.2 : Résultats de l'exemple 1.2.

	GMRES	QMR	CMRH
itération	150	150	150
norme	$2.13\ 10^{-5}$	$7.25\ 10^{-8}$	$5.99\ 10^{-5}$
coût	$2.5\ 10^{7}$	$2.1\ 10^{7}$	$2.0\ 10^{7}$

La méthode QMR donne des résultats meilleurs que GMRES et CMRH, les courbes de ces deux méthodes ont la même allure.

• **Exemple 1.3 :** $\beta = 1.1$, $\alpha = 0$. Dans ce cas $\kappa(A) \approx 2.05\ 10^{13}$. La figure 1.3 décrit les courbes obtenues et le tableau 1.3 donne les résultats associés à ce choix.

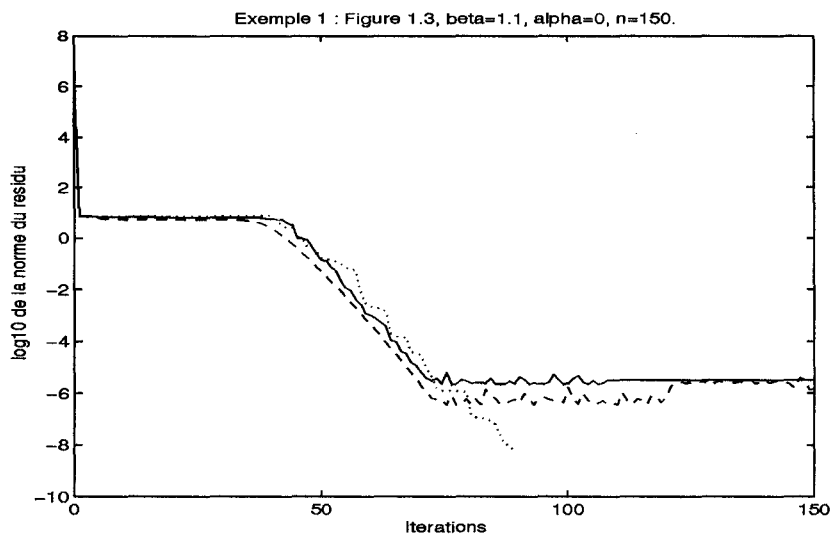


Tableau 1.3 : Résultats de l'exemple 1.3.

	GMRES	QMR	CMRH
itération	150	89	150
norme	$1.51\ 10^{-6}$	$5.4\ 10^{-9}$	$3.29\ 10^{-6}$
coût	$2.5\ 10^{7}$	$1.3\ 10^{7}$	$2.\ 10^{7}$

Comme pour le choix précédent, QMR est plus performant, les méthodes GMRES et CMRH n'arrivent pas à atteindre la précision voulue.

Exemple 2.

Les exemples suivants sont donnés afin d'illustrer la stagnation de la classe des méthodes MRSe. Nous considérons les deux exemples suivants :

• **Exemple 2.1 :** Cet exemple a été donné par Brown [8] afin d'illustrer la stagnation

du GMRES, on prend $n = 40$ et on considère la matrice $n \times n$ suivante :

$$A = \begin{pmatrix} \epsilon & 1 & & & \\ -1 & \epsilon & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & \epsilon & 1 \\ & & & -1 & \epsilon \end{pmatrix}$$

avec $\epsilon = 0.1$. Le conditionnement de cette matrice est $\kappa(A) \approx 16$. Nous obtenons la figure 2.1 ainsi que les résultats donnés dans le tableau 2.1.

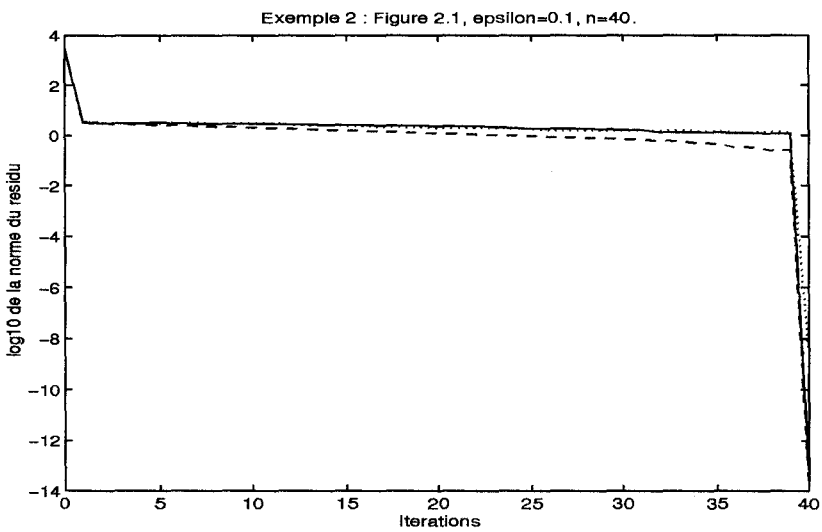


Tableau 2.1 : Résultats de l'exemple 2.1.

	GMRES	QMR	CMRH
itération	40	40	40
norme	$1.22\ 10^{-14}$	$2.68\ 10^{-9}$	$4.26\ 10^{-14}$
coût	$5.0\ 10^{5}$	$4.4\ 10^{5}$	$4.1\ 10^{5}$

Le tracé des courbes permet de voir que les trois méthodes stagnent et n'arrivent à obtenir la solution qu'à la dernière itération. La performance de la méthode QMR est la moins bonne.

• **Exemple 2.2 :** La matrice de cet exemple est une matrice Toeplitz construite à l'aide de la fonction MATLAB *toeplitz(c,d)* où c et d sont deux vecteurs de $\mathbb{R}^n$

générés aléatoirement. La taille de la matrice est $n = 50$, son conditionnement est $\kappa(A) \approx 9.17\ 10^2$. Les courbes, respectivement les résultats de cet exemple sont décrits par la figure 2.2, respectivement le tableau 2.2. Nous obtenons les résultats suivants :

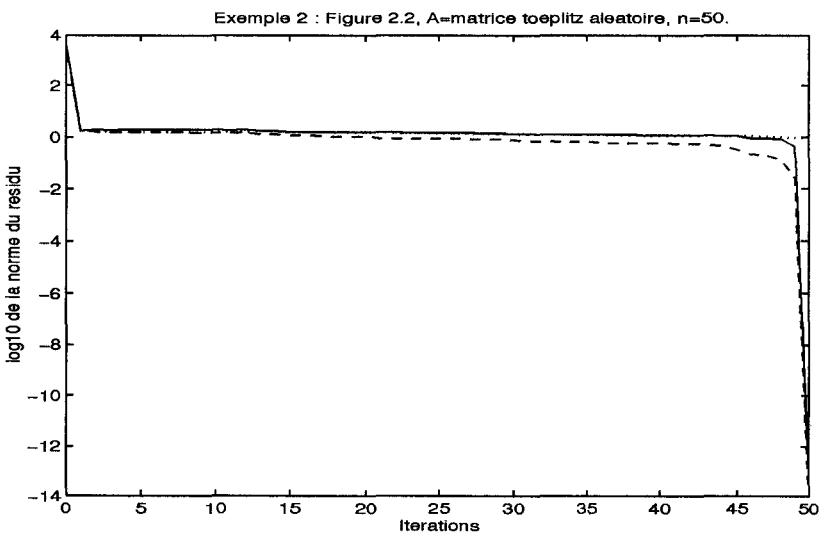


Tableau 2.2 : Résultats de l'exemple 2.2.

	GMRES	QMR	CMRH
itération	50	50	50
norme	$1.11\ 10^{-14}$	$9.41\ 10^{-1}$	$7.35\ 10^{-14}$
coût	$9.7\ 10^5$	$8.4\ 10^5$	$7.9\ 10^5$

La méthode QMR n'arrive pas à converger, les deux autres méthodes présentent des courbes similaires et voisines.

Notons que dans le cas ou $m < 40$ les méthodes GMRES(m), QMR(m) et CMRH(m) ne convergent pas. Ces derniers exemples montrent donc que la technique du redémarrage n'est pas toujours efficace.

Exemple 3.

Cet exemple a été donné dans [60], la matrice A est symétrique définie positive. C'est une matrice creuse $n \times n$ aléatoire construite par éléments finis sur un quadrillage NX par NY. Le programme qui permet de générer une telle matrice a été recupéré de

mltemplates par "anonymous ftp " (ftp netlib2.cs.utk.edu). Trois différents maillages sont considérés :

• **Exemple 3.1 :** $NX = 8$, $NY = 7$. La matrice obtenue est de taille $n = 199$, elle a pour conditionnement $\kappa(A) \approx 75$. Les courbes et les résultats sont donnés par la figure 3.1 et le tableau 3.1.

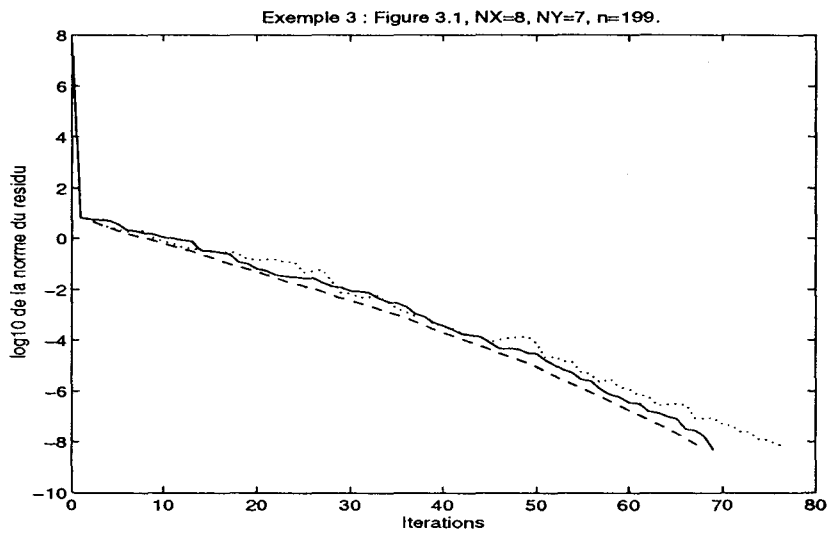


Tableau 3.1 : Résultats de l'exemple 3.1.

	GMRES	QMR	CMRH
itération	68	77	69
norme	$4.93\ 10^{-9}$	$7.07\ 10^{-9}$	$5.00\ 10^{-9}$
coût	$1.4\ 10^{7}$	$1.9\ 10^{7}$	$1.3\ 10^{7}$

• **Exemple 3.2 :** $NX = 6$, $NY = 6$. La taille de la matrice est $n = 133$, le conditionnement de A est $\kappa(A) \approx 111$. Nous obtenons la figure 3.2, ainsi que le tableau des résultats 3.2.

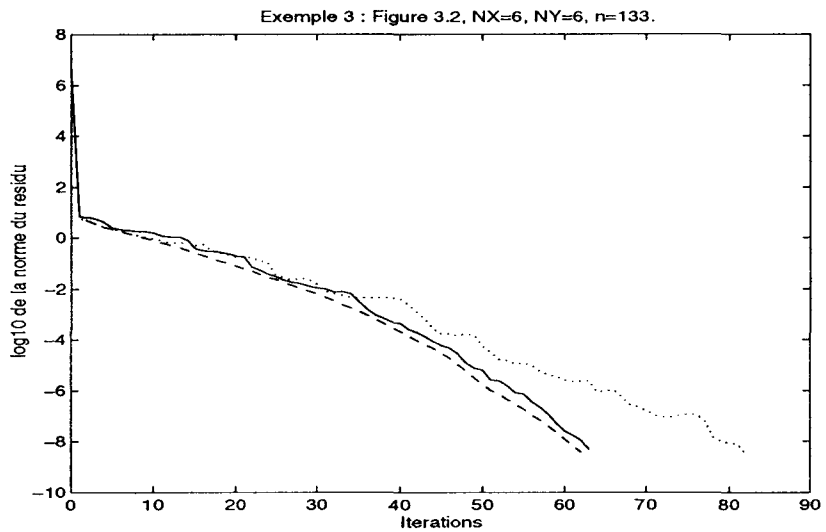


Tableau 3.2 : Résultats de l'exemple 3.2.

	GMRES	QMR	CMRH
itération	62	82	63
norme	$3.73\ 10^{-9}$	$3.36\ 10^{-9}$	$4.70\ 10^{-9}$
coût	$6.2\ 10^{6}$	$9.1\ 10^{6}$	$5.7\ 10^{6}$

• **Exemple 3.3 :** $NX = 10$, $NY = 4$. La taille du système est $n = 149$. La matrice a pour conditionnement $\kappa(A) \approx 544$. Nous obtenons les courbes de la figure 3.3 et les résultats du tableau 3.3.

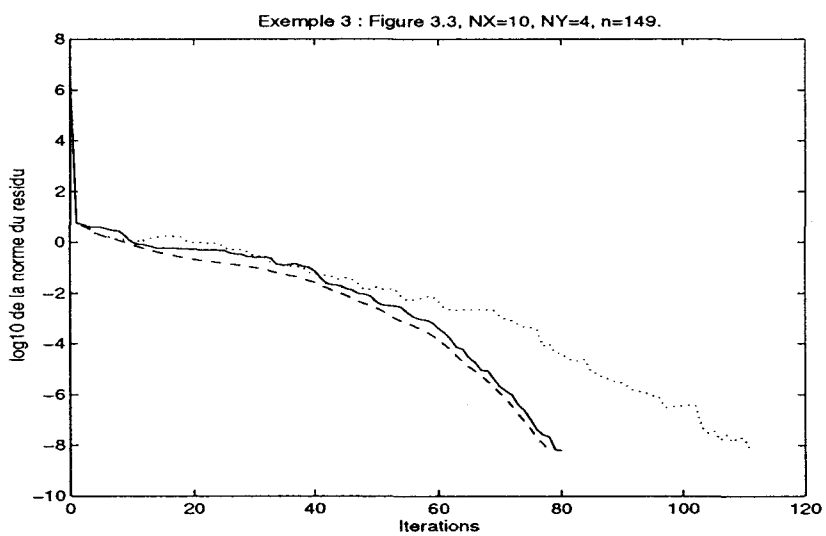


Tableau 3.3 : Résultats de l'exemple 3.3.

	GMRES	QMR	CMRH
itération	78	111	80
norme	$6.12\ 10^{-9}$	$6.49\ 10^{-9}$	$6.27\ 10^{-9}$
coût	$1.0\ 10^{7}$	$1.5\ 10^{7}$	$9.2\ 10^{6}$

Pour les trois choix, les méthodes GMRES et CMRH donnent de meilleurs résultats que la méthode QMR. L'allure de la courbe GMRES est similaire à celle de la courbe CMRH.

Les exemples donnés dans cette partie, pour la plupart, montrent que la méthode GMRES converge plus rapidement que les deux autres méthodes. La rapidité du GMRES par rapport au QMR est plus visible que celle par rapport au CMRH (en général la convergence de la méthode CMRH est obtenue, au maximum, en deux itérations de plus que celle de la méthode GMRES). Le coût opératoire est à l'avantage du CMRH, ceci est dû à la nature de la base construite par le processus de Hessenberg et à l'absence de produit scalaire dans ce même processus.

2.5.2 Algorithmes redémarrés

Dans cette section, nous comparons les résultats des méthodes GMRES(m) et CMRH(m). L'implémentation des algorithmes est celle décrite dans [42] et [46]. Les tests ont étés réalisés sur une machine "Sun SparcStation20" dont la précision machine est égale à $2.22\ 10^{-16}$, à l'aide du Compilateur FORTRAN : f77 avec l'option "-fast" .

Les matrices testées sont obtenues par discrétisation d'équations aux dérivées partielles à l'aide des différences finies. Pour cela on utilise une discrétisation uniforme contenant $(nx+2)\times(ny+2)\times(nz+2)$ points y compris les points du bord du domaine de discrétisation. La matrice ainsi obtenue est de taille $n = nx \times ny \times nz$.

Les critères d'arrêt utilisés sont :

- Pour GMRES : $\dfrac{\|r_k^{\text{GMRES}}\|_2}{\|r_0\|_2} \leq 10^{-8}$. La norme $\|r_k^{\text{GMRES}}\|_2$ étant égale à $|\mu_{k+1}^{(k)}|$ qu'on calcule lors de la factorisation $\mathcal{QR}$ de $\widetilde{H}_m$.

- Pour CMRH : $\dfrac{|\mu_{k+1}^{(k)}|}{\|r_0\|_2} \leq 10^{-9}$.

- Les deux algorithmes sont arrêtés si le nombre de produits matrice vecteur dépasse pmv_{max} un nombre maximal de produits matrice vecteur fixé ($pmv_{max} \simeq 1000$).

Les résultats sont reportés dans un tableau. Dans la première ligne G(m), respectivement C(m) désigne la méthode GMRES(m), respectivement la méthode CMRH(m). La seconde ligne donne le temps CPU nécessité pour la convergence de chaque méthode. La troisième ligne donne la norme du résidu final, la quatrième représente le nombre de redémarrages effectués ainsi que l'itération finale ($j//k$ s'interprète par : l'algorithme s'est arrêté à la $k^{ème}$ itération lors du $j^{ème}$ redémarrage). La dernière ligne donne pmv le nombre de produits matrice vecteur effectués par chaque méthode. Les deux dernières colonnes de chaque tableau donnent les résultats obtenus pour les méthodes GMRES et CMRH sans redémarrage. La notation $\star$ signifie qu'il n'y a pas eu convergence de la méthode.

Exemple 4.

La matrice test a été obtenue en discrétisant sur le carré unité Ω l'équation aux dérivées partielles suivante :

$$-u_{xx}(x,y) - u_{yy}(x,y) + \xi u_x(x,y) = t(x,y) \quad \text{sur } \Omega$$

$$u(x,y) = 1 + xy \quad \text{sur } \partial\Omega.$$

La fonction $t(x,y)$ est choisie de façon que la solution exacte du problème soit $u = 1+xy$. Pour cet exemple nous prenons $nx = ny = 50$, la matrice obtenue a pour taille $n = 2500$. Nous considérons les choix suivants du paramètre ξ.

- **Exemple 4.1** : $\xi = 10$. Les tableaux 4.1.1, 4.1.2 et 4.1.3 donnent dans l'ordre les résultats obtenus sans préconditionnement, avec préconditionnement *ILU0* et préconditionnement *MILU0*.

Tableau 4.1.1 : Résultats obtenus sans préconditionnement pour l'exemple 4.1.

	G(10)	C(10)	G(20)	C(20)	GMRES	CMRH
CPU	2.69	2.87	3.26	4.12	12.13	8.98
norme	$1.69\ 10^{-7}$	$3.49\ 10^{-7}$	$1.77\ 10^{-7}$	$1.84\ 10^{-7}$	$1.63\ 10^{-7}$	$2.01\ 10^{-7}$
$j//k$	25//5	31//4	10//17	17//20	0//144	0//151
pmv	307	378	239	396	146	153

Tableau 4.1.2 : Résultats obtenus avec précond. *ILU0* pour l'exemple 4.1.

	G(10)	C(10)	G(20)	C(20)	GMRES	CMRH
CPU	1.08	1.41	1.20	0.99	1.51	1.18
norme	$1.41\ 10^{-7}$	$2.92\ 10^{-7}$	$1.59\ 10^{-7}$	$3.18\ 10^{-7}$	$1.35\ 10^{-7}$	$1.37\ 10^{-7}$
$j//k$	7//5	10//9	3//6	3//6	0//46	0//47
pmv	91	131	74	74	48	49

Tableau 4.1.3 : Résultats obtenus avec précond. *MILU0* pour l'exemple 4.1.

	G(10)	C(10)	G(20)	C(20)	GMRES	CMRH
CPU	0.38	0.37	0.43	0.39	0.48	0.44
norme	$1.11\ 10^{-7}$	$4.98\ 10^{-8}$	$1.45\ 10^{-7}$	$8.39\ 10^{-8}$	$1.02\ 10^{-7}$	$6.68\ 10^{-8}$
$j//k$	2//7	2//9	1//4	1//7	0//23	0//25
pmv	33	35	28	31	25	27

• **Exemple 4.2 :** $\xi = 1000$. Le tableau 4.2 donne les résultats obtenus sans préconditionnement pour ce second choix de ξ. En ce qui concerne le préconditionnement, les deux méthodes convergent rapidement. Nous obtenons :

⋆ Préconditionnement *ILU0* :

$$\|r_{15}^{\text{GMRES}}\|_2 = 1.09\ 10^{-6} \text{ en } 0.270 \text{ CPU et } \|r_{16}^{\text{CMRH}}\|_2 = 4.15\ 10^{-7} \text{ en } 0.25 \text{ CPU.}$$

⋆ Préconditionnement *MILU0* :

$$\|r_{12}^{\text{GMRES}}\|_2 = 4.61\ 10^{-7} \text{ en } 0.20 \text{ CPU et } \|r_{13}^{\text{CMRH}}\|_2 = 1.57\ 10^{-7} \text{ en } 0.19 \text{ CPU.}$$

Tableau 4.2 : Résultats obtenus sans préconditionnement pour l'exemple 4.2.

	G(20)	C(20)	G(50)	C(50)	GMRES	CMRH
CPU	5.04	7.18	11.24	12.42	23.05	17
norme	$1.2\ 10^{-6}$	$1.03\ 10^{-6}$	$1.2\ 10^{-6}$	$3.57\ 10^{-6}$	$1.04\ 10^{-6}$	$1.8\ 10^{-6}$
$j//k$	16//4	29//20	7//26	11//36	0//200	0//209
pmv	358	660	392	610	202	211

• **Exemple 4.3 :** $\xi = 10000$. Les résultats obtenus pour ce choix sont ceux donnés par le tableau 4.3. Comme pour le choix précédent, les méthodes GMRES et CMRH convergent rapidement lorsqu'il y a préconditionnement.

⋆ Préconditionnement *ILU0* :

$$\|r_{15}^{\text{GMRES}}\|_2 = 5.33\ 10^{-6} \text{ en } 0.27 \text{ CPU et } \|r_{16}^{\text{CMRH}}\|_2 = 3.56\ 10^{-6} \text{ en } 0.24 \text{ CPU.}$$

⋆ Préconditionnement *MILU0* :

$$\|r_{8}^{\text{GMRES}}\|_2 = 4.9\ 10^{-6} \text{ en } 0.12 \text{ CPU et } \|r_{9}^{\text{CMRH}}\|_2 = 1.34\ 10^{-6} \text{ en } 0.12 \text{ CPU.}$$

Tableau 4.3 : Résultats obtenus sans préconditionnement pour l'exemple 4.3.

	G(20)	C(20)	G(50)	C(50)	GMRES	CMRH
CPU	⋆	⋆	24.28	26.05	138.51	105.79
norme	⋆	⋆	$1.25\ 10^{-5}$	$2.81\ 10^{-5}$	$1.21\ 10^{-5}$	$1.08\ 10^{-5}$
$j//k$	67//16	94//4	15//16	23//38	0//488	0//528
pmv	> 1000	> 1000	798	1040	490	530

Les trois tableaux 4.1.1, 4.2 et 4.3 montrent que sans préconditionnement la méthode GMRES(m) est plus performante que CMRH(m) pour des petites valeurs de m. Par contre les deux méthodes GMRES et CMRH donnent des résultats similaires avec un avantage pour la méthode CMRH en ce qui concerne le temps CPU.

Lorsque il y a préconditionnement les deux méthodes ont des performances sensiblement égales puisqu'on remarque qu'elle convergent, à une itération près, en même temps.

Exemple 5.

La matrice test a été obtenue en discrétisant sur le carré unité Ω l'équation aux dérivées partielles donnée dans [3]:

$$-u_{xx}(x,y) - u_{yy}(x,y) + 2p_1u_x(x,y) + 2p_2u_y(x,y) - p_3u(x,y) = F(x,y) \text{ sur } \Omega$$

$$u(x,y) = 1 + xy \text{ sur } \partial\Omega.$$

La fonction F est choisie de telle façon que la solution exacte du problème soit $u = 1+xy$. Les constantes p_1, p_2, p_3 sont positives, notons que plus la valeur de p_3 est grande plus le problème est difficile à résoudre. Pour cet exemple nous prenons $nx = ny = 32$, la matrice obtenue a pour taille $n = 1024$. Nous considérons les choix suivants des paramètres p_1, p_2 et p_3.

• **Exemple 5.1** : $p_1 = p_2 = 2$, $p_3 = 10$. Le tableau 5.1 donne les résultats obtenus sans préconditionnement pour cet exemple. En ce qui concerne le préconditionnement, les deux méthodes convergent rapidement. Nous obtenons :

⋆ Préconditionnement *ILU0* :

$\|r_{33}^{\text{GMRES}}\|_2 = 6.6\ 10^{-8}$ en 0.3 CPU et $\|r_{33}^{\text{CMRH}}\|_2 = 1.16\ 10^{-7}$ en 0.25 CPU.

⋆ Préconditionnement *MILU0* :

$\|r_{30}^{\text{GMRES}}\|_2 = 6.97\ 10^{-8}$ en 0.26 CPU et $\|r_{32}^{\text{CMRH}}\|_2 = 5.14\ 10^{-8}$ en 0.24 CPU.

Tableau 5.1 : Résultats obtenus sans préconditionnement pour l'exemple 5.1.

	G(20)	C(20)	G(40)	C(40)	GMRES	CMRH
CPU	1.29	1.74	1.73	1.76	2.19	1.82
norme	$1.34\ 10^{-7}$	$2.02\ 10^{-7}$	$1.36\ 10^{-7}$	$1.36\ 10^{-7}$	$1.41\ 10^{-7}$	$1.38\ 10^{-7}$
$j//k$	11//15	18//10	4//37	6//9	0//103	0//107
pmv	259	408	207	263	105	109

• **Exemple 5.2** : $p_1 = p_2 = 2$, $p_3 = 1000$. Sans préconditionnement, les résultats obtenus sont ceux du tableau 5.2.

Notons que pour cet exemple les deux préconditionnement ne sont pas efficaces même lorsqu'il y a redémarrage des deux algorithmes. De plus, le préconditionnement retarde la convergence des deux méthodes. Nous obtenons :

⋆ Préconditionnement *ILU0* :

$\|r_{142}^{\text{GMRES}}\|_2 = 1.94\ 10^{-7}$ en 3.95 CPU et $\|r_{146}^{\text{CMRH}}\|_2 = 1.11\ 10^{-7}$ en 3.32 CPU.

⋆ Préconditionnement *MILU0* :

$\|r_{409}^{\text{GMRES}}\|_2 = 3.24\ 10^{-7}$ en 31.39 CPU et $\|r_{416}^{\text{CMRH}}\|_2 = 1.20\ 10^{-7}$ en 25.79 CPU.

Comme les préconditionneurs *ILU0* et *MILU0* n'ont pas étés efficaces pour cet exemple, nous avons essayé un préconditionnement par *ILUT(k, ϵ)* [44]. Lorsque $k \leq 20$, le préconditionnement $ILUT(k, 10^{-8})$ n'améliore pas la convergence des deux méthodes, par contre en prenant $k = 30$ les deux méthodes convergent rapidement et nous obtenons :

$\|r_7^{\text{GMRES}}\|_2 = 6.89\ 10^{-8}$ en 0.15 CPU et $\|r_8^{\text{CMRH}}\|_2 = 1.12\ 10^{-9}$ en 0.17 CPU.

Tableau 5.2 : Résultats obtenus sans préconditionnement pour l'exemple 5.2.

	G(100)	C(100)	G(250)	C(250)	GMRES	CMRH
CPU	$\star$	$\star$	33.81	31.22	20.16	17
norme	$\star$	$\star$	$1.63\ 10^{-5}$	$1.15\ 10^{-4}$	$3.15\ 10^{-7}$	$2.9\ 10^{-7}$
$j//k$	$\star$	$\star$	3//250	3//250	0//330	0//342
pmv	> 1000	> 1000	1008	1008	332	344

Comme pour l'exemple précédent, nous remarquons que sans préconditionnement et pour des petites valeurs de m, la méthode GMRES(m) est légèrement meilleure que CMRH(m). Notons que le choix de petites valeurs pour m n'est pas toujours possible, en effet le tableau 5.2 montre que même lorsque $m = 100$, les méthodes GMRES(m) et CMRH(m) n'arrivent pas à converger. En fait pour le choix $p_3 = 1000$ le redémarrage n'est efficace que pour des grandes valeurs de m ($m \geq 250$).

Dans le cas du préconditionnement et pour $p_3 = 1000$, le redémarrage n'est d'aucune efficacité, de plus le préconditionnement *MILU0* handicape la convergence des deux méthodes. Seul le précondionnement *ILUT(k, ϵ)* avec $k \geq 30$ est efficace pour cet exemple.

Exemple 6.

La matrice test a été obtenue en discrétisant sur le cube unité Ω l'équation aux dérivées partielles donnée dans [35] :

$$-\Delta u(x,y,z) + \theta(xu_x(x,y,z) + yu_y(x,y,z) + zu_z(x,y,z)) + \lambda u(x,y,z) = F(x,y,z),$$

$$u(x,y,z) = 0 \quad \text{sur } \partial\Omega.$$

Le second membre f a été calculé de telle façon que la solution exacte du système soit $x_* = (1, \cdots, 1)^T$. Le nombre de points utilisés dans chaque direction pour la discrétisation est de 27, i.e. $nx = ny = nz = 25$. La taille du système ainsi obtenu est $n = 15625$. Pour les paramètres θ et λ nous considérons les choix suivants :

• **Exemple 6.1** : $\theta = 40$, $\lambda = -250$. Avec préconditionnement les deux méthodes convergent rapidement et même à la première itération dans le cas d'un préconditionnement *MILU0*. Pour le préconditionnement *ILU0* nous avons :

$$\|r_{43}^{\text{GMRES}}\|_2 = 4.77\ 10^{-7} \quad \text{en} \quad 10.91 \quad \text{CPU} \quad \text{et} \quad \|r_{44}^{\text{CMRH}}\|_2 = 3.68\ 10^{-7} \quad \text{en} \quad 9.06 \quad \text{CPU.}$$

Le tableau 6.1 ci dessous donne les résultats obtenus sans préconditionnement pour l'exemple 6.1.

Tableau 6.1 : Résultats obtenus sans préconditionnement pour l'exemple 6.1.

	G(40)	C(40)	G(80)	C(80)	GMRES	CMRH
CPU	$\star$	$\star$	96.37	76.1	62.01	43.2
norme	$\star$	$\star$	$4.68\ 10^{-7}$	$2.04\ 10^{-6}$	$4.96\ 10^{-7}$	$1.28\ 10^{-6}$
$j//k$	$\star$	$\star$	3//59	3//79	0//124	0//125
pmv	> 1000	> 1000	307	327	126	127

Exemple 6.2 : $\theta = -40$, $\lambda = 250$. Nous obtenons des résultats analogues à ceux de l'exemple préceédent. Il y a convergence à la première itération dans le cas d'un préconditionnement *MILU0*. Pour le préconditionnement *ILU0* nous obtenons :

$$\|r_{17}^{\text{GMRES}}\|_2 = 1.05\ 10^{-6} \quad \text{en} \quad 2.79 \quad \text{CPU et} \quad \|r_{18}^{\text{CMRH}}\|_2 = 6.32\ 10^{-7} \quad \text{en} \quad 2.67 \quad \text{CPU.}$$

Le tableau 6.2 donne les résultats obtenus sans préconditionnement.

Tableau 6.2 : Résultats obtenus sans préconditionnement pour l'exemple 6.2.

	G(20)	C(20)	G(40)	C(40)	GMRES	CMRH
CPU	10.02	9.27	15.11	11.2	20.98	15.12
norme	$9.05\ 10^{-7}$	$1.38\ 10^{-6}$	$1.05\ 10^{-6}$	$2.23\ 10^{-7}$	$5.82\ 10^{-7}$	$2.7\ 10^{-7}$
$j//k$	3//20	4//14	1//38	1//38	0//69	0//70
pmv	88	104	82	82	71	72

Pour le premier choix $\theta = 40$, $\lambda = -250$: sans préconditionnement, la méthode GMRES(m) stagne alors que la méthode CMRH(m) présente de petites oscillations sans converger et cela pour $m \leq 40$. En fait les deux méthodes ne commencent à converger que lorsque m dépasse 50. Pour les longs redémarrages ainsi que pour les algorithmes non redémarrés, CMRH(m) respectivement CMRH est plus rapide que GMRES(m) respectivement GMRES. Notons de plus qu'il n'y a pas de supériorité d'une des deux méthodes sur l'autre dans le cas de préconditionnement.

Le second choix $\theta = -40$, $\lambda = 250$ rend l'équation aux dérivées partielles plus facile à résoudre par les deux méthodes, dans ce cas CMRH respectivement CMRH(m) est plus rapide que GMRES respectivement GMRES(m).

Exemple 7.

La matrice test a été obtenue en discrétisant sur le carré unité Ω l'équation aux dérivées partielles :

$$-u_{xx}(x,y) - u_{yy}(x,y) + \delta e^{xy} u_x(x,y) + \delta e^{-xy} u_y(x,y) + \theta u(x,y) = F(x,y).$$

Le second membre f a été calculé de façon que la solution exacte du système soit $x_* = (1, \cdots, 1)^T$. Pour cet exemple nous prenons $nx = ny = 30$, ainsi la taille de système obtenu est $n = 900$. Pour les paramètres δ et θ, trois choix sont considérés :

• **Exemple 7.1 :** $\theta = -50$, $\delta = 10$. Le tableau 7.1 donne les résultats obtenus sans préconditionnement.

Tableau 7.1.1 : Résultats obtenus sans préconditionnement pour l'exemple 7.1.

	G(20)	C(20)	G(40)	C(40)	GMRES	CMRH
CPU	3.4	3.3	2.36	2.34	2.3	1.82
norme	0.23	$5.41\ 10^{-2}$	$9.59\ 10^{-8}$	$4.94\ 10^{-8}$	$9.6\ 10^{-8}$	$1.15\ 10^{-7}$
$j//k$	45//20	45//20	7//39	10//12	0//119	0//126
pmv	1012	1012	335	434	121	128

Les deux méthodes, avec préconditionnement *MILU0*, convergent à la première itération. Avec un préconditionnement *ILU0*, nous obtenons :

$\|r_{37}^{\text{GMRES}}\|_2 = 1.01\ 10^{-7}$ en 0.3 CPU et $\|r_{39}^{\text{CMRH}}\|_2 = 6.31\ 10^{-8}$ en 0.27 CPU.

• **Exemple 7.2 :** $\theta = 50$, $\delta = -10$. Les résultats obtenus, sans préconditionnement, sont ceux donnés par le tableau 7.2. Dans le cas d'un préconditionnement *ILU0* nous obtenons :

$\|r_{25}^{\text{GMRES}}\|_2 = 8.42\ 10^{-8}$ en 0.17 CPU et $\|r_{26}^{\text{CMRH}}\|_2 = 4.72\ 10^{-8}$ en 0.15 CPU.

Avec un préconditionnement *MILU0*, les deux méthodes convergent à la première itération.

Tableau 7.2 : Résultats obtenus sans préconditionnement pour l'exemple 7.2.

	G(10)	C(10)	G(20)	C(20)	GMRES	CMRH
COU	0.37	0.41	0.50	0.47	1.15	0.85
norme	$1.12\ 10^{-7}$	$8.55\ 10^{-8}$	$1.21\ 10^{-7}$	$6.64\ 10^{-8}$	$9.87\ 10^{-8}$	$8.81\ 10^{-8}$
$j//k$	10//7	13//2	5//8	6//3	0//82	0//84
pmv	129	160	120	137	84	86

• **Exemple 7.3 :** $\theta = -50$, $\delta = 1000$. Les tableaux 7.3.1 et 7.3.2 résument les résultats obtenus sans préconditionnement et avec préconditionnement *ILU0*. Comme pour les deux choix précédents, le préconditionnement *MILU0* permet aux deux méthodes de converger à la première itération.

Tableau 7.3.1 : Résultats obtenus sans préconditionnement pour l'exemple 7.3.

	G(20)	C(20)	G(50)	C(50)	GMRES	CMRH
CPU	1.88	$\star$	3.78	7.09	14.17	11.90
norme	$2.05\ 10^{-6}$	$\star$	$2.04\ 10^{-6}$	$3.00\ 10^{-6}$	$1.94\ 10^{-6}$	$1.71\ 10^{-6}$
$j//k$	20//2	$\star$	8//37	19//50	0//300	0//320
pmv	444	> 1000	455	1040	302	322

Tableau 7.3.2 : Résultats obtenus avec précond. $ILU0$ pour l'exemple 7.3.

	G(20)	C(20)	G(50)	C(50)	GMRES	CMRH
CPU	$\star$	$\star$	3.35	3.15	1.98	1.58
norme	$\star$	$\star$	$1.90\ 10^{-6}$	$1.38\ 10^{-6}$	$1.41\ 10^{-6}$	$1.88\ 10^{-5}$
$j//k$	$\star$	$\star$	6//40	7//45	0//108	0//112
pmv	> 1000	> 1000	354	411	110	114

Pour le premier choix et dans le cas où les deux méthodes ne sont pas préconditionnées, GMRES(20) ne commence à converger qu'a partir du $73^{\text{ème}}$ redémarrage alors que CMRH(20) converge à partir du $42^{\text{ème}}$. En fait nous obtenons $\|r^{\text{GMRES}}_{90//20}\|_2 = 6.26\ 10^{-6}$ à un coût de 2002 produits matrice vecteur et $\|r^{\text{CMRH}}_{63//13}\|_2 = 3.97\ 10^{-8}$ à un coût de 1445 produits matrice vecteur. Pour $m > 20$, les deux méthodes avec et sans préconditionnement donnent des résultats similaires.

En ce qui concerne le second choix, nous obtenons des résultats comparables pour les deux méthodes.

Pour le dernier choix, la méthode GMRES(m) est plus performante que CMRH(m) s'il n'y a pas préconditionnement, dans le cas contraire les deux méthodes ont des performances similaires.

2.5.3 Algorithmes tronqués

Nous comparons dans cette section la performance des deux méthodes décrites dans le dernier paragraphe du présent chapitre à savoir DQGMRES(k) et DTCMRH(k). L'implémentation de l'algorithme DQGMRES est celle décrite par l'algorithme 3.3 dans [43], en remplaçant dans cet algorithme la procédure d'Orthogonalisation Incomplète par le processus de Hessenberg Incomplet nous obtenons l'implémentation de la méthode DTCMRH(k). Dans tous les exemples le vecteur second membre f est égal à $A\,e$, où $e = (1, \ldots, 1)^T$.

Les tests ont étés réalisés sur un "Sun SparcStation10" à l'aide du Compilateur FORTRAN : f77 avec l'option "-fast".

En ce qui concerne les critères d'arrêt, nous avons utilisé :

- Pour DQGMRES(k) : $\frac{|\mu_{k+1}^{(k)}|}{\|r_0\|_2} \leq 10^{-7}$.
- Pour DTCMRH(k) : $\frac{|\mu_{k+1}^{(k)}|}{\|r_0\|_2} \leq 10^{-8}$.

Afin de comparer les résidus, nous calculons les résidus des deux méthodes à chaque itération, la norme des résidus est alors tracée dans une échelle logarithmique. Dans chaque figure les courbes des méthodes DTCMRH(20), DTCMRH(10), DTCMRH(5) sont représentées par la ligne continue, les tirets, la ligne faite avec des '++' respectivement. La norme euclidienne des résidus des méthodes DQGMRES(20), DQGMRES(10), DQGMRES(5) respectivement est représentée par des tirets-points, pointillés, points respectivement.

Les résultats sont reportés dans un tableau, la première ligne donne le temps CPU nécessité pour la convergence de chaque méthode, la seconde donne le résidu final, la troisième représente l'estimation $|\mu_{k+1}^{(k)}|$. La dernière ligne donne l'itération finale. Dans chaque tableau DQG(m), respectivement DTC(m) désigne DQGMRES(m), respectivement DTCMRH(m).

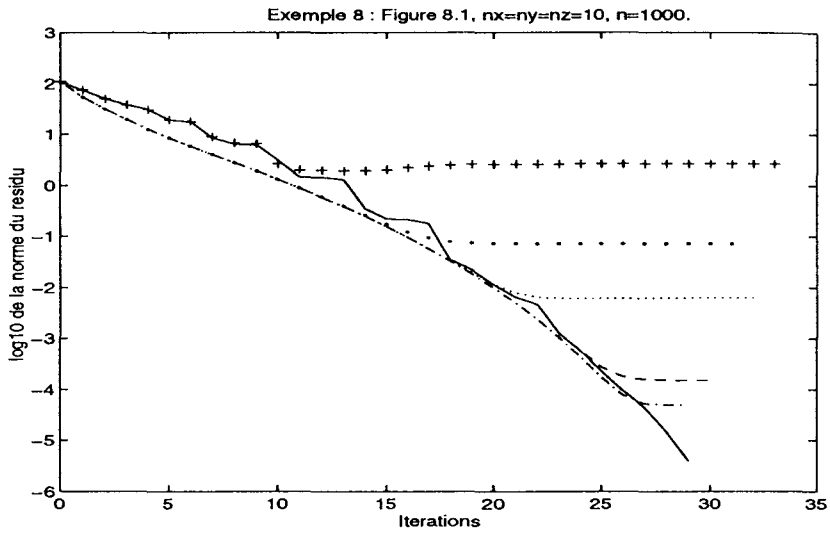
Exemple 8

La matrice test est celle obtenue par discrétisation de l'équation aux dérivées partielles de l'exemple 6 avec le choix des paramètres $\theta = -40$, et $\lambda = 250$. Nous considérons les deux maillages suivants :

Exemple 8.1 : $nx = 10$, $ny = 10$, $nz = 10$, la matrice ainsi obtenue est de taille $n = 1000$. Les résultats (cf Tableau 8.1) sont ceux obtenus en utilisant les algorithmes sans préconditionnement et sont donnés par le tableau 8.1.

Tableau 8.1 : Résultats obtenus sans préconditionnement pour l'exemple 8.1.

	DQG(5)	DTC(5)	DQG(10)	DTC(10)	DQG(20)	DTC(20)
CPU	0.54	0.51	0.72	0.59	0.83	0.74
norme	$7.11\ 10^{-2}$	2.62	$6.17\ 10^{-3}$	$1.49\ 10^{-4}$	$4.94\ 10^{-5}$	$3.94\ 10^{-6}$
$\mu_{k+1}^{(k)}$	$5.67\ 10^{-6}$	$7.66\ 10^{-6}$	$6.31\ 10^{-6}$	$1.51\ 10^{-7}$	$7.63\ 10^{-6}$	$1.07\ 10^{-6}$
itération	31	33	32	30	29	29



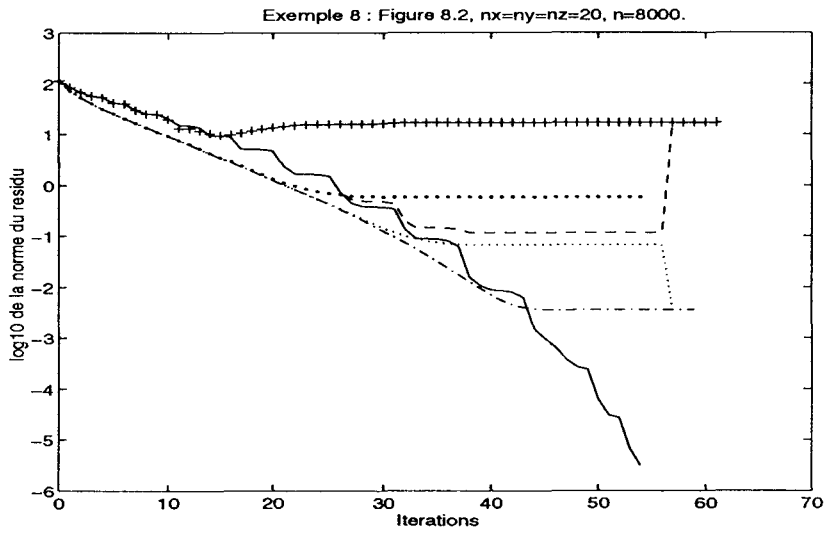
Pour cet exemple, nous observons que pour $l = 10$ ou $l = 20$ DTCMRH(l) donne des résultats meilleurs que ceux de DQGMRES(l), par contre DQGMRES(5) a des performances légèrement meilleures que DTCMTH(5). Notons que nous obtenons pour les méthodes CMRH et GMRES les résultats suivants :

$$\|r_{33}^{\text{GMRES}}\|_2 = 1.00\,10^{-6} \text{ et } \|r_{34}^{\text{CMRH}}\|_2 = 7.28\,10^{-7}.$$

Exemple 8.2 : $nx = 20$, $ny = 20$, $nz = 20$, la matrice ainsi obtenue est de taille $n = 8000$. Comme pour l'exemple précédent, les résultats (cf Tableau 8.2) sont ceux obtenus en utilisant les algorithmes sans préconditionnement.

Tableau 8.2 : Résultats obtenus sans préconditionnement pour l'exemple 8.2.

	DQG(5)	DTC(5)	DQG(10)	DTC(10)	DQG(20)	DTC(20)
CPU	8.05	7.87	11.20	9.47	17.30	13.03
norme	0.58	17.14	$6.64\,10^{-2}$	0.11	$3.61\,10^{-3}$	$3.14\,10^{-6}$
$\mu_{k+1}^{(k)}$	$1.13\,10^{-5}$	$5.23\,10^{-7}$	$9.10\,10^{-6}$	$9.68\,10^{-7}$	$8.48\,10^{-6}$	$7.36\,10^{-7}$
itération	54	61	56	56	59	54



En appliquant les algorithmes GMRES et CMRH, nous obtenons :

$$\|r_{53}^{\mathrm{GMRES}}\|_2 = 6.01\,10^{-6} \text{ et } \|r_{54}^{\mathrm{CMRH}}\|_2 = 3.14\,10^{-6}.$$

Pour $l = 5$ ou $l = 10$ DQGMRES(l) est meilleur que DTCMRH(l). Cependant, les résultats donnés par DTCMRH(20) sont légèrement meilleurs que ceux de la méthode DQGMRES(20).

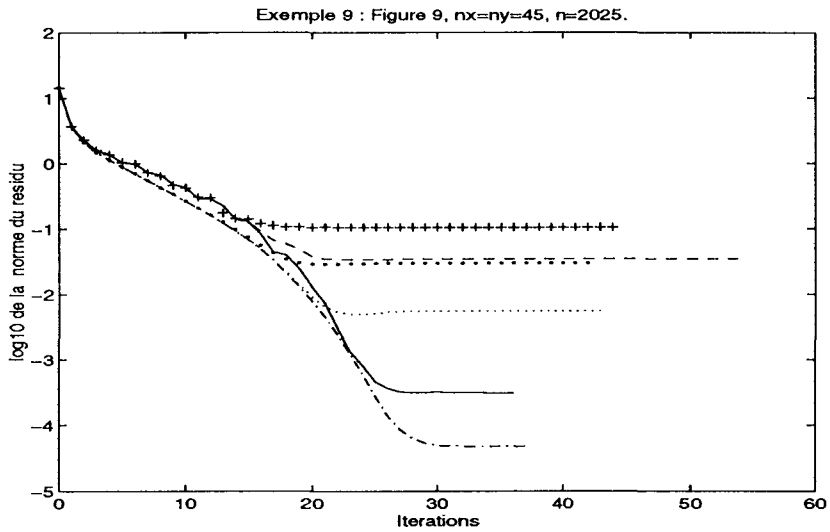
Exemple 9

La matrice test est celle obtenue par discrétisation de l'équation aux dérivées partielles de l'exemple 7, les paramètres étant $\theta = 50$, $\delta = -10$. On considère le maillage $nx = ny = 45$, la matrice ainsi obtenue est de taille $n = 2025$.

Les deux méthodes sans préconditionnement ne convergent pas. Les résultats donnés (cf Tableau 9) sont ceux obtenus en utilisant les algorithmes avec le préconditionnement *ILU0*.

Tableau 9 : Résultats obtenus avec précond. *ILU0* pour l'exemple 9.

	DQG(5)	DTC(5)	DQG(10)	DTC(10)	DQG(20)	DTC(20)
CPU	1.88	1.93	2.58	2.99	2.84	2.48
norme	$2.98\,10^{-2}$	$1.04\,10^{-1}$	$5.59\,10^{-3}$	$3.44\,10^{-2}$	$4.82\,10^{-5}$	$3.15\,10^{-4}$
$\mu_{k+1}^{(k)}$	$9.47\,10^{-7}$	$8.83\,10^{-8}$	$1.35\,10^{-6}$	$1.23\,10^{-7}$	$1.30\,10^{-6}$	$9.71\,10^{-8}$
itération	42	44	43	54	37	36



Sans préconditionnement les méthodes GMRES et CMRH convergent respectivement en 111 et 114 itérations, ce qui peut expliquer que les méthodes DQGMRES(l) et DTCMRH(l) (sans préconditionnement) ne convergent pas pour $l \leq 20$. Pour cet exemple, la performance de DQGMRES(l) est meilleure que celle de DTCMRH(l). Notons que pour le préconditionnement ILU0 nous avons :

$$\|r_{33}^{\text{GMRES}}\|_2 = 1.10\,10^{-6} \text{ et } \|r_{54}^{\text{CMRH}}\|_2 = 7.28\,10^{-7}.$$

Dans les trois exemples précédents, consacrés à la comparaison des deux méthodes DQGMRES(l) et DTCMRH(l), nous remarquons que le paramètre $\mu_{k+1}^{(k)}$ utilisé dans le critère d'arrêt des deux méthodes n'est pas une bonne estimation de la norme du résidu. Saad [43] a montré que l'on peut calculer le résidu exact, ou une meilleure estimation de sa norme en utilisant la technique précédemment décrite dans le paragraphe 2.2.3.

2.5.4 Analyse et discussion des résultats

Lorsque les matrices sont pleines, les différents tests effectués pour les méthodes GMRES, QMR et CMRH non redémarrées, non tronquées et non préconditionnées montrent que ces trois méthodes ont des performances similaires. En général, GMRES gagne au maximum deux itérations sur CMRH, mais le coût algorithmique est en faveur du CMRH.

Pour des matrices creuses de grande taille, sans préconditionnement la méthode GMRES(m) est plus performante que CMRH(m) pour des petites valeurs de m car CMRH(m) redémarre plus de fois que GMRES(m). Cette remarque n'est en général plus valable lorsque il y a préconditionnement. Pour des grandes valeurs de m, CMRH(m) est meilleur que GMRES(m) vu que les deux méthodes convergent en faisant presque toujours le même nombre d'itérations, cependant le temps CPU est en faveur de la méthode CMRH(m).

En ce qui concerne la méthode DQGMRES(k), nous n'avons pas observé les mêmes résultats que ceux obtenus par Saad et Wu [43]. En effet pour les exemples précédents, la méthode GMRES(2k) donne de meilleurs résultats que DQGMRES(k). Le préconditionnement améliore en général la convergence de la méthode. Notons aussi que lorsque la méthode GMRES ne converge pas rapidement pour un problème donné, la méthode DQGMRES(k) ne convergera pas, en général, pour des petites valeurs de k. Cette dernière remarque reste valable pour la méthode DTCMRH(k).

Conclusion

Dans ce chapitre, nous avons décrit deux classes de méthodes pour la résolution des systèmes d'équations linéaires. Ces deux classes font partie de la méthode de Hessenberg Généralisée qui peut être considérée comme un cas particulier des méthodes de sous espace de Krylov pour la résolution des systèmes linéaires. Les deux classes de méthodes sont la méthode de Galerkin obtenue par imposition de la condition de Pertrov-Galerkin et la méthode MRSe obtenue par imposition de la condition de semi-minimisation.

Différentes méthodes connues pour la résolution de systèmes linéaires peuvent être retrouvées comme cas particuliers de la méthode de Hessenberg Généralisée. En adoptant respectivement les choix Arnoldi, Hessenberg et Lanczos nous obtenons :

- les méthodes FOM, IOM, DIOM, Hessenberg, BCG qui font partie de la méthode de Galerkin.

- les méthodes GMRES, CMRH, QMR, DQGMRES et DTCMRH qui font partie de la méthode MRSe.

Nous verrons dans le chapitre 4 le lien qui existe entre la méthode de Galerkin et la méthode MRSe. Ce lien n'est autre que le "smoothing" variable qui est un cas particulier de la procédure hybride définie par C. Brezinski et M.R. Zaglia [5] et étudiée en détail par Abkowicz [1].

Appendix : Le tableau ci dessous permet de comparer le nombre d'opérations, ainsi que la place mémoire nécessaires à chaque iération j de différentes méthodes vues dans ce chapitre.

méthode	produit scalaire	SAXPY	produit matrice vecteur	place mémoire
IOM(l)	$min(l,j)+1$	$min(l,j)+1$	1/0	$(min(l,j)+3)n$
DIOM(l)	$min(l,j)+1$	$2min(l,j)+1$	1/0	$(2min(l,j)+3)n$
FOM	$j+1$	$j+1$	1/0	$(j+3)n$
GMRES	$j+1$	$j+1$	1/0	$(j+5)n$
BCG	2	5	1/1	$9n$
QMR	2	12	1/1	$15n$
CMRH	–	$1^{(*)}$	$1/0^{(\dagger)}$	$(j+6)n$
DQGMRES(l)	$min(l,j)+3$	$2min(l,j)+2$	1/0	$(2min(l,j)+8)n$
DTCMRH(l)	–	$min(l,j)+3^{(*)}$	$1/0^{(\dagger)}$	$(min(l,j)+8)n$

- La notation i/j veut dire qu'il y a i produits avec la matrice A et j produits avec sa transposée A^T.
- Pour le méthodes GMRES, BCG et QMR, nous avons considéré les algorithmes donnés dans TEMPLATES [4]. Pour les autres méthodes, nous avons considéré des schémas analogues.
- Dans la place mémoire, nous n'avons pas tenu compte de la matrice A.
- Dans la méthode CMRH, si on accepte de détruire la matrice A, alors il est possible de stocker les itérés construits par le processus de Hessenberg dans A (voir les remarques de §1.2.2). Dans les autres méthodes, il n'est pas possible de faire pareil.
- (*) Il n'y a pas d'opérations SAXPY dans le processus de Hessenberg utilisé dans ces deux méthodes. Cependant le nombre de multiplications (additions) nécessaires à chaque itération j du CMRH (respectivement DTCMRH) est $n-j$ (respectivement $n-l$).
- (†) Etant donné que le vecteur b_j construit par le processus de Hessenberg (respectivement Hessenberg Incomplet) admet $j-1$ (respectivement l) composantes nulles, le produit matrice vecteur Ab_j à l'itération j dans la méthode CMRH nécessite $n\nu_j$ multiplications. Si la matrice est dense $n\nu_j = n-j$, et si la matrice est creuse $n\nu_j \leq N_{nz}$ où N_{nz} est le nombre déléments non nuls de la matrice A. Pour les autres méthodes le coût d'un produit matrice vecteur est $n\nu$ mulitiplications, où $\nu = n$ pour une matrice dense et $\nu = N_{nz}$ pour une matrice creuse.

Chapitre 3

Sur une procédure de "smoothing" variable

Dans le chapitre 2, nous avons défini deux classes de méthodes, la première est celle des méthodes de Galerkin et la seconde est celle de type MRSe. Dans la pratique, il est connu que la convergence des méthodes de Galerkin peut être très irrégulière. En effet, dans les méthodes de Galerkin, la norme euclidienne des résidus $\|r_k\|_2$, $k = 1, 2, \ldots$ n'est pas décroissante en fonction de l'indice d'itération. Afin d'éliminer ce problème nous pouvons utiliser :

- soit la méthode de semi-minimisation du résidu, c'est à dire que la solution approchée x_k est choisie de telle façon que :

$$|r_k|_{Z_k} = \min_{x \in x_0 + K_k(r_0, A)} |f - Ax|_{Z_k}$$

avec Z_k et $|u|_{Z_k}$ définies comme précédemment pour la méthode MRSe ;

- soit l'algorithme MRS (minimal residual smoothing algorithm) introduit par Schönauer [48] afin de rendre la convergence des méthodes Gradient Conjugué Généralisés plus régulière.

Soit $\{x_k\}_{k=1,\ldots}$ une suite des itérés générés par une méthode itérative pour la résolution du système linéaire $Ax = f$. La suite $\{x'_k\}_{k=1,\ldots}$ obtenue par une technique de lissage (en anglais : "smoothing") appliquée à $\{x_k\}$ est définie comme suit :

$$\begin{cases} x'_0 &= x_0 \\ x'_{k+1} &= x'_k + \lambda_k (x_{k+1} - x'_k) \qquad \text{pour} \quad k = 0, 1, \ldots . \end{cases}$$

Différents types de lissage peuvent être obtenus suivant le choix du coefficient λ_k

- Si λ_k est telle que:

$$\lambda_k = \min_{\lambda \in \mathbb{R}} \|f - A(x_k' + \lambda(x_{k+1} - x_k'))\|_Z, \tag{3.1}$$

où Z une matrice symétrique définie positive, dans ce cas nous obtenons l'algorithme MRS dont des résultats théoriques ont étés établies par Weiss [56] et par Weiss et Schönauer [58]. Notons que dans ce cas nous avons:

$$\lambda_k = -\frac{(r_k', r_{k+1} - r_k')_Z}{(r_{k+1} - r_k', r_{k+1} - r_k')_Z}, \tag{3.2}$$

avec $r_k = f - Ax_k$ et $r_k' = f - Ax_k'$.

- Si λ_k est choisi de façon que:

$$\lambda_k = \frac{\tau_{k+1}^2}{\|r_{k+1}\|_2^2} \quad \text{avec} \quad \tau_{k+1}^2 = \frac{1}{\frac{1}{\tau_k^2} + \frac{1}{\|r_{k+1}\|_2^2}} \quad \text{et} \quad \tau_0 = \|r_0\|_2, \tag{3.3}$$

nous obtenons l'algorithme QMRS (quasi minimal residual smoothing algorithm) introduit par L. Zhou et H.F. Walker [62].

Rappelons que L. Zhou et H.F. Walker [62] ont établi que la méthode QMR peut être obtenue à partir de la méthode BCG par application de l'algorithme QMRS, en effet:

$$x_{k+1}^{\text{QMR}} = x_k^{\text{QMR}} + \frac{\tau_{k+1}^2}{\rho_{k+1}^2}(x_{k+1}^{\text{BCG}} - x_k^{\text{QMR}}), \tag{3.4}$$

où $\tau_k = \sqrt{\dfrac{1}{\sum_{j=0}^{k} 1/\rho_j^2}}$ et $\rho_j = \|r_j^{\text{BCG}}\|_2$.

La comparaison de la méthode BICO (obtenue par application de l'algorithme MRS à la méthode BCG) avec la méthode QMR (obtenue par application de l'algorithme QMRS à la méthode BCG) est donnée dans [57]. Notons aussi que Weiss [56] a montré que l'application de l'algorithme MRS à des méthodes dont les résidus sont orthogonaux permet de retrouver des méthodes dont la norme du résidu est minimale dans le sous espace de Krylov considéré. Ainsi, en appliquant l'algorithme MRS à la méthode FOM nous retrouvons la méthode GMRES.

Dans ce chapitre, nous montrons que la méthode MRSe peut être obtenue en appliquant l'algorithme SMRS (semi minimal residual smoothing algorithm) [27] à la méthode

de Galerkin. Notons aussi que l'algorithme SMRS est un cas particulier d'une procédure de lissage variable, cette procédure est obtenue en remplaçant dans l'équation (3.2) la matrice fixe Z par une matrice Z_{k+1} qui dépend de l'indice de l'itération k. Lorsque nous prenons $Z_{k+1} = (B_{k+2}^L)^T B_{k+2}^L$ où B_{k+2} est la matrice construite par le processus de Hessenberg Généralisé et B_{k+2}^L sa matrice inverse généralisée à gauche définie par (1.8) nous obtenons l'algorithme SMRS.

Nous débuterons ce chapitre par quelques résultats concernant la procédure du lissage variable. Au paragraphe 3.2 nous établissons des relations entre les itérés de la méthode de Galerkin et ceux de la méthode MRSe, nous serons alors en mesure de donner le principal résultat de ce chapitre. Au paragraphe 3.3, nous établissons quelques propriétés verifiées par la Z_k-semi-norme et les résidus des deux méthodes de Galerkin et MRSe. Finalement au paragraphe 3.4 nous concluons ce chapitre en appliquant les résultats obtenus aux cas particuliers des méthodes de Galerkin et MRSe.

3.1 La procédure de "smoothing" variable

3.1.1 Définition

Soit $\{x_k\}_{k=1,\ldots}$ une suite de solutions approchées calculées par une méthode itérative appliquée au système linéaire

$$Ax = f.$$

Soit $r_k = f - Ax_k$ le résidu associé à la solution x_k et Z_{k+1} une matrice symétrique définie positive.

Définition 6

La suite $\{x'_k\}_{k=1,\ldots}$ obtenue par la procédure de "smoothing" variable appliquée à la suite $\{x_k\}_{k=1,\ldots}$ est définie par:

$$\begin{cases} x'_0 &= x_0 \\ x'_{k+1} &= x'_k + \lambda_k (x_{k+1} - x'_k) \qquad \textit{pour} \quad k = 0, 1, \ldots. \end{cases}$$

Le coefficient λ_k étant déterminé par $\|r'_{k+1}\|_{Z_{k+1}} = min$, où $r'_{k+1} = f - Ax'_{k+1}$.

Il est clair que nous avons aussi:

$$\begin{aligned} r'_0 &= r_0 \\ r'_{k+1} &= r'_k + \lambda_k (r_{k+1} - r'_k) \qquad \text{pour} \quad k = 0, 1, \ldots, \end{aligned} \tag{3.5}$$

et donc en dérivant $\|r_k' + \lambda\,(r_{k+1} - r_k')\|_{Z_{k+1}}$ par rapport à λ nous avons :

$$\lambda_k = -\frac{(r_k', r_{k+1} - r_k')_{Z_{k+1}}}{(r_{k+1} - r_k', r_{k+1} - r_k')_{Z_{k+1}}}. \tag{3.6}$$

Une conséquence directe de la définition est que :

$$\|r_{k+1}'\|_{Z_{k+1}} \leq min(\|r_k'\|_{Z_{k+1}}, \|r_{k+1}\|_{Z_{k+1}}).$$

Par contre, nous ne pouvons rien dire en ce qui concerne la comparaison des normes : $\|r_{k+1}'\|_{Z_{k+1}}$, $\|r_k'\|_{Z_k}$ et $\|r_{k+1}\|_{Z_k}$.

3.1.2 Propriétés de la procédure de "smoothing" variable

Les résultats donnés dans cette section ont étés initialement établis par Weiss [56] pour l'algorithme MRS, quelques résultats restent encore vérifiés par l'algorithme de la procédure de "smoothing" variable.

Propriété 4

Soit r_k' la suite obtenue par la procédure de "smoothing" variable appliquée à la suite r_k alors :

$$r_k' = \sum_{i=0}^{k} \delta_{i,k} r_i, \tag{3.7}$$

avec

$$\sum_{i=0}^{k} \delta_{i,k} = 1. \tag{3.8}$$

Preuve :

Nous établissons ce résultat par récurrence sur k.
Nous avons $r_0' = \delta_{0,0}\, r_0$ avec $\delta_{0,0} = 1$. Supposons que le résultat soit vrai pour k.

$$\begin{aligned} r_{k+1}' &= r_k' + \lambda_k\,(r_{k+1} - r_k') \\ &= \sum_{i=0}^{k} \delta_{i,k} r_i + \lambda_k\, r_{k+1} - \lambda_k \sum_{i=0}^{k} \delta_{i,k} r_i \\ &= \sum_{i=0}^{k+1} \delta_{i,k+1} r_i \end{aligned}$$

avec $\delta_{k+1,k+1} = \lambda_k$ et $\delta_{i,k+1} = (1-\lambda_k)\delta_{i,k}$ pour $i = 0, \ldots, k$. De plus nous avons :

$$\sum_{i=0}^{k+1} \delta_{i,k+1} = \delta_{k+1,k+1} + \sum_{i=0}^{k}(1-\lambda_k)\delta_{i,k} = \lambda_k + (1-\lambda_k)\sum_{i=0}^{k}\delta_{i,k} = 1.$$

■

Propriété 5

Soit $r_k^{'}$ la suite obtenue par la procédure de "smoothing" variable appliquée à la suite r_k alors :

$$r_{k+1}^{'\,T} Z_{k+1}(r_{k+1} - r_k^{'}) = 0, \tag{3.9}$$

$$r_{k+1}^{'\,T} Z_{k+1} r_{k+1}^{'} = r_{k+1}^{'\,T} Z_{k+1} r_k^{'}, \tag{3.10}$$

$$r_{k+1}^{'\,T} Z_{k+1} r_{k+1}^{'} = r_{k+1}^{'\,T} Z_{k+1} r_{k+1}. \tag{3.11}$$

Preuve :

L'équation (3.9) découle directement de l'expression même de $r_{k+1}^{'}$ et de celle de λ_k données dans la définition 6.

Si $\lambda_k = 0$ alors $r_{k+1}^{'} = r_k^{'}$, la seconde expression (3.10) s'obtient alors directement. Si $\lambda_k \neq 0$, de la première équation, nous pouvons déduire :

$$\frac{1}{\lambda_k} r_{k+1}^{'\,T} Z_{k+1}(r_{k+1}^{'} - r_k^{'}) = 0$$

et ainsi, nous obtenons (3.10).

En combinant les deux premières équations, nous obtenons la derniére égalité (3.11).

■

Propriété 6

Soit $r_k^{'}$ la suite obtenue par la procédure de "smoothing" variable appliquée à r_k et soit $d_k = r_{k+1} - r_k^{'}$ le vecteur direction de recherche alors :

$$\|r_{k+1}^{'}\|_{Z_{k+1}}^2 = \|r_k^{'}\|_{Z_{k+1}}^2 - \frac{(r_k^{'\,T} Z_{k+1} d_k)^2}{d_k^T Z_{k+1} d_k}. \tag{3.12}$$

Preuve :

En utilisant successivement les relations (3.10) et (3.5) nous avons :

$$\begin{aligned} \|r'_{k+1}\|^2_{Z_{k+1}} &= r'^{T}_{k+1} Z_{k+1} r'_k \\ &= r'^{T}_k Z_{k+1} r'_k + \lambda_k r'^{T}_k Z_{k+1}(r_{k+1} - r'_k), \end{aligned}$$

et par l'équation (3.6) et la définition de la direction de recherche d_k nous obtenons :

$$\|r'_{k+1}\|^2_{Z_{k+1}} = \|r'_k\|^2_{Z_{k+1}} - \frac{r'^{T}_k Z_{k+1} d_k}{d_k^T Z_{k+1} d_k} r'^{T}_k Z_{k+1} d_k.$$

■

Propriété 7

Soit r'_k la suite obtenue par le "smoothing" variable appliqué à la suite r_k et soit $d_k = r_{k+1} - r'_k$ le vecteur direction de recherche alors :

$$r'_{k+1} = \mathcal{D}_k r'_k, \tag{3.13}$$

avec

$$\mathcal{D}_k = I_n^{(n)} - \frac{d_k d_k^T Z_{k+1}}{d_k^T Z_{k+1} d_k}. \tag{3.14}$$

De plus, $\mathcal{D}_k$ est un projecteur vérifiant $\mathcal{D}_k d_k = 0$ et $\mathcal{D}_k v = v$ pour tout $v \perp_{Z_{k+1}} d_k$.

Preuve :

Par définition nous avons :

$$\begin{aligned} r'_{k+1} &= r'_k + \lambda_k d_k \\ &= r'_k - \frac{r'^{T}_k Z_{k+1} d_k}{d_k^T Z_{k+1} d_k} d_k \\ &= r'_k - d_k \frac{d_k^T Z_{k+1} r'_k}{d_k^T Z_{k+1} d_k} \\ &= (I_n^{(n)} - \frac{d_k d_k^T Z_{k+1}}{d_k^T Z_{k+1} d_k}) r'_k. \end{aligned}$$

De plus, soit $v \in \mathbb{R}^n$ alors

$$\begin{aligned} \mathcal{D}_k v &= (I_n^{(n)} - \frac{d_k d_k^T Z_{k+1}}{d_k^T Z_{k+1} d_k}) v \\ &= v - d_k \frac{d_k^T Z_{k+1} v}{d_k^T Z_{k+1} d_k}, \end{aligned}$$

ainsi, si $v \perp_{Z_{k+1}} d_k$ alors $\mathcal{D}_k v = v$ et pour $v = d_k$ nous avons $\mathcal{D}_k v = d_k - d_k = 0$.

■

Propriété 8

Soit r'_k la suite obtenue par la procédure de "smoothing" variable appliquée à la suite r_k. Les vecteurs direction de recherche $d_k = r_{k+1} - r'_k$ et $\hat{d}_k = \eta d_k$ produisent le même itéré r'_{k+1}.

Preuve :

Soit $\hat{\mathcal{D}}_k = I_n^{(n)} - \dfrac{\hat{d}_k \hat{d}_k^T Z_{k+1}}{\hat{d}_k^T Z_{k+1} \hat{d}_k}$ alors $\hat{\mathcal{D}}_k = I_n^{(n)} - \dfrac{\eta^2 d_k d_k^T Z_{k+1}}{\eta^2 d_k^T Z_{k+1} d_k} = \mathcal{D}_k$.

■

3.2 Liens entre les méthodes GAL et MRSe

Dans ce paragraphe nous allons établir certaines relations entre les itérés des deux méthodes discutées au cours du chapitre 2, à savoir la méthode de Galerkin (GAL) et la méthode de Semi-minimisation du résidu (MRSe). Commençons alors par rappeler quelques relations dont nous aurons besoin dans la suite.

Les itérés de la méthode GAL respectivement MRSe s'écrivent :

$$x_k^{\mathrm{GAL}} = x_0 + B_k H_k^{-1} e_1^{(k)} \tag{3.15}$$

$$x_k^{\mathrm{MRSe}} = x_0 + B_k \widetilde{H}_k^{+} e_1^{(k+1)}, \tag{3.16}$$

avec $\widetilde{H}_k$ et B_k les deux matrices construites par le processus de Hessenberg Généralisé et vérifiant :

$$AB_k = B_k \widetilde{H}_k. \tag{3.17}$$

Notons que pour plus de simplicité dans les calculs, nous prendrons le facteur normalisant le vecteur b_{k+1} égal à l'unité (i.e. $h_{k+1,k} = 1$).

Partitionnons la matrice $\widetilde{H}_k$ sous la forme :

$$\widetilde{H}_k = \begin{pmatrix} \widetilde{H}_{k-1} & \tilde{h}_k \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} H_k \\ e_k^{(k)T} \end{pmatrix} \in \mathbb{R}^{(k+1)\times k}, \tag{3.18}$$

où $\widetilde{h}_k = (h_{1,k}, \ldots, h_{k,k})^T$.

Notons que les itérés de la méthode MRSe sont déterminés par la condition de semi-minimisation :

$$\min_{x \in x_0 + K_k(r_0, A)} |f - Ax|_{Z_k}. \qquad (\mathcal{M}_0)$$

où $Z_k = (B_{k+1}^L)^T B_{k+1}^L$. La matrice Z_k étant définie positive sur $K_{k+1}(r_0, A)$, elle définit donc une norme sur $K_{k+1}(r_0, A)$ et par suite la condition $(\mathcal{M}_0)$ est équivalente à :

$$r_k^{\mathrm{MRSe}} \perp_{Z_k} AK_k(r_0, A). \qquad (3.19)$$

Maintenant, en utilisant les formules (8) et (16) données dans [24], nous obtenons :

$$H_k^{-1} = (\widetilde{H}_{k-1}, \widetilde{h}_k)^{-1} = (\widetilde{H}_{k-1}, \widetilde{h}_k)^+ = \begin{pmatrix} \widetilde{H}_{k-1}^+ - \widetilde{H}_{k-1}^+ \widetilde{h}_k \delta_k^T \\ \delta_k^T \end{pmatrix}, \qquad (3.20)$$

avec

$$\delta_k = \frac{u_k}{\|u_k\|_2^2} \quad \text{et} \quad u_k = (I_k^{(k)} - \widetilde{H}_{k-1} \widetilde{H}_{k-1}^+) \widetilde{h}_k. \qquad (3.21)$$

En utilisant la relation (3.20) adaptée à l'indice $k+1$, nous obtenons le résultat suivant :

Lemme 10

Soit t_k le vecteur de $\mathbb{R}^n$ suivant :

$$t_k = b_{k+1} - B_k \widetilde{H}_k^+ \widetilde{h}_{k+1},$$

alors

$$x_{k+1}^{\mathrm{GAL}} = x_k^{\mathrm{MRSe}} + \theta_k t_k$$

et

$$r_{k+1}^{\mathrm{GAL}} = r_k^{\mathrm{MRSe}} - \theta_k A t_k,$$

où $\theta_k = \delta_{k+1}^T e_1^{(k+1)}$.

Preuve :

En écrivant (3.20) et (3.21) pour l'indice $k+1$ nous avons :

$$\begin{aligned} x_{k+1}^{\mathrm{GAL}} &= x_0 + (B_k, b_{k+1}) \begin{pmatrix} \widetilde{H}_k^+ e_1^{(k+1)} - \widetilde{H}_k^+ \tilde{h}_{k+1} \delta_{k+1}^T e_1^{(k+1)} \\ \delta_{k+1}^T e_1^{(k+1)} \end{pmatrix} \\ &= x_0 + B_k \widetilde{H}_k^+ e_1^{(k+1)} + \theta_k(-B_k \widetilde{H}_k^+ \tilde{h}_{k+1} + b_{k+1}) \\ &= x_k^{\mathrm{MRSe}} + \theta_k t_k, \end{aligned}$$

où

$$\theta_k = \delta_{k+1}^T e_1^{(k+1)} \text{ et } t_k = b_{k+1} - B_k \widetilde{H}_k^+ \tilde{h}_{k+1}.$$

De même

$$r_{k+1}^{\mathrm{GAL}} = r_k^{\mathrm{MRSe}} - \theta_k A t_k.$$

■

Nous obtenons un résultat analogue pour x_{k+1}^{MRSe} et x_k^{MRSe} en calculant $\widetilde{H}_{k+1}^+$ en fonction de $\widetilde{H}_k^+$.

Lemme 11

Soit λ_k le scalaire suivant :

$$\lambda_k = \frac{\|u_{k+1}\|_2^2}{1 + \|u_{k+1}\|_2^2},$$

alors

$$x_{k+1}^{\mathrm{MRSe}} = x_k^{\mathrm{MRSe}} + \lambda_k \theta_k t_k$$

et

$$r_{k+1}^{\mathrm{MRSe}} = r_k^{\mathrm{MRSe}} - \lambda_k \theta_k A t_k.$$

Peuve :

Comme $\widetilde{H}_{k+1} = (C_k, c_{k+1})$ où $C_k = \begin{pmatrix} \widetilde{H}_k \\ 0 \end{pmatrix}$ and $c_{k+1} = \begin{pmatrix} \tilde{h}_{k+1} \\ 1 \end{pmatrix}$, nous avons

$$\widetilde{H}_{k+1}^+ = \begin{pmatrix} C_k^+ - C_k^+ c_{k+1} \gamma_{k+1}^T \\ \gamma_{k+1}^T \end{pmatrix}$$

avec

$$\gamma_{k+1} = \frac{(I_{k+2}^{(k+2)} - C_k C_k^+) c_{k+1}}{\|(I_{k+2}^{(k+2)} - C_k C_k^+) c_{k+1}\|_2^2}.$$

Or $C_k^+ = (\widetilde{H}_k^+, 0)$, donc

$$(I_{k+2}^{(k+2)} - C_k C_k^+)c_{k+1} = \begin{pmatrix} I_{k+1}^{(k+1)} - \widetilde{H}_k \widetilde{H}_k^+ & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \tilde{h}_{k+1} \\ 1 \end{pmatrix} = \begin{pmatrix} u_{k+1} \\ 1 \end{pmatrix},$$

ce qui donne

$$\gamma_{k+1} = \frac{\begin{pmatrix} u_{k+1} \\ 1 \end{pmatrix}}{1 + \|u_{k+1}\|_2^2}.$$

En calculant x_{k+1}^{MRSe} par (3.16) et en écrivant $B_{k+1} = (B_k, b_{k+1})$, il vient que :

$$\begin{aligned}
x_{k+1}^{\text{MRSe}} &= x_0 + (B_k, b_{k+1}) \begin{pmatrix} (\widetilde{H}_k^+, 0) - (\widetilde{H}_k^+, 0) \begin{pmatrix} \tilde{h}_{k+1} \\ 1 \end{pmatrix} \gamma_{k+1}^T \\ \gamma_{k+1}^T \end{pmatrix} e_1^{(k+2)} \\
&= x_0 + (B_k, b_{k+1}) \begin{pmatrix} (\widetilde{H}_k^+, 0) e_1^{(k+2)} \\ 0 \end{pmatrix} + \gamma_{k+1}^T e_1^{(k+2)} (B_k, b_{k+1}) \begin{pmatrix} -\widetilde{H}_k^+ \tilde{h}_{k+1} \\ 1 \end{pmatrix} \\
&= x_0 + B_k \widetilde{H}_k^+ e_1^{(k+1)} + \gamma_{k+1}^T e_1^{(k+2)} (B_k, b_{k+1}) \begin{pmatrix} -\widetilde{H}_k^+ \tilde{h}_{k+1} \\ 1 \end{pmatrix} \\
&= x_k^{\text{MRSe}} + \gamma_{k+1}^T e_1^{(k+2)} (b_{k+1} - B_k \widetilde{H}_k^+ \tilde{h}_{k+1}) \\
&= x_k^{\text{MRSe}} + \gamma_{k+1}^T e_1^{(k+2)} t_k,
\end{aligned}$$

or

$$\gamma_{k+1}^T e_1^{(k+2)} = \frac{u_{k+1}^T e_1^{(k+1)}}{1 + \|u_{k+1}\|_2^2},$$

et

$$\theta_k = \delta_{k+1}^T e_1^{(k+1)} = \frac{u_{k+1}^T e_1^{(k+1)}}{\|u_{k+1}\|_2^2}$$

ainsi

$$\gamma_{k+1}^T e_1^{(k+2)} = \frac{\theta_k \|u_{k+1}\|_2^2}{1 + \|u_{k+1}\|_2^2}.$$

En posant

$$\lambda_k = \frac{\|u_{k+1}\|_2^2}{1 + \|u_{k+1}\|_2^2},$$

nous obtenons

$$x_{k+1}^{\text{MRSe}} = x_k^{\text{MRSe}} + \lambda_k \theta_k t_k$$

et

$$r_{k+1}^{\text{MRSe}} = r_k^{\text{MRSe}} - \lambda_k \theta_k A t_k.$$

■

Les résultats des deux lemmes précédents permettent d'établir le théorème suivant qui constitue le principal résultat de ce chapitre.

Théorème 22

Les itérés x_k^{GAL}, x_k^{MRSe} *ainsi que leur vecteur résidu* r_k^{GAL}, r_k^{MRSe} *sont tels que :*

$$x_{k+1}^{\mathrm{MRSe}} - x_k^{\mathrm{MRSe}} = \lambda_k (x_{k+1}^{\mathrm{GAL}} - x_k^{\mathrm{MRSe}})$$

et

$$r_{k+1}^{\mathrm{MRSe}} - r_k^{\mathrm{MRSe}} = \lambda_k (r_{k+1}^{\mathrm{GAL}} - r_k^{\mathrm{MRSe}}),$$

avec

$$\lambda_k = -\frac{(r_k^{\mathrm{MRSe}}, r_{k+1}^{\mathrm{GAL}} - r_k^{\mathrm{MRSe}})_{Z_{k+1}}}{(r_{k+1}^{\mathrm{GAL}} - r_k^{\mathrm{MRSe}}, r_{k+1}^{\mathrm{GAL}} - r_k^{\mathrm{MRSe}})_{Z_{k+1}}}.$$

Preuve :

Les relations établies dans les deux lemmes précédents permettent d'écrire :

$$x_{k+1}^{\mathrm{MRSe}} - x_k^{\mathrm{MRSe}} = \lambda_k (x_{k+1}^{\mathrm{GAL}} - x_k^{\mathrm{MRSe}})$$

et

$$r_{k+1}^{\mathrm{MRSe}} - r_k^{\mathrm{MRSe}} = \lambda_k (r_{k+1}^{\mathrm{GAL}} - r_k^{\mathrm{MRSe}}),$$

ainsi

$$\lambda_k = \frac{(r_{k+1}^{\mathrm{MRSe}} - r_k^{\mathrm{MRSe}}, r_{k+1}^{\mathrm{GAL}} - r_k^{\mathrm{MRSe}})_{Z_{k+1}}}{(r_{k+1}^{\mathrm{GAL}} - r_k^{\mathrm{MRSe}}, r_{k+1}^{\mathrm{GAL}} - r_k^{\mathrm{MRSe}})_{Z_{k+1}}},$$

or

$$r_{k+1}^{\mathrm{GAL}} - r_k^{\mathrm{MRSe}} = -\theta_k A t_k \quad \text{et} \quad t_k \in K_{k+1}(r_0, A),$$

donc $r_{k+1}^{\mathrm{GAL}} - r_k^{\mathrm{MRSe}} \in AK_{k+1}(r_0, A)$ et grâce à la condition d'orthogonalité (3.19) nous avons :

$$r_{k+1}^{\mathrm{MRSe}} \perp_{Z_{k+1}} AK_{k+1}(r_0, A)$$

et donc

$$(r_{k+1}^{\mathrm{MRSe}}, r_{k+1}^{\mathrm{GAL}} - r_k^{\mathrm{MRSe}})_{Z_{k+1}} = 0.$$

Finalement nous obtenons :

$$\lambda_k = -\frac{(r_k^{\mathrm{MRSe}}, r_{k+1}^{\mathrm{GAL}} - r_k^{\mathrm{MRSe}})_{Z_{k+1}}}{(r_{k+1}^{\mathrm{GAL}} - r_k^{\mathrm{MRSe}}, r_{k+1}^{\mathrm{GAL}} - r_k^{\mathrm{MRSe}})_{Z_{k+1}}}.$$

■

Le théorème 22 montre donc que la méthode MRSe est obtenue en appliquant l'algorithme SMRS à la méthode GAL en choisissant $Z_{k+1} = (B_{k+2}^T)^L B_{k+2}^T$.

Avant d'appliquer le résultat du théorème aux principaux cas particuliers des méthodes GAL et MRSe, donnons quelques propriétés vérifiées par la Z_k-semi-norme et les résidus de ces méthodes.

3.3 Quelques propriétés de la Z_k-semi-norme

Avant d'établir quelques propriétés de la semi-norme associée à la matrice Z_k, nous avons besoin de donner l'expression de B_{k+1}^L en fonction de B_k^L. Rappelons que nous avons :

$$B_{k+1} = (B_k, b_{k+1}), \quad Y_{k+1} = (Y_k, y_{k+1}), \quad et \quad Y_k^T b_{k+1} = 0.$$

En calculant $B_{k+1}^L = (Y_{k+1}^T B_{k+1})^{-1} Y_{k+1}^T$ nous obtenons :

$$B_{k+1}^L = \begin{pmatrix} B_k^L \\ \dfrac{y_{k+1}^T (I_n^{(n)} - B_k B_k^L)}{y_{k+1}^T b_{k+1}} \end{pmatrix}. \tag{3.22}$$

Ainsi, il est possible d'exprimer Z_k en fonction de Z_{k-1}, en effet :

$$Z_k = (B_{k+1}^L)^T B_{k+1}^L = Z_{k-1} + \frac{\tilde{y}_k \tilde{y}_k^T}{(y_{k+1}^T b_{k+1})^2}. \tag{3.23}$$

avec $\tilde{y}_k = (I_n^{(n)} - B_k B_k^L)^T y_{k+1}$.

Le résultat du lemme suivant est particulièrement utile pour établir les autres résultats qui vont suivre.

Lemme 12

La matrice $I_n^{(n)} - B_k B_k^L$ est un projecteur verifiant :

$$\forall \ v \ \in \ K_k(r_0, A) \ \ (I_n^{(n)} - B_k B_k^L) v = 0.$$

Preuve :

Rappelons que d'après la définition de l'inverse à gauche de B_k^L, nous avons (§1.1 definition 4) $B_k B_k^L B_k = B_k$, ainsi

$$(I_n^{(n)} - B_k B_k^L)^2 = I_n^{(n)} - B_k B_k^L - B_k B_k^L + B_k B_k^L B_k B_k^L = I_n^{(n)} - B_k B_k^L.$$

Soit v un vecteur de $K_k(r_0, A)$, il existe $s \in \mathbb{R}^n$ tel que $v = B_k s$, d'où

$$(I_n^{(n)} - B_k B_k^L) v = (I_n^{(n)} - B_k B_k^L) B_k s = B_k s - B_k B_k^L B_k s = B_k s - B_k s = 0.$$

■

La relation (3.22) nous permet d'établir le résultat suivant :

Propriété 9

Les semi-normes $|.|_{Z_k}$ *et* $|.|_{Z_{k+i}}$, $i = 1, 2, \ldots,$ *vérifient :*

1. $|x|_{Z_k} = 0, \ \forall\ x \in span\{b_{k+2}, \ldots, b_n\}$.
2. $|x|_{Z_k} = |x|_{Z_{k+i}}, \ \forall\ x \in K_{k+1}(r_0, A)$.

Preuve :

1. Le résultat s'obtient immédiatement en remarquant que chaque élément de la famille $\{b_{k+2}, \ldots, b_n\}$ est orthogonal à Y_{k+1}, i.e. $b_i \perp Y_{k+1}, \ \forall\ i = k+2, \ldots, n$.
2. Il suffit de montrer que les semi-normes $|.|_{Z_k}$ et $|.|_{Z_{k+1}}$ sont égales sur $K_{k+1}(r_0, A)$. Soit $x \in K_{k+1}(r_0, A)$, alors par le lemme 3 nous avons $(I_n^{(n)} - B_{k+1}B_{k+1}^L)x = 0$ et donc par la relation (3.22) nous obtenons :

$$B_{k+2}^L x = (B_{k+1}^L x, 0)^T,$$

c'est à dire que $\forall x \in K_{k+1}(r_0, A), |x|_{Z_{k+1}} = |x|_{Z_k}$.

■

Les résultats suivants découlent immédiatement de la propriété précédente.

Propriété 10

Les résidus des méthodes MRSe et GAL vérifient :

1. $|r_{k+i}^{\mathrm{GAL}}|_{Z_k} = 0, \quad pour \ \ i = 1, \ldots, k.$
2. $|r_{k+1}^{\mathrm{GAL}}|_{Z_i} = 0, \quad pour \ \ i = 0, \ldots, k.$
3. $|r_{k+1}^{\mathrm{GAL}}|_{Z_{k+1}} = \dfrac{\|r_{k+1}^{\mathrm{GAL}}\|_2}{\|b_{k+2}\|_2}.$
4. $|r_k^{\mathrm{GAL}}|_{Z_k} = |r_k^{\mathrm{GAL}}|_{Z_{k+i}}$ *et* $|r_k^{\mathrm{MRSe}}|_{Z_k} = |r_k^{\mathrm{MRSe}}|_{Z_{k+i}}$, *pour* $i = 0, \ldots, k$.

Preuve :

1. Soit $i \geq 1$, le résidu r_{k+i}^{GAL} étant orthogonal à Y_{k+1}, il vient alors que

$$B_{k+1}^L r_{k+i}^{\text{GAL}} = (Y_{k+1}^T B_{k+1})^{-1} Y_{k+1}^T r_{k+i}^{\text{GAL}} = 0.$$

2. Le résidu r_{k+1}^{GAL} étant orthogonal à Y_{k+1}, il est orthogonal à Y_i pour $1 \leq i \leq k+1$, ainsi

$$\|B_{i+1}^L r_{k+1}^{\text{GAL}}\|_2 = 0, \quad \text{pour} \quad i = 0, \ldots, k.$$

3. D'après la propriété 1 du chapitre 2, r_{k+1}^{GAL} est égal à b_{k+2} à une constante multiplicative près ϱ^{GAL} ($r_{k+1}^{\text{GAL}} = \varrho^{\text{GAL}} b_{k+2}$, avec $\varrho^{\text{GAL}} = -h_{k+2,k+1} d_{k+1}^{\text{GAL}\,T} e_{k+1}^{(k+1)}$), or pour $i = 1, \ldots, k+1$, $B_i^L r_{k+1}^{\text{GAL}} = 0$ et d'après (3.22) il vient que :

$$B_{k+2}^L r_{k+1}^{\text{GAL}} = \begin{pmatrix} B_{k+1}^L r_{k+1}^{\text{GAL}} \\ \dfrac{y_{k+2}^T (I_n^{(n)} - B_{k+1} B_{k+1}^L) r_{k+1}^{\text{GAL}}}{y_{k+2}^T b_{k+2}} \end{pmatrix} \tag{3.24}$$

$$= \begin{pmatrix} 0 \\ \dfrac{y_{k+2}^T r_{k+1}^{\text{GAL}}}{y_{k+2}^T b_{k+2}} \end{pmatrix}. \tag{3.25}$$

En calculant la norme euclidienne, nous obtenons :

$$|r_{k+1}^{\text{GAL}}|_{Z_{k+1}} = \frac{|y_{k+2}^T r_{k+1}^{\text{GAL}}|}{|y_{k+2}^T b_{k+2}|} = \frac{|\varrho^{\text{GAL}}| \|y_{k+2}^T b_{k+2}|}{|y_{k+2}^T b_{k+2}|} = |\varrho^{\text{GAL}}| = \frac{\|r_{k+1}^{\text{GAL}}\|_2}{\|b_{k+2}\|_2}.$$

4. Les résidus r_k^{GAL} et r_k^{MRSe} sont dans $K_{k+1}(r_0, A)$, or le second résultat de la propriété précédente montre que les normes $|.|_{Z_{k+i}}$, $i = 0, 1, \cdots$ sont égales sur $K_{k+1}(r_0, A)$.

■

Maintenant, nous avons besoin d'établir le lemme suivant qui sera utile dans la suite.

Lemme 13

Soient r_{k+1}^{GAL} et r_k^{MRSe} les résidus des méthodes GAL et MRSe, alors le vecteur r_{k+1}^{GAL} est Z_{k+j}-orthogonal à r_k^{MRSe}, pour $j = 1, 2, \ldots,$.

Preuve :

Comme r_{k+1}^{GAL} et r_k^{MRSe} sont des éléments du sous espace de Krylov $K_{k+2}(r_0, A)$ et puisque les normes $|.|_{Z_{k+1}}$ et $|.|_{Z_{k+j}}$ sont égales sur ce même sous espace, il suffit de montrer que r_{k+1}^{GAL} est Z_{k+1}-orthogonal à r_k^{MRSe}.

D'après la relation (3.23), nous avons :

$$r_{k+1}^{\mathrm{GAL}^T} Z_{k+1} r_k^{\mathrm{MRSe}} = r_{k+1}^{\mathrm{GAL}^T} Z_k r_k^{\mathrm{MRSe}} + \frac{r_{k+1}^{\mathrm{GAL}^T} \tilde{y}_{k+1} \tilde{y}_{k+1}^T r_k^{\mathrm{MRSe}}}{(y_{k+2}^T b_{k+2})^2}.$$

D'une part, nous avons $\tilde{y}_{k+1}^T r_k^{\mathrm{MRSe}} = 0$ puisque $r_k^{\mathrm{MRSe}} \in K_{k+1}(r_0, A)$ et d'après le lemme 3 $(I_n^{(n)} - B_{k+1} B_{k+1}^L) r_k^{\mathrm{MRSe}} = 0$.

D'autre part $B_{k+1}^L r_{k+1}^{\mathrm{GAL}} = 0$ et donc $r_{k+1}^{\mathrm{GAL}^T} Z_{k+1} r_k^{\mathrm{MRSe}} = 0$. D'où $r_{k+1}^{\mathrm{GAL}} \perp_{Z_{k+1}} r_k^{\mathrm{MRSe}}$.

■

Propriété 11

Les résidus r_k^{MRSe}, r_{k+1}^{MRSe} et r_{k+1}^{GAL} vérifient :

$$\frac{1}{|r_{k+1}^{\mathrm{MRSe}}|_{Z_{k+j}}^2} = \frac{1}{|r_k^{\mathrm{MRSe}}|_{Z_{k+j}}^2} + \frac{1}{|r_{k+1}^{\mathrm{GAL}}|_{Z_{k+j}}^2}, \quad \textit{pour } j = 1, 2, \ldots, . \tag{3.26}$$

Preuve :

Comme il y a égalité des normes $|.|_{Z_{k+j}}$ sur $K_{k+2}(r_0, A)$, pour $j = 1, 2, \ldots,$ il suffit de montrer que le résultat est vrai pour $j = 1$. D'après la propriété 6 nous avons :

$$\begin{aligned}
|r_{k+1}^{\mathrm{MRSe}}|_{Z_{k+1}}^2 &= |r_k^{\mathrm{MRSe}}|_{Z_{k+1}}^2 - \frac{\left(r_k^{\mathrm{MRSe}^T} Z_{k+1}(r_{k+1}^{\mathrm{GAL}} - r_k^{\mathrm{MRSe}})\right)^2}{(r_{k+1}^{\mathrm{GAL}} - r_k^{\mathrm{MRSe}})^T Z_{k+1}(r_{k+1}^{\mathrm{GAL}} - r_k^{\mathrm{MRSe}})} \\
&= |r_k^{\mathrm{MRSe}}|_{Z_{k+1}}^2 - \frac{\left(r_k^{\mathrm{MRSe}^T} Z_{k+1} r_k^{\mathrm{MRSe}}\right)^2}{r_{k+1}^{\mathrm{GAL}^T} Z_{k+1} r_{k+1}^{\mathrm{GAL}} + r_k^{\mathrm{MRSe}^T} Z_{k+1} r_k^{\mathrm{MRSe}}} \\
&= |r_k^{\mathrm{MRSe}}|_{Z_{k+1}}^2 - \frac{|r_k^{\mathrm{MRSe}}|_{Z_{k+1}}^2}{|r_{k+1}^{\mathrm{GAL}}|_{Z_{k+1}}^2 + |r_k^{\mathrm{MRSe}}|_{Z_{k+1}}^2} \\
&= \frac{|r_k^{\mathrm{MRSe}}|_{Z_{k+1}}^2 |r_{k+1}^{\mathrm{GAL}}|_{Z_{k+1}}^2}{|r_{k+1}^{\mathrm{GAL}}|_{Z_{k+1}}^2 + |r_k^{\mathrm{MRSe}}|_{Z_{k+1}}^2}.
\end{aligned}$$

La seconde égalité s'obtient en utilisant le résultat du lemme 12.

■

En appliquant itérativement le résultat de la propriété 11, et en utilisant l'égalité des normes nous obtenons le théorème suivant :

Théorème 23

Soit r_{k+1}^{MRSe} *et* r_i^{GAL}, $i = 0, \ldots, k+1$, *les résidus des méthodes GAL et MRSe. Alors :*

$$\frac{1}{|r_{k+1}^{\mathrm{MRSe}}|_{Z_{k+1}}^2} = \sum_{i=0}^{k+1} \frac{1}{|r_i^{\mathrm{GAL}}|_{Z_i}^2}. \tag{3.27}$$

La classe des méthodes orthogonales est une classe de méthodes itératives pour la résolution d'uns système linéaire. Elle est caractérisée par l'orthogonalité des résidus, i.e. les résidus verifient $r_i^T Z r_j = 0$ pour $i \neq j$, où Z est une matrice symétrique définie positive. Le résultat du théorème 23 a été établi par Weiss pour cette classe de méthodes. En fait, Weiss [56, 57] a montré que si r_k^{MR} est la suite des résidus obtenus par application de l'algorithme MRS à la suite des résidus r_k^{OR} d'une méthode orthogonale, alors les relations suivantes sont équivalentes.

$$\|r_{k+1}^{\mathrm{MR}}\|_Z = \min_{\lambda_k} \|r_k^{\mathrm{MRSe}} + \lambda_k (r_{k+1}^{\mathrm{OR}} - r_k^{\mathrm{MR}})\|_Z \tag{3.28}$$

$$\|r_{k+1}^{\mathrm{MR}}\|_Z = \min_{\lambda_1, \cdots, \lambda_k} \|r_k^{\mathrm{MR}} + \sum_{i=0}^{k+1} \lambda_i (r_i^{\mathrm{OR}} - r_{i-1}^{\mathrm{MR}})\|_Z \tag{3.29}$$

$$r_{k+1}^{\mathrm{MR}} = \frac{1}{\sum_{i=0}^{k+1} \frac{1}{\|r_i^{\mathrm{OR}}\|_Z^2}} \sum_{j=0}^{k+1} \frac{1}{\|r_j^{\mathrm{OR}}\|_Z^2} r_j^{\mathrm{OR}} \tag{3.30}$$

$$\frac{1}{\|r_{k+1}^{\mathrm{MR}}\|_Z^2} = \sum_{i=0}^{k+1} \frac{1}{\|r_i^{\mathrm{OR}}\|_Z^2}. \tag{3.31}$$

L'orthogonalité des résidus est nécessaire pour établir l'orthogonalité entre le résidu r_{k+1}^{MR} et les vecteurs de direction $d_j = r_{j+1}^{\mathrm{OR}} - r_j^{\mathrm{MR}}$ $j \leq k$ (§lemme 4.12 [56, p.77]), ce qui permet d'obtenir l'équivalence entre les deux premières équations.

Dans notre cas, r_{k+1}^{MRSe} est Z_{k+1}-orthogonal au sous espace $AK_{k+1}(r_0, A)$ et comme $d_j = r_{j+1}^{\mathrm{GAL}} - r_j^{\mathrm{MRSe}}$ est un élément de $AK_{j+1}(r_0, A)$, nous avons donc :

Lemme 14

Soit r_{k+1}^{MRSe} *respectivement* $d_j = r_{j+1}^{\mathrm{GAL}} - r_j^{\mathrm{MRSe}}$, $j = 0, \ldots, k$ *le résidu obtenu par la méthode MRSe respectivement les vecteurs de direction alors* r_{k+1}^{MRSe} *est* Z_{k+1}*-orthogonal à* d_j, *pour* $j = 1, 2, \ldots, k$.

Le résultat du lemme précédent montre donc que l'équivalence entre les relations (3.28),(3.29) et (3.31) reste vérifiée pour les méthodes MRSe, GAL et pour la matrice $Z_{k+1} = (B_{k+1}^L)^T B_{k+1}^L$.

Le résultat du lemme suivant nous permet d'établir que la relation (3.30) est elle aussi valable pour les deux méthodes considérées GAL et MRSe.

Lemme 15

Les résidus de la méthode de Galerkin vérifient

$$r_i^{\mathrm{GAL}\,T} Z_{k+1} r_j^{\mathrm{GAL}} = 0 \quad \forall \; i \neq j \leq k, \tag{3.32}$$

i.e. les résidus r_i^{GAL} $(i \leq k)$ *sont* Z_{k+1}*-orthogonaux.*

Preuve :

D'après (3.24) nous avons

$$\begin{aligned} B_{k+2}^L r_i^{\mathrm{GAL}} &= \begin{pmatrix} B_{k+1}^L r_i^{\mathrm{GAL}} \\ \dfrac{y_{k+2}^T (I_n^{(n)} - B_{k+1} B_{k+1}^L) r_i^{\mathrm{GAL}}}{y_{k+2}^T b_{k+2}} \end{pmatrix} \\ &= \begin{pmatrix} B_{k+1}^L r_i^{\mathrm{GAL}} \\ 0 \end{pmatrix}. \end{aligned}$$

La seconde égalité est une conséquence du lemme 12, ainsi il est facile de voir que :

$$\begin{aligned} B_{k+2}^L r_i^{\mathrm{GAL}} &= \begin{pmatrix} B_i^L r_i^{\mathrm{GAL}} \\ \dfrac{y_{i+1}^T (I_n^{(n)} - B_i B_i^L) r_i^{\mathrm{GAL}}}{y_{i+1}^T b_{i+1}} \\ 0 \\ \vdots \\ 0 \end{pmatrix} \\ &= \begin{pmatrix} 0 \\ \vdots \\ 0 \\ \dfrac{y_{i+1}^T r_i^{\mathrm{GAL}}}{y_{i+1}^T b_{i+1}} \\ 0 \\ \vdots \\ 0 \end{pmatrix} \end{aligned}$$

car $B_i^L r_i^{\mathrm{GAL}} = 0$. Remarquons que toutes les composantes du vecteur $B_{k+2}^L r_i^{\mathrm{GAL}}$ sont nulles sauf la composante $i+1$ qui est égale à $\dfrac{\|r_i\|_2}{\|b_{i+1}\|_2}$. Il est alors clair que le Z_{k+1}-produit scalaire de r_i^{GAL} par r_j^{GAL} pour $i \neq j$ et $i, j \leq k+1$ est nul.

■

Ainsi nous obtenons le théorème suivant :

Théorème 24

Les résidus des méthodes GAL et MRSe vérifient :

$$|r_{k+1}^{\mathrm{MRSe}}|_{Z_{k+1}} = \min_{\lambda_k} |r_k^{\mathrm{MRSe}} + \lambda_k (r_{k+1}^{\mathrm{GAL}} - r_k^{\mathrm{MRSe}})|_{Z_{k+1}} \tag{3.33}$$

$$|r_{k+1}^{\mathrm{MRSe}}|_{Z_{k+1}} = \min_{\lambda_1,\cdots,\lambda_k} \left| r_k^{\mathrm{MRSe}} + \sum_{i=0}^{k+1} \lambda_i (r_i^{\mathrm{GAL}} - r_{i-1}^{\mathrm{MRSe}}) \right|_{Z_{k+1}} \tag{3.34}$$

$$r_{k+1}^{\mathrm{MRSe}} = \frac{1}{\sum_{i=0}^{k+1} \frac{1}{|r_i^{\mathrm{GAL}}|_{Z_i}^2}} \sum_{j=0}^{k+1} \frac{1}{|r_j^{\mathrm{GAL}}|_{Z_j}^2} r_j^{\mathrm{GAL}} \tag{3.35}$$

$$\frac{1}{|r_{k+1}^{\mathrm{MRSe}}|_{Z_{k+1}}^2} = \sum_{i=0}^{k+1} \frac{1}{|r_i^{\mathrm{GAL}}|_{Z_i}^2}. \tag{3.36}$$

La relation (3.35) montre que le résidu r_{k+1}^{MRSe} est donné comme combinaison barycentrique des résidus r_i^{GAL} affectés des cœfficients α_i ($\alpha_i = \frac{1}{\|r_i^{\mathrm{GAL}}\|_{Z_i}^2} / \sum_{j=0}^{k+1} \frac{1}{\|r_j^{\mathrm{GAL}}\|_{Z_j}^2}$) pour $i = 0, \ldots, k+1$. Cette relation donne aussi un aperçu sur l'influence des normes des résidus de la méthode de Galerkin sur celle des résidus de la méhode de semi-minimisation du résidu. Si à l'itération $k+1$ le norme euclidienne de r_{k+1}^{GAL} est petite, alors r_{k+1}^{GAL} est affécté d'un poids large dans la combinaison barycentrique (3.35) est $\|r_{k+1}^{\mathrm{MRSe}}\|_2$ est petite. Par contre une croissance de la norme de r_{k+1}^{GAL} se traduit par un poids relativement petit dans la combinaison (3.35) et ainsi la croissance de $\|r_{k+1}^{\mathrm{MRSe}}\|_2$ est relativement petite.

Les équations (3.26), (3.27) et (3.35) sont équivalentes, mais donnent deux aspects presque distincts sur la façon dont sont liées les normes $\|r_{k+1}^{\mathrm{GAL}}\|_2$ et $\|r_{k+1}^{\mathrm{MRSe}}\|_2$. Les équations (3.27) et (3.35) expliquent clairement la dépendance de r_{k+1}^{MRSe} par rapport à $r_0^{\mathrm{GAL}}, \ldots, r_{k+1}^{\mathrm{GAL}}$; notons que la norme $\|r_{k+1}\|_2^{\mathrm{MRSe}}$ est petite si et seulement si une des normes $\|r_0\|_2^{\mathrm{GAL}}, \ldots, \|r_{k+1}\|_2^{\mathrm{GAL}}$ est petite. Et en écrivant l'équation (3.26) sous la forme équivalente suivante :

$$|r_{k+1}^{\mathrm{GAL}}|_{Z_{k+1}} = \frac{|r_{k+1}^{\mathrm{MRSe}}|_{Z_{k+1}}}{\sqrt{1 - \left(|r_{k+1}^{\mathrm{MRSe}}|_{Z_{k+1}} / |r_k^{\mathrm{MRSe}}|_{Z_k}\right)^2}}, \tag{3.37}$$

nous voyons que cette équation fait apparaitre la dépendance locale de $\|r_{k+1}^{\mathrm{GAL}}\|_2$ par rapport à $\|r_k^{\mathrm{MRSe}}\|_2$ et $\|r_{k+1}^{\mathrm{MRSe}}\|_2$. Il y'a égalité entre les résidus des deux méthodes GAL et MRSe si et seulement si $\|r_{k+1}^{\mathrm{GAL}}\|_2 = \|r_{k+1}^{\mathrm{MRSe}}\|_2 = 0$, c'est à dire lorsque les deux méthodes ont atteint la solution. Si la solution n'est pas encore atteinte, le facteur $\left(|r_{k+1}^{\mathrm{MRSe}}|_{Z_{k+1}} / |r_k^{\mathrm{MRSe}}|_{Z_k}\right)^2$ qui mesure le progrés accompli à l'itération $k+1$ par la méthode

MRSe est un indicateur sur la différence qui existe entre les normes $\|r_{k+1}^{\text{MRSe}}\|_2$ et $\|r_{k+1}^{\text{MRSe}}\|_2$. De plus la relation (3.37) montre clairement, comme cela a été obsérvé pour les méthodes GMRES et FOM par Brown [8], que si une des deux méthodes (GAL et MRSe) réalise de bonnes ou mauvaises performances pour un problème donné, alors il en est de même pour l'autre méthode.

3.4 Conclusion

Dans ce chapitre, nous avons établi que la méthode MRSe peut être considérée comme une méthode obtenue par application de l'algorithme SMRS (forme particulière de la technique de "smoothing") à la méthode GAL. Cette technique est un cas particulier de la procédure de "smoothing" variable dans laquelle la matrice (et donc la norme associée) change à chaque itération. Les propriétés démontrées par Weiss [56, 57] pour l'algorithme MRS appliqué aux méthodes GCG (generalized conjugate gradient methods) restent valables pour l'algorithme SMRS appliquée à la méthode de Galerkin.

A l'aide de la formulation générale adoptée pour définir la méthode de Hessenberg Généralisée, nous pouvons retrouver des résultats précédemment établis pour les méthodes Arnoldi/GMRES et BCG/QMR. Nous concluons ce chapitre en appliquant le résultat du théorème 21 aux principaux cas particuliers de la méthode Hessenberg Généralisée.

3.4.1 Le cas Arnoldi/GMRES

Nous avons vu §2.4.1 que si la base B_{k+1} est construite par le processus d'Arnoldi, alors la Z_k-semi-norme est égale à la norme euclidienne sur $K_{k+1}(r_0, A)$. Il est alors clair que dans ce cas les algorithmes SMRS et MRS coincident lorsque ils sont appliqués à la méthode d'Arnoldi. Ainsi nous retrouvons le résultat donné par Weiss [56] : la méthode GMRES est obtenue par application de l'algorithme MRS à la méthode d'Arnoldi.

3.4.2 Le cas BCG/QMR

Si nous utilisons le processus de Lanczos (sans stratégie de look-ahead) pour générer la base B_{k+1}, la méthode GAL respectivement MRSe n'est autre que le méthode BCG respectivement QMR. Nous avons rappelé au début de ce chapitre que L. Zhou et H. F. Walker ont montré qu'en appliquant l'algorithme QMRS à la méthode BCG nous

retrouvons la méthode QMR. Le résultat du théorème 21 nous permet donc de réécrire le résultat de Zhou et Walker, précédemment cité, sous une formulation similaire à celle de Schönauer. Cette formulation diffère de la formulation classique dans le sens ou la matrice qui définit la norme est une matrice qui dépend de l'indice d'itération. Finalement nous pouvons affirmer que les algorithmes SMRS et QMRS coincident lorsque ils sont appliqués à la méthode BCG.

3.4.3 Le cas Hessenberg/CMRH

L'utilisation du processus de Hessenberg dans la méthode Hessenberg Généralisée permet de définir la méthode Hessenberg §2.1.2, respectivement la méthode CMRH §2.4.3, qui est un cas particulier de la méthode GAL, respectivement MRSe. Comme pour le cas BCG/QMR, le théorème 21 s'applique au cas Hessenberg/CMRH, c'est à dire que la méthode CMRH peut être considérée comme étant la méthode obtenue par application de l'algorithme SMRS à la méthode Hessenberg.

3.4.4 Autres cas

Il existe d'autres méthodes qui font partie de la méthode Hessenberg Généralisée pour lesquelles s'applique le résultat du théorème 21. Ces méthodes sont :

- La méthode IOM(l) (ou sa version directe DIOM(l)) qui est un cas particulier de la méthode de Galerkin §2.1.2. A cette méthode correspond la méthode QGMRES(l) [43] (où sa version directe DQGMRES(l)) §2.4.4 qui est une méthode de semi-minimisation du résidu.
- La méthode THM(l) (ou sa version directe DTHM(l)), cette méthode fait partie de la classe des méthodes de Galerkin §2.1.2. La méthode MRSe correspondante est la méthode TCMRH(l) (ou sa version directe DTCMRH(l)) §2.4.5.

Chapitre 4

La méthode de Newton Hessenberg Généralisée

4.1 Introduction

Dans le présent chapitre, nous nous intéressons à la résolution des systèmes non linéaires ayant la forme générale :

$$F(u) = 0 \tag{4.1}$$

où $F : \mathbb{R}^n \longrightarrow \mathbb{R}^n$ est une application différentiable et $u \in \mathbb{R}^n$.

Une des méthodes classiques les plus utilisées pour la résolution de tels systèmes est la méthode de Newton. Cette méthode est une méthode itérative, elle peut être considérée comme une procédure de linéarisation. L'application F est localement approchée par une fonction linéaire. Nous obtenons alors un système d'équations linéaires dont la solution nous permet de calculer l'itéré suivant.

Soient u_{k+1} et u_k deux itérés successifs de la méthode de Newton. La fonction linéaire qui approche localement F est obtenue à partir du développement en série de Taylor à l'ordre 1 de l'application F au point $x_k = u_{k+1} - u_k$.

$$F(u_{k+1}) = F(u_k + x_k) \simeq F(u_k) + F'(u_k)x_k + \cdots$$

L'itéré u_{k+1} étant construit dans l'espoir que $F(u_{k+1}) = 0$, on voit alors que l'itéré x_k doit être calculé comme solution du système linéaire :

$$F'(u_k)x = -F(u_k). \tag{4.2}$$

Ainsi l'algorithme de la méthode de Newton peut être décrit comme suit :

Algorithme : *Méthode de Newton.*

1. *Choisir* u_0 *une approximation initiale.*
2. *Pour* $k = 0, 1, \cdots$ *jusqu'à convergence :*
   résoudre $F'(u_k)x_k = -F(u_k)$;
   prendre $u_{k+1} = u_k + x_k$.

La méthode de Newton est très attractive car elle converge rapidement si nous pouvons choisir une bonne approximation u_0 de la solution exacte u_*. Notons cependant que nous devons, à chaque itération, résoudre le système linéaire (4.2).

Pour des petites valeurs de n, la résolution du système (4.2) peut être réalisée à l'aide des factorisations $\mathcal{LU}$, $\mathcal{QR}$ ou Cholesky (si la matrice Jacobienne est symétrique définie positive). Comme le coût des méthodes directes citées précédemment est de l'ordre de $\mathcal{O}(n^3)$, elles ne peuvent être utilisées lorsque la taille du système (4.1) est grande, ou lorsque l'approximation u_0 n'est pas suffisamment proche d'une solution du système (4.1). Dans ce cas, la résolution du système (4.2) doit être accomplie à l'aide de méthodes itératives telles que les méthodes étudiées au chapitre 2.

Généralement, lorsque une méthode itérative est utilisée pour la résolution d'un système linéaire, la solution x_k n'est obtenue qu'avec une certaine précision. Dans le cas du système linéaire (4.2), cette précision s'écrit : $r_k = -F(u_k) - F'(u_k)x_k$. Ainsi la méthode de Newton où on résout le système (4.2) par une telle technique fait partie de la classe des méthodes de Newton Inexactes [16, 9].

La classe des méthodes de Newton Inexactes nécessite l'utilisation d'une suite auxiliaire $\{\eta_k\}$. Cette suite permet de controler le calcul de chaque solution x_k du système linéaire associé à l'itération k de la méthode de Newton [16, 9, 10]. La question qui peut se poser alors est comment choisir $\{\eta_k\}$ de telle façon que la méthode garde le caractère de convergence locale que possède la méthode de Newton.

Dans la suite, nous considérons la méthode de Newton Inexacte dont l'algorithme est décrit comme suit :

Algorithme : *Méthode de Newton Inexacte.*

1. *Choisir* u_0 *une approximation initiale*

2. *Pour* $k = 0, 1, \cdots$ *jusqu'à convergence :*

$$\begin{aligned} &\textit{trouver } x_k \textit{ vérifiant } : F'(u_k)x_k = -F(u_k) + r_k, && (4.3)\\ &\textit{avec} \qquad \|r_k\| \leq \eta_k \|F(u_k)\|; && (4.4)\\ &\textit{prendre} \quad : \qquad u_{k+1} = u_k + x_k. \end{aligned}$$

Dans (4.4) η_k représente la précision voulue sur le calcul de x_k solution de (4.2) par une méthode itérative et $r_k = -F'(u_k)x_k - F(u_k)$ étant le résidu associé à x_k. La norme $\|.\|$ est une norme quelconque de $\mathbb{R}^n$.

Avant de caractériser l'ordre de convergence de la méthode de Newton Inexacte, nous avons besoin d'imposer quelques conditions sur l'application F. Ces conditions sont :

- Il existe u_* tel que $F(u_*) = 0$. (4.5)
- F est de classe C^1au voisinage de u_*. (4.6)
- $J(u_*) \equiv F'(u_*)$ est inversible. (4.7)

Dans [16], Dembo, Eisenstat et Steihaug ont montré que si la suite $\{\eta_k\}_{k\geq 0}$ est majorée par 1, alors la méthode de Newton Inexacte décrite par l'algorithme précédent préserve le caractère de convergence locale. De plus l'ordre de convergence des itérés de la méthode de Newton Inexacte est caractérisé par l'ordre de convergence des résidus relatifs $\{r_k/\|F(u_k)\|\}$. Plus précisément le résultat suivant est donné dans [16, 9].

Théorème 25

Soit $F : \mathbb{R}^n \longrightarrow \mathbb{R}^n$ *une application vérifiant les conditions (4.5), (4.6) et (4.7). Soit* $\{\eta_k\}$ *vérifiant* $0 \leq \eta_k \leq \eta_{max} < t < 1$. *Alors il existe* $\varepsilon > 0$ *tel que : si* $\{u_k\}$ *est une suite d'itérés donnés par la méthode de Newton Inexacte vérifiant :*

$$\|u_0 - u_*\| < \varepsilon$$

alors $\{u_k\}$ *converge linéairement vers* u_*, *c'est à dire que :*

$$\|u_{k+1} - u_*\|_* \leq t\|u_k - u_*\|_*, \qquad (4.8)$$

avec $\|u\|_* = \|F'(u_*)u\|$.

Si de plus η_k tend vers 0 alors $\{u_k\}$ converge superlinéairement vers u_*, c'est à dire que

$$\|u_{k+1} - u_*\| = o(\|u_k - u_*\|), \qquad (quand\ k \to \infty). \tag{4.9}$$

De même si F' est lipschitzienne dans un voisinage de u_* et $\eta_k = \mathcal{O}(\|F(u_k)\|^{p-1})$, alors la convergence est d'ordre p, c'est à dire que :

$$\|u_{k+1} - u_*\| = \mathcal{O}(\|u_k - u_*\|^p), \qquad (quand\ k \to \infty). \tag{4.10}$$

Le résultat donnant les conditions pour la convergence linéaire ou superlinéaire a été établi dans [16]. Le résultat concernant la convergence quadratique a été établi par P.N Brown [9].

Les méthodes itératives considérées pour la résolution du système (4.2) sont les méthodes GMRES [42] et CMRH [46]. Ce sont deux méthodes de projection sur des sous espaces de Krylov et sont des cas particuliers de la méthode de Hessenberg Généralisée étudiée et décrite dans le chapitre 2. En général les méthodes de projection sur des espaces de Krylov pour la résolution d'un système linéaire nécessitent seulement l'action de la matrice (associée au système) sur un vecteur v, ce qui nous évite le stockage de la matrice. De plus, comme nous ne savons pas toujours, ou nous ne voulons pas calculer (à chaque itération de la méthode de Newton) les n^2 éléments de la matrice $F'(u)$, on préfère remplacer l'action de la matrice jacobienne $F'(u)$ sur un vecteur v par le quotient approché :

$$F'(u)v \simeq (F(u + \sigma v) - F(u))/\sigma \tag{4.11}$$

où σ est un scalaire assez petit. Rappelons que Brown [9] a donné l'appellation suivante : *" Inexacte Newton/Finite Difference Projections Methods."* pour la méthode de Newton Inexacte où on résout le système (4.2) en y approchant le produit $F'(u)x$ par le quotient (4.11).

4.2 La méthode de Newton Hessenberg Généralisée

Dans ce paragraphe, nous nous intéressons à la résolution du système linéaire (4.2) associé à l'itération k de la méthode de Newton. Ce système peut être reformulé sous la forme usuelle suivante :

$$Ax = b, \tag{4.12}$$

où $A = J(u_k) = F'(u_k)$ et $b = -F(u_k)$.

Pour résoudre le système (4.12), nous considérons la méthode de Hessenberg Généralisée vue au chapitre 2. Cette méthode peut être décrite par l'algorithmes 8 de la

méthode de Galerkin (GAL) ou par l'algorithme 9 de la méthode de semi-minimisation de la norme du résidu (MRSe). Ces deux méthodes, rappelons le, sont des cas particuliers de la méthode de Hessenberg Généralisée. D'ou l'algorithme 10 suivant :

Algorithme 10 : *Méthode de Hessenberg Généralisée.*

1- Initialisation :
Soit x_0 *une approximation initale ;*
calculer : $r_0 = b - Ax_0$; $\beta = \|r_0\|$; $b_1 = r_0/\beta$;

2- Processus de Hessenberg Généralisé :
pour $j = 1, \ldots, m$
$\omega = Ab_j$;
pour $i = 1, \ldots, j$
$h_{i,j} = (y_i, \omega)/(y_i, b_i)$; $\omega = \omega - h_{i,j} b_i$;
fin du pour ;
$\omega_{j+1} = \omega$; $h_{j+1,j} = \|\omega_{j+1}\|$;
$b_{j+1} = \omega_{j+1}/h_{j+1,j}$;
fin du pour.

3- Former l'approximation x_m *:*
Par la méthode de Galerkin : chercher d_m *solution de :* $H_m d = \beta e_1^{(m)}$;
Par la méthode MRSe : chercher d_m *solution de :* $\min_{d \in \mathbb{R}^m} \|\beta e_1^{(m+1)} - \widetilde{H}_m d\|_2$;
calculer $x_m = x_0 + B_m d_m$.

4.2.1 Version différence finie de la méthode de Hessenberg Généralisée

L'algorithme 10 sera utilisé pour la résolution du système linéaire associé à chaque itération de la méthode de Newton. Cependant, nous avons vu qu'on préferait remplacer l'action da la matrice Jacobienne (et donc l'action de la matrice A) sur un vecteur par le quotient approché (4.11). Nous devons donc voir comment se transforment les différents itérés de l'algorithme 10 lorsque tout produit matrice vecteur Av est approché par (4.11). Ainsi, nous obtenons alors un nouvel algorithme appelé version différence finie de la méthode de Hessenberg Généralisée (DFHG). Suivant que l'on utilise la méthode de Galerkin ou la méthode MRSe, nous obtenons soit la version différence finie de la méthode de Galerkin, soit la version différence finie de la méthode MRSe.

Dans l'algorithme DFHG, nous désignerons par q_0 le quotient approché (4.11) qui ap-

proche $A\,\widehat{x}_0$, où $\widehat{x}_0$ est une solution initiale quelconque. Les vecteurs de la base construite seront notés $\widehat{b}_1, \ldots, \widehat{b}_k$. Les quotients approchant les produits matrice vecteur $A\widehat{b}_j$ seront notés q_{j+1} pour $j = 1, \ldots, m$. En résumé, les itérés de l'algorithme DFHG seront notés avec des "chapeaux" afin de les différencier de ceux de l'algorithme 10, et nous avons :

$$q_0 = \left(F(u_k + \sigma_0 \widehat{x}_0) - F(u_k)\right)/\sigma_0,$$

$$q_{j+1} = \left(F(u_k + \sigma_j \widehat{b}_j) - F(u_k)\right)/\sigma_j, \quad j = 1 \ldots, m.$$

Dans la suite et par souci de simplicité, nous allons noter u_k par u.

Algorithme 11 : *version différence finie de la méthode de Hessenberg Généralisée (DFHG) .*

1- Initialisation :

Soit $\widehat{x}_0$ *une solution initiale.*
calculer $q_0 = (F(u + \sigma_0 \widehat{x}_0) - F(u))/\sigma_0$; $\widehat{r}_0 = b - q_0$;
$\widehat{\beta} = \|\widehat{r}_0\|$; $\widehat{b}_1 = \widehat{r}_0/\widehat{\beta}$; $q_1 = \widehat{b}_1$;

2- Processus de Hessenberg Généralisé :

pour $j = 1, \ldots, m$
$q_{j+1} = (F(u + \sigma_j \widehat{b}_j) - F(u))/\sigma_j$; $\widehat{\omega} = q_{j+1}$;
pour $i = 1, \ldots, j$
$\widehat{h}_{i,j} = (y_i, \widehat{\omega})/(y_i, \widehat{b}_i)$; $\widehat{\omega} = \widehat{\omega} - \widehat{h}_{i,j}\widehat{b}_i$;
fin du pour ;
$\widehat{\omega}_{j+1} = \widehat{\omega}$; $\widehat{h}_{j+1,j} = \|\widehat{\omega}_{j+1}\|$;
$\widehat{b}_{j+1} = \widehat{\omega}_{j+1}/h_{j+1,j}$;
fin du pour.

3- Former l'approximation x_m :

Par la méthode de Galerkin : chercher $\widehat{d}_m$ *solution de :* $\widehat{H}_m d = \widehat{\beta} e_1^{(m)}$;
Par la méthode MRSe : chercher $\widehat{d}_m$ *solution de :* $\min_{d \in \mathbb{R}^m} \|\widehat{\beta} e_1^{(m+1)} - \widetilde{\widehat{H}}_m d\|_2$;
calculer $\widehat{x}_m = x_0 + \widehat{B}_m \widehat{d}_m$.

Avant de voir quelques propriétés de la version différence finie de la méthode de Hessenberg Généralisée, nous allons étudier l'analyse de la propagation de l'erreur dans la méthode de Hessenberg Généralisée dans le cas où on suppose que les plus grandes erreurs sont commises lors des produits matrice vecteur.

4.2.2 Analyse de la propagation de l'erreur dans la méthode de Hessenberg Généralisée

Nous savons que dans la pratique chaque opération arithmétique d'un algorithme donné est effectuée avec une certaine erreur. Cela est dû au fait que la précision de la machine sur laquelle est implémenté l'algorithme est une précision finie. Dans la suite, nous supposons que les plus importantes erreurs commises dans l'implémentation de l'algorithme sont commises dans les produits matrice vecteur. Les systèmes linéaires considérés dans cette section ne sont pas forcément associés à une itération de la méthode de Newton, mais sont quelconques.

Nous supposons que l'erreur ε_v commise sur chaque produit matrice vecteur Av est due au fait qu'on approche A par un operateur $\widehat{A}$, i.e. $Av = \widehat{A}v + \varepsilon_v$. Notons que l'erreur ε_v peut être due soit à la précision finie de la machine, soit à ce que l'action de A sur v est approchée par $\widehat{A}v$ (par exemple: dans le cas où A est une matrice jacobienne $F'(u)$, alors $\widehat{A}v$ peut être donné par: $\widehat{A}v = (F(u+\sigma v) - F(u))/\sigma$ ou par $\widehat{A}v = (F(u+\sigma v) - F(u-\sigma v))/2\sigma$). En utilisant les mêmes notations que dans l'algorithme 11, nous obtenons l'algorithme 12.

Algorithme 12 :

1- Initialisation :

Soit $\widehat{x}_0$ *une solution initiale.*

calculer $q_0 = \widehat{A}\widehat{x}_0$; $\widehat{r}_0 = b - q_0$; $\widehat{\beta} = \|\widehat{r}_0\|$; $\widehat{b}_1 = \widehat{r}_0/\widehat{\beta}$; $q_1 = \widehat{b}_1$;

2- Processus de Hessenberg Généralisé :

pour $j = 1, \ldots, m$

$q_{j+1} = \widehat{A}\widehat{b}_j$; $\widehat{\omega} = q_{j+1}$;

pour $i = 1, \ldots, j$

$\widehat{h}_{i,j} = (y_i, \widehat{\omega})/(y_i, \widehat{b}_i)$; $\widehat{\omega} = \widehat{\omega} - \widehat{h}_{i,j}\widehat{b}_i$;

fin du pour ;

$\widehat{\omega}_{j+1} = \widehat{\omega}$; $\widehat{h}_{j+1,j} = \|\widehat{\omega}_{j+1}\|$;

$\widehat{b}_{j+1} = \widehat{\omega}_{j+1}/h_{j+1,j}$;

fin du pour.

3- Former l'approximation x_m *:*

Par la méthode de Galerkin : chercher $\widehat{d}_m$ *solution de :* $\widehat{H}_m d = \widehat{\beta} e_1^{(m)}$;

Par la méthode MRSe : chercher $\widehat{d}_m$ *solution de :* $\min_{d \in \mathbb{R}^m} \|\widehat{\beta} e_1^{(m+1)} - \widetilde{H}_m d\|_2$;

calculer $\widehat{x}_m = x_0 + \widehat{B}_m \widehat{d}_m$.

Dans la suite, nous généralisons certains résultats donnés par Brown [9]. Ces résultats concernent la comparaison des algorithmes des méthodes FOM (respectivement GMRES) et la version différence finie de la méthode FOM (respectivement GMRES).

Supposons que m étapes soient réalisables dans l'algorithme précédent, dans ce cas les m vecteurs $q_1, \ldots, q_m$ sont linéairement indépendants. Ainsi, par application du théorème 3 (§1.1), il vient que la famille $\{\widehat{b}_1, \ldots, \widehat{b}_m\}$ construite par l'algorithme précédent est une base du sous espace $\widehat{K}_m = span\{q_1, \cdots, q_m\}$.

Dans la suite nous définissons :

$$\varepsilon_i = q_{i+1} - A\widehat{b}_i \qquad i = 1, \ldots, m, \tag{4.13}$$

ε_i représente l'erreur commise en approchant $A\widehat{b}_i$ par $\widehat{A}\,\widehat{b}_i$,

$$\varepsilon^{(m)} = (\varepsilon_1, \cdots, \varepsilon_m) \in \mathbb{R}^{n\times m}, \tag{4.14}$$

$$E_m = \varepsilon^{(m)} \widehat{B}_m^L \in \mathbb{R}^{n\times n}, \tag{4.15}$$

avec $\widehat{B}_m^L = (Y_m^T \widehat{B}_m)^{-1} Y_m^T$ étant une matrice inverse à gauche de $\widehat{B}_m$ définie par (1.7) et (1.8), ainsi nous obtenons :

$$(A + E_m)\widehat{b}_i = A\widehat{b}_i + \varepsilon^{(m)} \widehat{B}_m^L \widehat{b}_i = A\widehat{b}_i + \varepsilon_i = \widehat{A}\widehat{b}_i = q_{i+1} \tag{4.16}$$

d'où le théorème suivant dont le résultat montre que l'application de l'algorithme 12 au système :

$$A\widehat{x} = b, \text{ avec } \widehat{x}_0 \text{ approximation initiale quelconque,} \tag{4.17}$$

en commettant les erreurs ε_i définies par (4.13), est équivalente à l'application de l'algorithme 10 au système perturbé :

$$(A + E_m)x = b, \text{ avec } x_0 \text{ approximation initiale tel que } (A + E_m)x_0 = q_0, \tag{4.18}$$

Théorème 26

Supposons que m étapes de l'algorithme 12 soient appliquées au système (4.17), avec des erreurs ε_i comme en (4.13), et soit E_m la matrice définie par (4.15). Alors m étapes de l'algorithme 10 peuvent être appliquées au problème perturbé (4.18). De plus, si $\widehat{B}_m = (\widehat{b}_1, \ldots, \widehat{b}_m)$, $\widetilde{\widehat{H}}_m = (\widehat{h}_{i,j})$ et $B_m = (b_1, \ldots, b_m)$, $\widetilde{H}_m = (h_{i,j})$ sont les matrices construites par ces deux algorithmes, alors :

$$\widehat{B}_m = B_m \quad et \quad \widetilde{\widehat{H}}_m = \widetilde{H}_m.$$

Preuve :

Le résultat est établi par récurrence. En choisissant x_0 tel que $(A + E_m)x_0 = q_0$, il vient que $\widehat{r}_0 = b - q_0 = b - (A + E_m)x_0 = r_0$, et donc nous obtenons :

$$\widehat{b}_1 = \frac{\widehat{r}_0}{\|\widehat{r}_0\|} = \frac{r_0}{\|r_0\|} = b_1.$$

Par définition $\widehat{\omega}_2 = q_2 - \widehat{h}_{1,1}\widehat{b}_1$, ainsi :

$$\widehat{h}_{1,1} = \frac{(y_1, q_2)}{(y_1, \widehat{b}_1)} = \frac{(y_1, (A + E_m)\widehat{b}_1)}{(y_1, \widehat{b}_1)} = \frac{(y_1, (A + E_m)b_1)}{(y_1, b_1)} = h_{1,1},$$

et donc $\widehat{\omega}_2 = (A + E_m)b_1 - h_{1,1}b_1 = \omega_2$, ce qui entraine que :

$$\widehat{h}_{2,1} = \|\widehat{\omega}_2\| = \|\omega_2\| = h_{2,1} \text{ et } \widehat{b}_2 = b_2.$$

Supposons que nous ayons $\widehat{b}_i = b_i$ et $\widehat{h}_{i,k} = h_{i,k}$ pour $i = 1, \cdots, j$ et $k = 1, \cdots, j-1$. A l'étape $j < m$, nous avons $\widehat{\omega}_{j+1} = q_{j+1} - \sum_{i=1}^{j} \widehat{h}_{i,j}\widehat{b}_i$, or

$$\widehat{h}_{i,j} = \frac{(y_i, q_{j+1})}{(y_i, \widehat{b}_i)} = \frac{(y_i, (A + E_m)\widehat{b}_j)}{(y_i, \widehat{b}_i)} = \frac{(y_i, (A + E_m)b_j)}{(y_i, b_i)} = h_{i,j}, \text{ pour } i = 1, \cdots, j$$

et donc $\widehat{\omega}_{j+1} = (A + E_m)b_j - \sum_{i=1}^{j} h_{i,j}b_i = \omega_{j+1}$. Ainsi :

$$\widehat{h}_{j+1,j} = \|\widehat{\omega}_{j+1}\| = \|\omega_{j+1}\| = h_{j+1,j} \text{ et } \widehat{b}_{j+1} = b_{j+1}.$$

Finalement après m étapes , nous obtenons

$$\widehat{B}_m = B_m \quad et \quad \widetilde{\widehat{H}}_m = \widetilde{H}_m.$$

■

Dans la suite nous noterons $\widehat{z}_m = \widehat{B}_m\widehat{d}_m$ et $z_m = B_m d_m$. Le théorème 26 montre donc que les itérés $\widehat{d}_m$ et $\widehat{z}_m$ construits par l'algorithme 12 appliqué au système (4.17) sont égaux aux itérés d_m et z_m construits par l'algorithme 10 appliqué au système pérturbé (4.18). Ainsi les solutions approchées de ces deux systèmes s'écrivent :

$$\widehat{x}_m = \widehat{x}_0 + \widehat{z}_m = \widehat{x}_0 + z_m,$$

$$x_m = x_0 + z_m = x_0 + \widehat{z}_m.$$

Ces deux dernières égalités nous permettent de donner une relation entre les résidus associés aux solutions approchées des deux systèmes considérés.

Soit $\bar{r}_m = b - A\widehat{x}_m$ le résidu associé à $\widehat{x}_m$ et $\tilde{r}_m = b - (A + E_m)x_m$ le résidu associé à x_m, nous avons alors :

$$\begin{aligned} \bar{r}_m &= b - A\widehat{x}_0 - A\widehat{z}_m \\ &= b - A\widehat{x}_0 - Az_m \\ &= ((b - A\widehat{x}_0) - \widehat{r}_0) + (\widehat{r}_0 - (A + E_m)z_m) + E_m z_m. \end{aligned}$$

En notant $\varepsilon_0 = (b - A\widehat{x}_0) - \widehat{r}_0$ et en remarquant que :

$$\widehat{r}_0 - (A + E_m)z_m = b - q_0 - (A + E_m)z_m = b - (A + E_m)x_m = \tilde{r}_m,$$

il vient que :

$$\bar{r}_m = \varepsilon_0 + \tilde{r}_m + E_m z_m. \tag{4.19}$$

Ainsi, par majoration, nous obtenons

$$\|\bar{r}_m\|_2 \leq \|\varepsilon_0\|_2 + \|\tilde{r}_m\|_2 + \|E_m z_m\|_2. \tag{4.20}$$

Le premier terme de l'inégalité (4.20) représente l'erreur commise en considérant $\widehat{r}_0$ comme résidu associé à la solution initiale $\widehat{x}_0$ (c'est l'erreur commise en approchant le produit $A\widehat{x}_0$ par $q_0 = \widehat{A}\widehat{x}_0$). Le second terme étant la norme du résidu associé à la solution x_m du système perturbé (4.18), et le troisième terme représente l'erreur faite sur le calcul des différents produits approchés $\widehat{A}\widehat{b}_j$.

En utilisant la définition de E_m et l'égalité des matrices $\widehat{B}_m$ et B_m, nous obtenons, d'après l'inégalité précédente :

$$\|\bar{r}_m\|_2 \leq \|\varepsilon_0\|_2 + \|\tilde{r}_m\|_2 + \|\varepsilon^{(m)} B_m^L B_m d_m\|_2,$$

or $B_m^L B_m = I_m^{(m)}$, il vient donc que :

$$\|\bar{r}_m\|_2 \leq \|\varepsilon_0\|_2 + \|\tilde{r}_m\|_2 + \|\varepsilon^{(m)}\|_2 \|d_m\|_2. \tag{4.21}$$

Il est souhaitable que la norme des erreurs ε_i , $i = 0, \cdots, m$ soit suffisamment petite afin de considérer $\|\tilde{r}_m\|_2$ comme une bonne estimation de la norme $\|\bar{r}_m\|_2$. Rappelons que pour la méthode de Galerkin $\|\tilde{r}_m\|_2$ est égal à $\|b_{m+1}\|_2 = \|\widehat{b}_{m+1}\|_2$ (à une constante multiplicative près §2.1.1). Et en ce qui concerne la méthode MRSe, nous pouvons facilement obtenir une majoration de $\|\tilde{r}_m\|_2$ (§2.2.3 et §2.4).

4.2.3 Propriétés de la méthode version différence finies de la méthode de Hessenberg Généralisée

Dans la suite, le système linéaire considéré est un système associé à une itération de la méthode de Newton. L'analyse de la propagation de l'erreur faite dans la section précédente, ainsi que la comparaison des algorithmes 10 et 12 faite dans le théorème 26 permet d'obtenir le résultat suivant :

Théorème 27

Supposons que m étapes de l'algorithme 11 soient appliquées au système (4.17), avec des erreurs ε_i comme en (4.13), et soit E_m la matrice définie par (4.15). Alors m étapes de l'algorithme 10 peuvent être appliquées au problème perturbé (4.18). De plus, si $\widehat{B}_m = (\widehat{b}_1, \ldots, \widehat{b}_m)$, $\widetilde{\widehat{H}}_m = (\widehat{h}_{i,j})$ et $B_m = (b_1, \ldots, b_m)$, $\widetilde{H}_m = (h_{i,j})$ sont les matrices construites par ces deux algorithmes, alors :

$$\widehat{B}_m = B_m \quad et \quad \widetilde{\widehat{H}}_m = \widetilde{H}_m.$$

Notons que nous avons aussi une comparaison entre $\bar{r}_m = b - A\widehat{x}_m$ le résidu associé à $\widehat{x}_m$ solution du système (4.17), et $\tilde{r}_m = b - (A + E_m)x_m$ associé à la solution x_m du système (4.18) avec $A = J(u) = F'(u)$ et $b = -F(u)$. Cette comparaison est donnée par (4.20) et (4.21).

Afin d'estimer l'importance des erreurs ε_i $i = 1, \cdots, m$, (ε_i est l'erreur commise en approchant le produit matrice vecteur $A\widehat{b}_j$ par q_{j+1} le quotient approché (4.11) (i.e. $q_{j+1} = \left(F(u + \sigma_j \widehat{b}_j) - F(u)\right)/\sigma_j$, nous avons besoin de supposer que l'application F' est γ-lipschitzienne sur un voisinage convexe $D \subset \mathbb{R}^n$ de u_*.

Définition 7

Soient G : $\mathbb{R}^n \longrightarrow \mathbb{R}^n$ une application et γ un réel strictement positif. L'application G est γ-lipschitzienne sur un domaine D de $\mathbb{R}^n$ si

$$\forall x,\ y \in D \quad \|G(x) - G(y)\|_2 \leq \gamma \|x - y\|_2.$$

D'après le lemme 4.1.12 donné dans [17], nous avons pour tout $i = 1, \cdots, m$:

$$\|F(u + \sigma_i \widehat{b}_i) - F(u) - F'(u)\sigma_i \widehat{b}_i\|_2 \leq \frac{\gamma}{2} |\sigma_i|^2 \|\widehat{b}_i\|_2^2,$$

les σ_i étant choisis de façon que $u + \sigma_i \widehat{b}_i \in D$. Ainsi, il vient que :

$$\|\varepsilon_i\|_2 = \|q_{i+1} - A\widehat{b}_i\|_2 \leq \frac{\gamma}{2} |\sigma_i| \|\widehat{b}_i\|_2^2,$$

et comme $\|\varepsilon^{(m)}\|_2 \leq \|\varepsilon^{(m)}\|_F \equiv (\|\varepsilon_1\|_2^2 + \ldots + \|\varepsilon_m\|_2^2)^{1/2}$, nous obtenons

$$\|\varepsilon^{(m)}\|_2 \leq \frac{\gamma}{2}(\sigma_1^2\|\hat{b}_1\|_2^4 + \cdots + \sigma_m^2\|\hat{b}_m\|_2^4)^{1/2} \leq \frac{\gamma}{2}\|\sigma^{(m)}\|_2\|\hat{B}_m\|_2^2, \tag{4.22}$$

avec $\sigma^{(m)} = (\sigma_1, \cdots, \sigma_m)^T \in \mathbb{R}^m$.

De même, si $u + \sigma_0\hat{x}_0 \in D$, nous avons :

$$\|\varepsilon_0\|_2 \leq \frac{\gamma}{2}|\sigma_0|\|\hat{x}_0\|_2^2.$$

Finalement, en utilisant les deux dernières inégalités, (4.21) devient :

$$\|\bar{r}_m\|_2 \leq \frac{\gamma}{2}(|\sigma_0|\|\hat{x}_0\|_2^2 + \|\sigma^{(m)}\|_2\|\hat{B}_m\|_2^2\|d_m\|_2) + \|\tilde{r}_m\|_2. \tag{4.23}$$

L'inégalité (4.23) est particulièrement utile pour établir le résultat suivant :

Théorème 28

Soit u une approximation de u_* une solution de $F(u) = 0$ avec F' inversible et γ-lipschitzienne sur $D \subset \mathbb{R}^n$ un voisinage convexe contenant u_* et u. Considérons le système linéaire (4.17) avec $A = F'(u)$ et $b = -F(u)$. Soit N le degré du polynôme minimal associé à A et $\hat{r}_0$. Soit p, η donnés et choisissons $\delta > 0$ suffisamment petit pour que :

$$\delta\gamma\|A^{-1}\|_2\|B_i^L\|_2 < 1, \quad i = 1, \ldots, N. \tag{4.24}$$

$$u + \delta t \in D \quad \forall \ t \in \mathbb{R}^n, \tag{4.25}$$

$$\frac{\gamma}{2}\delta(\|\hat{x}_0\|_2^2 + \|B_i\|_2^2\|d_i\|_2) \leq \frac{1}{2}\eta\|F(u)\|_2^p, \quad i = 1, \ldots, N. \tag{4.26}$$

Si on choisit σ_0, $\sigma^{(n)} = (\sigma_1, \cdots, \sigma_n)^T \in \mathbb{R}^n$ de telle façon que $\sigma_0 < \delta$ et $\|\sigma^{(n)}\|_2 < \delta$, alors il existe $m \in \{0, 1, \ldots, N\}$ tel que l'itéré $\hat{x}_m = \hat{x}_0 + \hat{B}_m\hat{d}_m$ construit par l'algorithme 11 (DFHG) existe et vérifie

$$\|\bar{r}_m\|_2 = \|b - A\hat{x}_m\|_2 \leq \eta\|F(u)\|_2^p.$$

Preuve :

Comme $\sigma_0 < \delta$ et $\|\sigma^{(m)}\|_2 < \delta$, alors par utilisation de l'inégalité (4.23) nous avons :

$$\|\bar{r}_m\|_2 \leq \frac{\gamma}{2}\delta(\|\hat{x}_0\|_2^2 + \|B_m\|_2^2\|d_m\|_2) + \|\tilde{r}_m\|_2,$$

et grâce à (4.26), nous obtenons

$$\|\bar{r}_m\|_2 \leq \frac{1}{2}\eta\|F(u)\|_2^p + \|\tilde{r}_m\|_2.$$

Il reste donc à montrer qu' il existe $m \in \{0, \cdots, n\}$ tel que l'itéré x_m vérifie

$$\|\tilde{r}_m\|_2 = \|b - (A + E_m)x_m\|_2 \leq \frac{1}{2}\eta\|F(u)\|_2^p.$$

Commencons alors par vérifier que $(A + E_m)$ est inversible. Pour cela, supposons que m vecteurs $\hat{b}_1, \ldots, \hat{b}_m$ aient été construit par l'algorithme 11 et soit $E_m = \varepsilon^{(m)} B_m^L$ comme il a été défini précédemment. Alors :

$$\begin{aligned} \|A^{-1}E_m\|_2 &\leq \|A^{-1}\|_2 \, \|E_m\|_2 \\ &\leq \|A^{-1}\|_2 \, \|B_m^L\|_2 \, \|\varepsilon^{(m)}\|_2, \end{aligned}$$

or d'après (4.22) nous avons $\|\varepsilon^{(m)}\|_2 \leq \frac{\gamma}{2}\|\sigma^{(m)}\|_2 \leq \frac{\gamma}{2}\delta$, et en utilisant (4.24) nous obtenons

$$\|A^{-1}E_m\|_2 \leq \|A^{-1}\|_2 \, \|B_m^L\|_2 \frac{\gamma}{2}\delta < \frac{1}{2}.$$

Comme $\|A^{-1}E_m\|_2 < 1$, il vient que $(I + A^{-1}E_m)$ est inversible, et donc la matrice $(A + E_m) = A(I + A^{-1}E_m)$ est elle même inversible.

Finalement, d'après la propriété de finitude de la méthode de Hessenberg Généralisée, on sait qu'il existe $m \in \{0, \cdots, N\}$ tel que $\hat{b}_{m+1} = b_{m+1} = 0$, et dans ce cas $x_m = x_0 + B_m d_m$ est la solution exacte du système $(A + E_m)b = x$ et donc le résidu associé à x_m est tel que $\|\tilde{r}_m\|_2 = 0 \leq \frac{1}{2}\|F(u)\|_2^p$.

■

Le théorème précédent est utile pour l'étude de la convergence locale de la méthode de Newton Hessenberg Généralisée.

4.2.4 Convergence locale de la version différence finie de la méthode NHG

Commençons ce paragraphe par donner l'algorithme de la version différence finie de la méthode de Newton Hessenberg Généralisée.

Algorithme 13 : *Version différence finie de la méthode NHG (DFNHG).*

1- Choisir u_0 une approximation initiale.

2- Pour $k = 0, 1, \cdots$ jusqu'à convergence :

Utiliser l'algorithme 11 pour trouver x_k vérifiant :

$$\begin{aligned} F'(u_k)x_k &= -F(u_k) + r_k, \\ \text{avec} \qquad \|r_k\|_2 &\leq \eta_k \|F(u_k)\|_2. \\ \text{Poser} \quad : \qquad u_{k+1} &= u_k + x_k. \end{aligned}$$

Cet algorithme sera plus explicité en détail lorsque nous donnerons les algorithmes des versions différences finies des méthodes Newton-GMRES (NG) et Newton-CMRH (NC). Avant cela, nous pouvons donner le résultat suivant obtenu en combinant les résultats des théorèmes 25 et 28.

Théorème 29

Soit $F : \mathbb{R}^n \longrightarrow \mathbb{R}^n$ une application vérifiant les conditions (4.5), (4.6) et (4.7). Supposons de plus que F soit γ-lipschitzienne sur $D \subset \mathbb{R}^n$ un voisinage de u_*. Soit $\{\eta_k\}$ vérifiant $0 \leq \eta_k \leq \eta_{max} < t < 1$. Alors il existe $\varepsilon > 0$ tel que pour tout $u_0 \in \mathbb{R}^n$ vérifiant $\|u_0 - u_*\| < \varepsilon$ alors les itérés $u_1, u_2, \cdots$ construits par l'algorithme 13 (DFNHG) sont bien définis et convergent linéairement vers u_*, c'est à dire que :

$$\|u_{k+1} - u_*\|_* \leq t\|u_k - u_*\|_*,$$

avec $\|u\|_* = \|F'(u_*)u\|$.
Si de plus $\eta_k = \mathcal{O}(\|F(u_k)\|^{p-1})$, alors la convergence est d'ordre p , c'est à dire que :

$$\|u_{k+1} - u_*\| = \mathcal{O}(\|u_k - u_*\|^p) \qquad (quand\ k \to \infty).$$

Nous avons déja remarqué que lorsque les erreurs ε_i, $i = 0, \cdots, m$ sont suffisamment petites, alors l'inégalité (4.21) permet de considérer $\|\tilde{r}_m\|_2$ (dont une estimation est facile à calculer) comme une estimation de $\|\bar{r}_m\|_2 = \|b - A\hat{x}_m\|_2$. Notons cependant que si les erreurs ε_i, $i = 0, \cdots, m$ ne sont pas négligeables, alors même si $\|\tilde{r}_m\|_2 \simeq 0$, l'inégalité (4.4) peut ne pas être vérifiée. De plus, étant donné qu'un nombre maximum d'itérations est fixé dans l'algorithme 11 (DFHG), il peut arriver que $m = m_{max}$ sans que la précision imposée sur le calcul de $\hat{x}_m$ ne soit satisfaite. Dans ce cas, nous pouvons soit redémarrer l'algorithme 11 (DFHG) ou accepter l'itéré $\hat{x}_m$. Un tel choix se trouve

justifié par l'observation des résultats numériques, ainsi que par le fait que $\widehat{x}_m$ peut être une direction de descente pour la fonction f définie par:

$$f(u) = \frac{1}{2}\|F(u)\|_2^2. \tag{4.27}$$

En fait, en faisant appel à une stratégie de minimisation globale de la fonction f, nous pouvons améliorer la convergence de la méthode de Newton lorsque l'approximation u_0 n'est pas dans un voisinage de u_* qui permet d'assurer la convergence locale de la méthode Inexacte de Newton. Dennis et Schnabel [17] ont suggéré d'utiliser une méthode à convergence globale pour la résolution du problème:

$$\min_{u \in \mathbb{R}^n} f(u). \tag{4.28}$$

Définition 8

Soient $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ une fonction et $p, u \in \mathbb{R}^n$. Le vecteur p est une direction de descente pour la fonction f au point u si il existe $\lambda_0 \in \mathbb{R}^*$ tel que:

$$\forall \lambda \in]0, \lambda_0[\qquad f(u + \lambda p) < f(u).$$

Rappelons que pour une fonction f différentiable dans un voisinage de u, le vecteur direction de descente est caractérisé par le théorème suivant:

Théorème 30

Soient $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ une fonction différentiable dans un voisinage de $u \in \mathbb{R}^n$. Le vecteur $p \in \mathbb{R}^n$ est une direction de descente pour f en u si et seulement si

$$\nabla f(u)^T p < 0.$$

Soit x_m une solution approchée du système $Ax = b$ avec A étant la matrice jacobienne de F, $A = J(u) = F'(u)$, $b = -F(u)$ et $r_m = b - Ax_m = -(F(u) + J(u)x_m)$. D'après l'expression de f, nous avons

$$\nabla f(u) = J(u)^T F(u), \tag{4.29}$$

la solution approchée x_m est une direction de descente pour f en u si:

$$F(u)^T J(u) x_m < 0,$$

or $F(u)^T J(u) x_m = F(u)^T(-r_m - F(u)) = -F(u)^T r_m - F(u)^T F(u)$, et donc x_m est une direction de descente si

$$|F(u)^T r_m| < F(u)^T F(u). \tag{4.30}$$

En particulier si $\|r_m\|_2 < \|F(u)\|_2$. Notons que ce résultat est indépendant de la méthode itérative utilisée pour la résolution du système linéaire associé à la méthode de Newton. Dans le cas des méthodes Newton-Arnoldi et Newton-GMRES, Brown [9] et Brown et Saad [10] ont établi les résultats suivants :

Propriété 12

Soit x_m une solution approchée non nulle donnée par la méthode d'Arnoldi en prenant comme approximation initiale $x_0 = 0$. Alors x_m est une direction de descente pour f en u et

$$F(u)^T J(u) x_m = -F(u)^T F(u).$$

Propriété 13

Soit x_m une solution approchée non nulle donnée par la méthode GMRES en prenant comme solution initiale $x_0 = 0$. Alors x_m est une direction de descente pour f en u et

$$F(u)^T J(u) x_m = -F(u)^T F(u) + \|r_m\|_2^2,$$

r_m étant le résidu associé à l'itéré x_m.

Les résultats des deux propriétés précédentes sont dûs essentiellement au fait que la base B_m construite par le processus d'Arnoldi est orthonormale ainsi qu'au fait que r_m^{Arnoldi} est égale à b_{m+1} (à un facteur multiplicatif près), et que r_m^{GMRES} est orthogonal à $AK_m(r_0, A)(= J(u)K_m)$. Nous pouvons généraliser ces résultats aux autres cas particuliers de la méthode de Hessenberg Généralisée. En effet la base B_m construite par le processus de Hessenberg Généralisé est Z_m-orthogonale, le résidu r_m^{GAL} de la méthode de Galerkin est lui aussi égale à b_{m+1} à un facteur multiplicatif près, et de même le résidu r_m^{MRSe} de la méthode de semi-minimisation du résidu est Z_m-orthogonale à $AK_m(r_0, A)$. La matrice Z_m étant la matrice symétrique définie positive sur $K_{m+1}(r_0, A)$, et définie par :

$$Z_m = (B_{m+1}^L)^T B_{m+1}^L,$$

avec B_{m+1}^L étant une matrice inverse à gauche de B_{m+1} comme elle a été définie au chapitre 1. La généralisation des deux résultats précédents est basée sur l'étude de la fonction $\tilde{f}$ définie par :

$$\tilde{f}(u) = \frac{1}{2}|F(u)|_{Z_m}^2 = \frac{1}{2}F(u)^T Z_m F(u). \tag{4.31}$$

En utilisant les différentes observations faites après les deux propriétés précédentes, nous pouvons établir les résultats suivants :

Propriété 14

Soit x_m une approximation non nulle donnée par la méthode de Galerkin en prenant comme approximation initiale $x_0 = 0$. Alors x_m est une direction de descente pour $\tilde{f}$ en u et

$$\nabla \tilde{f}(u)^T x_m = -F(u)^T Z_m F(u).$$

Preuve :

D'après l'expression de $\tilde{f}$, nous avons

$$\nabla \tilde{f}(u) = J(u)^T Z_m F(u), \tag{4.32}$$

ainsi $\nabla \tilde{f}(u)^T x_m = F(u)^T Z_m J(u) x_m$, or $J(u)x_m = -F(u) - r_m$, et donc

$$\begin{aligned} \nabla \tilde{f}(u)^T x_m &= F(u)^T Z_m(-F(u) - r_m) \\ &= -F(u)^T Z_m F(u) - F(u)^T Z_m r_m. \end{aligned}$$

L'hypothèse $x_0 = 0$ entraîne que $b = -F(u) = r_0 =$ or $r_0^T Z_m r_m = 0$ (§4.3 lemme 14), ainsi :

$$\nabla \tilde{f}(u)^T x_m = -F(u)^T Z_m F(u).$$

■

Propriété 15

Soit x_m une approximation non nulle donnée par la méthode MRSe en prenant comme approxmation initiale $x_0 = 0$. Alors x_m est une direction de descente pour $\tilde{f}$ en u et

$$\nabla \tilde{f}(u)^T x_m = -F(u)^T Z_m F(u) + |r_m|^2_{Z_m},$$

r_m étant le résidu associé à l'itéré x_m.

Preuve :

Nous avons déja établi dans la démonstration de la propriété précédente que

$$\nabla \tilde{f}(u)^T x_m = -F(u)^T Z_m F(u) - F(u)^T Z_m r_m.$$

Il nous reste donc à montrer que $|r_m|^2_{Z_m} = -F(u)^T Z_m r_m$.

La méthode de Hessenberg Généralisée minimise $|J(u)x + F(u)|_{Z_m}$ pour x dans le sous espace de Krylov $K_m(r_0, A)$, et dans ce cas r_m le résidu correspondant à la solution

approchée x_m est Z_m-orthogonale à $J(u)K_m(r_0, A)$, ainsi nous avons :

$$\begin{aligned} F(u)^T Z_m r_m &= (-r_m - J(u)x_m)^T Z_m r_m \\ &= -r_m^T Z_m r_m - r_m^T Z_m J(u)x_m \\ &= -r_m^T Z_m r_m. \end{aligned}$$

Comme $x_0 = 0$. alors $r_0 = b = -F(u) = \|r_0\|b_1 = \|r_0\|B_{m+1}e_1^{(m+1)}$, or la solution approchée x_m s'écrit $x_m = x_0 + B_m\widetilde{H}_m^+ e_1^{(m+1)} = B_m\widetilde{H}_m^+ e_1^{(m+1)}$, ce qui permet d'avoir :

$$\begin{aligned} \nabla\tilde{f}(u)^T x_m &= F(u)^T Z_m J(u)x_m \\ &= -\|r_0\|(B_{m+1}e_1^{(m+1)})^T Z_m J(u)B_m\widetilde{H}_m^+ e_1^{(m+1)} \\ &= -\|r_0\|e_1^{(m+1)^T} B_{m+1}^T Z_m B_{m+1}\widetilde{H}_m\widetilde{H}_m^+ e_1^{(m+1)}, \end{aligned}$$

or $B_{m+1}^T Z_m B_{m+1} = B_{m+1}^T (B_{m+1}^L)^T B_{m+1}^L B_{m+1} = I_{m+1}^{(m+1)}$ et $\widetilde{H}_m\widetilde{H}_m^+ = \widetilde{H}_m(\widetilde{H}_m^T\widetilde{H}_m)^{-1}\widetilde{H}_m^T$. Supposons $h_{m+1,1} \neq 0$ (si $h_{m+1,1} = 0$ alors $x_m = x_* = A^{-1}b$, et on a $\nabla\tilde{f}(u)^T x_m = -b^T b < 0$), la matrice $\widetilde{H}_m$ étant une matrice Hessenberg supérieure, elle est donc de rang maximal. Ainsi

$$\nabla\tilde{f}(u)^T x_m = -\|r_0\|e_1^{(m+1)^T}\widetilde{H}_m\widetilde{H}_m^+ e_1^{(m+1)} < 0.$$

■

Dans le chapitre précédent, nous avons montré que pour les cas particuliers des méthodes Arnoldi et GMRES, la semi-norme $|.|_{Z_m}$ est égale à la norme euclidienne. Ainsi, les propriétés 14 et 15 permettent de retrouver les propriétés 12 et 13.

L'expression de $\nabla\tilde{f}(u)^T x_m$ donnée dans la propriété 15 montre que lorsque l'apporoximation initiale x_0 est non nulle, alors une condition suffisante pour que la solution approchée x_m^{MRSe} soit une direction de descente pour $\tilde{f}$ en u est que $|r_m^{\text{MRSe}}|_{Z_m} < |F(u)|_{Z_m}$. Cette condition est en particulier réalisée lorsque x_0 est telle que $|r_0|_{Z_m} < |F(u)|_{Z_m}$. Dans ce cas x_m^{MRSe} est une direction de descente pour $\tilde{f}$ en u pour tout $m \geq 1$.

En pratique. la semi-norme $|.|_{Z_m}$ est difficile à calculer, il apparaît alors que l'introduction de la fonction $\tilde{f}$ a essentiellement pour but d'établir des résultats théoriques. Ajoutons à cela qua la méthode qui nous intéresse est la version différence finie de la méthode NHG. de plus la norme usuelle étant la norme euclidienne, il faudrait donc donner des conditions sur les σ_i, $i = 0, \cdots, m$ pour que la solution $\hat{x}_m$ construite par l'algorithme 11 (DFHG) soit une direction de descente pour f en u (et non pas pour $\tilde{f}$).

Soit $\hat{x}_m$ une solution approchée construite par l'algorithme 11 (DFHG). D'après l'équation (4.30), nous savons toujours que $\hat{x}_m$ est une direction de descente pour f en u si $\bar{r}_m = b - A\hat{x}_m$ est tel que :

$$|F(u)^T\bar{r}_m| = |b^T\bar{r}_m| < |F(u)^T F(u)| = \|b\|_2^2.$$

Or, de l'équation (4.19), nous déduisons que:

$$\bar{r}_m = \varepsilon_0 + \varepsilon^{(m)} d_m + \tilde{r}_m,$$

avec $\tilde{r}_m = b - (A + E_m)x_m$ et x_m la solution du système pérturbé (4.18). Ce qui permet d'écrire:

$$|b^T \bar{r}_m| = |b^T(\varepsilon_0 + \varepsilon^{(m)} d_m + \tilde{r}_m)|.$$

Ainsi, si les erreurs ε_i, $i = 0, \cdots, m$ sont suffisamment petites, alors une condition suffisante pour que $\hat{x}_m$ soit une direction de descente pour f en u est que:

$$|b^T \tilde{r}_m| < \|b\|_2^2. \tag{4.33}$$

Cette dernière condition est facile à vérifier puisque d'après les résultats du chapitre 2 concernant le calcul recursif du résidu des méthodes GAL et MRSe, nous savons calculer facilement $\tilde{r}_m$ (§2.1.1 équation (2.6) et §2.2.3 équation (2.17)). Notons aussi que la condition (4.33) est en particulier vérifiée si $\|\tilde{r}_m\|_2 < \|b\|_2^2$.

Finalement le résultat du théorème 28 ainsi que l'étude faite sur la fonction f, montrent que nous pouvons toujours choisir les σ_i, $i = 0, \cdots, m$ de façon à avoir une solution approchée $\hat{x}_m$ qui soit une direction de descente pour f en u, et que la méthode version différence finie de la méthode NHG garde le caractère de convergence locale caractéristique de la méthode de Newton. Notons cependant qu'il serait utile d'associer à la méthode version différence finie da la méthode NHG un algorithme de minimisation de la fonction f afin qu'elle y ait convergence globale. De tels procédés ont étés étudiés par Brown et Saad [10, 11] pour les méthodes Newton-Arnoldi et Newton-GMRES. Notons aussi que les résultats donnés dans [11] sont plus généraux puisque ils sont valables pour des méthodes de type Newton Inexacte (où la solution du système linéaire est obtenue par une méthode de projection) combinée avec des méthodes de recherche linéaire telles que "linesearch techniques" ou "model trust region algorithms".

4.3 Résultats numériques

Dans ce paragraphe, nous allons donner quelques résultats numériques qui permettent de comparer la performance des méthodes NG (Newton-GMRES) et NC (Newton-CMRH). Nous commençons par donner les algorithmes des deux méthodes. Les exemples numériques seront traités juste après.

4.3.1 La version différence finie de la méthode NG

L'algorithme ci dessous décrit la méthode de Newton Inexacte où la solution du système linéaire (4.2) est obtenue par la version différence finie de la méthode GMRES.

Algorithme 14 : *Newton-GMRES (NG).*

(I)- Choisir u_0 *une approximation initiale du système non linéaire; poser* $k = 0$ *; choisir une précision* ϵ_0.

(II)- Résolution du système linéaire $Ax = b$, *avec* $A = F'(u_k)$ *et* $b = -F(u_k)$.

(1)- Initialisation :

choisir $\widehat{x}_0$ *une approximation initiale ;*

calculer $q_0 = (F(u_k + \sigma_0 \widehat{x}_0) - F(u_k))/\sigma_0$; $\widehat{r}_0 = b - q_0$;

$\widehat{\beta} = \|\widehat{r}_0\|_2$; $\widehat{b}_1 = \widehat{r}_0/\widehat{\beta}$; $q_1 = \widehat{b}_1$.

(2)- Processus d'Arnoldi :

pour $j = 1, \cdots, m$

(a)- $q_{j+1} = (F(u_k + \sigma_j \widehat{b}_j) - F(u))/\sigma_j$; $\widehat{\omega} = q_{j+1}$;

pour $i = 1, \cdots, j$

$\widehat{h}_{i,j} = (\widehat{b}_i, \widehat{\omega})$; $\widehat{\omega} = \widehat{\omega} - \widehat{h}_{i,j}\widehat{b}_i$;

fin du pour ;

$\widehat{\omega}_{j+1} = \widehat{\omega}$; $\widehat{h}_{j+1,j} = \|\widehat{\omega}_{j+1}\|_2$:

$\widehat{b}_{j+1} = \widehat{\omega}_{j+1}/\widehat{h}_{j+1,j}$;

(b)- calculer une estimation de $\rho_j = \|b - (F(u_k) + \sigma \widehat{x}_j) - F(u_k))/\sigma\|_2$;

si $\rho_j \leq \epsilon_k$ *aller à (3) en posant* $m = j$;

fin du pour.

(3)- Former la solution $\widehat{x}_m$:

chercher $\widehat{d}_m$ *solution de :* $\min_{d \in \mathbb{R}^m} \|\widehat{\beta} e_1^{(m+1)} - \widetilde{\widehat{H}}_m d\|_2$:

calculer $\widehat{x}_m = \widehat{x}_0 + \widehat{B}_m \widehat{d}_m$.

(4)- Redémarrage du GMRES.

calculer une estimation de $\rho_m = \|b - (F(u_k) + \sigma \widehat{x}_m) - F(u_k))/\sigma\|_2$;

si $\rho_m > \epsilon_k$ *aller à (1) en posant* $\widehat{x}_0 = \widehat{x}_m$;

(III)- Calculer $u_{k+1} = u_k + \widehat{x}_m$.

(IV)- Si $\|F(u_{k+1})\|_2$ *est assez petit stop.*

sinon $k = k + 1$, *choisir une nouvelle précision* ϵ_k *et retourner à (II).*

4.3.2 La version différence finie de la méthode NC

L'algorithme ci dessous décrit la méthode de Newton Inexacte où la solution du système linéaire (4.2) est obtenue par la version différence finie de la méthode CMRH.

Algorithme 15 : *Newton-CMRH (NC).*

(I)- Choisir u_0 *une approximation initiale du système non linéaire; poser* $k = 0$ *; choisir une précision* ϵ_0.

(II)- Résolution du système linéaire $Ax = b$, *avec* $A = F'(u_k)$ *et* $b = -F(u_k)$.

(1)- Initialisation :

choisir $\widehat{x}_0$ *une approximation initiale ; poser* $p = (1, 2, \cdots, n)^T$ *;*
calculer $q_0 = (F(u_k + \sigma_0 \widehat{x}_0) - F(u_k))/\sigma_0$ *;* $\widehat{r}_0 = b - q_0$ *;*
déterminer i_0 *tel que* $|(\widehat{r}_0)_{i_0}| = \|\widehat{r}_0\|_\infty$ *;* $p(1) \longleftrightarrow p(i_0)$ *;*
$\widehat{\beta} = (\widehat{r}_0)_{i_0}$; $\widehat{b}_1 = \widehat{r}_0/\widehat{\beta}$; $q_1 = \widehat{b}_1$.

(2)- Processus de Hesseneberg :

pour $j = 1, \cdots, m$

(a)- $q_{j+1} = (F(u_k + \sigma_j \widehat{b}_j) - F(u))/\sigma_j$; $\widehat{\omega} = q_{j+1}$;

pour $i = 1, \cdots, j$

$\widehat{h}_{i,j} = (\widehat{b}_i, \widehat{\omega})$; $\widehat{\omega} = \widehat{\omega} - \widehat{h}_{i,j}\widehat{b}_i$;

fin du pour;

$\widehat{\omega}_{j+1} = \widehat{\omega}$;

déterminer i_0 *tel que* $|(\widehat{\omega}_{j+1})_{i_0}| = \|\widehat{\omega}_{j+1}\|_\infty$ *;* $p(j+1) \longleftrightarrow p(i_0)$ *;*

$\widehat{h}_{j+1,j} = (\widehat{\omega}_{j+1})_{i_0}$ *:*

$\widehat{b}_{j+1} = \widehat{\omega}_{j+1}/\widehat{h}_{j+1,j}$ *;*

(b)- calculer une estimation de $\rho_j = \|b - (F(u_k) + \sigma \widehat{x}_j) - F(u_k))/\sigma\|_2$ *;*
si $\rho_j \leq \epsilon_k$ *aller à (3) en posant* $m = j$ *;*

fin du pour.

(3)- Former la solution $\widehat{x}_m$ *:*

chercher $\widehat{d}_m$ *solution de :* $\min_{d \in \mathbb{R}^m} \|\widehat{\beta} e_1^{(m+1)} - \widetilde{\widehat{H}}_m d\|_2$ *:*

calculer $\widehat{x}_m = \widehat{x}_0 + \widehat{B}_m \widehat{d}_m$.

(4)- Redémarrage du CMRH.

calculer une estimation de $\rho_m = \|b - (F(u_k) + \sigma \widehat{x}_m) - F(u_k))/\sigma\|_2$ *;*
si $\rho_m > \epsilon_k$ *aller à (1) en posant* $\widehat{x}_0 = \widehat{x}_m$ *;*

(III)- Calculer $u_{k+1} = u_k + \widehat{x}_m$.

(IV)- Si $\|F(u_{k+1})\|_2$ *est assez petit stop.*
sinon $k = k + 1$, *choisir une nouvelle précision* ϵ_k *et retourner à (II).*

4.3.3 Implémentation des algorithmes et choix des paramètres

Les tests décrits dans la suite ont été réalisés sur une machine "Sun SparcStation 10", en double précision. Les programmes ont étés compilés par le compilateur f77. Dans tous les tests, l'approximation initiale $\widehat{x}_0$ du système linéaire associé à la méthode de Newton est initialisée à $(0, \cdots, 0)^T$.

Un point important dans l'implémentation des algorithmes décrits précédemment concerne le choix des paramètres σ_i et des précisions ϵ_k imposées sur le calcul de la solution $\widehat{x}_m$.

• **choix des σ_i.** Dans les algorithmes Newton-GMRES et Newton-CMRH, nous avons approché l'action du Jacobien de l'application F sur un vecteur v par le quotient

$$J(u).v \simeq \frac{F(u+\sigma v) - F(u)}{\sigma}.$$

En précision finie, le calcul de $F(u)$ n'est pas exact. En fait, au lieu de la valeur réelle nous obtenons la valeur approchée $F(u) + \epsilon(u)$. En supposant que l'erreur $\epsilon(u)$ est majorée, i.e. $\|\epsilon(u)\|_2 \leq \bar{\epsilon}\ \forall\ u$, alors

$$J(u).v - \frac{F(u+\sigma v) + \epsilon(u+\sigma v) - F(u) - \epsilon(u)}{\sigma} = \mathcal{O}(\sigma + \bar{\epsilon}/\sigma). \tag{4.34}$$

La quantité à l'intérieur du terme $\mathcal{O}$ est minimale pour $\sigma = \sqrt{\bar{\epsilon}}$. En général l'erreur ϵ est due aux erreurs d'arrondis, et donc $\bar{\epsilon}$ est sensiblement égale à la précision machine $\epsilon \simeq 10^{-16}$. L'equation (4.34) montre qu'il est inaproprié de prendre les paramètres σ_i très petits. En fait l'analyse parécédente montre qu'un choix raisonnable de σ serait $\sigma \simeq 10^{-8}$. Ce choix est valable pour des itérés u et v ayant sensiblement la même norme. Pour des vecteurs u et v quelconques nous considérons le choix

$$\sigma = \sqrt{\bar{\epsilon}}\frac{\|u\|_2}{\|v\|_2}.$$

• **choix des ϵ_k.** En ce qui concerne les critères d'arrêt associés au système linéaire nous considérons le choix $\epsilon_k = \eta_k \|F(u_k)\|_2$, où η_k est une suite décroissante majorée par l'unité. Le choix de η_k est très important, en effet si la valeur de η_k est trop petite, alors il est possible que des calculs inutiles soient effectués lors des premières itérations de la méthode de Newton. Un bon choix de η_k permet d'éviter donc un calcul de $\widehat{x}_m$ avec une trop grande précision par rapport à la précision nécessaire à la convergence. Dans les exemples numériques qui vont suivre, nous avons considéré $\eta_k = (\frac{1}{2})^k$.

• **choix des autres paramètres** . Nous avons pris pour m la valeur $m_{max} = 10$ ou $m_{max} = 20$. Si $m = m_{max}$ et $\rho_m > \epsilon_k$ alors on redémarre l'algorithme CMRH ou

GMRES une et une seule fois. En ce qui concerne le critère d'arrêt pour le problème non linéaire, l'algorithme est arrêté lorsque une solution approchée u_k, du problème non linéaire $F(u) = 0$, vérifie $\|F(u_k)\|_2 \leq 10^{-8}\|F(u_0)\|_2$.

Différentes approximations de départ ont été testées, nous avons considéré six vecteurs choisis aléatoirement dont les composantes sont distribuées uniformément sur un intervalle donné. Le tableau suivant donne l'intervalle I de distribution des composantes des six solutions initiales.

u_0	$rand_1$	$rand_2$	$rand_3$	$rand_4$	$rand_5$	$rand_6$
I	$[0,1]$	$[-1,1]$	$[-1,0]$	$[-2,-1]$	$[-2,2]$	$[-2,0]$

Les résultats obtenus sont donnés sous forme de courbes qui représentent la norme de $F(u)$ dans une échelle logarithmique en fonction du nombre d'itérations. Dans chaque figure, la courbe de la méthode de Newton-CMRH est représentée par la ligne continue et celle de la méthode Newton-GMRES est représentée par des tirets.

4.3.4 Description des exemples et résultats

Exemple 1. Problème de Bratu [10].

Il s'agit de résoudre l'équation aux dérivées partielles non linéaire suivante :

$$-\Delta u + \alpha u_x + \lambda e^u = f,$$

définie sur le carré unité de $\mathbb{R}^2$, avec des conditions aux bords de type Dirichlet. Lorsqu'on discrétise cette équation à l'aide de différences finies, nous obtenons un système non linéaire de taille $n = n_x.n_y$, avec $n_x + 2$, $n_y + 2$ étant le nombre de points de discrétisation utilisés dans chaque direction. La fonction f est choisie de façon que la solution du problème discrétisé soit égale à $(1, \cdots, 1)^T$. Ainsi ce problème admet au moins une solution. Nous avons considéré différentes valeurs pour λ, par contre la valeur de α est fixée à $\alpha = 100$. Tous les exemples sont de taille $n = 2500$ ($n_x = n_y = 50$).

Pour chaque choix du paramètre λ nous donnons les résultats obtenus pour quatre approximations initiales.

1. $\lambda = 1$ (page. 130).
2. $\lambda = 5$ (page. 131).

3. $\lambda = 10$ (page. 132).

4. $\lambda = -5$ (page. 133).

5. $\lambda = -10$ (page. 134).

Discussion des résultats de l'exemple 1.

Nous savons que pour $\lambda \geq 0$, il existe une unique solution pour le problème de Bratu (dans notre cas $u_* = e = (1, \cdots, 1)^T$). La plupart des exemples montrent que les courbes des méthodes NG et NC ont la même allure et que ces deux méthodes donnent des résultats similaires.

Du point de vue convergence, les deux méthodes, en général, convergent à la même itération (cela n'est pas vrai pour $\lambda < 0$). Nous avons observé numériquement que lorsque la solution du système linéaire n'est pas obtenue avec la précision désirée, il y'a une légère modification de l'allure de la courbe d'une des deux méthodes. C'est le cas notamment pour la méthode NG dans les exemples $\lambda = 1, u_0 = rand_5$ à l'itération 6 et $\lambda = 5, u_0 = rand_5$ à l'itération 9, ainsi que pour la méthode NC dans les exemples $\lambda = 5, u_0 = rand_1$ à l'itération 5 et $\lambda = 5, u_0 = rand_2$ à l'itération 7.

Lorsque le paramètre λ est négatif, l'application linéaire F, associée au problème de Bratu, n'admet pas une solution unique. Ce résultat est vérifié dans les exemples $\lambda = -5$ ou $\lambda = -10$ en prenant pour solution initiale $u_0 = rand_5$ ou $u_0 = rand_6$. Nous remarquons aussi que, lorsque les deux méthodes convergent vers la même solution $u_* = e$, les deux courbes ont la même allure dans les deux cas $\lambda = -5$ avec $u_0 = rand_1$ ou $u_0 = rand_4$ et $\lambda = -10$ avec $u_0 = rand_2$ ou $u_0 = rand_4$. Notons un léger avantage, pour la méthode NC sur la méthode NG, en ce qui concerne le nombre d'itérations dans le cas $\lambda = -10$.

Les exemples $\lambda = -5$ et $\lambda = -10$ avec $u_0 = rand_5$ ou $u_0 = rand_6$ illustrent bien la difficulté de résoudre un système non linéaire. En effet les deux méthodes convergent vers des solutions distinctes, bien que la solution initiale soit la même dans les deux cas. De plus, pour $\lambda = -1, -2, -3$, nous ne sommes pas arrivés à trouver une solution du système pour les choix $u_0 = rand_1$, $rand_2$, $rand_3$, $rand_5$. En fait il y a oscillation ou stagnation dans les deux méthodes, et cela même en partant d'une solution très proche de la solution e (exemple $u_0 = 0.9999.e$). Cependant, pour $u_0 = rand_4$ ou $u_0 = rand_6$, nous obtenons les résultats et les courbes données ci dessous. Ces résultats ne vont pas dans le sens des observations faites sur les exemples précédents, à savoir que les deux méthodes Newton-GMRES et Newton-CMRH donnent des résultats similaires. En effet les courbes ci dessous montrent que seule la méthode NC approche une des solutions du

problème sans atteindre cependant la précision voulue.

Courbes de l'exemple 1, $\lambda = -1$.

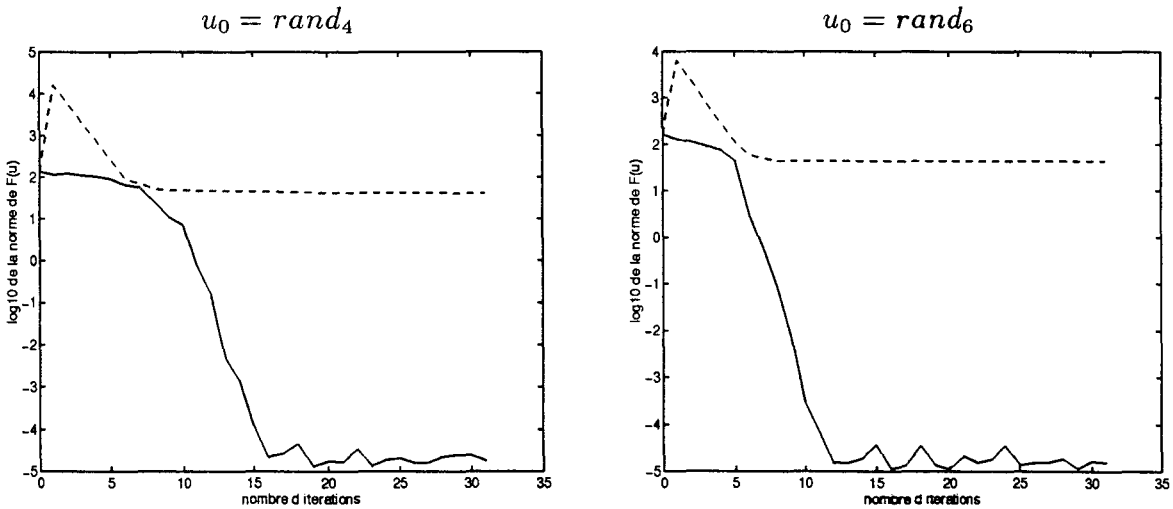


Tableau 4.0 : Résultats de l'exemple 1, $\lambda = -1$.

	$u_0 = rand_4$				$u_0 = rand_6$			
	itération	NEF	$\|F(u)\|_2$	CPU	itération	NEF	$\|F(u)\|_2$	CPU
NG	31	$\star$	$\star$	$\star$	31	$\star$	$\star$	$\star$
NC	31	714	$1.81\,10^{-5}$	51.09	31	714	$1.53\,10^{-5}$	50.61

Exemple 2

Pour cet exemple, l'application F est obtenue par discrétisation, sur le carré unité, de l'équation aux dérivées partielles non linéaire suivante :

$$-\Delta u + \alpha u_x + \gamma u^2 = f.$$

Comme pour l'exemple 1, la fonction f est choisie de façon que la solution du problème soit égale à $e = (1, \cdots, 1)^T$. Le système non linéaire ainsi obtenu admet plusieurs solutions. Nous prenons $\alpha = 100$. Le nombre de points utilisé pour la discrétisation est égale à 32, ainsi le système obtenu est de taille $n = 1024$. Nous considérons les deux choix suivants pour le paramètre γ :

1. $\gamma = 100$ (page. 135)

2. $\gamma = 10^6$ (page. 136)

Notons que pour le choix $\gamma = 10^6$, nous avons $\|u_0\|_2 \simeq 10^7$, c'est pour cela qu'on considère le nouveau test d'arrêt $\|F(u_k)\|_2 \leq 10^{-14}\|F(u_0)\|_2$.

Discussion des résultats de l'exemple 2.

Les solutions trouvées u ont toutes leurs composantes égales en valeur absolue à l'unité, i.e. $u = (\pm 1, \cdots, \pm 1)^T$. Comme pour l'exemple précédent, les deux méthodes donnent des résultats similaires. Notons u^{NG} (respectivement u^{NC}) la solution obtenue par la méthode Newton-GMRES, (respectivement Newton-CMRH). Il peut y avoir convergence des deux méthodes vers deux solutions distinctes, c'est les cas des exemples $\gamma = 100$ avec $u_0 = rand_2$ ou $u_0 = rand_5$ et $\gamma = 10^6$ avec $u_0 = rand_2$ ou $u_0 = rand_5$. Par contre pour les exemples $\gamma = 100$ avec $u_0 = rand_3$ ou $u_0 = rand_4$ et $\gamma = 10^6$ avec $u_0 = rand_3$ ou $u_0 = rand_4$, les deux méthodes convergent vers la même solution $-e = (-1, \cdots, -1)^T$. Dans le cas de convergence vers la même solution, les deux courbes ont la même allure et sont assez voisines. A l'inverse, lorsqu'il y a convergence vers deux solutions différentes, les deux courbes ont des allures différentes. De plus, on observe, numériquement, que cette différence dans l'allure est plus visible lorsque la norme de la différence entre les deux solutions $\|u^{\text{NC}} - u^{\text{NG}}\|_2$ est plus importante.

Exemple 3. Generalized function of Rosenbrock [13].

Cet exemple est pris de [13, p.I.5]. On considère l'application F définie par:

$$\begin{aligned} F_1(u) &= -4\zeta((u)_2 - (u)_1^2)(u)_1 - 2(1 - (u)_1) \\ F_i(u) &= 2\zeta((u)_i - (u)_{i-1}) - 4((u)_{i+1} - (u)_i^2)(u)_i - 2(1 - (u)_{i-1}) \quad i = 2, \ldots, n-1 \\ F_n(u) &= 2\zeta((u)_n - (u)_{n-1}^2) \end{aligned}$$

La matrice jacobienne de l'application F est une matrice tridiagonale. Le système non linéaire $F(u) = 0$ a pour solution exacte la solution unique: $u_* = (1, 1, \ldots, 1)^T$. Nous considérons un système de taille $n = 5000$ et les deux choix suivants du paramètre ζ:

1. $\zeta = 1$ (page. 137).
2. $\zeta = 10$ (page. 138).

Discussion des résultats de l'exemple 3.

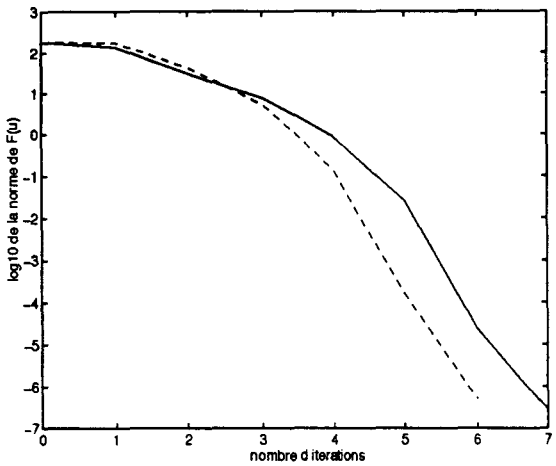
Dans le cas ou $\zeta = 1$, les deux méthodes arrivent à converger vers la solution du problème, et cela pour tous les différents choix des solutions initiales considérées. Par

contre lorsque la valeur de ζ grandit, les deux méthodes ne convergent pas toujours vers la solution. La convergence n'est pas obtenue notamment pour les exemples $\zeta = 10$ avec $u_0 = rand_4$ pour la méthode NG et $\zeta = 10$ avec $u_0 = rand_6$ pour la méthode NC.

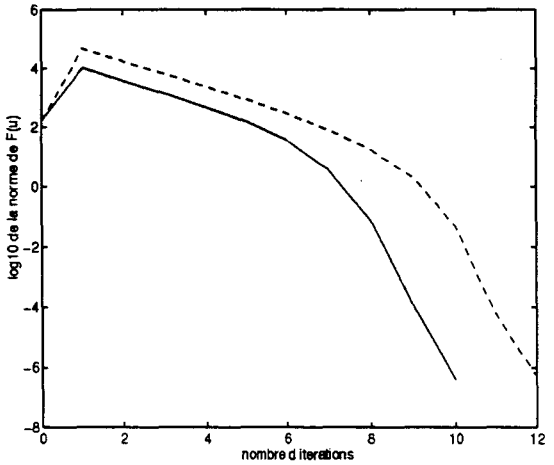
Les courbes obtenues pour cet exemple présentent plus d'irrégularités par rapport à celles obtenues pour les deux exemples précédents. Cela peut être expliqué par le choix de la précision imposée sur le calcul de la solution $\widehat{x}_m$ du système linéaire associé à la méthode de Newton. En effet, en choisissant de nouvelles valeurs du paramètre η_k (§4.3.3) on remarque que la convergence est soit perturbée, soit au contraire améliorée comme le montre les exemples donnés dans la page 139.

Courbes de l'exemple 1, $\lambda = 1$.

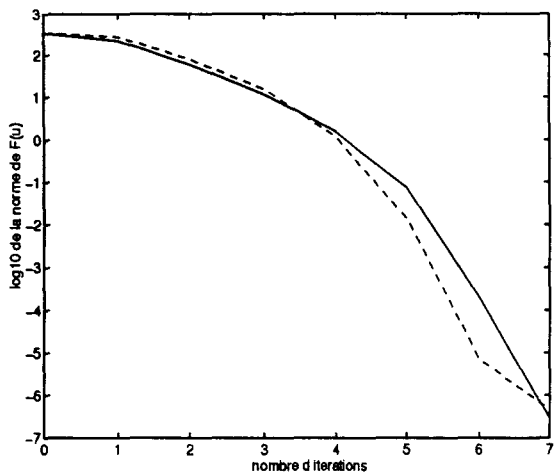
$u_0 = rand_2$



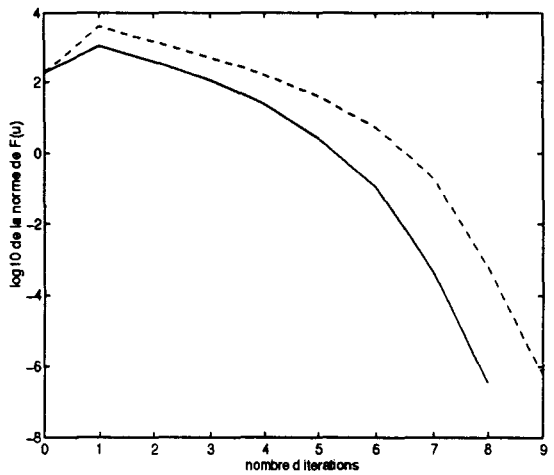
$u_0 = rand_4$



$u_0 = rand_5$



$u_0 = rand_6$

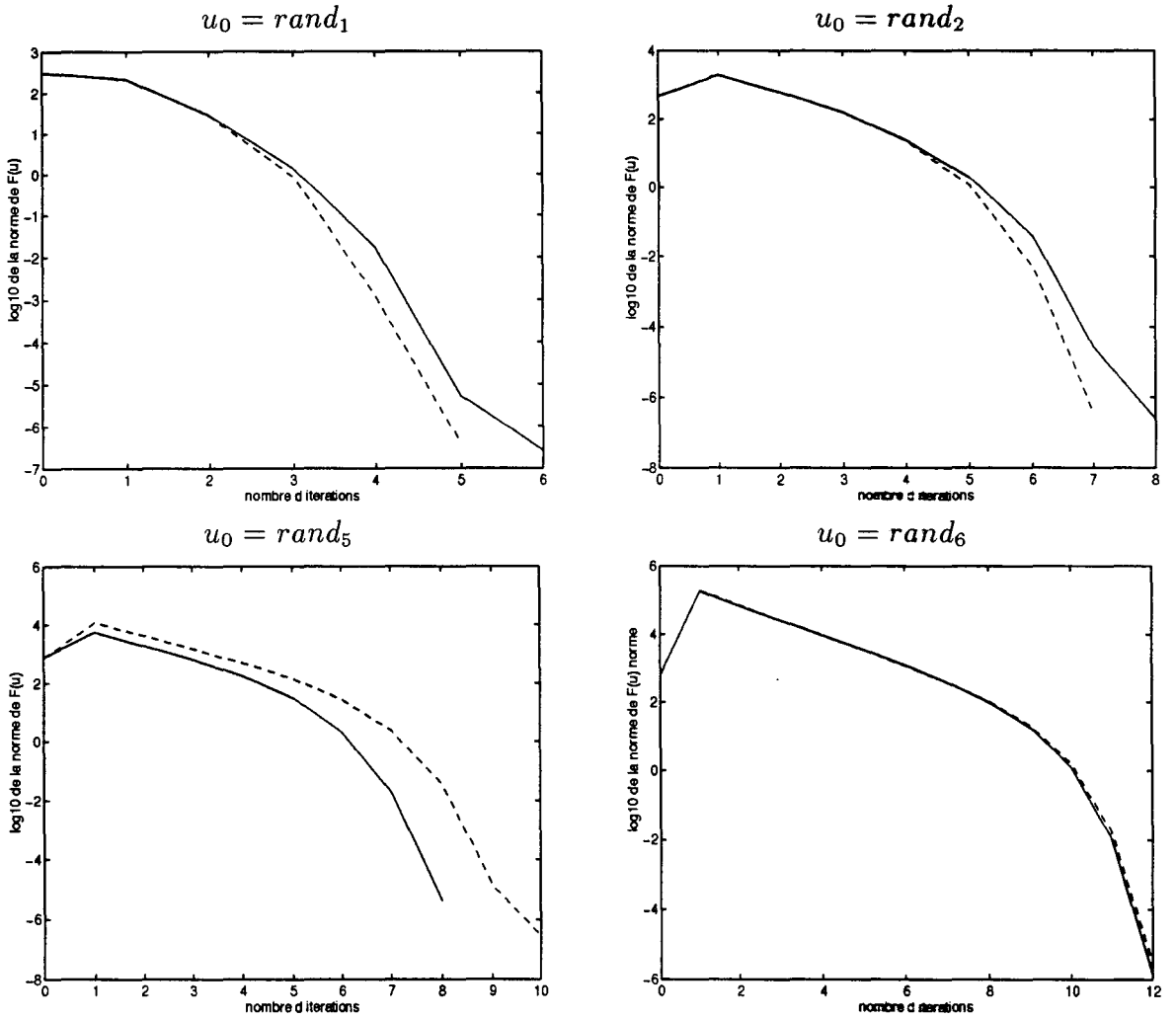


TAB. 4.1 – *Résultats de l'exemple 1,* $\lambda = 1$.

	$u_0 = rand_2$				$u_0 = rand_4$			
	itération	NEF	$\|F(u)\|_2$	CPU	itération	NEF	$\|F(u)\|_2$	CPU
NG	6	73	$4.99\,10^{-7}$	5.60	12	156	$5.07\,10^{-7}$	11.58
NC	7	129	$2.67\,10^{-7}$	9.69	10	187	$3.97\,10^{-7}$	13.65

	$u_0 = rand_5$				$u_0 = rand_6$			
	itération	NEF	$\|F(u)\|_2$	CPU	itération	NEF	$\|F(u)\|_2$	CPU
NG	7	96	$4.64\,10^{-7}$	7.35	10	109	$5.77\,10^{-7}$	8.29
NC	7	129	$2.75\,10^{-7}$	9.76	8	141	$3.49\,10^{-7}$	10.23

Courbes de l'exemple 1, $\lambda = 5$.



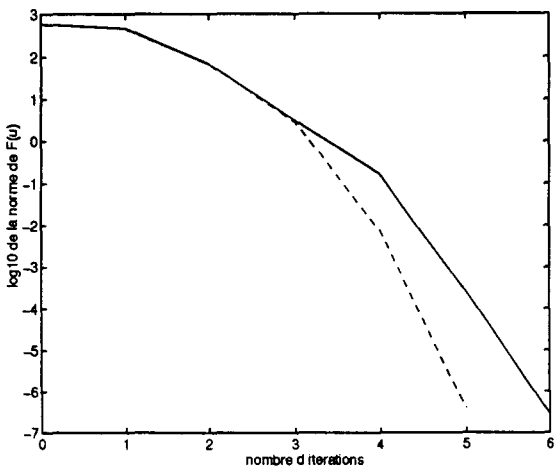
TAB. 4.2 – *Résultats de l'exemple 1, $\lambda = 5$.*

	$u_0 = rand_1$				$u_0 = rand_2$			
	itération	NEF	$\|F(u)\|_2$	CPU	itération	NEF	$\|F(u)\|_2$	CPU
NG	5	61	$3.75\,10^{-7}$	4.58	7	85	$3.18\,10^{-7}$	6.27
NC	6	106	$2.65\,10^{-7}$	7.76	8	130	$2.45\,10^{-7}$	9.27

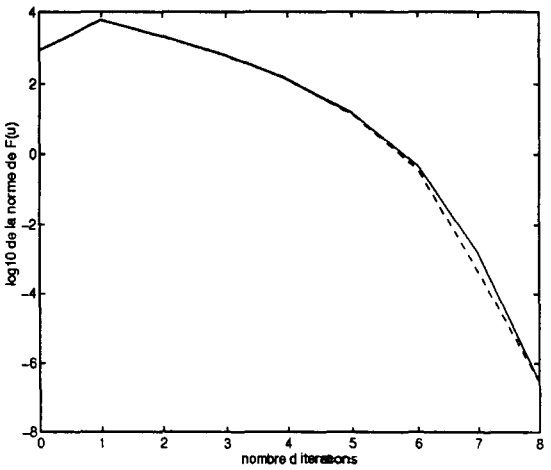
	$u_0 = rand_5$				$u_0 = rand_6$			
	itération	NEF	$\|F(u)\|_2$	CPU	itération	NEF	$\|F(u)\|_2$	CPU
NG	10	132	$2.78\,10^{-7}$	9.62	12	145	$2.86\,10^{-6}$	10.48
NC	8	130	$4.21\,10^{-6}$	9.31	12	200	$1.42\,10^{-6}$	14.21

Courbes de l'exemple 1, $\lambda = 10$.

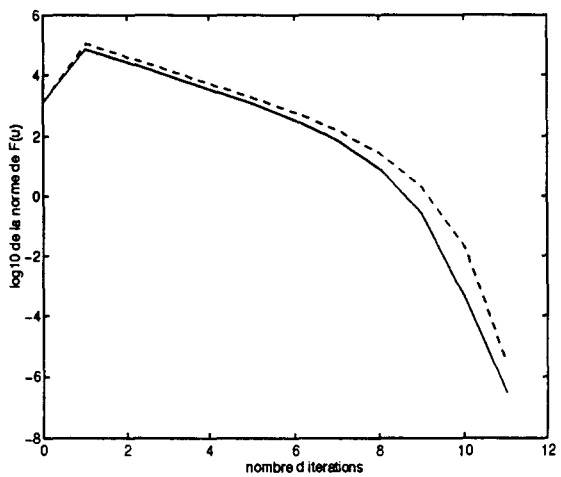
$u_0 = rand_1$



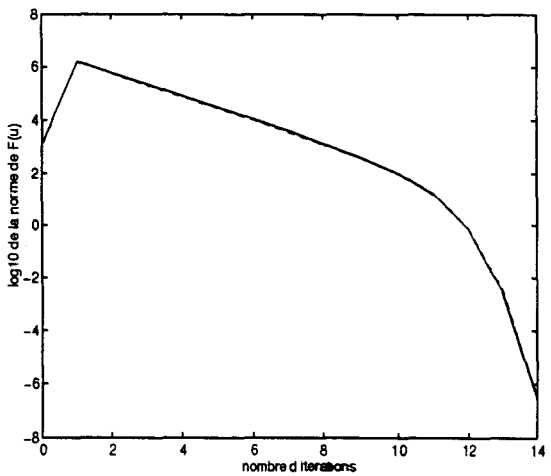
$u_0 = rand_2$



$u_0 = rand_5$



$u_0 = rand_6$



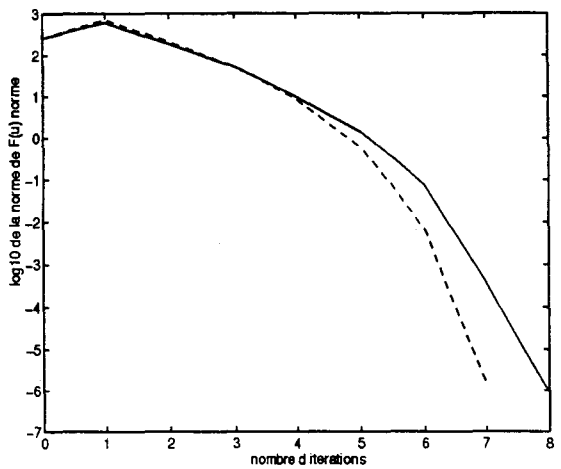
TAB. 4.3 - *Résultats de l'exemple 1,* $\lambda = 10$.

	$u_0 = rand_1$				$u_0 = rand_2$			
	itération	NEF	$\|F(u)\|_2$	CPU	itération	NEF	$\|F(u)\|_2$	CPU
NG	5	61	$3.91\,10^{-7}$	4.54	8	97	$2.96\,10^{-7}$	7.06
NC	6	95	$2.62\,10^{-7}$	6.82	8	130	$2.90\,10^{-7}$	9.31

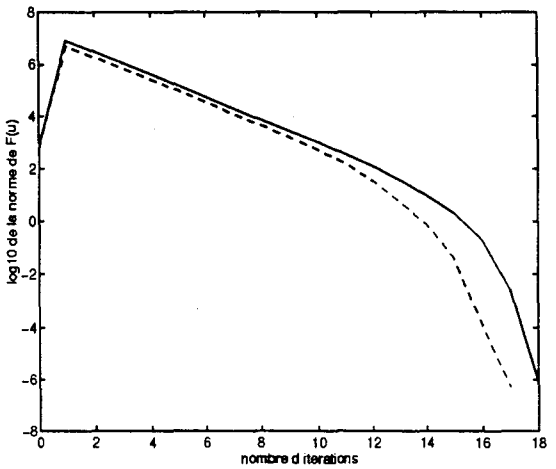
	$u_0 = rand_5$				$u_0 = rand_6$			
	itération	NEF	$\|F(u)\|_2$	CPU	itération	NEF	$\|F(u)\|_2$	CPU
NG	11	133	$2.70\,10^{-6}$	9.58	14	180	$2.78\,10^{-7}$	12.92
NC	11	177	$2.95\,10^{-7}$	12.55	14	279	$3.1\,10^{-7}$	19.79

Courbes de l'exemple 1, $\lambda = -5$.

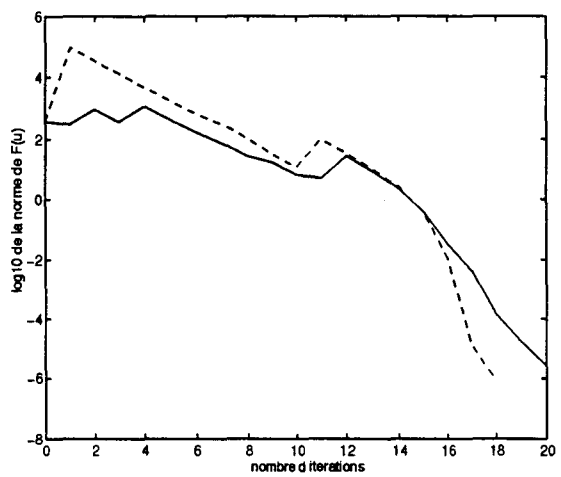
$u_0 = rand_1$



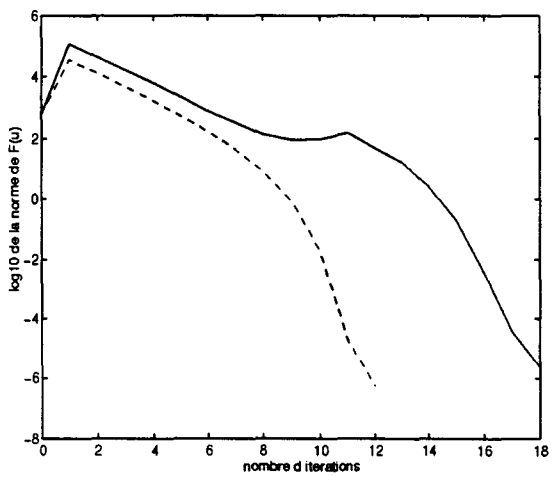
$u_0 = rand_4$



$u_0 = rand_5$



$u_0 = rand_6$



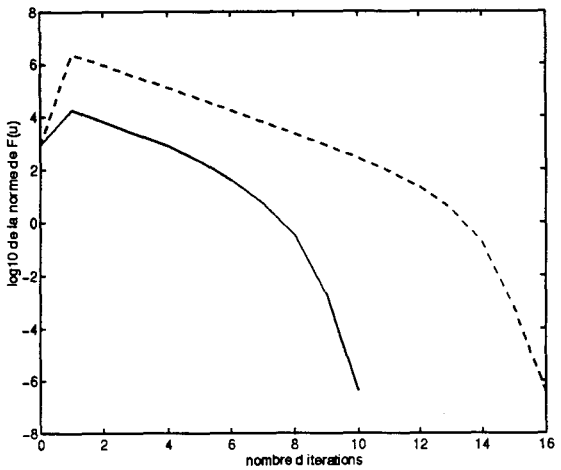
TAB. 4.4 – *Résultats de l'exemple 1,* $\lambda = -5$.

	$u_0 = rand_1$				$u_0 = rand_4$			
	itération	NEF	$\|F(u)\|_2$	CPU	itération	NEF	$\|F(u)\|_2$	CPU
NG	7	85	$1.62\,10^{-6}$	6.22	17	282	$5.19\,10^{-7}$	20.61
NC	8	152	$8.85\,10^{-7}$	10.90	18	371	$8.37\,10^{-7}$	26.28

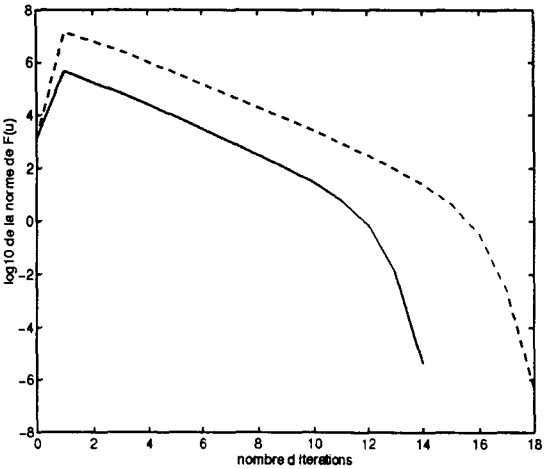
	$u_0 = rand_5$				$u_0 = rand_6$			
	itération	NEF	$\|F(u)\|_2$	CPU	itération	NEF	$\|F(u)\|_2$	CPU
NG	18	371	$7.87\,10^{-7}$	26.79	12	156	$5.37\,10^{-7}$	11.23
NC	20	450	$2.55\,10^{-6}$	31.87	18	371	$2.21\,10^{-6}$	26.25

Courbes de l'exemple 1, $\lambda = -10$.

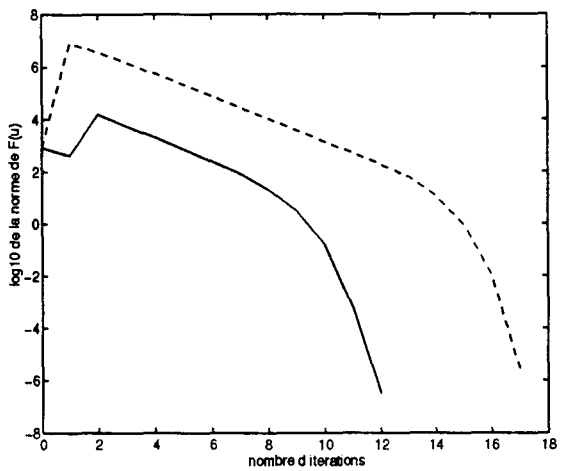
$u_0 = rand_2$



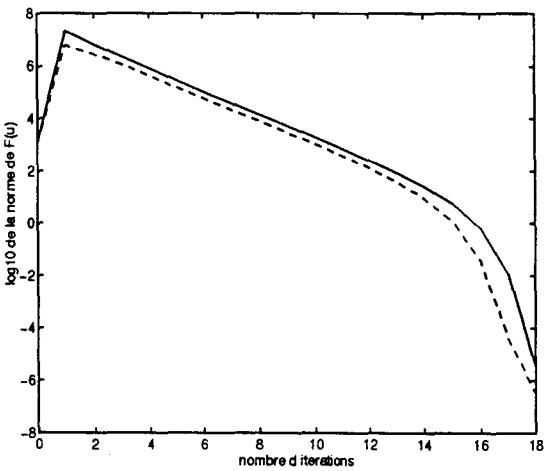
$u_0 = rand_4$



$u_0 = rand_5$



$u_0 = rand_6$



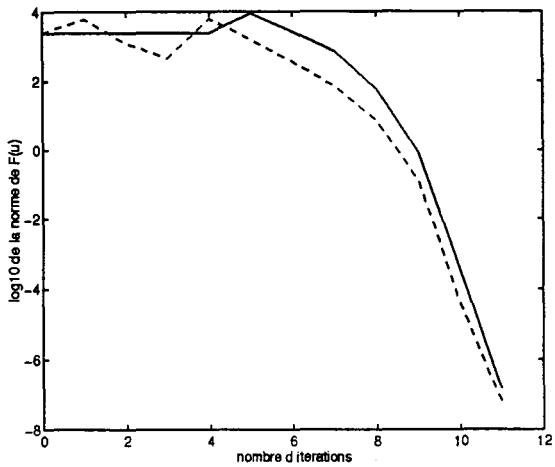
TAB. 4.5 – *Résultats de l'exemple 1,* $\lambda = -10$.

	$u_0 = rand_2$				$u_0 = rand_4$			
	itération	NEF	$\|F(u)\|_2$	CPU	itération	NEF	$\|F(u)\|_2$	CPU
NG	16	237	$3.52\,10^{-7}$	17.05	18	349	$3.92\,10^{-7}$	25.26
NC	10	187	$4.10\,10^{-7}$	13.29	14	257	$4.45\,10^{-6}$	18.36

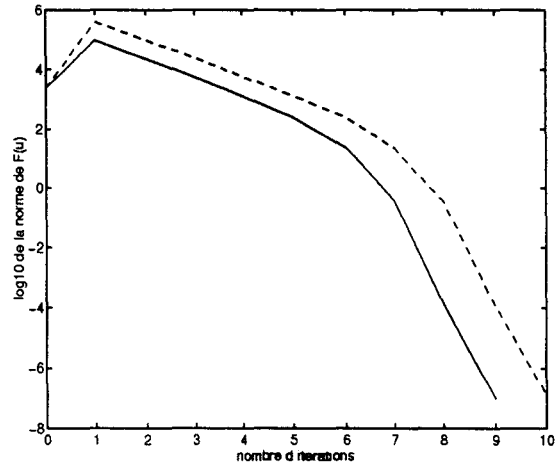
	$u_0 = rand_5$				$u_0 = rand_6$			
	itération	NEF	$\|F(u)\|_2$	CPU	itération	NEF	$\|F(u)\|_2$	CPU
NG	17	337	$3.08\,10^{-6}$	24.30	18	261	$3.03\,10^{-7}$	18.73
NC	12	211	$3.34\,10^{-7}$	15.01	18	349	$3.32\,10^{-6}$	24.68

Courbes de l'exemple 2, $\gamma = 100$.

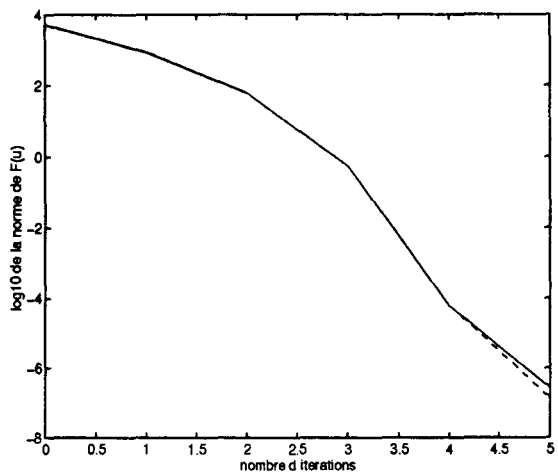
$u_0 = rand_2$



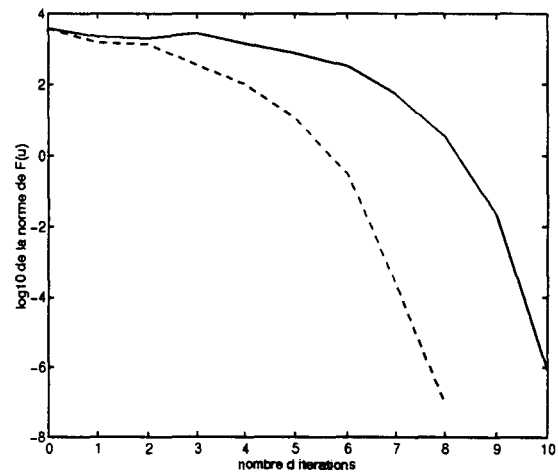
$u_0 = rand_3$



$u_0 = rand_4$



$u_0 = rand_5$

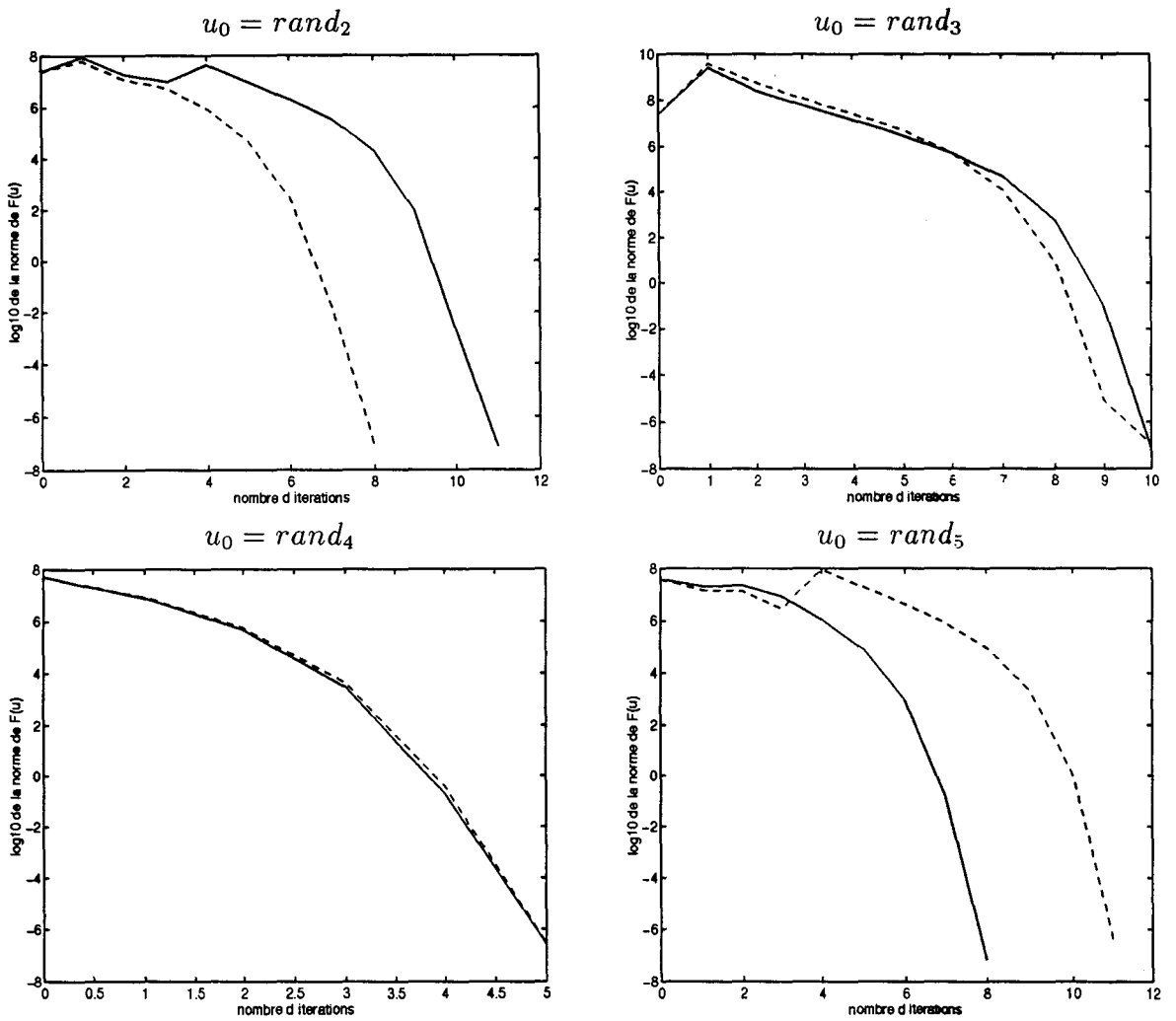


TAB. 4.6 – *Résultats de l'exemple 2,* $\gamma = 100$.

	$u_0 = rand_2$				$u_0 = rand_3$			
	itération	NEF	$\|F(u)\|_2$	CPU	itération	NEF	$\|F(u)\|_2$	CPU
NG	11	144	$6.21\,10^{-8}$	3.41	10	121	$1.24\,10^{-7}$	2.91
NC	11	210	$1.51\,10^{-7}$	4.99	10	120	$9.66\,10^{-8}$	2.90

	$u_0 = rand_4$				$u_0 = rand_5$			
	itération	NEF	$\|F(u)\|_2$	CPU	itération	NEF	$\|F(u)\|_2$	CPU
NG	5	61	$1.42\,10^{-7}$	1.57	8	97	$9.27\,10^{-8}$	2.36
NC	5	61	$2.64\,10^{-7}$	1.59	10	176	$7.86\,10^{-7}$	4.11

Courbes de l'exemple 2, $\gamma = 10^6$.

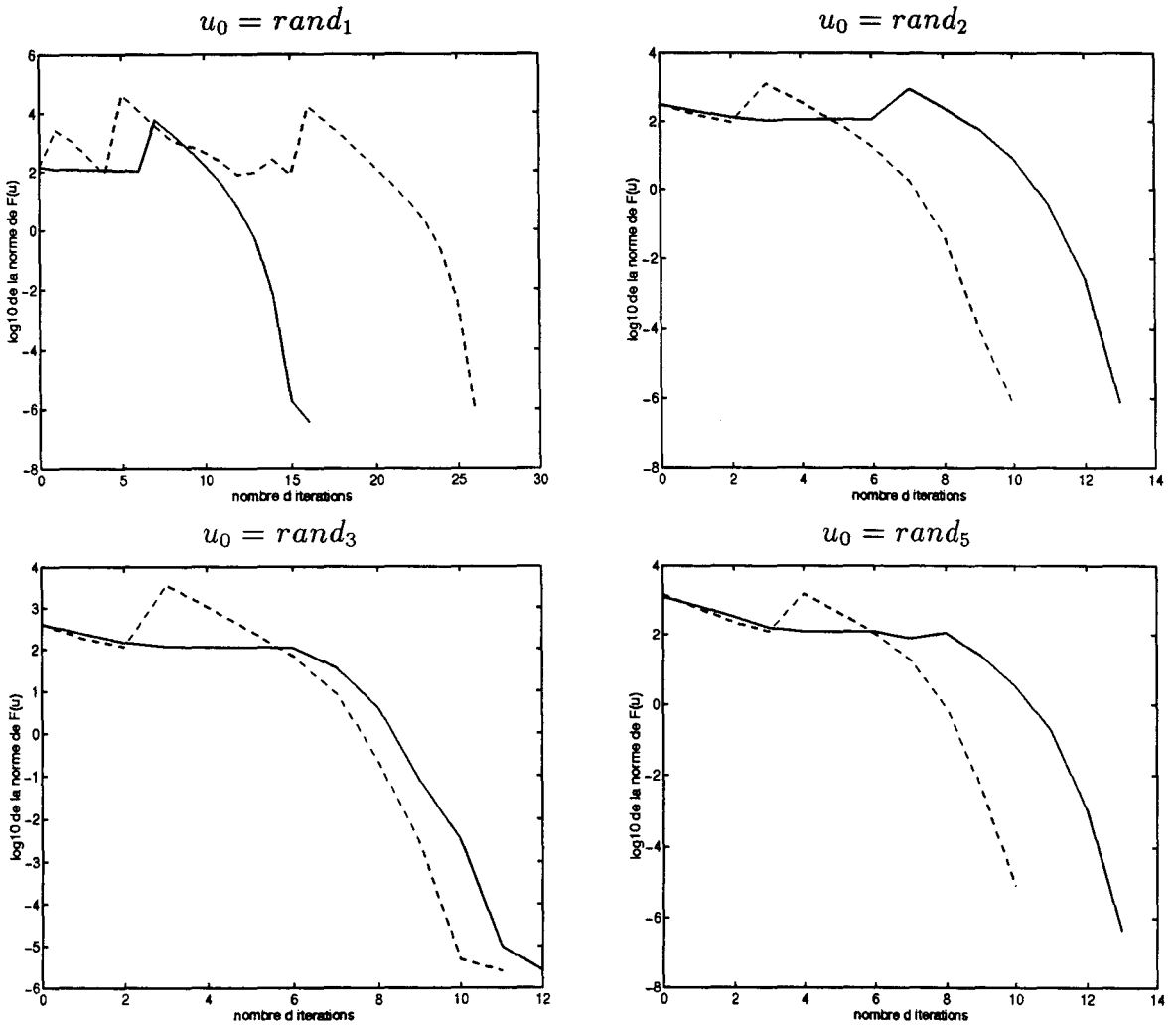


TAB. 4.7 - *Résultats de l'exemple 2,* $\gamma = 10^6$.

	$u_0 = rand_2$				$u_0 = rand_3$			
	itération	NEF	$\|F(u)\|_2$	CPU	itération	NEF	$\|F(u)\|_2$	CPU
NG	8	97	$9.21\,10^{-8}$	2.38	10	132	$1.08\,10^{-7}$	3.08
NC	11	166	$7.68\,10^{-8}$	3.98	10	132	$7.51\,10^{-8}$	3.09

	$u_0 = rand_4$				$u_0 = rand_5$			
	itération	NEF	$\|F(u)\|_2$	CPU	itération	NEF	$\|F(u)\|_2$	CPU
NG	5	61	$2.81\,10^{-7}$	1.54	11	188	$3.65\,10^{-7}$	4.33
NC	5	61	$2.54\,10^{-7}$	1.53	8	130	$6.91\,10^{-8}$	3.05

Courbes de l'exemple 3, $\zeta = 1$.

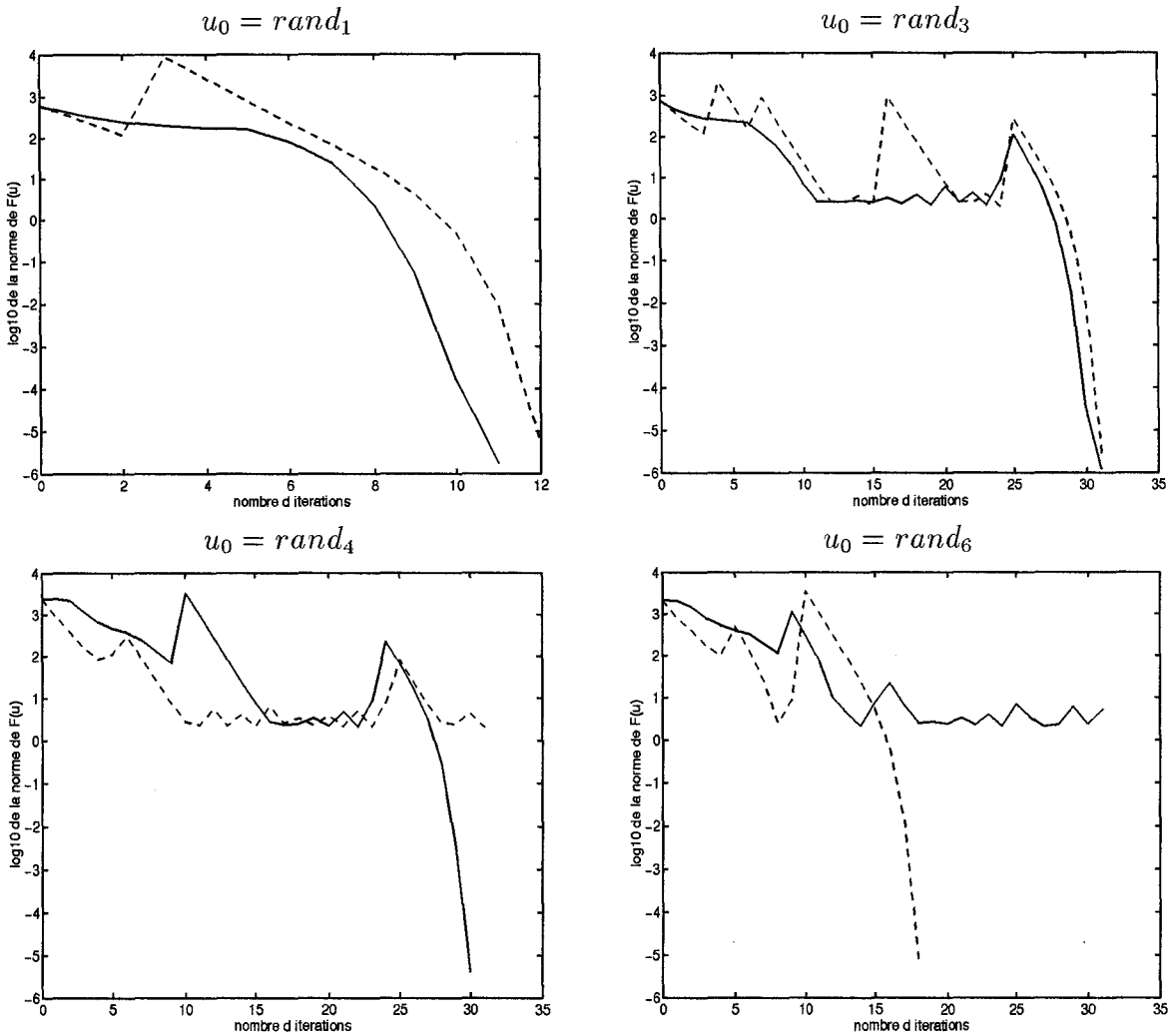


TAB. 4.8 – *Résultats de l'exemple 3, $\zeta = 1$.*

	$u_0 = rand_1$				$u_0 = rand_2$			
	itération	NEF	$\|F(u)\|_2$	CPU	itération	NEF	$\|F(u)\|_2$	CPU
NG	26	398	$9.26\,10^{-7}$	58.49	10	77	$6.32\,10^{-7}$	7.31
NC	16	223	$3.53\,10^{-7}$	23.89	13	136	$7.68\,10^{-7}$	12.84

	$u_0 = rand_3$				$u_0 = rand_5$			
	itération	NEF	$\|F(u)\|_2$	CPU	itération	NEF	$\|F(u)\|_2$	CPU
NG	11	81	$2.56\,10^{-6}$	7.60	10	61	$2.29\,10^{-7}$	5.58
NC	13	141	$2.66\,10^{-6}$	14.45	13	137	$4.45\,10^{-7}$	12.67

Courbes de l'exemple 3, $\zeta = 10$.

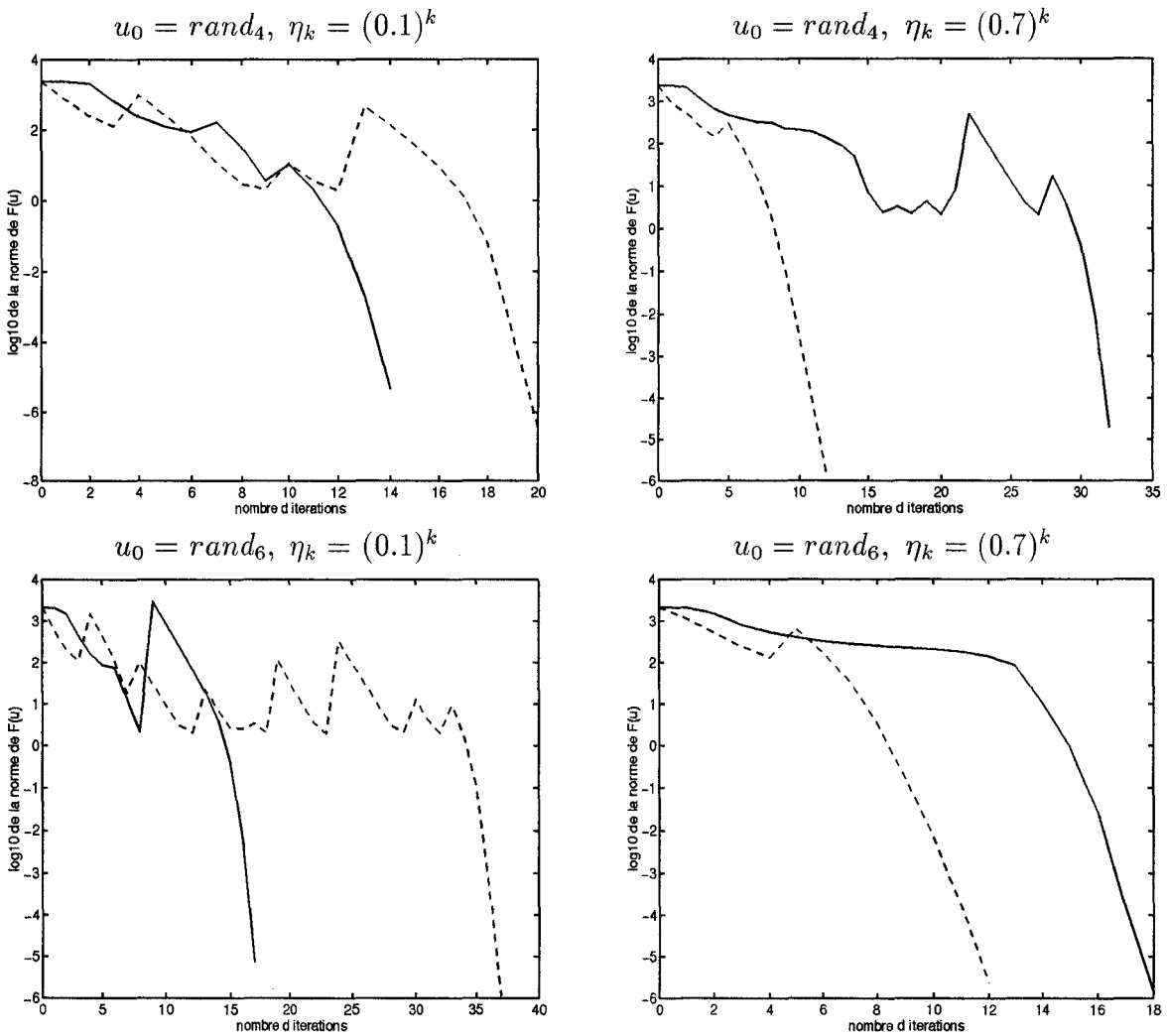


TAB. 4.9 – *Résultats de l'exemple 3,* $\zeta = 10$.

	$u_0 = rand_1$				$u_0 = rand_3$			
	itération	NEF	$\|F(u)\|_2$	CPU	itération	NEF	$\|F(u)\|_2$	CPU
NG	12	155	$5.18\,10^{-6}$	22.40	31	1041	$2.77\,10^{-6}$	190.54
NC	11	190	$1.73\,10^{-6}$	25.45	31	1060	$1.22\,10^{-6}$	178.40

	$u_0 = rand_4$				$u_0 = rand_6$			
	itération	NEF	$\|F(u)\|_2$	CPU	itération	NEF	$\|F(u)\|_2$	CPU
NG	31	$\star$	$\star$	$\star$	18	368	$7.32\,10^{-6}$	58.55
NC	30	950	$4.07\,10^{-6}$	157.3	31	$\star$	$\star$	$\star$

Courbes de l'exemple 3, autres choix de η_k.



TAB. 4.10 – *Résultats de l'exemple 3, autres choix de* η_k.

	$u_0 = rand_4,\ \eta_k = (0.1)^k$				$u_0 = rand_4,\ \eta_k = (0.7)^k$			
	itération	NEF	$\|F(u)\|_2$	CPU	itération	NEF	$\|F(u)\|_2$	CPU
NG	20	749	$3.66\,10^{-7}$	140.17	12	109	$1.62\,10^{-6}$	12.14
NC	14	456	$4.52\,10^{-6}$	75.11	32	874	$1.96\,10^{-5}$	140.9

	$u_0 = rand_6,\ \eta_k = (0.1)^k$				$u_0 = rand_6,\ \eta_k = (0.7)^k$			
	itération	NEF	$\|F(u)\|_2$	CPU	itération	NEF	$\|F(u)\|_2$	CPU
NG	37	1491	$1.14\,10^{-6}$	279.51	12	108	$2.36\,10^{-6}$	12.14
NC	17	607	$7.20\,10^{-6}$	101.47	18	252	$1.43\,10^{-6}$	31.23

4.3.5 Analyse et discussion des résultats

L'analyse des exemples et des résultats du précédent paragraphe montre qu'en général les deux méthodes Newton-GMRES et Newton-CMRH ont des performances comparables. Dans le cas où les deux méthodes convergent à la même itération, nous avons remarqué que le nombre d'évaluations de l'application F de la méthode NC est souvent supérieure à celui de la méthode NG. Cela est du au fait que la méthode CMRH(m), pour des petites valeurs de m, redémarre en général plus de fois que la méthode GMRES(m) (voir les exemples du chapitre 2 §2.5.2).

Notons que si on compare les algorithmes NC et NG à ceux obtenus par les algorithmes des méthodes Newton-GMRES et Newton-CMRH où la matrice jacobienne est approchée par une discrétisation (i.e. $(J(u_k))_{i,j} = (F_i(u_k + \sigma_j e_j^{(n)}) - F_i(u_k))/\sigma_j$, où $e_j^{(n)}$ est le $j^{\text{ème}}$ vecteur de la base canonique), on obtient, dans la pluspart des cas, des résultats similaires. De plus les itérés calculés par les deux versions de chaque méthode sont en général identiques. D'où l'utilité d'approcher l'action du jacobien sur un vecteur par le quotient approché (4.11) puisque dans ce cas il y'a un gain de temps appréciable.

Dans plusieurs exemples, les méthodes NC et NG ne convergent pas vers la même solution, même si la solution de départ est la même pour les deux méthodes. Ce cas de figure peut être expliqué par le fait qu'une ou plusieurs solutions $\widehat{x}_m$ n'ont pas étés calculées avec la même précision. Dans ce cas, il se peut que les solutions $\widehat{x}_m^{\text{GMRES}}$ et $\widehat{x}_m^{\text{CMRH}}$ soient suffisamment distantes pour que les itérés des deux méthodes de Newton-GMRES u_k^{NG} et Newton-CMRH u_k^{NC} soient distinctes et appartiennent à des voisinages différents de deux solutions distinctes du système non linéaire $F(u) = 0$.

Les méthodes Newton-GMRES et Newton-CMRH décrites par les algorithmes NG et NC ne convergent que lorsque la solution initiale u_0 est assez proche d'une solution u_* du système $F(u) = 0$. La convergence de ces méthodes est donc locale. Afin que ces méthodes obtiennent le caractère de convergence globale, il faut leur associer des algorithmes d'optimisation qui permettent, soit de savoir si une solution $\widehat{x}_m$ est une direction de descente pour la fonction $f(u) = \frac{1}{2}\|F(u)\|_2^2$, soit d'obtenir une solution u qui soit assez proche d'une solution u_*, et ainsi il suffit de prendre $u_0 = u$. Ces méthodes ainsi que d'autres ont étés étudiées par Brown et Saad [11], de plus dans [10] ces auteurs ont préconditionné les algorithmes afin d'améliorer et d'accélerer la convergence des méthodes Newton-GMRES et Newton-Arnoldi. Notons que les résultats donnés dans [11] sont plus généraux et sont valables pour la méthode de Newton où la solution du système linéaire est obtenue par des méthodes de projection sur des espaces quelconques et non pas uniquement sur des sous espaces de Krylov.

Bibliographie

[1] A. ABKOWICZ, C. BREZINSKI, *Acceleration property of the hybrid procedure for solving linear systems*, Applications Mathematicae, to appear.

[2] W. ARNOLDI, *The principle of minimized iterations in the solution of the matrix eigenvalue problem.* Quart. Appl. Math., 9 (1951), pp. 17–29.

[3] Z. BAI, D. HU, L. REICHEL, *A Newton basis GMRES implementation*, IMA J.Numer. Anal., 14 (1994), pp. 563–581.

[4] R. BARRET ET AL., *TEMPLATES for the solution of linear systems : building blocks for iterative methods.*

[5] C. BREZINSKI, M. REDIVO ZAGLIA, *Hybrid procedures for solving linear systems*, Numer. Math., 67 (1994), pp. 1–19.

[6] C. BREZINSKI, M. REDIVO ZAGLIA, H. SADOK, *Avoiding breakdown and near-breakdown in Lanczos type algorithms*, Numer. Algorithms., 1 (1991), pp. 261–284.

[7] C. BREZINSKI, M. REDIVO ZAGLIA, H. SADOK, *A breakdown-free Lanczos type algorithm for solving linear systems*, Numer. Math., 63 (1992), pp. 29–38.

[8] P.N. BROWN, *A theoretical comparison of the Arnoldi and GMRES algorithms*, SIAM J. Sci. Stat. Comput., 12 (1992), pp. 58–78.

[9] P.N. BROWN, *A local convergence theory for combined inexact Newton/finite difference projection methods*, SIAM J. Numer. Anal., 24 (1987), pp. 407–434.

[10] P.N. BROWN, Y. SAAD, *Hybrid Krylov methods for nonlinear systems of equations*, SIAM J. Sci. Stat. Comput., 11 (1990), pp. 450–481.

[11] P.N. BROWN, Y. SAAD, *Convergence theory of nonlinear Krylov algorithms*, SIAM J. Optimization., 4 (1994), pp. 297–330.

[12] A. M. BRUASET, *A survey of preconditionned iterative methods.* Pitman Research in Mathematics Series 328 (1995).

[13] J. C. P. Bus, *Numerical solution of systems of non linear equations*, Mathematisch Centrum, Amsterdam, 1980.

[14] E. Craig, *The N-step iteration procedures*, J. Math. Phys., 34 (1955), pp. 64–73.

[15] J. Cullum, A. Greenbaum, *Relations between Galerkin and norm-minimizing iterative methods for solving linear systems*, SIAM J. Matrix Anal. Appl. Comput., 17 (1996), pp. 223–247.

[16] R.S. Dembo, S.C. Eisenstat, and T. Steihaug, *Inexact Newton methods*, SIAM J. Numer. Anal., 19 (1982), pp. 400–408.

[17] J. E. Dennis, Jr. and R. B. Schnabel, *Numerical methods for unconstrained optimization and nonlinear equations*, Prentice-Hall Series in Computational Mathematics, Englewood Cliffs, NJ, 1983.

[18] H. C. Elman, *Iterative methods for large sparse non symmetric systems of linear equations.*, PhD thesis, Computer Science Dept., Yale univ., New Haven, CT, 1982.

[19] R. Fletcher, *Conjugate Gradient methods for indefinite systems, in Numerical Analysis*, G. A. Watson ed. LNM 506, Springer Verlag, Berlin, 1976, pp. 73–89.

[20] R.W. Freund, *Conjugate Gradient type for linear systems with complex symmetric coefficient matrices*, SIAM J. Sci. Stat. Comput., 13 (1992), pp. 425–448.

[21] R.W. Freund, N.M. Nachtigal, *QMR: A quasi minimal residual method for non-Hermitian linear systems*, Numer. Math., 60 (1991), pp. 315–339.

[22] R.W. Freund, N.M. Nachtigal, *An implementation of the QMR method based on coupled two-term recurrences*, SIAM J. Sci. Stat. Comput., 15 (1994), pp. 425–448.

[23] R.W. Freund, M.H Gutknecht, N.M. Nachtigal, *An implementation of the look-ahead Lanczos algorithm for non-Hermitian matrices*, part I. technical Report 90.45, RIACS, NASA Ames Research Center, Moffet Field, CA, Nov. 1990.

[24] T. N. E. Greville, *Some applications of the pseudoinverse of a matrix*, SIAM Review, 2 (1960), pp. 15–22.

[25] I. Gustafsson, *A class of first order factorization methods*, BIT, 18 (1978), pp. 142–156.

[26] M.H Gutknecht, *The unsymmetric Lanczos algorithms and their relations to Padé approximation, continued fractions, and the qd algorithm.* In Proceddings of the Copper Mountain Conference on iterative methods, 1990.

[27] M. HEYOUNI, H. SADOK, *On a variable smoothing procedure for Krylov subspaces methods*, submitted to Linear Algebra Appl.

[28] M.R. HESTNES, *The conjugate gradient method for solving linear systems*, In Proceedings of the Symposium in Applied Mathematics, Volume 6 of Numerical Analysis, pp. 83–102. American Mathematical Society, New York, NY, 1956.

[29] M.R. HESTNES, E.L. STIEFEL, *Methods of conjugate gradients for solving linear systems*, J. Res. Nat. Bur. stand., 49 (1952), pp. 409–436.

[30] A.S. HOUSEHOLDER, F.L. BAUER, *On certain methods for expanding the characteristic polynomial*, Numer. Math., 1 (1959), pp. 29–37.

[31] K.C. JEA, D.M. YOUNG, *Generalized conjugate gradient acceleration of nonsymmetrizable iterative methods*, Linear Algebra Appl., 34 (1980), pp. 159–194.

[32] C. LANCZOS, *An iteration method for the solution of the eigenvalue problem of linear differential and integral operators*, J. Res. Nat. Bur. stand., 45 (1950), pp. 255–282.

[33] C. LANCZOS, *Solution of systems of linear equations by minimized iterations*, J. Res. Nat. Bur. stand., 49 (1952), pp. 33–53.

[34] J. A. MEIJERNIK, H. A. VAN DER VORST, *An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix*, Math. Comp., 31 (1977), pp. 148–162.

[35] N.M. NACHTIGAL, *A look-ahead variant of the Lanczos algorithm and its application to the quasi minimal residual method for nonhermitian linear systems*, PhD thesis, Massachusetts Institute of Technology, 1991.

[36] C.C. PAIGE, M.A. SAUNDERS, *Solution of sparse indefinite systems of linear equations unsymmetric matrices*, SIAM J. Numer. Anal., 12 (1975), pp. 617–624.

[37] B.N. PARLETT, *A new look at the Lanczos algorithm for solving symmetric systems of linear equations*, Linear Algebra Appl., 29 (1980), pp. 323–346.

[38] B.N. PARLETT, D.R. TAYLOR, Z.A. LIU, *A look-ahead Lanczos algorithm for unsymmetric matrices*, Mathematics of computations., 44 (1985), pp. 105–124.

[39] Y. SAAD, *Variations on Arnoldi's method for computing eigenelements of large unsymmetric matrices*, Linear Algebra Appl., 34 (1980), pp. 269–295.

[40] Y. SAAD, *Krylov subspace methods for solving large unsymmetric linear systems*, Mathematics of computations., 37 (1981), pp. 105–126.

[41] Y. SAAD, *Practical use of some Krylov subspace methods for solving indefinite and nonsymmetric linear systems*, SIAM J. Sci. Stat. Comput., 5 (1984), pp. 203–228.

[42] Y. SAAD, M.H. SCHULTZ, *GMRES: A Generalized Minimal Residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Stat. Comput., 7 (1986), pp. 856–869.

[43] Y. SAAD, K. WU, *DQGMRES: A Direct Quasi-Minimal Residual algorithm Based on Incomplete Orthogonalization*, to appear.

[44] Y. SAAD, *ILUT: a dual thersold incomplete ILU factorisation*, Num. Lin. Alg. with Applications., 1 (1994), pp. 387–402.

[45] Y. SAAD, *a flexible inner-outer precondioned GMRES algorithm.*, SIAM J. Sci. Stat. Comput., 14 (1993), pp. 461–469.

[46] H. SADOK, *CMRH: A new method for solving nonsymmetric linear systems based on the Hessenberg reduction algorithm*, submitted.

[47] H. SADOK, *Analysis of the convergence of the minimal and the orthogonal residual methods*, submitted.

[48] W. SCHÖNAUER, *Scientific Computing on Vectors Computers*, North–Holland, Amsterdam, New York, Oxford, Tokyo, 1987.

[49] A. SIDI, W. F. FORD, D.A. SMITH, *Acceleration of convergence of vector sequences*, SIAM J. Numer. Anal., 23 (1986), pp. 178–196.

[50] K. TURNER, H. F. WALKER, *Efficient accuracy solutions with GMRES(m)*, SIAM J. Sci. Stat. Comput., 13 (1992), pp. 815–825.

[51] C. VUIK, *Further experiences with GMRESR*, text presented at Copper mountain Conference, 1992.

[52] C. VUIK, H.A VAN DER VORST, *A comparison of some GMRES-like methods*, Linear Algebra Appl., 160 (1992), pp. 131–162.

[53] H.A VAN DER VORST, C. VUIK, *The superlinear convergence of GMRES*, J. Comput. Appl. Math., 48 (1993), pp. 327–341.

[54] H.A VAN DER VORST, C. VUIK, *GMRESR: a family of nested GMRES methods*, Num. Lin. Alg. with Appl., 1 (1994), pp. 369–386.

[55] P. K. W. VINSOME *ORTHOMIN An iterative method for solving sparse sets of simultaneous linear equations*, In Proc. Fourth Symposium on reservoir simulation, Society of Petroleum engineers of AIME., pp. 149–159, 1976.

[56] R. WEISS, *Convergence behavior of Generalized Conjugate Gradient methods*, PhD thesis, University of Karlsruhe, 1990.

[57] R. WEISS, *Relations between Smoothing and QMR*, Internal report 53/94, University of Karlsruhe, Computing center, 1994.

[58] R. WEISS, W. SCHÖNAUER *Accelerating Generalized Conjugate Gradient methods by smoothing* In Iterative Methods in Linear Algebra, R. Beauwens and P. de Groen eds., pp. 283–292, North-Holland, 1992.

[59] H. F. WALKER, *Implementation of the GMRES method using Householder transformations*, SIAM J. Sci. Stat. Comput., 9 (1988), pp. 152–163.

[60] A. J. WATHEN, *Realistic eigenvalue bounds for the Galerkin mass matrix*, IMA J. Numer. Anal., 7 (1987), pp. 449-457.

[61] J.H. WILKINSON, *The algebraic Eigenvalue Problem*, Claredon Press, Oxford, England, 1965.

[62] L. ZHOU, H.F. WALKER, *Residual smoothing techniques for iterative methods*, SIAM J. Sci. Stat. Comput., 15 (1994), pp. 297–312.