N° d'ordre : 2296 Année 2005

# THÈSE

présentée pour obtenir le titre de

## DOCTEUR DE L'INSTITUT NATIONAL POLYTECHNIQUE DE TOULOUSE

École Doctorale : Informatique & Télécommunications
Spécialité : Informatique & Mathématiques Appliquées

par

**Ahmed Touhami**

# UTILISATION DES FILTRES DE TCHEBYCHEFF ET CONSTRUCTION DE PRÉCONDITIONNEURS SPECTRAUX POUR L'ACCÉLÉRATION DES MÉTHODES DE KRYLOV

Soutenue publiquement le 25 Novembre 2005 devant le jury composé de :

| | | | |
|---|---|---|---|
| M. | Gene H. Golub | Professeur, Université de Stanford | *Président* |
| MM. | Mario Arioli | Directeur de Recherches, RAL | *Rapporteurs* |
| | Jocelyne Erhel | Directrice de Recherches, INRIA | |
| | Gérard Meurant | Directeur de Recherches, CEA | |
| MM. | Iain S. Duff | Professeur, RAL et CERFACS | *Examinateurs* |
| | Luc Giraud | Professeur, ENSEEIHT | |
| M. | Daniel Ruiz | Maître de Conférences, ENSEEIHT | *Co-encadreur* |
| M. | Michel Daydé | Professeur, ENSEEIHT | *Directeur de thèse* |

*À mes parents*
*À mon frère et À mes sœurs*
*À mes tantes*

Merci pour tous vos sacrifices

# Remerciements

Cette thèse a été effectuée entre 2002 et 2005 à l'**I**nstitut de **R**echerche en **I**nformatique de **T**oulouse (I.R.I.T) site de l'**E**cole **N**ationale **S**upérieure d'**E**lectrotechnique, d'**E**lectronique, d'**I**nformatique, d'**H**ydraulique et des **T**élécommunications de Toulouse (E.N.S.E.E.I.H.T). L'ensemble de ces travaux ainsi que les activités de monitorat n'auraient pu avoir lieu sans les multiples échanges avec les acteurs de la recherche et de l'enseignement de l'E.N.S.E.E.I.H.T, du C.E.R.F.A.C.S et de l'I.P.S.T–C.N.A.M. Leurs conseils, leur aide, leur amitié, et en un mot leur présence ont fait de ces trois années l'une des périodes les plus enrichissantes de ma vie. Qu'ils soient tous ici chaleureusement remerciés.

Je tiens particulièrement à remercier :

Michel Daydé, Professeur à l'E.N.S.E.E.I.H.T, pour m'avoir accueilli dans son laboratoire, et d'avoir accepté de diriger mes travaux de thèse. Ses conseils avisés et sa clairvoyance m'ont été d'une aide précieuse.

Daniel Ruiz, Maître de Conférences à l'E.N.S.E.E.I.H.T, pour avoir accepté co-diriger mes travaux de thèse. Je tiens ici à lui exprimer toute ma reconnaissance pour son aide et ses encouragements. Ses qualités scientifiques et humaines ont toujours été pour moi, une source de motivation même dans les moments difficiles.

Gene H. Golub, Professeur Fletcher Jones à l'université de Stanford qui m'a fait l'honneur de participer et de présider le jury de cette thèse. Je veux aussi le remercier pour les conseils et remarques qu'ils m'a apportés lors de nos nombreux échanges et rencontres sans oublier son invitation à l'universié de Stanford.

Mario Arioli, Directeur de Recherches au Rutherford Appleton Laboratory (R.A.L), Jocelyne Erhel, Directrice de Recherches à l'I.N.R.I.A et Gérard Meurant Directeur de Recherches au C.E.A, qui se sont intéressés à mon travail et qui ont accepté de le juger. Leurs remarques pertinentes ont permis d'améliorer la présentation finale de ce manuscrit.

Iain S. Duff, Professeur au R.A.L, Luc Giraud, Professeur à l'E.N.S.E.E.I.H.T et Miloud Sadkane, Professeur à l'Université de Bretagne Occidentale, pour leurs encouragements et l'attention particulière qu'ils ont accordée à ce tra-

vail.

Serge Gratton, Senior Researcher dans l'équipe ALGO du C.E.R.F.A.C.S qui s'est toujours montré disponible pour d'innombrables discussions. J'ai pu profiter de la finesse de ses connaissances et de la pertinence de ses remarques.

Daniel Loghin, Senior Lecturer à l'université de Birmingham pour toutes les discussions scientifiques que nous avons pu avoir au C.E.R.F.A.C.S. Ses conseils et son aide m'ont toujours été précieux. J'ai éprouvé un réel plaisir à travailler avec lui.

Frédéric Messine, Maître de Conférences à l'E.N.S.E.E.I.H.T, qui m'a toujours amicalement conseillé et aidé. Les diverses discussions que nous avons pu avoir ont contribué à mon epanouissement scientifique.

Je remercie l'ensemble du personnel enseignant et administratif de la filière et du département Informatique de l'I.P.S.T–C.N.A.M pour la sympathie qu'ils m'ont témoignée durant la réalisation de ce travail, en particulier Abdelkrim Achaïbou qui a accepté de me parrainer durant mon monitorat et Hadj Batatia, responsable de la filière informatique au C.N.A.M Midi-Pyrénées pour la liberté qu'il m'a laissée dans mes choix d'enseignements ainsi que pour leurs conseils stimulants et précieux.

Ces trois années aurait été bien ternes sans la présence de tous les membres de l'équipe APO et du projet GRID, en particulier toutes les personnes du quatrième étage du laboratoire d'informatique : je les remercie pour leur gentillesse et leur bonne humeur.

Un grand merci à l'ensemble des thésards que j'ai croisés ou côtoyés, je leur souhaite tous une bonne continuation. Un merci particulier aux doctorants Aurélie Hurault et Pierre-Loïc Garoche avec qui j'ai eu le plaisir de partager des heures de labeur, à mes collègues Ming Chau et Pierre Martinon, docteurs matheux, sans oublier Clovis Tauber ne serait-ce que pour la "*relecture*" de cette page ! Nos discussions, pas toujours scientifiques, ont souvent brisé les dynamiques négatives lors des périodes difficiles.

Je remercie également le personnel du Service Édition de l'E.N.S.E.E.I.H.T pour le soin qu'il a apporté à la réalisation du tirage de ce document.

Je voudrais finalement exprimer ma profonde gratitude à mes parents, ma famille et mes amis dont le soutien a été sans faille.
Maman, Papa, je vous remercie de n'avoir pas douté, je vous dois la vie, et aujourd'hui, je vous dois ce que va être ma vie.

# Table des matières

# Introduction

La modélisation des problèmes que l'on rencontre en physique, en mécanique, etc., et d'une façon générale dans les sciences pour l'ingénieur, conduit, éventuellement après une étape de discrétisation, à la résolution de systèmes d'équation en dimension finie. Lorsque les systèmes sont non linéaires, leur résolution passe par une étape de linéarisation, ou est obtenue grâce à une extension des méthodes de résolution des systèmes linéaires. On peut donc dire que le calcul scientifique repose sur la résolution de systèmes linéaires, pour laquelle il est fondamental que soient développées des méthodes numériques rapides, robustes, et faciles à utiliser.

Par résolution, nous entendons soit la recherche des solutions du problème $\mathbf{Ax} = \lambda\mathbf{x}$, c.-à.-d. le calcul de valeurs et vecteurs propres où $\lambda$ est la valeur propre et $\mathbf{x}$ est le vecteur propre correspondant, soit la recherche de la solution $\mathbf{x}$ d'un système linéaire $\mathbf{Ax} = \mathbf{b}$, éventuellement au sens des moindres carrés (si le nombre d'équations est supérieur au nombre d'inconnues).

La détermination de valeurs propres et vecteurs propres a pris une place importante dans le calcul scientifique en raison par exemple, de l'information qu'elle apporte, sur la stabilité du problème physique considéré. Elle est d'une grande utilité dans le domaine de la chimie quantique pour la détermination d'ondes par exemple, en recherche opérationnelle pour trouver les distributions stationnaires des chaînes de Markov, dans le domaine de la mécanique des fluides pour étudier la vitesse d'écoulement ... Dans la pratique on a généralement besoin de ne calculer que quelques valeurs propres dans une région particulière du plan complexe, comme les plus petites ou les plus grandes en module, ou celles au voisinage d'une certaine valeur fixée. Pour réaliser cela, on peut réduire la taille du problème de départ, d'où le recours à des techniques itératives de projections orthogonales. Ces méthodes de projection consistent en général à construire une matrice dite projetée, de taille plus petite que celle de la matrice de départ. Parmi les méthodes de projection, on distingue deux types :

Méthodes asymptotiques : Ce sont des méthodes itératives, la projection se fait sur un sous-espace de type $\mathbf{G}_k = \mathbf{A}^k\mathbf{S}$, où $\mathbf{S}$ est un sous-espace engendré par un vecteur $\mathbf{u}$ ou par un ensemble de $r$ vecteurs indépendants

$\mathbf{S} = [\mathbf{u}_1, \ldots, \mathbf{u}_r]$. La dimension de $\mathbf{G}_k$ reste constante au cours des itérations. La convergence de ces méthodes est assurée, sous des hypothèses appropriées, quand $k$ tend vers l'infini [15, 34, 63, 74]. on peut citer parmi ces méthodes, la méthode du sous-espace itéré, la méthode de la puissance et la méthode des itérations inverses.

Méthodes de type Krylov : Pour les méthodes de projection de type Krylov, la projection se fait sur un sous-espace de Krylov de taille $m$, où $m$ croît avec les itérations. Contrairement aux méthodes asymptotiques, la terminaison des méthodes de type Krylov est assurée pour $m = n$ ( en arithmétique exacte), bien qu'on espère qu'elle se produise pour $m \ll n$ ($n$ étant la taille du problème). Parmi les méthodes de projection de type Krylov pour le calcul des éléments propres, on peut citer la méthode d'Arnoldi pour la résolution des problèmes non symétriques, ou celle de Lanczos pour les problèmes symétriques.

Pour la résolution des systèmes linéaires, il existe plusieurs méthodes possibles ; certaines sont directes et d'autres itératives. En règle générale, les méthodes directes (les factorisations LU et QR) sont des solveurs efficaces et fiables. Pour des problèmes de très grande taille, cependant ces méthodes directes peuvent devenir très chères, tant en occupation mémoire qu'en coût de calcul. Une alternative pourrait être l'utilisation des méthodes itératives. La plupart des méthodes itératives manipulent le système linéaire au travers de produits matrice-vecteur, ce qui réduit la place mémoire, en nécessitant de stocker que les éléments indispensables au calcul de ces produits. Dans certains cas, la matrice du système n'est pas explicitement connue, mais simplement donnée par l'appel à une fonction réalisant le produit de cette matrice par un vecteur quelconque. Dans ce cas, les méthodes itératives restent la seule alternative possible. Cependant ces méthodes restent généralement moins fiables que les méthodes directes, et plus ou moins efficaces sur des classes de problèmes bien spécifiques. L'augmentation de la taille des systèmes à résoudre conduit à avoir recours aux techniques de projection développées depuis les trente dernières années. On distingue donc deux types de méthodes itératives pour la résolution de systèmes linéaires :

Méthodes asymptotiques : Les méthodes asymptotiques, qui convergent à l'infini, sont les plus anciennes et les plus simples à implanter. Mais elles offrent souvent une convergence lente. Parmi ces méthodes, les méthodes de Jacobi, Gauss-Seidel et la méthode de Relaxation sont les plus connues. Actuellement ces méthodes ne sont plus guère utilisées en tant que solveurs linéaires, mais restent intéressantes en tant que préconditionneurs pour d'autres méthodes itératives.

Méthodes de type Krylov : Les méthodes de type Krylov, employées depuis une trentaine d'années, peuvent se révéler très efficaces. Leur analyse est plus compliquée que celle des méthodes asymptotiques. Ces méthodes se basent sur une technique de projection sur un sous-espace de Krylov de di-

mension $m \ll n$, qui constituent une suite de sous-espaces emboîtés de taille croissante, et permettant de construire une séquence de vecteurs convergeant vers la solution recherchée. Ces sous-espaces ont la forme suivante :

$$\mathcal{K}_m(\mathbf{A}, \mathbf{u}) = [\mathbf{u}, \mathbf{A}\mathbf{u}, \cdots, \mathbf{A}^{m-1}\mathbf{u}]$$

avec $\mathbf{u} \in \mathbb{R}^n$ et $m \leq n$. La convergence de ces méthodes est assurée en théorie pour $m = n$ au plus. Les méthodes de Krylov les plus utilisées pour la résolution des systèmes linéaires sont les méthode du résidu minimal ou GMRES, dans le cas général, et la méthode du Gradient Conjugué (CG), pour le cas symétrique.

Au Chapitre 1, nous faisons une présentation succincte de quelques méthodes itératives, asymptotiques ou de type Krylov pour résoudre ces deux problèmes d'algèbre linéaire creux et de grande taille.

L'approche que nous développons dans cette thèse se base sur l'utilisation des filtres de Tchebycheff et sur la construction de préconditionneurs spectraux pour l'accélération des méthodes de Krylov.

Dans une première partie, nous considérons le cas où le système linéaire est symétrique défini positif (SPD). Pour résoudre ce type de système, la méthode itérative habituellement utilisée est le gradient conjugué. Le but de cette méthode est de trouver $\mathbf{x}^{(m)} \in \mathbf{x}^{(0)} + \mathcal{K}_m(\mathbf{A}, \mathbf{r}^{(0)})$ minimisant la quantité $||\mathbf{x} - \mathbf{x}^{(m)}||_{\mathbf{A}}$, appelée erreur, et où $\mathbf{r}^{(0)} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(0)}$ est le résidu initial. Comme la matrice $\mathbf{A}$ est SPD et les espaces de Krylov sont emboîtés et de taille croissante, l'erreur ne peut que décroître au sens large au fur et à mesure de la construction de ces espaces. Le premier travail était motivé par le fait que la convergence de la méthode du gradient conjugué est souvent pénalisée dès que le système linéaire à résoudre possède des valeurs propres au voisinage de zéro. Plusieurs techniques sont proposées dans la littérature, pour atténuer l'effet néfaste de ces plus petites valeurs propres, en mettant à jour le préconditionneur ou en contraignant la méthode du gradient conjugué à travailler dans le complémentaire orthogonal du sous espace invariant associé aux plus petites valeurs propres.

Récemment, Arioli et Ruiz [4], ont proposé un algorithme de factorisation spectrale partielle (dénommé Tchebycheff-PSF) combinant les filtres polynomiaux de Tchebycheff et le processus de Lanczos, pour calculer, avec différents niveaux de filtrage $\varepsilon$, une base orthogonale $\mathbf{W}(\varepsilon)$ du sous espace invariant associé aux plus petites valeurs propres. Une partie de notre travail (cf. [28, 29]), relativement à cette méthode, a consisté à l'incorporer au sein de méthodes de résolution exploitant l'information spectrale du système à résoudre, et à évaluer son potentiel et ses limitations. Au Chapitre 2, nous introduisons les filtres polynomiaux de Tchebycheff et leur utilisation dans la construction de l'algorithme de factorisation spectrale partielle Tchebycheff-PSF. Nous décrirons également l'influence de certains paramètres sur le comportememt de cette technique. L'exploitation concrète de cette approche,

combinée à diverses méthodes de résolution est étudiée et validée exprimentalement au Chapitre 3. Parmis les méthodes de résolution, nous considérons la version du gradient conjugué avec déflation [77], qui consiste à augmenter l'espace de Krylov généré dans les itérations du gradient conjugué avec la base $\mathbf{W}(\varepsilon)$, et à projeter, à chaque itération, les vecteurs de Krylov sur le complémentaire orthogonal de cette base $\mathbf{W}(\varepsilon)$. Une autre technique consite à appliquer l'algorithme du gradient conjugué au système projeté [51]. Une autre approche encore est basée sur un préconditionnement spectral [10], qui vise à translater de 1 la position des plus petites valeurs propres de la matrice du système. Tous ces algorithmes ont été implantés en **Matlab**© pour l'évaluation de leur potentiel ou de leurs limitations.

Dans une deuxième partie, au Chapitre 4, nous proposons une nouvelle approche basée sur une combinaison de la méthode du gradient conjugué avec des filtres polynomiaux de Tchebycheff comme préconditionneurs. Cette technique vise à mettre la méthode du gradient conjugué dans un mode particulier, où le conditionnement du système linéaire n'est pas tellement réduit, mais son spectre en grande partie regroupé autour de 1. Notre préconditionneur consiste à appliquer les filtres polynomiaux de Tchebycheff à une partie seulement du spectre de la matrice d'itérations, pour pouvoir décaler la quasi totalité des valeurs propres du système linéaire prés de 1, sans dégrader la distribution des plus petites valeurs propres. La méthode résultant de cette combinaison Tchebycheff–Krylov se distingue par sa capacité à construire une base de Krylov de taille petite, très riche en ce qui concerne les plus petits vecteurs propres/valeurs propres. Cette base de Krylov peut être réutilisée dans un deuxième niveau de préconditionnement pour une multirésolution, c'est-à-dire la résolution d'une séquence de systèmes linéaires avec la même matrice mais différents seconds membres. La technique de préconditionnement à base de polynôme de Tchebycheff requiert principalement la multiplication d'un vecteur par la matrice initiale et quelques mises-à-jour de vecteur, mais aucun produit scalaire. Cette remarque est très importante dans le contexte du calcul parallèle, et en particulier dans les environnements à mémoire distribuée où le calcul des produits scalaires exige une attention particulière. Enfin, nos algorithmes sont testés sur des exemples extraits de la bibliothèque Harwell-Boeing et sur des problèmes d'Équations aux Dérivées Partielles (EDP). Ces tests confirment tous les résultats théoriques.

Dans une troisième partie (cf. Chapitre 5), nous examinons certains cas pathologiques pour la méthode du Chapitre 4 et proposons des remèdes comme par exemple l'utilisation du gradient conjugué par blocs. Nous introduisons également une nouvelle technique adaptative de préconditionnement dans le cadre d'une multirésolution issue d'un processus de linéarisation. Notre préconditionneur est basé sur l'information des sous espaces de Krylov produite aux étapes précédentes dans l'itération non-linéaire. Notre

approche a été validée sur des problèmes d'équations aux dérivées partielles non-linéaires en mécanique des fluides (Navier–Stokes) et sur des méthodes de décomposition de domaine pour la résolution de problèmes elliptiques.

# Chapitre 1

# Méthodes de Projection en Algèbre Linéaire

De nombreuses applications scientifiques et industrielles, conduisent à des problèmes d'algèbre linéaire avec des matrices creuses et de grande taille. Les deux principaux problèmes d'algèbre linéaire sont :

- Résoudre le système linéaire

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{1.1}$$

  où $\mathbf{A} \in \mathbb{R}^{n\times n}$ est une matrice inversible, creuse et de grande taille et $\mathbf{b} \in \mathbb{R}^n$ le second membre.
- Trouver les valeurs et vecteurs propres de $\mathbf{A}$, tels que $\mathbf{A}\mathbf{u} = \mu\mathbf{u}$, où $\mu \in \mathbb{C}$ est la valeur propre et $\mathbf{u} \in \mathbb{C}^n$ le vecteur propre correspondant.

Dans ce chapitre nous faisons une présentation de quelques méthodes itératives, asymptotiques ou de type Krylov pour résoudre des problèmes d'algèbre linéaire creux et de grande taille. Pour le calcul des valeurs propres on prendra la méthode de la puissance comme exemple de méthode asymptotique, la méthode d'Arnoldi comme exemple de méthode à terminaison finie dans le cas non symétrique et la méthode de Lanczos dans le cas symétrique. Pour la résolution de systèmes linéaires, on ne considérera que les méthodes de type Krylov, et on étudiera la méthode du résidu minimal GMRES comme exemple général et la méthode du Gradient Conjugué (CG), dans le cas symétrique. Le choix de ces méthodes a été dicté par la volonté de traiter des approches représentatives du calcul numérique actuel. Ce sont ces méthodes qui seront reprises pour les expérimentations dans les chapitres qui suivent.
Dans tout ce qui suit, on note $\mathbf{x}^{(m)}$ l'approximation de la solution à l'itération $m$, on note $\mathbf{r}^{(m)} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(m)}$ le résidu et $\mathbf{e}^{(m)} = \mathbf{x} - \mathbf{x}^{(m)}$ l'erreur, de sorte que $\mathbf{r}^{(m)} = \mathbf{A}\mathbf{e}^{(m)}$.
Sauf citation supplémentaire, tous les résultats et algorithmes présentés ici sont décrits, en particulier, dans [26, 34, 31, 46, 63, 74, 75].

## 1.1 Méthodes Asymptotiques de Projection sur un Sous-Espace

Les méthodes de projection sur un sous-espace, telles que la méthode d'itération de sous-espace et la méthode de la puissance, sont parmi les plus anciennes pour l'approximation de valeurs propres et vecteurs propres associés. On présentera comme exemple de ces méthodes, la méthode de la puissance itérée et sa variante, la méthode de la puissance itérée inverse.

### 1.1.1 Méthode de la Puissance

La méthode de la puissance est l'une des plus anciennes et des plus utilisées pour le calcul des valeurs propres. Elle s'appuie sur le fait que la suite des itérés $\mathbf{A}^k\mathbf{v}$, $k = 1,\ 2,\ \ldots$ où $\mathbf{v}$ est le vecteur initial, converge (sous de bonnes conditions [15, 34, 63, 74]) vers un vecteur propre dominant, c'est-à-dire un vecteur propre associé à la valeur propre de plus grand module de $\mathbf{A}$. La méthode d'itérations de sous-espace généralise ce principe pour l'obtention d'une base du sous-espace associé aux $m$ valeurs propres dominantes [74, § 5].

L'Algorithme 1.1 représente l'application de la méthode de la puissance pour une matrice $\mathbf{A}$ en partant d'un vecteur initial $\mathbf{v}^{(0)}$.

Algorithme 1.1 : Méthode de la puissance

**Début**

1. Choix de $\mathbf{v}^{(0)}$ tel que $||\mathbf{v}^{(0)}||_2 = 1$
2. **Pour** $k = 1, 2, \ldots,\ m$ **Faire**
   i. $\mathbf{x}^{(k)} = \mathbf{A}\mathbf{v}^{(k-1)}$
   ii. $\mathbf{v}^{(k)} = \mathbf{x}^{(k)}/||\mathbf{x}^{(k)}||_2$
   iii. $\lambda_k = {\mathbf{v}^{(k)}}^T\mathbf{A}\mathbf{v}^{(k)}$
3. **FinPour**

**Fin**

### 1.1.2 Méthode d'Itérations Inverses

La méthode d'itérations inverses est une variante de la méthode de la puissance ; c'est la méthode de la puissance appliquée à la matrice $\mathbf{A}^{-1}$ et non pas à la matrice $\mathbf{A}$. Donc cette méthode calcule la valeur propre de plus grand module de $\mathbf{A}^{-1}$ ainsi qu'un vecteur propre associé, ce qui correspond

au calcul de la valeur propre de plus petit module de $\mathbf{A}$ et à un vecteur propre associé.

L'Algorithme 1.2 résume la méthode d'itérations inverses appliquée à la matrice $\mathbf{A}$ en partant d'un vecteur initial $\mathbf{v}^{(0)}$.

Algorithme 1.2 : Itérations inverses

**Début**

1. Choix de $\mathbf{v}^{(0)}$ tel que $||\mathbf{v}^{(0)}||_2 = 1$
2. **Pour** $k = 1, 2, \ldots,$ $m$ **Faire**
   i. Résoudre $\mathbf{A}\mathbf{x}^{(k)} = \mathbf{v}^{(k-1)}$
   ii. $\mathbf{v}^{(k)} = \mathbf{x}^{(k)}/||\mathbf{x}^{(k)}||_2$
   iii. $\lambda_k = {\mathbf{v}^{(k)}}^T \mathbf{A}\mathbf{v}^{(k)}$
3. **FinPour**

**Fin**

## 1.2 Méthodes de Projection sur un Sous-Espace de Krylov

Les méthodes de projection sur un sous-espace de Krylov sont très populaires et ont été largement classifiées et étudiés [34, 31, 46, 75]. Ces méthodes se basent sur des techniques de projection orthogonale sur un sous-espace, appelé sous-espace de Krylov, et donnent naissance à toute une famille de méthodes dites polynomiales. Cette présentation, ainsi que les motivations résumées dans ce qui suit, sont fortement inspirées de [26, § 6].

Soit $\mathbf{x}^{(0)}$ une approximation initiale et $\mathbf{r}^{(0)} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(0)}$.

**Définition 1.1.** Les méthodes polynomiales sont définies par des itérations du type

$$\mathbf{x}^{(m)} = \mathbf{x}^{(0)} + \mathcal{P}_{m-1}(\mathbf{A})\mathbf{r}^{(0)},$$

où $\mathcal{P}_{m-1}$ est un polynôme de degré au plus $m-1$.
Le sous-espace $\mathcal{K}_m(\mathbf{A}, \mathbf{r}^{(0)}) = [\mathbf{r}^{(0)}, \mathbf{A}\mathbf{r}^{(0)}, \ldots, \mathbf{A}^{m-1}\mathbf{r}^{(0)}]$ est appelé espace de Krylov de dimension $m$ associé à $\mathbf{A}$ et $\mathbf{r}^{(0)}$.
Une méthode itérative est polynomiale si et seulement si elle vérifie la condition de sous-espace

$$\mathbf{x}^{(m)} \in \mathbf{x}^{(0)} + \mathcal{K}_m(\mathbf{A}, \mathbf{r}^{(0)}), \tag{1.2}$$

ou, de façon équivalente,

$$\mathbf{x}^{(m)} \in \mathbf{x}^{(m-1)} + \mathcal{K}_m(\mathbf{A}, \mathbf{r}^{(0)}).$$

C'est une méthode de sous-espace liée aux espaces de Krylov.

La motivation essentielle de ce type de méthodes, comme rappelé dans [26] en particulier, est que, partant du polynôme caractéristique d'une matrice inversible $\mathbf{A}$ donnée, il est facile de vérifier qu'il existe un polynôme $\mathcal{P}(\mathbf{X})$ de degré au plus $n-1$ ($n$ étant la dimension de $\mathbf{A}$) tel que $\mathbf{A}^{-1} = \mathcal{P}(\mathbf{A})$. L'objectif des méthodes polynomiales est d'approcher d'une certaine manière ce polynôme $\mathcal{P}(\mathbf{X})$.

Pour les grands systèmes, on cherche à arrêter les itérations avant d'aboutir à la solution exacte. À l'étape $m$, pour déterminer l'itéré $\mathbf{x}^{(m)}$, on peut chercher à satisfaire une condition de minimisation de l'erreur pour un certain produit scalaire. Cela n'est pas toujours possible, et la recherche dans ce domaine a abouti à une condition plus générale que nous abordons maintenant (voir [26]) : une méthode de projection de Krylov est une méthode polynomiale définie par une matrice $\mathbf{B}$ d'ordre $n$ et deux conditions :

- la condition de sous-espace (1.2),
- la condition dite de Petrov-Galerkin :

$$(\mathbf{B}\,\mathbf{e}^{(m)})^T\mathbf{u} = 0, \quad \forall \mathbf{u} \in \mathcal{K}_m(\mathbf{A}, \mathbf{r}^{(0)}), \tag{1.3}$$

où $\mathbf{e}^{(m)} = \mathbf{x} - \mathbf{x}^{(m)}$ constitue l'erreur à l'étape $m$.

Le choix de $\mathbf{B}$ dicte la construction de la méthode de projection. Lorsque la matrice $\mathbf{B}$ définit un produit scalaire, on obtient la propriété de minimisation souhaitée, à savoir :

**Proposition 1.1.** *Si $\mathbf{B}$ est une matrice symétrique définie positive alors, pour $\mathbf{x}^{(m)}$ satisfaisant la condition de sous-espace(1.2), la condition de Petrov-Galerkin (1.3) est équivalente à minimiser l'erreur suivante :*

$$||\mathbf{e}^{(m)}||_{\mathbf{B}} = \min_{\mathbf{y} \in \mathcal{K}_m(\mathbf{A}, \mathbf{r}^{(0)})} ||\mathbf{e}^{(0)} - \mathbf{y}||_{\mathbf{B}}\,, \tag{1.4}$$

*sachant que $\mathbf{e}^{(m)} = \mathbf{x} - \mathbf{x}^{(m)} = \mathbf{x} - (\mathbf{x}^{(0)} + \mathbf{y}) = \mathbf{e}^{(0)} - \mathbf{y}$. Dans ce cas $\mathbf{x}^{(m)}$ est déterminé de manière unique.*

Pour la démonstration de cette proposition, on pourra se reporter à [26, § 6].

### 1.2.1 Méthode d'Arnoldi

Comme rappelé par Saad [75], la méthode d'Arnoldi est une méthode de projection orthogonale sur un sous-espace de Krylov, généralement appliquée aux matrices non symétriques. Cette méthode a été introduite par Arnoldi en 1950 [5], dans le but de factoriser une matrice sous forme Hessenberg supérieure [61]. Y. Saad a mis en évidence expérimentalement que cette

stratégie est utile pour l'approximation de valeurs propres de matrices de grande taille. Si l'algorithme arrive à la $m^{\text{ème}}$ itération, $m \ll n$, on obtient une matrice $\mathbf{H}_m$ de forme Hessenberg supérieure de taille $m \times m$, et une matrice orthonormale $\mathbf{V}_m$ de taille $n \times m$ dont les colonnes sont définies par des vecteurs $\mathbf{v}_1, \ldots, \mathbf{v}_m$. Ces vecteurs forment une base orthonormale du sous-espace de Krylov $\mathcal{K}_m(\mathbf{A}, \mathbf{v}_1) = [\mathbf{v}_1, \mathbf{A}\mathbf{v}_1, \ldots, \mathbf{A}^{m-1}\mathbf{v}_1]$.

**Définition 1.2 (matrice d'Hessenberg).** $\mathbf{H} = (h_{ij})_{1 \leq i,\ j \leq n}$ est une matrice sous forme Hessenberg supérieure (resp. inférieure) si la condition (1.5) (resp. (1.6)) est vérifiée.

$$\forall\ i > j+1,\ h_{ij} = 0 \tag{1.5}$$
$$\forall\ j > i+1,\ h_{ij} = 0 \tag{1.6}$$

La construction des vecteurs $\mathbf{v}_j$ orthonormés et de la matrice $\mathbf{H}_m$ est implémentée par l'Algorithme 1.3. Cet algorithme a comme paramètres :

- $\mathbf{v}_1$, un vecteur initial de norme 1 ;
- $\mathbf{A}$, la matrice du système aux valeurs propres considéré ;
- $m$, la taille de l'espace de Krylov.

Dans l'Algorithme 1.3 le calcul des vecteurs $\mathbf{v}_j$, $1 \leq j \leq m$, est effectué par des multiplications successives. Ensuite, chaque vecteur $\mathbf{v}_j$ est orthogonalisé par rapport à tous les vecteurs $\mathbf{v}_i$ déjà calculés ($i \in \{1, \ldots, j-1\}$), puis normé. L'orthogonalisation (étape 2.ii dans l'Algorithme 1.3) correspond à une procédure de Gram-Schmidt modifiée [34].

**Proposition 1.2.** *Dans l'*Algorithme 1.3 *d'Arnoldi, on a, à l'étape m, les relations suivantes :*

$$\mathbf{A}\mathbf{V}_m = \mathbf{V}_m\mathbf{H}_m + h_{m+1,m}\mathbf{v}_{m+1}\mathbf{e}_m^T \tag{1.7}$$
$$= \mathbf{V}_{m+1}\overline{\mathbf{H}}_m \tag{1.8}$$
$$\mathbf{V}_m^T\mathbf{A}\mathbf{V}_m = \mathbf{H}_m, \tag{1.9}$$

*où* $\mathbf{e}_m$ *est le* $m^{\text{ème}}$ *vecteur de la base canonique de* $\mathbb{R}^m$, $m = 1, \ldots, n$.

Pour la démonstration de cette proposition, on pourra se reporter à [75, § 6.3].

La matrice $\overline{\mathbf{H}}_m$ correspond à la matrice $\mathbf{H}_m$, définie ci-dessus, augmentée de la ligne $m+1$, le terme $h_{m+1,m}$ étant le $m^{\text{ème}}$ et seul élément non nul de cette ligne, si $m < n$. Si $m = n$, alors $\mathbf{A}\mathbf{V}_n = \mathbf{V}_n\mathbf{H}_n$ et $\mathbf{H}_n$ est unitairement semblable à $\mathbf{A}$.

**Définition 1.3.** Le polynôme minimal associé à la matrice $\mathbf{A}$ et au vecteur $\mathbf{v}_1$ est le polynôme de plus petit degré qui vérifie : $\mathcal{P}(\mathbf{A})\,\mathbf{v}_1 = 0$.

**Proposition 1.3.** $\mathbf{w}_j = 0$ *si et seulement si le polynôme minimal associé à* $\mathbf{A}$ *et à* $\mathbf{v}_1$ *est de degré* $j$.

Algorithme 1.3 : Procédure d'Arnoldi
Construction d'une base orthonormale pour l'espace de $\mathcal{K}_m(\mathbf{A}, \mathbf{v}_1)$ et de la matrice d'Hessenberg associée $\mathbf{H}_m$

**Début**

1. Choix de $\mathbf{v}_1$ tel que $||\mathbf{v}_1||_2 = 1$
2. **Pour** $j = 1, 2, \ldots, m$ **Faire :**
   i. $\mathbf{w}_j = \mathbf{A}\mathbf{v}_j$
   ii. **Pour** $i = 1, 2, \ldots, j$ **Faire :**
   $h_{i,j} = \mathbf{v}_i^T \mathbf{w}_j$
   $\mathbf{w}_j = \mathbf{w}_j - h_{i,j}\mathbf{v}_i$
   iii. **FinPour**
   iv. $h_{j+1,j} = ||\mathbf{w}_j||_2$
   v. **Si** $h_{j+1,j} = 0$ **Alors**
   Stop
   **Sinon**
   $\mathbf{v}_{j+1} = \mathbf{w}_j / h_{j+1,j}$
   vi. **FinSi**
3. **FinPour**

**Fin**

Pour la démonstration, on pourra se reporter à [75, § 6.3]. Une fois que la matrice $\mathbf{H}_m$ de forme de Hessenberg supérieure est construite, il est facile de calculer ses valeurs propres par l'algorithme QR, ce qui est peu coûteux tant que cette matrice est de taille beaucoup plus petite que celle de la matrice de $\mathbf{A}$.
Soit $\{\lambda_i^{(m)},\ i = 1, 2, \ldots, m\}$ les valeurs propres de $\mathbf{H}_m$ et $\{\mathbf{y}_i^{(m)},\ i = 1, 2, \ldots, m\}$ des vecteurs propres associés, $||\mathbf{y}_i^{(m)}||_2 = 1$.
On définit alors des vecteurs propres approchés pour $\mathbf{A}$ par : $\mathbf{u}_i^{(m)} = \mathbf{V}_m \mathbf{y}_i^{(m)}$.

**Proposition 1.4.** *Soit* $\mathbf{y}_i^{(m)}$, $||\mathbf{y}_i^{(m)}||_2 = 1$ *un vecteur propre de* $\mathbf{H}_m$ *associé à la valeur propre* $\lambda_i^{(m)}$, *et soit* $\mathbf{u}_i^{(m)}$ *le vecteur défini par :* $\mathbf{u}_i^{(m)} = \mathbf{V}_m \mathbf{y}_i^{(m)}$, *alors*

$$(\mathbf{A} - \lambda_i^{(m)}\mathbf{I})\mathbf{u}_i^{(m)} = h_{m+1,m}\, \mathbf{e}_m^T\, \mathbf{y}_i^{(m)}\, \mathbf{v}_{m+1} \tag{1.10}$$

$$||(\mathbf{A} - \lambda_i^{(m)}\mathbf{I})\mathbf{u}_i^{(m)}||_2 = h_{m+1,m}\, |\mathbf{e}_m^T\, \mathbf{y}_i^{(m)}|. \tag{1.11}$$

*Remarque* 1.1. La norme du résidu est égale à la dernière composante du vecteur propre $\mathbf{y}_i^{(m)}$ multipliée par $h_{m+1,m}$. On appelle ce terme le résidu d'Arnoldi.

*Remarque* 1.2. Si $h_{m+1,m} = 0$, le sous-espace $\mathcal{K}_m$ est invariant par $\mathbf{A}$.
On appelle ce phénomène "*lucky breakdown*" car les valeurs propres $\lambda_i^{(m)}$ de

$\mathbf{H}_m$, sont exactement $m$ valeurs propres de $\mathbf{A}$ et les vecteurs $\mathbf{u}_i^{(m)}$ forment un ensemble de $m$ vecteurs propres de $\mathbf{A}$.

### Méthode d'Arnoldi pour les systèmes linéaires

La méthode d'Arnoldi peut être utilisée pour la résolution de systèmes linéaires, en calculant une projection sur un sous-espace de Krylov $\mathcal{K}_m$ (voir § 1.2) qui consiste à calculer une solution approchée $\mathbf{x}^{(m)} = \mathbf{x}^{(0)} + \mathbf{z}^{(m)}$ du système (1.1), où $\mathbf{x}^{(0)}$ est un vecteur initial et $\mathbf{z}$ est un vecteur de l'espace $\mathcal{K}_m(\mathbf{A}, \mathbf{v}_1)$.

Si $\mathbf{v}_1 = \mathbf{r}^{(0)}/||\mathbf{r}^{(0)}||_2$ où $\mathbf{r}^{(0)} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(0)}$ et $\beta = ||\mathbf{r}^{(0)}||_2$, alors après $m$ itérations de l'Algorithme 1.3 on a une base orthonormale de $\mathcal{K}_m(\mathbf{A}, \mathbf{v}_1)$ formée à partir du vecteur initial $\mathbf{v}_1$, $\mathbf{V}_m = [\mathbf{v}_1, \ldots, \mathbf{v}_m]$, et une matrice d'Hessenberg supérieure $\mathbf{H}_m$ de taille $m \times m$, tels que $\mathbf{V}_m^T \mathbf{A} \mathbf{V}_m = \mathbf{H}_m$ (voir **Proposition** 1.2) et $\mathbf{V}_m^T \mathbf{r}^{(0)} = \mathbf{V}_m^T(\beta\, \mathbf{v}_1) = \beta\, \mathbf{e}_1$.

Algorithme 1.4 : Procédure d'Arnoldi Orthogonalisation Totale

**Début**

1. Choix d'un vecteur $\mathbf{x}^{(0)}$
2. Calcul de $\mathbf{r}^{(0)} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(0)}$
3. $\beta = ||\mathbf{r}^{(0)}||_2$ et $||\mathbf{v}_1||_2 = \mathbf{r}^{(0)}/\beta$
4. **Pour** $j = 1, 2, \ldots, m$ **Faire :**
   i. $\mathbf{w}_j = \mathbf{A}\mathbf{v}_j$
   ii. **Pour** $i = 1, 2, \ldots, j$ **Faire :**
   $h_{i,j} = \mathbf{v}_i^T \mathbf{w}_j$
   $\mathbf{w}_j = \mathbf{w}_j - h_{i,j}\mathbf{v}_i$
   iii. **FinPour**
   iv. $h_{j+1,j} = ||\mathbf{w}_j||_2$
   v. **Si** $h_{j+1,j} = 0$ **Alors**
   Stop
   **Sinon**
   $\mathbf{v}_{j+1} = \mathbf{w}_j / h_{j+1,j}$
   vi. **FinSi**
5. **FinPour**
6. **Résoudre** $\mathbf{H}_m \mathbf{y}^{(m)} = (\beta \mathbf{e}_1)$
7. $\mathbf{x}^{(m)} = \mathbf{x}^{(0)} + \mathbf{V}_m \mathbf{y}^{(m)}$

**Fin**

La correction $\mathbf{z}^{(m)}$ étant un vecteur de $\mathcal{K}_m(\mathbf{A}, \mathbf{v}_1)$ peut s'écrire comme $\mathbf{z}^{(m)} = \mathbf{V}_m\, \mathbf{y}^{(m)}$ où $\mathbf{y}^{(m)}$ est un vecteur de taille $m$. La solution approchée $\mathbf{x}^{(m)}$ est calculée comme la combinaison linéaire des vecteurs $\mathbf{v}_j$ (matrice

$\mathbf{V}_m$) selon la formule $\mathbf{x}^{(m)} = \mathbf{x}^{(0)} + \mathbf{V}_m\,\mathbf{y}^{(m)}$ avec $\mathbf{y}^{(m)}$ donné par $\mathbf{y}^{(m)} = \mathbf{H}_m^{-1}(\beta\,\mathbf{e}_1)$.

L'Algorithme 1.4 présente la méthode d'Arnoldi appliquée au système linéaire (1.1) en partant d'un vecteur initial quelconque $\mathbf{x}^{(0)}$ et qui calcule le vecteur solution $\mathbf{x}$ du système (1.1). La littérature spécialisée [34, 75] utilise la dénomination de Méthode d'Orthogonalisation Totale pour la méthode d'Arnoldi appliquée à la solution des systèmes linéaires.

### Considération d'implantation – Orthogonalisation des vecteurs

La procédure de Gram-Schmidt [34] utilisée pour orthogonaliser les vecteurs $\mathbf{v}_j$, $(j \in \{1, \ldots, m\})$ dans les Algorithmes 1.3 et 1.4 est une version modifiée de la procédure standard. Cette version modifiée présente une plus grande robustesse que la procédure de Gram-Schmidt standard. Cependant, elle présente quand même certains problèmes d'ordre numérique [38]. La méthode d'Arnoldi peut devenir plus robuste d'un point de vue numérique, lorsque l'algorithme de Householder [90] est employé. Cette procédure est basée sur une décomposition QR de la matrice $\mathbf{V}_i$ de dimension $n \times i$ composée de vecteurs colonnes $\mathbf{v}_j$ avec $j \in \{1, \ldots, i\}$. Ceci permet une orthogonalisation moins sensible à l'accumulation des erreurs d'arrondi, mais d'un coût légèrement supérieur. La description de l'algorithme de Householder et de Gram-Schmidt ainsi que leur utilisation dans la méthode d'Arnoldi sont décrites en détail dans [75, § 6.3].

### Considération d'implantation – Arnoldi avec Redémarrage

D'un point de vue pratique, la méthode d'Arnoldi présente de sérieuses limitations à cause de la taille $m$ de l'espace de Krylov $\mathcal{K}_m(\mathbf{A}, \mathbf{v}_1)$. La complexité de calcul due à la procédure de Gram-Schmidt est de l'ordre de $\mathcal{O}(m^2\,n)$ (orthogonalisation de $m$ vecteurs de taille $n$). L'utilisation mémoire est de l'ordre de $\mathcal{O}(m\,n)$ (stockage de $m$ vecteurs de taille $n$). L'option naturelle pour réduire la taille de l'espace de Krylov est d'exécuter des redémarrages périodiques de la procédure. Ceci revient à appliquer la méthode d'Arnoldi pour un nombre $m$ de pas, et utiliser ensuite la solution obtenue comme un vecteur initial d'un nouvel ensemble de $m$ pas.
L'Algorithme 1.5 montre la modification en conséquence de l'Algorithme 1.4. Dans l'Algorithme 1.5 on appelle la procédure d'Arnoldi comme décrit dans l'Algorithme 1.3.

Nous notons que cette notion de redémarrage a été introduite également pour le calcul de valeurs propres qui peut être effectuée, soit de manière explicite ou implicite. Le redémarrage explicite, a été introduit dans [41] pour le cas symétrique et développé par Saad (cf. [70, 71, 74]) dans le cas non symétrique. Cette approche est appelée "*Explicitly Restarted Arnoldi*

*Method*" (ERAM) dans laquelle le vecteur de redémarrage est construit en utilisant une combinaison linéaire entre les $k$ vecteurs de Ritz calculés ($\mathbf{v}_1 = \sum_{i=1}^{k} \alpha_i \mathbf{y}_i$). Le principe de cette stratégie est de faire converger le sous-espace de Krylov vers un sous-espace propre associé aux éléments propres recherchés. Cette stratégie essaie d'obtenir la convergence simultanée de tous ses éléments propres (par exemple quand les coefficients de la combinaison linéaire sont identiques $\alpha_i$ = Constante). Mais en pratique la convergence des vecteurs propres ne se fait pas à la même vitesse. Quant au redémarrage implicite, une autre variante a été proposée par Sorensen (cf. [81]), appelée "*Implicitly Restarted Arnoldi Method*" (IRAM). Cette technique combine la factorisation de Arnoldi avec le mécanisme de la méthode QR avec translation implicite [34, § 7.5].

Notons que dans le cas où $\mathbf{A}$ est une matrice symétrique, la procédure d'Arnoldi se simplifie en procédure de Lanczos symétrique (voir § 1.2.3) et que dans ce cas, la matrice d'Hessenberg $\mathbf{H}_m$ est tridiagonale symétrique notée $\mathbf{T}_m$.

Algorithme 1.5 : Procédure d'Arnoldi avec Redémarrage

**Début**

1. Choix d'un vecteur $\mathbf{x}^{(0)}$
2. Calcul de $\mathbf{r}^{(0)} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(0)}$
3. $\beta = ||\mathbf{r}^{(0)}||_2$ et $||\mathbf{v}_1||_2 = \mathbf{r}^{(0)}/\beta$
4. **Pour** $k = 1, 2, \ldots,$ **Faire :**
   i. Arnoldi (Algorithme 1.3)
   ii. **Résoudre** $\mathbf{y}^{(m)} = \mathbf{H}_m^{-1}(\beta\, \mathbf{e}_1)$
   iii. $\mathbf{x}^{(m)} = \mathbf{v}^{(m)} + \mathbf{V}_m \mathbf{y}^{(m)}$
   iv. **Si** Convergence **Alors**
   stop
   **Sinon**
   $\mathbf{x}^{(0)} = \mathbf{x}^{(m)}$
   calcul de $\mathbf{r}^{(0)} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(0)}$
   $\beta = ||\mathbf{r}^{(0)}||_2$ et $||\mathbf{v}_1||_2 = \mathbf{r}^{(0)}/\beta$
   v. **FinSi**
5. **FinPour**

**Fin**

### 1.2.2 Méthode GMRES

La méthode GMRES (Generalized Minimum Residual Method) [76] est aussi une méthode de projection sur un espace de Krylov. De façon iden-

tique à la méthode d'Arnoldi, un espace $\mathcal{K}_m(\mathbf{A}, \mathbf{v}_1)$ est engendré et une matrice $\mathbf{H}_m$ est calculée. Les mêmes considérations à propos des procédures d'orthogonalisation (Gram-Schmidt ou Householder) peuvent être faites. La seule différence entre la méthode d'Arnoldi (Algorithme 1.4) et la méthode de GMRES se situe dans le calcul de la solution approchée. La méthode GMRES consiste à calculer une solution approchée $\mathbf{x}^{(m)} = \mathbf{x}^{(0)} + \mathbf{z}^{(m)}$ du système (1.1), où $\mathbf{x}^{(0)}$ est un vecteur initial et $\mathbf{z}^{(m)}$ est un vecteur de l'espace $\mathcal{K}_m$. Après $m$ itérations de l'Algorithme 1.4, on a une base orthogonale de $\mathcal{K}_{m+1}$ formée à partir du vecteur initial $\mathbf{v}_1 = \mathbf{r}^{(0)}/||\mathbf{r}^{(0)}||_2$ où $\mathbf{r}^{(0)} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(0)}$, $\mathbf{V}_{m+1} = [\mathbf{v}_1, \ldots, \mathbf{v}_{m+1}]$, et une matrice Hessenberg supérieure $\overline{\mathbf{H}}_m$ de taille $(m+1) \times m$.
La correction $\mathbf{z}^{(m)}$ étant un vecteur de $\mathcal{K}_m$ peut s'écrire comme $\mathbf{z}^{(m)} = \mathbf{V}_m \mathbf{y}^{(m)}$, où $\mathbf{y}^{(m)}$ est un vecteur de taille $m$. On cherche donc une solution approchée sous la forme $\mathbf{x}^{(m)} = \mathbf{x}^{(0)} + \mathbf{V}_m \mathbf{y}^{(m)}$, de telle sorte que

$$\begin{aligned} \mathbf{b} - \mathbf{A}\mathbf{x}^{(m)} &= \mathbf{b} - \mathbf{A}(\mathbf{x}^{(0)} + \mathbf{V}\mathbf{y}^{(m)}) = \mathbf{r}^{(0)} - \mathbf{A}\mathbf{V}_m\mathbf{y}^{(m)} \\ &= \beta\,\mathbf{v}_1 - \mathbf{V}_{m+1}\overline{\mathbf{H}}_m\mathbf{y}^{(m)} = \mathbf{V}_{m+1}(\beta\,\mathbf{e}_1 - \overline{\mathbf{H}}_m\mathbf{y}^{(m)}). \end{aligned}$$

On définit alors la fonctionnelle suivante :

$$\mathbf{J}(\mathbf{y}) = ||\mathbf{b} - \mathbf{A}\mathbf{x}^{(m)}||_2 = ||\mathbf{b} - \mathbf{A}(\mathbf{x}^{(0)} + \mathbf{V}_m\mathbf{y})||_2.$$

Comme les colonnes de $\mathbf{V}_{m+1}$ sont orthonormales on a,

$$\mathbf{J}(\mathbf{y}) = ||\mathbf{V}_{m+1}(\beta\,\mathbf{e}_1 - \overline{\mathbf{H}}_m\mathbf{y})||_2 = ||\beta\,\mathbf{e}_1 - \overline{\mathbf{H}}_m\mathbf{y}||_2,$$

et la solution approchée $\mathbf{x}^{(m)} = \mathbf{x}^{(0)} + \mathbf{V}_m\mathbf{y}^{(m)}$ est choisie en minimisant le résidu $\mathbf{b} - \mathbf{A}\mathbf{x}^{(m)}$ sous la contrainte $\mathbf{z}^{(m)} \in \mathcal{K}_m$ : donc $\mathbf{x}^{(m)} = \mathbf{x}^{(0)} + \mathbf{V}_m\mathbf{y}^{(m)}$ où $\mathbf{y}^{(m)}$ est la solution au sens des moindres carrés de $\min_{\mathbf{y}} ||\beta\,\mathbf{e}_1 - \overline{\mathbf{H}}_m\mathbf{y}||_2$. Une solution pour ce problème est proposée dans [75, § 6.5], et consiste à effectuer une factorisation QR de la matrice $\overline{\mathbf{H}}_m$ à l'aide de rotations de Givens qui permettent d'annuler les termes sur la sous-diagonale de $\overline{\mathbf{H}}_m$ [34]. L'approximation fournie par GMRES est donc l'unique vecteur qui minimise $\mathbf{J}(\mathbf{y})$ et qu'on peut noter $\mathbf{y}^{(m)}$. La solution approchée $\mathbf{x}^{(m)}$ à l'itération $m$ s'écrit donc :

$$\mathbf{x}^{(m)} = \mathbf{x}^{(0)} + \mathbf{V}_m\mathbf{y}^{(m)} \text{ où } \mathbf{y}^{(m)} = \mathsf{arg}(\min_{\mathbf{y}} ||\beta\,\mathbf{e}_1 - \overline{\mathbf{H}}_m\mathbf{y}||_2).$$

L'Algorithme 1.6 présente l'implémentation de la méthode de GMRES de façon analogue à la méthode d'orthogonalisation totale (Algorithme 1.4). La méthode GMRES revient en fait à la méthode de projection de Krylov dans laquelle on considérerait $\mathbf{B} = \mathbf{A}^T\mathbf{A}$ comme définie en § 1.2. Elle est caractérisée par :

$$\left\{ \begin{array}{ll} \mathbf{x}^{(m)} & \in \mathbf{x}^{(0)} + \mathcal{K}_m(\mathbf{A}, \mathbf{r}^{(0)}) \\ \mathbf{r}^{(m)} & \perp \mathbf{A}\mathcal{K}_m(\mathbf{A}, \mathbf{r}^{(0)}). \end{array} \right. \tag{1.12}$$

**Algorithme 1.6 : Méthode de GMRES Totale**

**Début**

1. Choix d'un vecteur $\mathbf{x}^{(0)}$
2. Calcul de $\mathbf{r}^{(0)} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(0)}$
3. $\beta = ||\mathbf{r}^{(0)}||_2$ et $||\mathbf{v}_1||_2 = \mathbf{r}^{(0)}/\beta$
4. **Pour** $j = 1, 2, \ldots, m$ **Faire :**
   i. $\mathbf{w}_j = \mathbf{A}\mathbf{v}_j$
   ii. **Pour** $i = 1, 2, \ldots, j$ **Faire :**
   $h_{i,j} = \mathbf{v}_i^T \mathbf{w}_j$
   $\mathbf{w}_j = \mathbf{w}_j - h_{i,j}\mathbf{v}_i$
   iii. **FinPour**
   iv. $h_{j+1,j} = ||\mathbf{w}_j||_2$
   v. **Si** $h_{j+1,j} = 0$ **Alors**
   Stop
   **Sinon**
   $\mathbf{v}_{j+1} = \mathbf{w}_j / h_{j+1,j}$
   vi. **FinSi**
5. **FinPour**
6. $\mathbf{y}^{(m)} = \mathsf{argmin}_{\mathbf{y}} ||\beta\, \mathbf{e}_1 - \overline{\mathbf{H}}_m \mathbf{y}||_2$
7. $\mathbf{x}^{(m)} = \mathbf{x}^{(0)} + \mathbf{V}_m \mathbf{y}^{(m)}$

**Fin**

La **Proposition** 1.1 se traduit immédiatement par trouver $\mathbf{x}^{(m)} \in \mathbf{x}^{(0)} + \mathcal{K}_m(\mathbf{A}, \mathbf{r}^{(0)})$ minimisant $||\mathbf{r}^{(m)}||_2$, c'est-à-dire

$$||\mathbf{r}^{(m)}||_2 = \min_{\mathbf{x}^{(m)} \in \mathbf{x}^{(0)} + \mathcal{K}_m(\mathbf{A}, \mathbf{r}^{(0)})} ||\mathbf{b} - \mathbf{A}\mathbf{x}^{(m)}||_2. \tag{1.13}$$

### Convergence de GMRES

Nous avons vu la construction de la méthode GMRES. Nous rappelons maintenant ses propriétés de convergence (cf. [75], par exemple).

**Théorème 1.5.** *Si* $\mathbf{A}$ *est diagonalisable, soit* $\mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{-1}$, *où les colonnes de* $\mathbf{U}$ *forment une base de vecteurs propres et où* $\mathbf{\Lambda} = \mathsf{diag}\,(\lambda_1, \ldots, \lambda_n)$ *et soit* $\kappa(\mathbf{U}) = ||\mathbf{U}||_2 ||\mathbf{U}^{-1}||_2$ *le conditionnement de* $\mathbf{U}$. *Les itérations de* GMRES *vérifient*

$$||\mathbf{r}^{(m)}||_2 \leq ||\mathbf{r}^{(0)}||_2\ \kappa(\mathbf{U}) \min_{\substack{q \in \mathbb{P}_m \\ q(0)=1}} \max_{i=1,\ldots,n} |q(\lambda_i)|. \tag{1.14}$$

Pour la démonstration de ce théorème, on pourra se reporter à [26, § 6], [75, § 6.11].

**Redémarrage de GMRES**

À l'instar de la méthode d'Arnoldi (Algorithme 1.3), la méthode GMRES peut aussi être utilisée avec redémarrage. Le principe est toujours le même, c'est-à-dire, réduire les coûts de traitement et d'utilisation de mémoire en limitant la dimension maximale des espaces de Krylov générés. La méthode GMRES($m$) effectue des cycles de $m$ itérations de GMRES, en redémarrant avec la dernière approximation $\mathbf{x}^{(m)}$. Cela permet de limiter le stockage à $\mathcal{O}(m)$ vecteurs et de réduire le temps de calcul. Toutefois, le choix de $m$ est délicat.
La méthode GMRES Totale (Algorithme 1.6) converge vers une solution exacte en au plus $n$ ($n$ étant la taille du problème) itérations. La méthode GMRES avec redémarrage, au contraire, peut stagner (cf. [26, § 6.7], [91]), i.e., peut ne plus réduire la norme du résidu. Ce phénomène de stagnation est un désavantage apporté par l'utilisation des redémarrages. Cependant, des valeurs de $m$ proches de la limite théorique de la méthode GMRES Totale ($n$) ne sont pas applicables pour des cas pratiques car les besoins de temps de calcul et place mémoire sont alors prohibitifs.

### 1.2.3 Méthode de Lanczos

La méthode de Lanczos symétrique consiste à construire une base orthonormée de l'espace de Krylov ayant un vecteur $\mathbf{v}_1$ comme vecteur de départ. On profite de la symétrie de $\mathbf{A}$ pour obtenir une récurrence à trois termes [75, § 6.6].
Le procédé de Lanczos, pour construire une base de taille $m$, est résumé dans l'Algorithme 1.7.

$\mathbf{V}_m = [\mathbf{v}_1, \ldots, \mathbf{v}_m]$ est une base orthonormée de $\mathcal{K}_m$. De plus, on note $\mathbf{T}_m$ la matrice tridiagonale avec pour diagonale les $\alpha_j$, $1 \leq j \leq m$ et pour sur- et sous-diagonale les $\beta_j$, $1 \leq j \leq m$.

$$\mathbf{T}_m = \begin{pmatrix} \alpha_1 & \beta_2 & 0 & \ldots & 0 \\ \beta_2 & \alpha_2 & \beta_3 & \ddots & \vdots \\ 0 & \beta_3 & \alpha_3 & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \beta_m \\ \vdots & & \ddots & \beta_m & \alpha_m \end{pmatrix}, \ \beta_j > 0, \ j = 2, \ldots, m$$

On a alors

$$\mathbf{AV} = \mathbf{V}_m \mathbf{T}_m + \beta_{m+1} \mathbf{v}_{m+1} \mathbf{e}_m^T,$$

avec $\mathbf{e}_m$, $m^{ème}$ vecteur de la base canonique de $\mathbb{R}^m$. On en déduit, par multiplication avec $\mathbf{V}_m^T$ que : $\mathbf{T}_m = \mathbf{V}_m^T \mathbf{A} \mathbf{V}_m$.

**Algorithme 1.7 : Méthode de Lanczos**

**Début**

1. Choix d'un vecteur $\mathbf{v}_1$ tel que $||\mathbf{v}_1||_2 = 1$
2. $\beta_1 = 1$ et $\mathbf{v}_0 = 0$
3. **Pour** $j = 1, 2, \ldots, m$ **Faire :**
   - i. $\mathbf{w}_j = \mathbf{A}\mathbf{v}_j - \beta_j \mathbf{v}_{j-1}$
   - ii. $\alpha_j = (\mathbf{w}_j, \mathbf{v}_j)$
   - iii. $\mathbf{w}_j = \mathbf{w}_j - \alpha_j \mathbf{v}_j$
   - iv. $\beta_{j+1} = ||\mathbf{w}_j||_2$
   - v. **Si** $\beta_{j+1} = 0$ **Alors**
     Stop
     **Sinon**
     $\mathbf{v}_{j+1} = \mathbf{w}_j / \beta_{j+1}$
   - vi. **FinSi**
4. **FinPour**

**Fin**

Le principal inconvénient de cette version de Lanczos est son manque de stabilité numérique. En effet, la récurrence à trois termes, qui permet de limiter la quantité de calculs, en exploitant l'orthogonalité structurelle entre les vecteurs de Krylov générés $\mathbf{v}_j$, de par la symétrie de $\mathbf{A}$, fait que d'un point de vue numérique, l'orthogonalité entre ces vecteurs n'est pas maintenue de manière explicite au niveau algorithmique. L'effet des erreurs d'arrondi conduit également à une perte d'orthogonalité assez rapide entre les vecteurs $\mathbf{v}_j$. Ceci a déjà été observé et analysé par C. Paige dans [57, 58, 59]. Il est donc utile de réorthogonaliser le vecteur $\mathbf{v}^{(j)}$ par rapport à l'ensemble où une partie de directions précédemment calculées. Divers algorithmes ont été proposés pour réaliser cela, comme par exemple, Lanczos avec réorthogonalisation complète [56], Lanczos avec réorthogonalisation partielle [79], Lanczos avec réorthogonalisation sélective [65]. Une bonne présentation de l'ensemble de ces techniques ainsi que des motivations qui ont conduit à leur développement peut être consultée dans [34, § 9.2]. De plus pour arriver à une précision correcte, il est souvent nécessaire de choisir $m$ assez grand. Si l'on veut garder en mémoire des vecteurs $\mathbf{v}_j$ (pour les réorthogonaliser) on aboutit à un algorithme qui requiert $\mathcal{O}(n\,m)$ en mémoire et augmenter $m$ peut devenir limitatif à terme. Une possibilité pour répondre à ces deux difficultés est d'utiliser une méthode de redémarrage comme dans le cas d'Arnoldi en § 1.2.1.

### Méthode de Lanczos pour les systèmes linéaires

La méthode de Lanczos pour les systèmes linéaires (Algorithme 1.8) est identique à la méthode d'Arnoldi (Algorithme 1.4). La solution approchée $\mathbf{x}^{(m)}$ à l'itération $m$ est obtenue par une méthode de projection orthogonale sur l'espace de Krylov $\mathcal{K}_m$ :

$$\mathbf{x}^{(m)} = \mathbf{x}^{(0)} + \mathbf{V}_m \mathbf{y}^{(m)} \text{ où } \mathbf{y}^{(m)} = \mathbf{T}_m^{-1}(\beta\, \mathbf{e}_1).$$

Algorithme 1.8 : Méthode de Lanczos pour les systèmes linéaires

**Début**

1. Choix d'un vecteur $\mathbf{x}^{(0)}$ et $\mathbf{r}^{(0)} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(0)}$
2. $\beta = ||\mathbf{r}^{(0)}||_2$, $\mathbf{v}_1 = \mathbf{r}^{(0)}/\beta$ et $\beta_1 \mathbf{v}_0 = 0$
3. **Pour** $j = 1, 2, \ldots, m$ **Faire :**
   i. $\mathbf{w}_j = \mathbf{A}\mathbf{v}_j - \beta_j \mathbf{v}_{j-1}$
   ii. $\alpha_j = (\mathbf{w}_j, \mathbf{v}_j)$
   iii. $\mathbf{w}_j = \mathbf{w}_j - \alpha_j \mathbf{v}_j$
   iv. $\beta_{j+1} = ||\mathbf{w}_j||_2$
   v. **Si** $\beta_{j+1} = 0$ **Alors**
      Stop
      **Sinon**
      $\mathbf{v}_{j+1} = \mathbf{w}_j/\beta_{j+1}$
   vi. **FinSi**
4. **FinPour**
5. $\mathbf{T}_m = \mathsf{tridiag}(\beta_i, \alpha_i, \beta_{i+1})$ et $\mathbf{V}_m = [\mathbf{v}_1, \ldots, \mathbf{v}_m]$
6. **Résoudre** $\mathbf{T}_m \mathbf{y}^{(m)} = (\beta \mathbf{e}_1)$
7. $\mathbf{x}^{(m)} = \mathbf{x}^{(0)} + \mathbf{V}_m \mathbf{y}^{(m)}$
8. **FinPour**

**Fin**

### 1.2.4 Méthode du Gradient Conjugué

La méthode du Gradient Conjugué (CG) correspond à la méthode de projection de Krylov que l'on obtient en considérant $\mathbf{B} = \mathbf{A}$ comme définie en § 1.2 et $\mathbf{A}$ est symétrique définie positive (SPD).

$$\begin{cases} \mathbf{x}^{(m)} & \in \mathbf{x}^{(0)} + \mathcal{K}_m(\mathbf{A}, \mathbf{r}^{(0)}) \\ \mathbf{r}^{(m)} & \perp \mathcal{K}_m(\mathbf{A}, \mathbf{r}^{(0)}). \end{cases} \tag{1.15}$$

La **Proposition** 1.1 se traduit immédiatement par : trouver $\mathbf{x}^{(m)} \in \mathbf{x}^{(0)} + \mathcal{K}_m(\mathbf{A}, \mathbf{r}^{(0)})$ minimisant $||\mathbf{x} - \mathbf{x}^{(m)}||_{\mathbf{A}}$.

Algorithme 1.9 : Gradient Conjugué (CG)

**Début**

1. Choix de $\mathbf{x}^{(0)}$, $\mathbf{r}^{(0)} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(0)}$ et $\mathbf{p}^{(0)} = \mathbf{r}^{(0)}$
2. **Pour** $k = 0, 1, \ldots, m$ **Faire**
   i. $\mathbf{q}^{(k)} = \mathbf{A}\mathbf{p}^{(k)}$
   ii. $\alpha_k = (\mathbf{r}^{(k)}, \mathbf{r}^{(k)})/(\mathbf{q}^{(k)}, \mathbf{p}^{(k)})$
   iii. $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{p}^{(k)}$
   iv. $\mathbf{r}^{(k+1)} = \mathbf{r}^{(k)} - \alpha_k \mathbf{q}^{(k)}$
   v. $\beta_{k+1} = (\mathbf{r}^{(k+1)}, \mathbf{r}^{(k+1)})/(\mathbf{r}^{(k)}, \mathbf{r}^{(k)})$
   vi. $\mathbf{p}^{(k+1)} = \mathbf{r}^{(k+1)} + \beta_{k+1} \mathbf{p}^{(k)}$
3. **FinPour**

**Fin**

Du fait des propriétés de $\mathbf{A}$, des relations de conjugaison (orthogonalité) apparaissent, conduisant à l'Algorithme 1.9 qui repose sur la construction de deux bases différentes de $\mathcal{K}_m(\mathbf{A}, \mathbf{r}^{(0)})$ :

- La base des résidus $\mathbf{R} = (\mathbf{r}^{(0)}, \ldots, \mathbf{r}^{(m)})$, qui est orthogonale,
- La base des directions de descente $\mathbf{P} = (\mathbf{p}^{(0)}, \ldots, \mathbf{p}^{(m)})$, qui est $\mathbf{A}$-orthogonale.

L'étape 2.v–2.viii de l'algorithme correspond ainsi à la $\mathbf{A}$-orthogonalisation de $\mathbf{p}^{(m+1)}$ vis-à-vis de $\mathbf{p}^{(m)}$ ce qui théoriquement implique l'orthogonalité de $\mathbf{p}^{(m+1)}$ vis-à-vis de toutes les directions de descente précédentes. Cependant, du point de vue numérique, cette conjugaison a tendance être perdue au fur et à mesure des itérations. Il convient alors de mettre en place une orthogonalisation complète (voire sélective) des directions de descente. Différentes mises en oeuvre sont d'ailleurs possibles (entre autre Gram-Schmidt, Gram-Schmidt modifié [34]) permettant d'obtenir des précisions plus ou moins élevées.

### Optimalité du Gradient Conjugué

Le théorème suivant montre que la solution produite par l'algorithme du gradient conjugué est optimale pour la norme $\mathbf{A}$, puis donne une borne supérieure grossière de convergence, en fonction du conditionnement de la matrice $\mathbf{A}$.

**Théorème 1.6.** *Soit* $\mathbf{A}$ *symétrique définie positive. L'erreur* $\mathbf{e}^{(m)} = \mathbf{x} - \mathbf{x}^{(m)}$

*dans l'*Algorithme 1.9 (CG) *vérifie les relations suivantes :*

$$\begin{aligned}
||\mathbf{e}^{(m)}||_{\mathbf{A}} &= \min_{\mathbf{z}\in\mathbf{x}^{(0)}+\mathcal{K}_m(\mathbf{A},\mathbf{r}^{(0)})} ||\mathbf{z}-\mathbf{x}||_{\mathbf{A}} \\
||\mathbf{e}^{(m)}||_{\mathbf{A}} &= \min_{q\in\mathbb{P}_{m-1}} \left|\left|\Big(\mathbf{I}-\mathbf{A}q(\mathbf{A})\Big)\mathbf{e}^{(0)}\right|\right|_{\mathbf{A}} \\
||\mathbf{e}^{(m)}||_{\mathbf{A}} &= \min_{\substack{q\in\mathbb{P}_m \\ q(0)=1}} ||q(\mathbf{A})\mathbf{e}^{(0)}||_{\mathbf{A}} \\
||\mathbf{e}^{(m)}||_{\mathbf{A}} &\leq 2\,||\mathbf{e}^{(0)}||_{\mathbf{A}} \left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^m,
\end{aligned}$$

$\kappa = \dfrac{\lambda_{\max}}{\lambda_{\min}}$ *étant le conditionnement de* $\mathbf{A}$ *et* $\mathbb{P}_m$ *étant l'ensemble des polynômes de degré* $\leq m$.

Pour la démonstration de ce théorème, on pourra se reporter à [16, 26, 34, 46, 75].

## 1.3 Étude de la convergence, préconditionnement

Les performances des solveurs de Krylov sont fortement liées au spectre de la matrice du système et plus précisément au spectre actif du système (ensemble des valeurs propres pour lesquelles le second membre a une projection non nulle sur le vecteur propre associé). On peut notamment remplacer le conditionnement $\kappa$ par le conditionnement actif $\kappa_{\text{act}}$ dans la dernière relation du Théorème. 1.6 obtenant ainsi un meilleur taux de convergence.

À partir de ces simples considérations, l'intérêt du préconditionnement doit sembler évident : l'idée est de résoudre le système équivalent $\mathbf{MAx} = \mathbf{Mb}$ avec $\mathbf{M}$ une matrice bien choisie qui confère au nouveau système de meilleures propriétés spectrales (notamment si $\mathbf{M} \approx \mathbf{A}^{-1}$ alors le conditionnement est excellent).

Dans le cadre du Gradient Conjugué, l'emploi d'un préconditionneur peut sembler problématique puisque la symétrie du problème est a priori rompue.

Soit $\mathbf{M}$ une matrice symétrique définie positive, approchant $\mathbf{A}$, on peut, par exemple, résoudre un des problèmes suivants :

$$\mathbf{M}^{-1}\mathbf{Ax} = \mathbf{M}^{-1}\mathbf{b}, \tag{1.16}$$

ou

$$\begin{cases} \mathbf{AM}^{-1}\mathbf{y} = \mathbf{b}, \\ \mathbf{x} = \mathbf{M}^{-1}\mathbf{y}. \end{cases} \tag{1.17}$$

Algorithme 1.10 : Gradient Conjugué Préconditionné (PCG)

**Début**

1. Choix de $\mathbf{x}^{(0)}$ et $\mathbf{r}^{(0)} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(0)}$
2. $\mathbf{z}^{(0)} = \mathbf{M}^{-1}\mathbf{r}^{(0)}$ et $\mathbf{p}^{(0)} = \mathbf{z}^{(0)}$
3. **Pour** $k = 0, 1, \ldots, m$ **Faire**
   - i. $\mathbf{q}^{(k)} = \mathbf{A}\mathbf{p}^{(k)}$
   - ii. $\alpha_k = (\mathbf{r}^{(k)}, \mathbf{z}^{(k)})/(\mathbf{q}^{(k)}, \mathbf{p}^{(k)})$
   - iii. $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{p}^{(k)}$
   - ii. $\mathbf{r}^{(k+1)} = \mathbf{r}^{(k)} - \alpha_k \mathbf{q}^{(k)}$
   - iv. $\mathbf{z}^{(k)} = \mathbf{M}^{-1}\mathbf{r}^{(k+1)}$
   - v. $\beta_{k+1} = (\mathbf{r}^{(k+1)}, \mathbf{z}^{(k+1)})/(\mathbf{r}^{(k)}, \mathbf{z}^{(k)})$
   - vi. $\mathbf{p}^{(k+1)} = \mathbf{r}^{(k+1)} + \beta_{k+1}\mathbf{p}^{(k)}$
4. **FinPour**

**Fin**

Malheureusement dans ce cas $\mathbf{M}^{-1}\mathbf{A}$ et $\mathbf{A}\mathbf{M}^{-1}$ ne sont plus symétriques. On peut préserver cette symétrie, par exemple, si $\mathbf{M}$ est disponible sous forme factorisée $\mathbf{M} = \mathbf{L}\mathbf{L}^T$, ce qui est possible par exemple dès que $\mathbf{M}$ est symétrique définie positive. On sépare alors la matrice de préconditionnement et on résout

$$\left\{ \begin{array}{rcl} \mathbf{L}^{-1}\mathbf{A}\mathbf{L}^{-T}\mathbf{y} = & \mathbf{L}^{-T}\mathbf{b}, \\ \mathbf{x} = & \mathbf{L}^{-T}\mathbf{y}. \end{array} \right. \tag{1.18}$$

qui est, lui, un système symétrique. Cela n'est pas cependant nécessaire car $\mathbf{M}^{-1}\mathbf{A}$ est auto-adjoint pour le produit scalaire $\mathbf{M}$. En effet

$$(\mathbf{M}^{-1}\mathbf{A}\mathbf{x}, \mathbf{y})_{\mathbf{M}} = (\mathbf{A}\mathbf{x}, \mathbf{y}) = (\mathbf{x}, \mathbf{M}^{-1}\mathbf{A}\mathbf{y})_{\mathbf{M}}.$$

Et donc, il suffit de résoudre (1.16) avec le produit scalaire $\mathbf{M}$ dans l'algorithme du Gradient Conjugué. Il n'est pas nécessaire non plus de calculer les produits scalaires avec $\mathbf{M}$ car avec $\mathbf{z}^{(m)} = \mathbf{M}^{-1}\mathbf{r}^{(m)}$,

$$\begin{array}{rcl} (\mathbf{z}^{(m)}, \mathbf{z}^{(m)})_{\mathbf{M}} & = & (\mathbf{z}^{(m)}, \mathbf{r}^{(m)}) \\ (\mathbf{M}^{-1}\mathbf{A}\mathbf{p}^{(m)}, \mathbf{p}^{(m)})_{\mathbf{M}} & = & (\mathbf{A}\mathbf{p}^{(m)}, \mathbf{p}^{(m)}) \end{array}$$

Si on injecte ces relations dans l'Algorithme 1.9 du gradient conjugué, on obtient l'Algorithme 1.10. Il est appelé algorithme du Gradient Conjugué préconditionné. Son avantage sur la version non préconditionnée est que la vitesse de convergence dépend maintenant du conditionnement de la matrice $\mathbf{M}^{-1}\mathbf{A}$.

Sa mise en oeuvre nécessite un vecteur intermédiaire supplémentaire et une résolution de système : $\mathbf{z}^{(m)} = \mathbf{M}^{-1}\mathbf{r}^{(m)}$. Il faut donc choisir $\mathbf{M}$ de telle

façon que ce surcoût ne soit pas trop important par rapport à la réduction effective du nombre d'itérations, et pour que le gain soit important, on choisit $\mathbf{M}$ telle que $\mathbf{M}^{-1}\mathbf{A}$ a de meilleures propriétés spectrales.

## 1.4 Conclusion

Dans ce chapitre, nous avons présenté quelques méthodes itératives, asymptotiques ou de type Krylov pour la résolution des problèmes de l'algèbre linéaire creux et de grande taille. Pour le calcul des valeurs propres, nous avons introduit la méthode de la puissance comme exemple de méthode asymptotique, la méthode d'Arnoldi comme exemple de méthode à terminaison finie dans le cas non symétrique et la méthode de Lanczos dans le cas symétrique. Quant à la résolution de systèmes linéaires, nous n'avons considéré que les méthodes de type Krylov. Dans le chapitre suivant nous introduisons une nouvelle technique combinant les filtres polynomiaux de Tchebycheff et le processus de Lanczos pour calculer avec différents niveaux de filtrage $\varepsilon$, une base orthogonale $\mathbf{W}(\varepsilon)$ du sous-espace invariant associé aux plus petites valeurs propres du problème à résoudre. Cette base sera incorporée au sein des méthodes exploitant une certaine information spectrale du système à résoudre.

# Chapitre 2

# Factorisation Spectrale Partielle

Récemment, un algorithme de factorisation spectrale partielle (dénommé Tchebycheff-PSF) combinant les filtres polynomiaux de Tchebycheff et le processus de Lanczos a été proposé par Arioli et Ruiz [4]. Une partie de notre travail (cf. [28, 29]), en relation avec cette méthode, a consisté à l'incorporer au sein de méthodes de résolution exploitant l'information spectrale du système à résoudre, et à évaluer son potentiel et ses limitations. L'objectif de ce chapitre est d'introduire les filtres polynomiaux de Tchebycheff et leur utilisation dans la construction de l'algorithme de factorisation spectrale partielle Tchebycheff-PSF. Nous décrirons également l'influence de certains paramètres sur le comportement de cette technique. L'exploitation concrète de cette approche combinée à diverses méthodes de résolution sera étudiée et validée expérimentalement au chapitre suivant.

## 2.1 Introduction

Dans l'ensemble de ce chapitre, le système linéaire considéré (1.1) est symétrique défini positif (SPD). Pour résoudre ce type de système, une des méthodes itératives de choix est le gradient conjugué [37]. Comme précédemment décrit, au Chapitre 1, le but de cette méthode est de calculer une approximation de la solution $\mathbf{x}^{(m)} \in \mathbf{x}^{(0)} + \mathcal{K}_m(\mathbf{A}, \mathbf{r}^{(0)})$, minimisant la quantité suivante (appelée erreur) $||\mathbf{x} - \mathbf{x}^{(m)}||_{\mathbf{A}}$, et où $\mathbf{r}^{(0)} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(0)}$ est le résidu initial. Comme $\mathbf{A}$ est SPD et les espaces de Krylov sont emboîtés et de dimension croissante, l'erreur ne peut que décroître au sens large au fur et à mesure de la construction de ces espaces.

Cependant, la convergence de la méthode de gradient conjugué est souvent pénalisée dès que le système linéaire à résoudre possède des valeurs

propres au voisinage de zéro. Plusieurs techniques ont été proposées dans la littérature pour atténuer l'effet néfaste de ces plus petites valeurs propres, en mettant par exemple à jour le préconditionneur [10], ou bien encore en contraignant la méthode du gradient conjugué à travailler dans le complémentaire orthogonal du sous-espace invariant associé aux plus petites valeurs propres [51, 77, 88]. Ces diverses approches exploitent la connaissance explicite d'un sous-espace invariant contenant les directions qui nuisent à la convergence. Typiquement, cet espace contient les directions propres associées aux plus petites valeurs propres qui, comme montré dans [85], sont les directions qui gênent le plus le gradient conjugué dans sa convergence, car elles induisent des paliers dans la décroissance monotone de l'erreur $||\mathbf{x} - \mathbf{x}^{(m)}||_{\mathbf{A}}$.

Dans ce chapitre, nous introduisons une technique basée sur les filtres polynomiaux de Tchebycheff combinés avec le processus de Lanczos ou de Lanczos par blocs (voir [33, 54, 84] pour plus de détails) permettant de calculer une base orthogonale du sous-espace invariant associé aux plus petites valeurs propres de $\mathbf{A}$.

Cette technique démarre par la génération aléatoire d'un ensemble de $s$ vecteurs ($s$ est appelé la taille du bloc) orthonormalisé. Des polynômes matriciels de Tchebycheff en $\mathbf{A}$ sont alors appliqués à l'ensemble de ces vecteurs de départ pour atténuer les composantes propres associées à toutes les valeurs propres dans un certain intervalle prédéterminé. On peut, par exemple, fixer un nombre positif $\mu \in ]0, \lambda_{\max}(\mathbf{A})[$, et décider de calculer tous les vecteurs propres associés aux valeurs propres dans l'intervalle $[\lambda_{\min}(\mathbf{A}), \mu]$. Ce paramètre $\mu$ sera appelé valeur de coupure, et servira explicitement dans la construction des polynômes de Tchebycheff. Le calcul de la plus grande valeur propre de $\mathbf{A}$, $\lambda_{\max}(\mathbf{A})$, n'est en général pas trop difficile, et dans certains cas, une borne supérieure très proche de $\lambda_{\max}(\mathbf{A})$ peut même être directement déduite grâce à quelques propriétés numériques de $\mathbf{A}$. Si les valeurs propres de la matrice $\mathbf{A}$ sont très bien regroupées, on peut espérer que le nombre de valeurs propres restantes dans l'intervalle $[\lambda_{\min}(\mathbf{A}), \mu]$, pour une valeur de coupure $\mu$ raisonnable, sera petit. Pour en revenir à l'algorithme lui même, après avoir filtré l'ensemble des $s$ vecteurs de départ, nous obtenons un ensemble de $s$ vecteurs dont les composantes propres sont proches de 0 pour toute valeur propre dans l'intervalle $[\mu, \lambda_{\max}(\mathbf{A})]$, à la suite de quoi il est possible d'utiliser la technique de Lanczos/Lanczos par blocs pour construire une approximation de l'espace invariant associé à toutes les valeurs propres dans l'intervalle complémentaire $[\lambda_{\min}(\mathbf{A}), \mu]$.

Dans ce qui suit, après avoir introduit les filtres polynomiaux de Tchebycheff, les étapes de la factorisation spectrale partielle seront détaillées. Nous décrirons enfin l'influence de certains paramètres sur le comportement de la méthode.

## 2.2 Filtrage Basé sur les Polynômes de Tchebycheff

Soit

$$\mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T = \mathbf{U}_1\mathbf{\Lambda}_1\mathbf{U}_1^T + \mathbf{U}_2\mathbf{\Lambda}_2\mathbf{U}_2^T,$$

la décomposition propre de la matrice $\mathbf{A}$, $\mathbf{\Lambda}_1$ étant la matrice diagonale formée des valeurs propres $\lambda_i$ de $\mathbf{A}$ inférieures au paramètre $\mu \in ]0, \lambda_{\max}[$, $\mathbf{U}_1$ la matrice rectangulaire dont les colonnes correspondent aux vecteurs propres orthonormalisés associés à ces valeurs propres, $\mathbf{\Lambda}_2$ et $\mathbf{U}_2$ étant les matrices complémentaires correspondantes.
Soit $\mathbf{P}$ la matrice de $s$ vecteurs orthonormalisés générés aléatoirement et $\mathcal{F}_m$ un polynôme de degré $m$. Sur la base de cette décomposition, on peut alors écrire

$$\mathcal{F}_m(\mathbf{A})\,\mathbf{P} = \mathbf{U}_1\mathcal{F}_m(\mathbf{\Lambda}_1)\mathbf{U}_1^T\,\mathbf{P} + \mathbf{U}_2\mathcal{F}_m(\mathbf{\Lambda}_2)\mathbf{U}_2^T\,\mathbf{P}. \tag{2.1}$$

Le but est de choisir un polynôme $\mathcal{F}_m$ permettant de rendre $\mathcal{F}_m(\mathbf{\Lambda}_2)$ très proche de zéro de manière contrôlée, de telle sorte que $\mathcal{F}_m(\mathbf{A})\,\mathbf{P}$ soit très colinéaire à $\mathbf{U}_1$.

Une famille de polynômes qui remplit cette contrainte est la famille des polynômes de Tchebycheff définis par la relation de récurrence suivante (voir [36, page 46]) :

$$\begin{cases} T_0(\omega) = 1\,, \quad T_1(\omega) = \omega, \\ T_{m+1}(\omega) = 2\,\omega\,T_m(\omega) - T_{m-1}(\omega) \qquad m \geq 1. \end{cases} \tag{2.2}$$

Les propriétés optimales des polynômes de Tchebycheff (cf. Théorème 4.2.1 dans [36, page 47]) peuvent être récapitulées comme suit :
Soit $|d| > 1$, et $Q \in \mathbb{P}_m$ l'ensemble des polynômes de degré $\leq m$, tel que $Q(d) = 1$, alors le polynôme $H_m(\omega) = \dfrac{T_m(\omega)}{T_m(d)}$ vérifie :

$$\begin{cases} \max\limits_{\omega\in[-1,1]} |H_m(\omega)| \leq \max\limits_{\omega\in[-1,1]} |Q(\omega)|, \quad \forall Q \in \mathbb{P}_m, \\ \max\limits_{\omega\in[-1,1]} |H_m(\omega)| = \dfrac{1}{|T_m(d)|}. \end{cases} \tag{2.3}$$

Soit $\omega_\mu$ la similitude directe définie par :

$$\lambda \in \mathbb{R} \longmapsto \omega_\mu(\lambda) = \frac{\lambda_{\max}(\mathbf{A}) + \mu - \mathbf{2}\lambda}{\lambda_{\max}(\mathbf{A}) - \mu},$$

où $\mu \in ]\lambda_{\min}(\mathbf{A}), \lambda_{\max}(\mathbf{A})[$ et $\lambda_{\min}(\mathbf{A})$ (resp. $\lambda_{\max}(\mathbf{A})$) est la plus petite valeur propre de $\mathbf{A}$ (resp. la plus grande valeur propre de $\mathbf{A}$).
Cette similitude transforme l'intervalle $[\mu, \lambda_{\max}(\mathbf{A})]$ en $[-1, 1]$, avec

$$\begin{cases} d_\mu = \omega_\mu(0) & = & \frac{\lambda_{\max}(\mathbf{A})+\mu}{\lambda_{\max}(\mathbf{A})-\mu} > 1 \\ \omega_\mu(\lambda_{\max}(\mathbf{A})) & = & -1 \\ \omega_\mu(\mu) & = & 1. \end{cases}$$

Soit alors

$$\mathcal{F}_m(\lambda) = \frac{T_m(\omega_\mu(\lambda))}{T_m(d_\mu)}, \tag{2.4}$$

notre filtre polynomial. D'après l'équation (2.3), $\mathcal{F}_m$ vérifie donc

$$\max_{\lambda\in[\mu,\lambda_{\max}(\mathbf{A})]} |\mathcal{F}_m(\lambda)| \leq \max_{\substack{\lambda\in[\mu,\lambda_{\max}(\mathbf{A})]\\ Q\in\mathbb{P}_m\\ Q(0)=1}} |Q(\lambda)|. \tag{2.5}$$

L'exemple de la figure 2.1 montre les valeurs du polynôme de Tchebycheff $\mathcal{F}_{16}(\lambda)$, $\lambda \in ]0,1]$, comme défini dans (2.4), avec $\lambda_{\min} = 10^{-16}$, $\lambda_{\max} = 1$, et $\mu = 10^{-1}$. Sur le bas de la figure 2.1, sont représentées les valeurs du polynôme $\mathcal{F}_{16}(\lambda)$ sur l'intervalle $[\mu, \lambda_{\max}]$. Cet exemple illustre comment utiliser les polynômes de Tchebycheff comme un filtre spectral pour atténuer uniformément, au dessous d'une valeur $\varepsilon = 10^{-4}$ relativement aux autres, les composantes propres associées à toutes les valeurs propres dans l'intervalle $[\mu, \lambda_{\max}]$. Sur le haut de la figure 2.1, on peut remarquer que les valeurs du polynôme $\mathcal{F}_{16}(\lambda)$ sur l'intervalle $[\lambda_{\min}, \mu]$ restent proches de 1 (surtout pour des valeurs $\lambda \in [\lambda_{\min}, 10^{-2}]$). Ceci est dû à la continuité des polynômes de Tchebycheff et au fait que $\mathcal{F}_{16}(0) = 1$.

Nous introduisons alors la notation algorithmique

$$\mathbf{Z} = \text{Tchebycheff–Filter}(\mathbf{P}, \varepsilon, [\mu, \lambda_{\max}], \mathbf{A})$$

pour représenter l'application du polynôme matriciel de Tchebycheff en $\mathbf{A}$ à l'ensemble de vecteurs $\mathbf{P}$,

$$\mathbf{Z} = \mathcal{F}_m(\mathbf{A})\,\mathbf{P},$$

où $\mathcal{F}_m$ est le filtre polynomial défini par l'équation (2.4). Le degré de $\mathcal{F}_m$ est fixé de façon à ce que $||\mathcal{F}_m||_\infty$ soit inférieure à $\varepsilon$ sur l'intervalle $[\mu, \lambda_{\max}]$, où $\varepsilon \ll 1$ est le niveau de filtrage choisi *a priori*. En effet, pour des valeurs données de $\mu$, $\lambda_{\max}(\mathbf{A})$ et $\varepsilon$, on peut fixer le degré $m$ de $T_m$ tel que $1/|T_m(d_\mu)| < \varepsilon$ sur $[\mu, \lambda_{\max}(\mathbf{A})]$, ce qui implique que$||\mathcal{F}_m(\mathbf{\Lambda}_2)||_\infty < \varepsilon$, et en utilisant l'équation (4.1) on peut alors écrire :

$$||\mathbf{U}_2^T\,\mathbf{Z}||_2 \leq ||\mathcal{F}_m(\mathbf{\Lambda}_2)||_2\,||\mathbf{U}_2^T\,\mathbf{P}||_2 \leq \varepsilon\,||\mathbf{U}_2^T\,\mathbf{P}||_2. \tag{2.6}$$

L'équation (2.6) montre explicitement comment le filtre matriciel de Tchebycheff en $\mathbf{A}$ appliqué à un ensemble de vecteurs pourra atténuer, dans ces vecteurs, les composantes propres associées à toutes les valeurs propres de l'intervalle $[\mu, \lambda_{\max}(\mathbf{A})]$ en dessous d'un niveau $\varepsilon$ relativement aux autres.

Le degré du polynôme de Tchebycheff permettant d'atténuer les composantes propres d'un facteur de filtrage $\varepsilon$, est directement lié au taux de convergence de l'itération de Tchebycheff sur l'intervalle $[\mu, \lambda_{\max}]$ (voir,

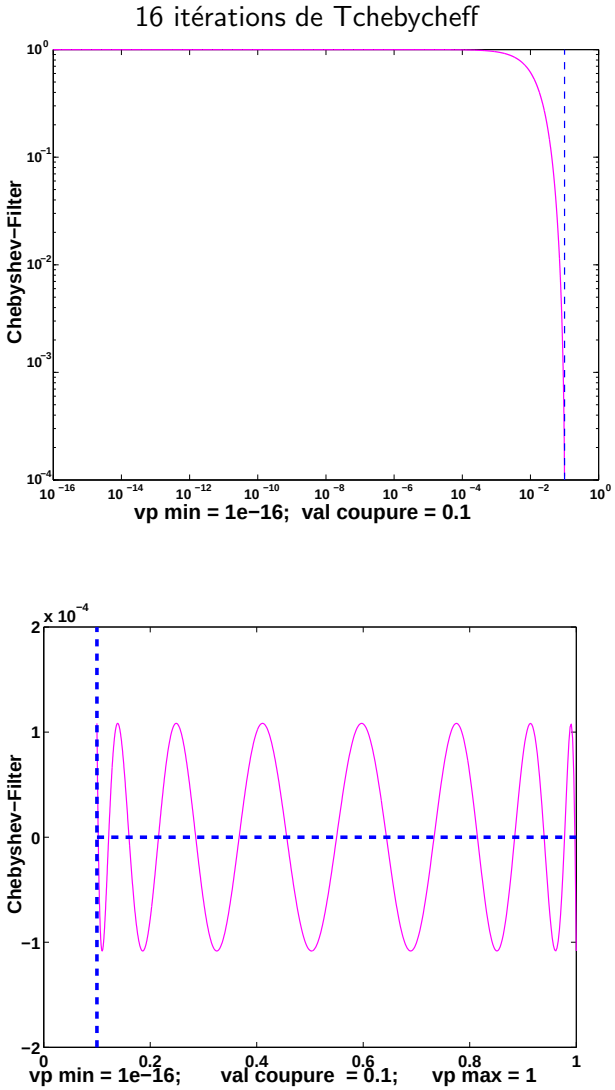



FIG. 2.1 – *Représentation des valeurs du filtre polynomial de Tchebycheff* $\mathcal{F}_{16}$ *dans les deux intervalles* $[10^{-16}, \mu]$ *et* $[\mu, 1]$*, avec* $\mu = 10^{-1}$.

e.g., [36, 30]), qui dépend seulement du rapport $\lambda_{\max}/\mu$. Comme illustré dans la figure 2.1, 16 itérations de Tchebycheff suffisent pour atteindre un niveau de filtrage $\varepsilon = 10^{-4}$ sur l'intervalle $[\mu, \lambda_{\max}]$ où $\lambda_{\max}/\mu = 10$. Si on prend par exemple $\lambda_{\max}/\mu = 100$ comme valeur de coupure, le nombre d'itérations de Tchebycheff pour atteindre le même niveau de filtrage devient alors égal à 50.

Le filtre polynomial de Tchebycheff est résumé dans l'Algorithme 2.1.

Algorithme 2.1 : Filtre Polynomial de Tchebycheff

**Début**

1. $\alpha_\mu = \dfrac{2}{\lambda_{\max}(\mathbf{A}) - \mu}$ et $d_\mu = \dfrac{\lambda_{\max}(\mathbf{A}) + \mu}{\lambda_{\max}(\mathbf{A}) - \mu}$
2. $\sigma_0 = 1$ et $\sigma_1 = d_\mu$
3. Choix d'un vecteur $\mathbf{x}^{(0)}$ et calcul de $\mathbf{r}^{(0)} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(0)}$
4. $\mathbf{x}^{(1)} = \mathbf{x}^{(0)} + \dfrac{\alpha_\mu}{d_\mu}\,\mathbf{r}^{(0)}$ et $\mathbf{r}^{(1)} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(1)}$
5. **Pour** $m = 1, 2, \ldots,$ jusqu'à $1/\sigma_m < \varepsilon$ **Faire :**
   i. $\sigma_{m+1} = 2\, d_\mu\, \sigma_m - \sigma_{m-1}$
   ii. $\mathbf{x}^{(m+1)} = 2\,\dfrac{\sigma_m}{\sigma_{m+1}}\Big(d_\mu \mathbf{x}^{(m)} + \alpha_\mu \mathbf{r}^{(m)}\Big) - \dfrac{\sigma_{m-1}}{\sigma_{m+1}}\,\mathbf{x}^{(m-1)}$
   iii. $\mathbf{r}^{(m+1)} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(m+1)}$
6. **FinPour**

**Fin**

Cet algorithme correspond à l'application du polynôme matriciel de Tchebycheff en $\mathbf{A}$ multipliant les composantes propres de $\mathbf{r}^{(0)}$ associées à toutes les valeurs propres dans l'intervalle $[\mu, \lambda_{\max}(\mathbf{A})]$ par le niveau de filtrage $\varepsilon$, et retournant donc le vecteur résidu $\mathbf{r}^{(m+1)} = \mathcal{F}_{m+1}(\mathbf{A})\,\mathbf{r}^{(0)}$ ainsi que l'itéré correspondant $\mathbf{x}^{(m+1)}$ tel que $\mathbf{b} - \mathbf{A}\mathbf{x}^{(m+1)} = \mathbf{r}^{(m+1)}$. Les étapes 5.ii et 5.iii de l'Algorithme 2.1 sont liées par la relation suivante :

$$\mathbf{r}^{(m+1)} = \mathcal{F}_{m+1}(\mathbf{A})\,\mathbf{r}^{(0)} = \frac{T_{m+1}(\omega_\mu(\mathbf{A}))\,\mathbf{r}^{(0)}}{T_{m+1}(d_\mu)} = \frac{1}{\sigma_{m+1}}\,T_{m+1}\Big(d_\mu\,\mathbf{I} - \alpha_\mu\,\mathbf{A}\Big)\,\mathbf{r}^{(0)}$$

où $d_\mu = \dfrac{\lambda_{\max} + \mu}{\lambda_{\max} - \mu}$, $\alpha_\mu = \dfrac{2}{\lambda_{\max} - \mu}$ et $\sigma_m = T_m(d_\mu)$ pour tout $m \geq 0$. L'étape 5.iii peut être en effet, remplacée par la relation de récurrence suivante

$$\mathbf{r}^{(m+1)} = 2\,\frac{\sigma_m}{\sigma_{m+1}}\Big(d_\mu \mathbf{r}^{(m)} - \alpha_\mu \mathbf{A}\mathbf{r}^{(m)}\Big) - \frac{\sigma_{m-1}}{\sigma_{m+1}}\,\mathbf{r}^{(m-1)}, \tag{2.7}$$

qui correspond à l'itération de Tchebycheff classique calculant $\mathbf{r}^{(m+1)}$ en fonction de $\mathbf{r}^{(m)}$ et $\mathbf{r}^{(m-1)}$.

## 2.3 Factorisation Spectrale Partielle

La factorisation spectrale partielle proposée dans [4] se base sur la combinaison des filtres polynomiaux de Tchebycheff, présentés au § 2.2, avec le processus de Lanczos/Lanczos par blocs [33, 59, 69, 84] pour calculer, pour un niveau de filtrage $\varepsilon$ fixé, une base orthogonale $\mathbf{W}$ du sous-espace invariant associé aux plus petites valeurs propres de la matrice $\mathbf{A}$.

Les étapes de cette technique sont résumées dans l'Algorithme 2.2, appelé aussi *Chebyshev–Partial Spectral Factorization* (Tchebycheff-PSF) [4, 28, 29].

Algorithme 2.2 : Tchebycheff-PSF

**Début**

**Calcul de $\lambda_{\max}(\mathbf{A})$ et choix de $\mu$**

1. Calcul d'une approximation de $\lambda_{\max}(\mathbf{A})$ par la méthode de la puissance
2. $\mu = \lambda_{\max}(\mathbf{A})/\alpha$, ($\alpha = 10$, ou 100 par exemple)

**Filtrage initial de la base générée**

3. $\mathbf{P}^{(0)} = \text{random}(n, s)$
4. $\mathbf{Z}^{(0)} = \text{Tchebycheff–Filter}(\mathbf{P}^{(0)}, \varepsilon, [\mu, \lambda_{\max}], \mathbf{A})$
5. $\mathbf{W}^{(0)} = \text{orthonormaliser}(\mathbf{Z}^{(0)})$
6. $\mathbf{W} = \mathbf{W}^{(0)}$ ; and $\delta_2 = 1$
7. **Pour** $k = 0, 1, \ldots,$ jusqu'à convergence **Faire :**

**Processus de Lanczos/Lanczos par blocs**

i. $\mathbf{P}^{(k+1)} = (\mathbf{I} - \mathbf{W}\mathbf{W}^T)\mathbf{A}\mathbf{W}^{(k)}$
ii. $[\mathbf{Q}^{(k+1)}, \mathbf{\Sigma}_1^{(k+1)}, \mathbf{V}] = \text{SVD}(\mathbf{P}^{(k+1)}, 0)$
iii. $\delta_1 = \min(\text{diag}(\mathbf{\Sigma}_1^{(k+1)}))$

**Maintien du niveau de filtrage**

iv. $\delta = \max(\varepsilon, \delta_1 \times \delta_2)$
v. $\mathbf{Z}^{(k+1)} = \text{Tchebycheff–Filter}(\mathbf{Q}^{(k+1)}, \delta, [\mu, \lambda_{\max}], \mathbf{A})$
vi. $\mathbf{Y}^{(k+1)} = (\mathbf{I} - \mathbf{W}\mathbf{W}^T)\mathbf{Z}^{(k+1)}$
vii. $[\mathbf{W}^{(k+1)}, \mathbf{\Sigma}_2^{(k+1)}, \mathbf{V}] = \text{SVD}(\mathbf{Y}^{(k+1)}, 0)$
viii. $\delta_2 = \min(\text{diag}(\mathbf{\Sigma}_2^{(k+1)}))$

**Incorporation des vecteurs dans la base courante**

ix. $\mathbf{W} = [\mathbf{W}; \mathbf{W}^{(k+1)}]$

8. **FinPour**

**Fin**

$\text{SVD}(\mathbf{Z}, 0)$ dénote la décomposition *"économique"* en valeurs singulières de $\mathbf{Z}$.

Pour démarrer l'algorithme, il est nécessaire de connaître la plus grande valeur propre de la matrice $\mathbf{A}$, $\lambda_{\max}(\mathbf{A})$. Ceci peut être obtenu, par exemple, avec quelques itérations de la méthode de la puissance introduite dans le chapitre 1, § 1.1. Mais il est possible également d'utiliser simplement une borne supérieure de $\lambda_{\max}(\mathbf{A})$, qui peut d'ailleurs dans certains cas être facilement déterminée à partir de propriétés de la matrice $\mathbf{A}$.
Pour définir l'intervalle de filtrage $[\mu, \lambda_{\max}(\mathbf{A})]$, qui sera incorporé dans les filtres polynomiaux de Tchebycheff précédemment décrits au § 2.2, nous fixons alors la valeur de coupure $\mu$ à $\lambda_{\max}(\mathbf{A})/\alpha$ (où $\alpha = 10$, ou 100 par exemple). Cela permet en particulier de contrôler explicitement le taux de convergence du filtre polynomial de Tchebycheff $\mathcal{F}_m$ sur l'intervalle $[\mu, \lambda_{\max}(\mathbf{A})]$, sachant que $\mathcal{F}_m(0)$ est fixé égal à 1. Cette valeur de coupure $\mu$, comme nous le verrons dans ce qui suit (chapitre 3), contrôle aussi le conditionnement réduit $\lambda_{\max}(\mathbf{A})/\mu$ qui sera exploité de manière implicite après calcul du sous-espace invariant associé à toutes les valeurs propres de $\mathbf{A}$ dans l'intervalle $]0, \mu]$.

Nous pouvons déjà insister sur le fait qu'il y a un compromis à atteindre dans le choix de $\mu$. En effet, un choix de $\mu$ trop grand peut induire beaucoup plus de valeurs propres (et vecteurs propres associés) dans l'intervalle $]0, \mu]$, et donc une dimension assez grande pour l'espace invariant qui sera calculé. A l'opposé, une valeur de $\mu$ très petite augmentera de manière substantielle le nombre d'itérations de Tchebycheff requises pour atteindre le niveau de filtrage prédéterminé $\varepsilon$ sur l'intervalle $[\mu, \lambda_{\max}(\mathbf{A})]$.

Les étapes 3 à 5 de l'algorithme Tchebycheff-PSF correspondent à l'application initiale du polynôme de Tchebycheff permettant de générer dans un premier temps un ensemble orthonormal $\mathbf{W}^{(0)}$ de $s$ vecteurs filtrés, où le paramètre $s \geq 1$ est la taille du bloc comme dans les techniques de Lanczos par blocs. Après cette première phase de filtrage, les composantes propres dans $\mathbf{W}^{(0)}$ qui correspondent aux valeurs propres de $\mathbf{A}$ dans l'intervalle $[\mu, \lambda_{\max}(\mathbf{A})]$ seront (en valeur absolue) au-dessous d'un certain niveau de filtrage $\varepsilon \ll 1$. Nous pouvons alors commencer l'itération de Lanczos par blocs, correspondant aux étapes 7.i et 7.ii dans l'algorithme Tchebycheff-PSF. Le processus de Lanczos permettra de construire, par l'intermédiaire des espaces de Krylov, un sous-espace invariant associé à toutes les valeurs propres de $\mathbf{A}$ dans l'intervalle $]0, \mu[$.

Plus précisément, ce processus revient à former

$$\mathbf{P}^{(k+1)} = \left(\mathbf{I} - \mathbf{W}\mathbf{W}^T\right)\mathbf{A}\mathbf{W}^{(k)}$$

et à orthonormaliser l'ensemble des $s$ vecteurs $\mathbf{P}^{(k+1)}$ pour calculer la prochaine entrée $\mathbf{W}^{(k+1)}$ dans la base orthonormale de Krylov $\mathbf{W}$. Comme discuté dans [4], la difficulté principale de l'itération de Lanczos/Lanczos par blocs est qu'elle détériore graduellement la séparation relative entre les composantes propres dans $\mathbf{W}$, initialement filtrées à $\varepsilon$ dans $\mathbf{W}^{(0)}$.

Cette détérioration peut être directement estimée (en termes de borne supérieure) par le calcul de la plus petite valeur singulière $\delta_1$ (déterminée à l'étape 7.iii) lorsqu'on orthonormalise le bloc de directions de Krylov $\mathbf{P}^{(k+1)}$ courant. Le but des étapes 7.iv à 7.v est de maintenir alors, sous le niveau $\varepsilon$, avec quelques itérations supplémentaires de filtrage, cette séparation relative entre les composantes propres associées à toutes les valeurs propres dans l'intervalle $[\mu, \lambda_{\max}(\mathbf{A})]$ dans les nouveaux vecteurs de Lanczos $\mathbf{W}^{(k+1)}$.

Les étapes 7.vi à 7.viii, visent à orthogonaliser les vecteurs récemment filtrés par rapport à la base orthonormale courante $\mathbf{W}$. Nous mentionnons qu'il est nécessaire d'exécuter une ré-orthogonalisation totale, puisque appliquer le filtre polynomial matriciel en $\mathbf{A}$ à l'ensemble des vecteurs $\mathbf{Q}^{(k+1)}$ détruit la propriété d'emboîtement intrinsèque des sous-espaces de Krylov (cf. Chapitre 1.)

Tester la convergence de l'algorithme Tchebycheff-PSF, ou l'invariance de la base $\mathbf{W}$, peut être réalisé par la génération aléatoire d'un vecteur supplémentaire au début, que l'on filtrera aussi au dessous du niveau $\varepsilon$ et que l'on normalisera. Ce vecteur sera utilisé (quand par exemple $\delta_2$ devient petit) pour tester l'invariance, en le projetant sur le complémentaire orthogonal de la base calculée $\mathbf{W}$, et en évaluant si la norme de ce projeté est proche de la valeur de filtrage $\varepsilon$ ou non. La convergence devrait être atteinte, *a priori*, quand la dimension de la base $\mathbf{W}$ est très proche du nombre de valeurs propres de $\mathbf{A}$ au-dessous de la valeur de coupure $\mu$.

### 2.3.1 Quelques Remarques

Comme expliqué dans la formule précédemment décrite (2.6), l'utilisation des filtres polynomiaux de Tchebycheff, dans le contexte de l'itération de Lanczos/Lanczos par blocs, peut être vue comme une sorte de déflation implicite dans le sous-espace invariant $\mathbf{U}_2$ associé aux plus grandes valeurs propres de $\mathbf{A}$ [42, 51, 52, 77]. Les filtres polynomiaux de Tchebycheff, ne présentent pas les mêmes propriétés que les projecteurs employés généralement dans une telle déflation (qui ont des valeurs propres explicites égales à 0 et 1), mais ces filtres polynomiaux imitent en partie les propriétés des projecteurs et offrent une alternative permettant d'obtenir le même type de comportement dans le processus de Lanczos/Lanczos par blocs que celui observé dans le cas des techniques de déflation. Pour autant, ces filtres polynomiaux évitent d'avoir à calculer les vecteurs propres liés à la mise en œuvre d'une telle déflation.

Cette combinaison des filtres de Tchebycheff avec la technique de Lanczos par blocs permet aussi de calculer une approximation des vecteurs propres associés à toutes les valeurs propres dans un intervalle donné $[\lambda_{\min}(\mathbf{A}), \mu[$ sans connaître *a priori* leur nombre. Naturellement, pour des raisons d'efficacité, il est souhaitable que le spectre de la matrice $\mathbf{A}$ soit fortement concentré

dans l'intervalle $[\mu, \lambda_{\max}(\mathbf{A})]$, pour une valeur de coupure $\mu$ raisonnable. Il est en effet important que $\mu$ ne soit pas trop petit par rapport à $\lambda_{\max}(\mathbf{A})$ pour que le nombre d'itérations de Tchebycheff exigées pour atteindre un certain niveau de filtrage $\varepsilon$ ne soit pas trop grand. D'autre part, il est aussi important que le nombre de valeurs propres de $\mathbf{A}$ inférieures à $\mu$ ne soit pas trop grand, pour que le nombre d'itérations de Lanczos et la dimension de la base $\mathbf{W}$ restent petits. À cet égard, un préconditionnement préliminaire appliqué à la matrice $\mathbf{A}$ peut également être utile pour transformer et mieux regrouper le spectre de la matrice d'itération.

D'autres techniques itératives, comme celles disponibles dans la bibliothèque ARPACK [44] par exemple, peuvent également être employées pour approcher un certain nombre de valeurs propres ou de vecteurs propres de la matrice $\mathbf{A}$. La différence, ici, est que l'algorithme Tchebycheff-PSF vise directement un conditionnement réduit ($\lambda_{\max}/\mu$) – avec le risque éventuel pour un mauvais choix de la valeur de coupure $\mu$, de construire une base de Krylov de plus ou moins grande taille – alors que les routines "*équivalentes*" dans ARPACK ciblent un nombre prédéterminé de valeurs propres et vecteurs propres – avec le risque cette fois d'avoir au final un conditionnement réduit relativement grand pour un nombre de valeurs propres ciblées trop petit –. Cette remarque n'a pas pour but de dire qu'une méthode est meilleure que l'autre, mais simplement que leurs objectifs sont différents, puisqu'avec l'algorithme Tchebycheff-PSF il est possible de contrôler directement un intervalle de valeurs propres, alors qu'avec les techniques de ARPACK, on peut mieux contrôler en général le coût des calculs et l'occupation en mémoire.

## 2.4 Essais Numériques

Dans cette section, nous analysons l'influence de certains paramètres sur le comportement numérique de Tchebycheff-PSF. Le spectre de notre cas test est extrait à partir de celui d'une matrice d'itération obtenue avec la méthode de Cimmino par blocs (voir [68]) appliquée à une matrice non symétrique de la collection des matrices creuses Harwell-Boeing [22]. Nous notons que ce problème test a un mauvais conditionnement (de l'ordre $10^{13}$), puisque la plus petite valeur propre est $1.08 \cdot 10^{-13}$ et la plus grande 2.59. Ce type de technique de préconditionnement (cf. [1]) transforme la matrice originale $\mathbf{A}$ en une matrice symétrique définie positive dont les valeurs propres sont en général regroupées dans un nombre relativement petit d'amas, mais sans amélioration particulière du conditionnement.

La figure 2.2 montre la distribution des valeurs propres de ce problème test, 33 valeurs propres sont à l'intérieur de $[\lambda_{\min}, \lambda_{\max}/5]$, 26 valeurs propres sont à l'intérieur de $[\lambda_{\min}, \lambda_{\max}/10]$, et 19 à l'intérieur de $[\lambda_{\min}, \lambda_{\max}/100]$.

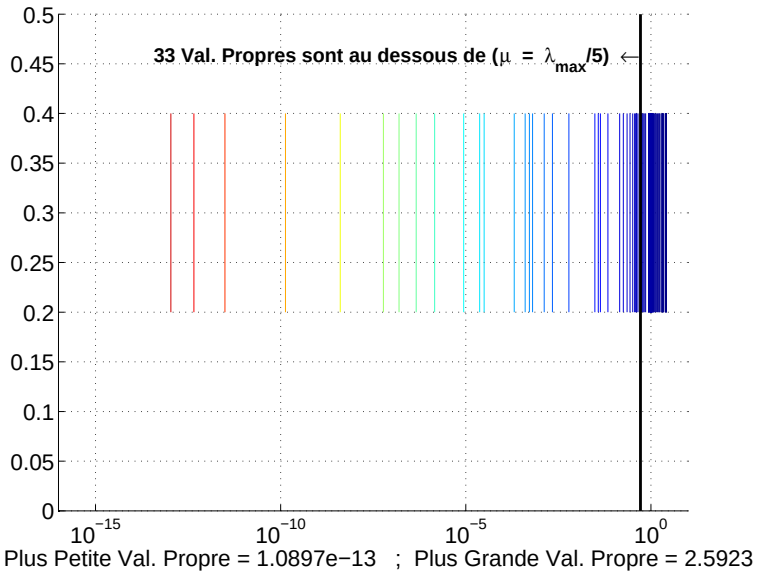



FIG. 2.2 – *Distribution des valeurs propres du problème test de taille 137.*

33 valeurs propres sont au dessous de $\mu = \lambda_{\max}/5$.
26 valeurs propres sont au dessous de $\mu = \lambda_{\max}/10$.
19 valeurs propres sont au dessous de $\mu = \lambda_{\max}/100$.

### 2.4.1 Influence des Différents Paramètres

L'algorithme Tchebycheff-PSF présenté au § 2.3 dépend de trois paramètres différents qui correspondent, pour le premier, au choix de la taille du bloc de départ $s$, pour le second, de la valeur de coupure $\mu$ et pour le dernier, du niveau de filtrage $\varepsilon$ en dessous duquel sont maintenues les composantes propres associées à toutes les valeurs propres plus grande que $\mu$. Dans le reste de cette section, nous discutons de l'influence de ces trois paramètres sur le comportement de l'Algorithme 2.2.

#### Choix de la Taille de Bloc $s$

La valeur de $s$ influence la plupart du temps le nombre d'étapes du processus de Lanczos par blocs, qui sera environ $k/s$, $k$ étant la dimension du sous-espace invariant $\mathbf{U}_1$ associé aux valeurs propres extérieures à l'intervalle d'atténuation $[\mu, \lambda_{\max}(\mathbf{A})]$. Dans [4], Arioli et Ruiz n'ont pas observé un impact important sur le nombre total de produits matrice-vecteur dans Tchebycheff-PSF lorsqu'on fait varier cette taille du bloc $s$. Fondamentalement, ce qu'ils ont observé est que, avant d'atteindre la convergence, le

produit de la taille du bloc $s$ par la somme, sur l'ensemble des itérations de Lanczos, du nombre de pas de refiltrage ne change pas beaucoup pour différentes valeurs de $s$. La taille du bloc $s$ peut par contre avoir une certaine influence sur la dernière étape de l'Algorithme 2.2 quand la convergence est atteinte, simplement parce qu'il faut filtrer à nouveau en dessous de $\varepsilon$ tous les $s$ vecteurs du dernier bloc construit $\mathbf{Q}^{(k+1)}$.

### Choix de la Valeur de Coupure $\mu$

Le deuxième paramètre $\mu$ divise le spectre de la matrice $\mathbf{A}$ en deux sous-ensembles, et fixe la dimension $k$ du sous-espace invariant $\mathbf{U}_1$. Ce paramètre détermine également le taux de convergence de l'itération de Tchebycheff utilisée pour filtrer les vecteurs de Lanczos dans Tchebycheff-PSF. $\mu$ devant être fixée relativement à la plus grande valeur propre de $\mathbf{A}$, une bonne approximation de $\lambda_{\max}$ peut être facilement obtenue avec quelques étapes de la méthode de la puissance.

Dans la table 2.1, nous pouvons observer ces effets combinés sur notre petit exemple d'essai, avec trois valeurs différentes du paramètre $\mu$. Le changement rapide du taux de convergence de l'itération de Tchebycheff, pour des valeurs décroissantes de $\mu$, induit plus d'étapes de filtrage à chaque itération de Lanczos. Ceci pourrait être éventuellement équilibré par une bonne réduction sur le nombre total d'itérations dans l'algorithme Tchebycheff-PSF. En d'autres termes, il vaut mieux réduire la valeur de $\mu$ seulement si les valeurs propres de la matrice $\mathbf{A}$ sont bien regroupées et si le changement de $\mu$ diminue la dimension du sous-espace invariant $\mathbf{U}_1$ qui sera approché au final.

À cet égard, le préconditionnement est un point clé, qui permet de bien regrouper les valeurs propres dans le spectre de la matrice d'itération. En outre, la méthode Tchebycheff-PSF s'adresse à des problèmes très mal conditionnés et offre la possibilité d'employer un éventail large de techniques de préconditionnement, qui regroupent essentiellement le spectre sans la contrainte de réduire le conditionnement en même temps.

### Choix du Niveau de Filtrage $\varepsilon$

Le choix du niveau de filtrage $\varepsilon$ a un impact direct sur la qualité du sous-espace invariant "approché", obtenu à terminaison de l'Algorithme 2.2.

Dans la table 2.1, nous comparons le nombre d'itérations de Tchebycheff requises à chaque étape de filtrage, pour trois valeurs de $\varepsilon$ différentes. Nous observons que des petites valeurs pour $\varepsilon$ induisent plus d'itérations de Tchebycheff dans la première et les dernières étapes de filtrage. En effet, le nombre d'itérations dans la première étape de filtrage est directement lié au choix du niveau de filtrage $\varepsilon$, puisque le but est de filtrer un certain ensemble

| | Nombre d'itérarions de Tchebycheff | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | $\mu = \lambda_{\max}/5$ | | | $\mu = \lambda_{\max}/10$ | | | $\mu = \lambda_{\max}/100$ | | |
| La taille de bloc de Lanczos $s = 6$ | Valeur de $\varepsilon$ | | | Valeur de $\varepsilon$ | | | Valeur de $\varepsilon$ | | |
| | $10^{-16}$ | $10^{-8}$ | $10^{-4}$ | $10^{-16}$ | $10^{-8}$ | $10^{-4}$ | $10^{-16}$ | $10^{-8}$ | $10^{-4}$ |
| 0 | 42 | 23 | 13 | 62 | 33 | 19 | 200 | 108 | 62 |
| 1 | 10 | 10 | 8 | 16 | 15 | 14 | 62 | 57 | 50 |
| 2 | 11 | 11 | 10 | 20 | 19 | 16 | 87 | 90 | 50 |
| 3 | 16 | 15 | 11 | 33 | 30 | 16 | 188 | 96 | 50 |
| 4 | 26 | 20 | 11 | 58 | 30 | 16 | - | - | - |
| 5 | 39 | 20 | 11 | - | - | - | - | - | - |

Dans le cas où $\mu = \lambda_{\max}/5$, il y a 33 vecteurs propres à capturer.
Dans le cas où $\mu = \lambda_{\max}/10$, il y a 26 vecteurs propres à capturer.
Dans le cas où $\mu = \lambda_{\max}/100$, il y a 19 vecteurs propres à capturer.

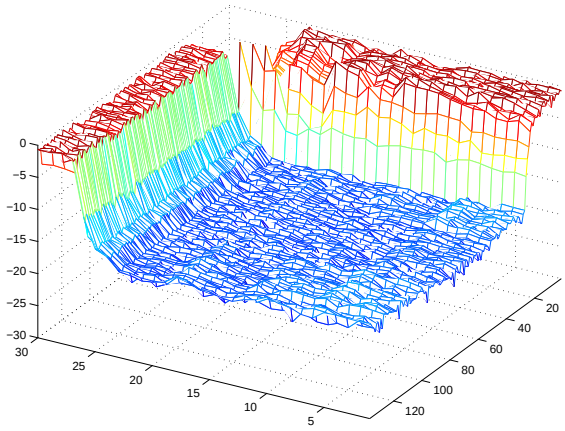
TAB. 2.1 – *Comparaison du nombre d'étapes de filtrage de Tchebycheff pour différentes valeurs du niveau de filtrage $\varepsilon$ et pour différentes valeur de coupure $\mu$. (la taille du Bloc Lanczos égale à 6).*

de vecteurs générés aléatoirement au dessous de ce niveau. A convergence, quand l'invariance du sous-espace calculé est presque atteinte, tout ou partie des vecteurs du dernier bloc de Krylov $\mathbf{P}^{(k+1)}$ sont fortement colinéaires au complémentaire orthogonal $\mathbf{U}_2$ du sous-espace invariant $\mathbf{U}_1$. La valeur $\delta_1$ calculée à l'étape 7.iii de Tchebycheff-PSF devient très petite, et un plus grand nombre d'itérations de refiltrage est alors nécessaire. Ce sont les deux raisons pour lesquelles des petites valeurs pour $\varepsilon$ impliquent plus d'itérations de Tchebycheff dans la première et les dernières étapes de filtrage.

La table 2.1 montre aussi que, pendant les étapes intermédiaires, le nombre d'itérations de Tchebycheff ne varie pas beaucoup avec le choix de $\varepsilon$. En effet, les étapes intermédiaires de refiltrage visent simplement à récupérer une certaine augmentation potentielle du niveau de filtrage après orthogonalisation des vecteurs de Lanczos par rapport aux précédents.

Les figures 2.3 et 2.4 montrent, pour le problème test décrit au début du § 2.4, l'influence du niveau de filtrage $\varepsilon$ sur la qualité et la structure numérique de la base $\mathbf{W}$. Dans ces figures, nous montrons, en échelle logarithmique, la décomposition propre de la base de Krylov de taille 30 obtenue après 5 étapes du processus Lanczos par blocs avec une taille de bloc égale à 6. Nous notons que le paramètre de coupure $\mu$ est fixé à $\lambda_{\max}/10$ et que

les vecteurs propres de $\mathbf{A}$ sont indexés sur l'axe X (de 1 jusqu'à 137).
les index (de 1 à 30) des vecteurs de Krylov sont indiqués sur l'axe Y.
L'axe Z indique le logarithme des valeurs absolues des composantes propres dans chaque vecteur de Lanczos/Orthodir.




les vecteurs propres de $\mathbf{A}$ sont indexés sur l'axe X (de 1 jusqu'à 137).
les index (de 1 à 30) des vecteurs de Krylov sont indiqués sur l'axe Y.
L'axe Z indique le logarithme des valeurs absolues des composantes propres dans chaque vecteur de Shur.

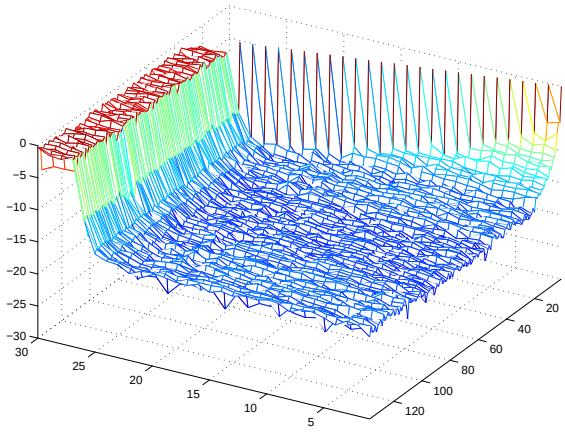
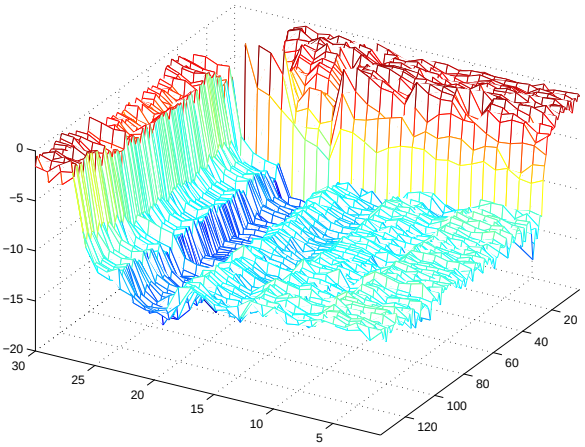



FIG. 2.3 – *La décomposition propre de la base de Krylov et les vecteurs de Schur obtenus avec un niveau de filtrage proche de la précision machine* ($\varepsilon = 10^{-16}$). *(5 étapes de Lanczos par blocs/Orthodir avec une taille du bloc* $s = 6$*).*

les vecteurs propres de $\mathbf{A}$ sont indexés sur l'axe X (de 1 jusqu'à 137).
les index (de 1 à 30) des vecteurs de Krylov sont indiqués sur l'axe Y.
L'axe Z indique le logarithme des valeurs absolues des composantes propres dans chaque vecteur de Lanczos/Orthodir.




les vecteurs propres de $\mathbf{A}$ sont indexés sur l'axe X (de 1 jusqu'à 137).
les index (de 1 à 30) des vecteurs de Krylov sont indiqués sur l'axe Y.
L'axe Z indique le logarithme des valeurs absolues des composantes propres dans chaque vecteur de Schur.

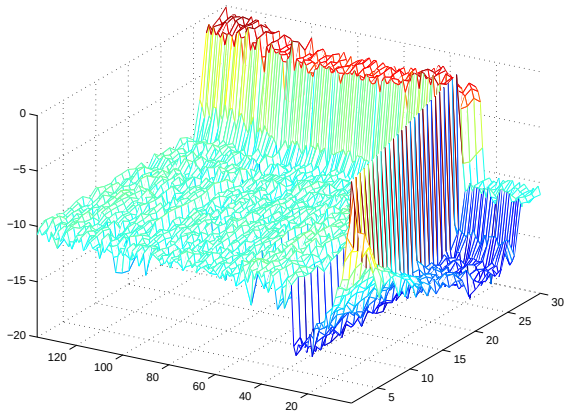



FIG. 2.4 – *La décomposition propre de la base de Krylov et les vecteurs de Schur correspondants obtenus avec un niveau de filtrage* $\varepsilon = 10^{-8}$. *(5 étapes de Lanczos par blocs/Orthodir avec une taille du bloc* $s = 6$*).*

Dans le bas de cette figure, on a fait une rotation de $90^{\circ}$ pour une bonne visualisation.

le sous-espace invariant visé est de taille 26 (cf. figure 2.2). Dans le cas où le niveau de filtrage est très proche de la précision machine $\varepsilon = 10^{-16}$ (cf. figure 2.3), nous pouvons voir la qualité numérique du sous-espace invariant associé aux $k = 26$ valeurs propres les plus petites. Cette qualité est illustrée par les composantes propres des vecteurs de Schur en ce qui concerne le complémentaire orthogonal $\mathbf{U}_2$ du sous-espace invariant $\mathbf{U}_1$, et qui sont très proche du niveau de filtrage $\varepsilon = 10^{-16}$. Dans le cas où le niveau de filtrage $\varepsilon = 10^{-8}$ (cf. figure 2.4), nous pouvons remarquer que la structure de la base calculée change par rapport à celle de la figure 2.3, mais sa qualité numérique reste bonne. Ceci est montré par les composantes propres des vecteurs de Schur (voir le bas de la figure 2.4).

## 2.5 Conclusion

Dans ce chapitre nous avons présenté les filtres polynomiaux de Tchebycheff et explicité les étapes de la factorisation spectrale partielle appelée aussi Tchebycheff-Partial Spectral Factorisation (Tchebycheff-PSF). Dans le chapitre suivant, la méthode Tchebycheff-PSF sera incorporée dans des techniques de résolution exploitant l'information spectrale afin d'établir une étude comparative de leur comportement et efficacité numériques.

# Chapitre 3

# Étude Comparative de Solveurs Itératifs Exploitant une Certaine Information Spectrale

Ce chapitre est entièrement consacré à une étude comparative du comportement numérique et de l'efficacité de techniques de résolution de systèmes linéaires (1.1), exploitant une certaine information spectrale de la matrice du système à résoudre. Parmi ces techniques, nous considérerons dans un premier temps l'algorithme Tchebycheff-PSF [4, 28, 29], précédemment décrit, permettant de réaliser une factorisation spectrale partielle en calculant, avec un niveau de filtrage $\varepsilon$ prédéterminé, une base orthogonale $\mathbf{W}(\varepsilon)$ du sous-espace invariant associé aux plus petites valeurs propres de $\mathbf{A}$. Cette information spectrale est ensuite exploitée dans des techniques de déflation ainsi que des algorithmes du type *deux grilles*, pour lesquels nous testerons différents lisseurs. Tous ces algorithmes seront implémentés en **Matlab**© pour l'évaluation de leur potentiel ou de leurs limitations.

## 3.1 Introduction

Le solveur itératif de choix pour résoudre un système linéaire (1.1) symétrique défini positif, est le gradient conjugué. Cependant, sa convergence est souvent pénalisée dès que la matrice $\mathbf{A}$ possède des valeurs propres au voisinage de zéro. De plus, en présence d'amas de valeurs propres aux extrémités du spectre de la matrice $\mathbf{A}$, combiné avec un mauvais conditionnement, le gradient conjugué classique peut exhiber une courbe de convergence avec un certain nombre de plateaux séparés par des sections de convergence su-

perlinéaire. Ces plateaux correspondent à la découverte de valeurs propres distinctes au sein d'un même amas, ce qui a déjà été observé et analysé dans [64, 85, 86]. Le problème est que ces plateaux peuvent être plutôt larges, en termes de nombre d'itérations, même lorsque ces amas aux extrêmes incorporent seulement quelques valeurs propres. Le gradient conjugué par blocs est une alternative pour réduire la taille des ces plateaux, mais qui en général ne permet pas de les éliminer complètement (voir, par exemple, [3, 13, 53]). Par conséquent, "*gommer*" l'effet néfaste des petites valeurs propres dans la matrice $\mathbf{A}$ devrait avoir un effet bénéfique sur le comportement et la vitesse de convergence du gradient conjugué.

Dans ce qui suit, nous supposons que la matrice $\mathbf{A}$ du système linéaire (1.1), qui est symétrique définie positive (SPD), présente aussi un spectre largement regroupé. Nous considérerons aussi le fait d'avoir à résoudre plusieurs systèmes linéaires avec la même matrice mais à second membres multiples et disponibles en séquence. Cette problématique est assez courante, comme dans le cas de la simulation numérique de problèmes d'évolution par exemple, et se prête assez bien à l'utilisation de solveurs directs, en particulier, qui factorisent initialement la matrice $\mathbf{A}$ et permettent ensuite de résoudre à moindre coût les systèmes linéaires issus de la variation des second membres. Cependant, quand la matrice $\mathbf{A}$ n'est elle même pas connue explicitement, la factorisation est rendue impossible et il est alors nécessaire de trouver une alternative du type itératif à coût raisonnable.

Plusieurs techniques ont été proposées pour améliorer la convergence du gradient conjugué, en mettant à jour le préconditionneur ou en contraignant le gradient conjugué à travailler dans le complémentaire orthogonal du sous espace invariant associé aux plus petites valeurs propres de $\mathbf{A}$. Le but de ce chapitre est de comparer le comportement numérique et l'efficacité de ces différentes techniques. Parmi ces techniques, nous considérerons, dans un premier temps, l'algorithme (Tchebycheff-PSF) [4, 28, 29], introduit au chapitre précédent, et qui permet de réaliser une factorisation spectrale partielle. Ce dernier est basé sur des filtres polynômiaux de Tchebycheff combinés avec le processus de Lanczos, pour calculer avec différents niveaux de filtrage $\varepsilon$, une base orthogonale approchée $\mathbf{W}(\varepsilon)$ du sous-espace invariant associé aux plus petites valeurs propres de la matrice $\mathbf{A}$. Cette information spectrale est ensuite exploitée dans des techniques de déflation ainsi que des algorithmes à deux grilles, pour lesquels nous manipulerons différents lisseurs.

En particulier, nous considérerons la version du gradient conjugué déflaté introduite par Saad et al. [77]. Cet algorithme exploite le lien entre l'algorithme de Lanczos et l'algorithme du gradient conjugué, et est mathématiquement équivalent à la version de Nicolaides [52], issue de la procédure de Lanczos déflaté. Kolotilina [42] emploie une technique de déflation double pour projeter simultanément les $k$ plus grandes et les $k$ plus petites valeurs propres en utilisant un sous-espace de déflation approprié, de

dimension $k$. Une autre alternative consiste à appliquer la déflation pour atténuer les effets néfastes des plus petites valeurs propres sur la convergence [20, 88]. Cette approche est utilisée dans [17] pour résoudre des problèmes d'électromagnétisme. Des idées similaires ont été également utilisées dans le cas des problèmes non symétriques pour modifier l'algorithme GMRES [7, 14, 24, 45, 47, 49]. Comme techniques exploitant l'information spectrale pour mettre à jour le préconditionneur, nous considérerons l'approche proposée dans [10], qui vise à translater de la valeur 1 la position des plus petites valeurs propres de la matrice du système.

Ce chapitre s'articule de la façon suivante. Nous consacrerons le § 3.2 aux diverses techniques de résolution de systèmes linéaires qui exploitent l'information spectrale extraite de la matrice $\mathbf{A}$, et nous indiquerons comment elles peuvent être combinées avec la factorisation spectrale partielle (Tchebycheff-PSF) pour construire des solveurs itératifs efficaces. Au § 3.3, nous comparerons ces techniques en termes de comportement numérique sur un ensemble de problèmes modèles provenant de la collection de matrices creuses Harwell-Boeing [22] ainsi que d'autres exemples issus de la discrétisation par éléments finis de problèmes d'Équations aux Dérivées Partielles (EDP), du type équation de diffusion. Dans le § 3.4, nous analyserons en détail la complexité algorithmique de ces différentes techniques de résolution, et nous étudierons comment peuvent-elles être employées pour résoudre efficacement des systèmes linéaires avec la même matrice mais à second membres multiples. En conclusion, au § 3.5, nous rappellerons les résultats nouveaux et donnerons une vue des perspectives de ce travail.

## 3.2 Techniques de Résolution de Systèmes Linéaires Exploitant l'Information Spectrale

Dans ce paragraphe, nous introduisons six techniques de résolution des systèmes linéaires (1.1), exploitant une certaine information spectrale extraite de la matrice $\mathbf{A}$. L'idée générale, commune à toutes ces techniques, est d'exploiter la base orthonormale $\mathbf{W}$ du sous-espace invariant associé aux plus petites valeurs propres de $\mathbf{A}$, en vue d'établir une projection sur le complémentaire orthogonal de $\mathbf{W}$. Une telle projection est utilisée, pour maintenir, implicitement ou explicitement dans les itérations du gradient conjugué, les résidus calculés dans le complémentaire orthogonal de $\mathbf{W}$.

La solution $\mathbf{x}$ de (1.1) peut être décomposée en deux parties $\mathbf{x}^{\text{Part1}}$ et $\mathbf{x}^{\text{Part2}}$, avec la première partie $\mathbf{x}^{\text{Part1}}$ obtenue directement par une projection oblique du second membre $\mathbf{b}$ sur le complémentaire orthogonal de $\mathbf{W}$. Une technique simple pour compléter la solution reviendrait alors à calculer la deuxième partie, $\mathbf{x}^{\text{Part2}}$, en exploitant la méthode du gradient conjugué avec $\mathbf{x}^{\text{Part1}}$ comme itéré initial (voir § 3.2.1).

Les trois autres techniques utilisent le projecteur oblique de manière explicite tout au long les itérations du gradient conjugué. La première, discutée au § 3.2.2, augmente implicitement l'espace de Krylov avec la base $\mathbf{W}$ en le déflatant sur le complémentaire orthogonal de $\mathbf{W}$, à chaque itération, les vecteurs de l'espace de Krylov généré dans les itérations du gradient conjugué. La seconde de ces techniques, introduite au § 3.2.3, utilise l'algorithme du gradient conjugué appliqué au système projeté $\mathcal{P}\mathbf{A}\mathbf{x} = \mathcal{P}\mathbf{b}$, où $\mathcal{P}$ est l'opérateur de projection sur le complémentaire orthogonal de $\mathbf{W}$, dont la particularité est de commuter avec $\mathbf{A}$. La dernière technique, détaillée au § 3.2.4, vise à établir un préconditionneur spectral permettant de translater de la valeur 1 la position des plus petites valeurs propres de la matrice du système linéaire.

Dans le § 3.2.5, nous indiquons comment l'itération de Tchebycheff peut également être employée pour calculer la partie de la solution liée aux plus grandes valeurs propres de $\mathbf{A}$. Quant à la deuxième partie, elle peut toujours être obtenue par une projection oblique du résidu sur la base $\mathbf{W}$. Au § 3.2.6, nous présenterons la méthode algébrique à deux grilles, qui fournit une interprétation alternative de certaines techniques discutées ci-dessus, et offre la possibilité de dériver toute une famille de techniques itératives basées sur le calcul d'une base approchée $\mathbf{W}$ du sous-espace invariant associé aux plus petites valeurs propres.

### 3.2.1 Gradient Conjugué avec Projection Initiale

Une première idée, introduite dans [62, 73, 87] pour la méthode de Lanczos, est de choisir un vecteur initial $\mathbf{x}^{(0)}$ tel que le résidu initial $\mathbf{r}^{(0)} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(0)}$ soit orthogonal à l'espace contenant les directions qui nuisent à la convergence. Typiquement cet espace contient les vecteurs propres associés aux plus petites valeurs propres, car, comme il est montré dans [85], ce sont ces directions qui gênent le plus souvent le gradient conjugué en induisant des paliers dans sa convergence.

Une fois que la base $\mathbf{W}$ du sous-espace invariant associé aux plus petites valeurs propres de $\mathbf{A}$ est obtenue, nous pouvons l'exploiter pour le calcul de la solution. L'idée est d'établir une projection oblique du résidu initial sur cet espace invariant afin d'obtenir les composantes propres de la solution qui correspondent aux plus petites valeurs propres :

$$\mathbf{x}^{(0)} = \mathbf{W}\left(\mathbf{W}^T\mathbf{A}\mathbf{W}\right)^{-1}\mathbf{W}^T\mathbf{b}. \tag{3.1}$$

Ensuite, la méthode du gradient conjugué classique est utilisée pour calculer la partie restante du vecteur solution.

On appelle gradient conjugué avec projection initiale, Init-CG [25, 28, 29, 77], cette variante du gradient conjugué. Elle est résumée dans l'Algorithme 3.1.

Algorithme 3.1 : Init-CG

**Entrée :** Une base orthonormale $\mathbf{W}$ du sous-espace invariant associé aux plus petites valeurs propres de $\mathbf{A}$. Cette base est calculée par l'Algorithme 2.2 (Tchebycheff-PSF)

**Début**

1. $\mathbf{x}^{(0)} = \mathbf{W}\left(\mathbf{W}^T\mathbf{A}\mathbf{W}\right)^{-1}\mathbf{W}^T\mathbf{b}$
2. $\mathbf{x} = \mathsf{CG}\left(\mathbf{A}, \mathbf{b}, \mathbf{x}^{(0)}, \mathrm{tol}\right)$.

**Fin**

### 3.2.2 Gradient Conjugué Déflaté

Dans ce paragraphe, nous considérons la version du gradient conjugué déflaté (Def-CG) décrite dans [77]. Cet algorithme exploite le lien entre l'algorithme de Lanczos et l'algorithme du gradient conjugué, et est mathématiquement équivalent à la version de Nicolaides, dérivée de la procédure de Lanczos déflaté [52].

Soit $\mathbf{A} \in \mathbb{R}^{n\times n}$, symétrique définie positive, $p$ vecteurs linéairement indépendants $\mathbf{w}^{(1)}, \mathbf{w}^{(2)}, \ldots, \mathbf{w}^{(p)}$ et $\mathbf{v}^{(1)}$ vecteur unitaire orthogonal à $\mathbf{w}^{(i)}$ pour $i = 1, 2, \ldots, p$. On pose $\mathbf{W} = [\mathbf{w}^{(1)}, \mathbf{w}^{(2)}, \ldots, \mathbf{w}^{(p)}]$. Comme $\mathbf{A}$ est définie positive, la matrice $\mathbf{W}^T\mathbf{A}\mathbf{W}$ est non-singulière. L'Algorithme 3.2, que nous appellerons algorithme du gradient conjugué déflaté (Def-CG), construit une suite de vecteurs $\{\mathbf{v}^{(k)}\}_{k=1,2,\ldots}$ telle que

$$\mathbf{v}^{(k+1)} \perp [\mathbf{W}, \mathbf{v}^{(1)}, \mathbf{v}^{(2)}, \ldots, \mathbf{v}^{(k)}] \;\text{ et }\; ||\mathbf{v}^{(k+1)}||_2 = 1.$$

On suppose aussi que le vecteur initial $\mathbf{x}^{(0)}$ est tel que $\mathbf{r}^{(0)} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(0)} \perp \mathbf{W}$. On pose $\mathbf{v}^{(1)} = \mathbf{r}^{(0)}/||\mathbf{r}^{(0)}||_2$ et

$$\mathcal{K}_{p,k}(\mathbf{A}, \mathbf{W}, \mathbf{r}^{(0)}) = [\mathbf{W}, \mathbf{v}^{(1)}, \mathbf{v}^{(2)}, \ldots, \mathbf{v}^{(k)}]$$

où $\mathbf{v}^{(k)}$ est le vecteur de l'espace de Krylov généré à l'étape $k$. À la $k^{\mathrm{e}}$ étape de la méthode, on cherche une solution approchée $\mathbf{x}^{(k)}$ telle que

$$\begin{cases} \mathbf{x}^{(k)} & \in \mathbf{x}^{(0)} + \mathcal{K}_{p,k}(\mathbf{A}, \mathbf{W}, \mathbf{r}^{(0)}), \\ \mathbf{r}^{(k)} & \perp \mathcal{K}_{p,k}(\mathbf{A}, \mathbf{W}, \mathbf{r}^{(0)}). \end{cases}$$

Cette technique repose sur la projection oblique du résidu initial $\mathbf{r}^{(0)}$ sur la base $\mathbf{W}$ et sur sa mise à jour :

$$\mathbf{x}^{(0)} = \mathbf{W}(\mathbf{W}^T\mathbf{A}\mathbf{W})^{-1}\mathbf{W}^T\mathbf{b} \quad \text{et} \quad \mathbf{r}^{(0)} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(0)}, \tag{3.2}$$

$$\mathbf{p}^{(0)} = \left(\mathbf{I} - \mathbf{W}(\mathbf{W}^T\mathbf{A}\mathbf{W})^{-1}(\mathbf{A}\mathbf{W})^T\right)\mathbf{r}^{(0)}, \tag{3.3}$$

$$= \mathcal{Q}\,\mathbf{r}^{(0)}. \tag{3.4}$$

Algorithme 3.2 : Def-CG

**Entrée :** Une base orthonormale $\mathbf{W}$ du sous-espace invariant associé aux plus petites valeurs propres de $\mathbf{A}$. Cette base est calculée par l'Algorithme 2.2 (Tchebycheff-PSF)

**Début**

1. Choix d'un vecteur $\mathbf{x}^{(0)}$ tel que $\mathbf{W}^T\mathbf{r}^{(0)} = 0$, où $\mathbf{r}^{(0)} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(0)}$.
2. Résoudre $\mathbf{W}^T\mathbf{A}\mathbf{W}\gamma^{(0)} = \mathbf{W}^T\mathbf{A}\mathbf{r}^{(0)}$.
3. $\mathbf{p}^{(0)} = \mathbf{r}^{(0)} - \mathbf{W}\gamma^{(0)}$.
4. **Pour** $k = 1, 2, \ldots, m$, **Faire :**
   - i. $\alpha_{k-1} = \dfrac{{\mathbf{r}^{(k-1)}}^T\mathbf{r}^{(k-1)}}{{\mathbf{p}^{(k-1)}}^T\mathbf{A}\mathbf{p}^{(k-1)}}$
   - ii. $\mathbf{x}^{(k)} = \mathbf{x}^{(k-1)} + \alpha_{k-1}\mathbf{p}^{(k-1)}$
   - iii. $\mathbf{r}^{(k)} = \mathbf{r}^{(k-1)} - \alpha_{k-1}\mathbf{A}\mathbf{p}^{(k-1)}$
   - iv. $\beta_{k-1} = \dfrac{{\mathbf{r}^{(k)}}^T\mathbf{r}^{(k)}}{{\mathbf{r}^{(k-1)}}^T\mathbf{r}^{(k-1)}}$
   - v. $\mathbf{z}^{(k)} = \mathbf{r}^{(k)} - \mathbf{W}(\mathbf{W}^T\mathbf{A}\mathbf{W})^{-1}\mathbf{W}^T\mathbf{A}\mathbf{r}^{(k)}$
   - vi. $\mathbf{p}^{(k)} = \beta_{k-1}\mathbf{p}^{(k-1)} + \mathbf{z}^{(k)}$
5. **FinPour**

**Fin**

A chaque itération, pour construire l'espace de Krylov augmenté, le nouveau résidu $\mathbf{r}^{(k)}$ est projeté sur le complémentaire $\mathbf{A}$-orthogonal de la base $\mathbf{W}$,

$$\mathbf{z}^{(k)} \quad = \quad \mathbf{r}^{(k)} - \mathbf{W}(\mathbf{W}^T\mathbf{A}\mathbf{W})^{-1}\mathbf{W}^T\mathbf{A}\mathbf{r}^{(k)} = \mathcal{Q}\,\mathbf{r}^{(k)}, \tag{3.5}$$

avant d'établir la prochaine entrée $\mathbf{p}^{(k)}$ dans la base $\mathbf{A}$-orthogonale de l'espace de Krylov. Pour une description assez complète de cette approche, on pourra se reporter à [77].

### 3.2.3 Gradient Conjugué sur le Système Projeté

Pour décrire cette approche, que nous appellerons Proj-CG, nous introduisons la projection suivante :

$$\mathcal{P} = \mathbf{I} - \mathbf{A}\mathbf{W}(\mathbf{W}^T\mathbf{A}\mathbf{W})^{-1}\mathbf{W}^T,\ \mathbf{W} \in \mathbb{R}^{n\times p} \tag{3.6}$$

qui est la projection $\mathbf{A}^{-1}$-orthogonale sur $[\mathbf{W}]^{\perp}$ le long de $[\mathbf{A}\mathbf{W}]$ ($\mathbf{I}$ étant la matrice identité). On suppose que $p \ll n$ et que $\mathbf{W}$ est de rang plein. On décompose le vecteur solution $\mathbf{x}$ en deux parties :

$$\mathbf{x} = (\mathbf{I} - \mathcal{P}^T)\mathbf{x} + \mathcal{P}^T\mathbf{x}.$$

La première partie $(\mathbf{I}-\mathcal{P}^T)\mathbf{x}$ est le composant de $\mathbf{x}$ contenu dans $\mathbf{W}$, quant à la seconde partie $\mathcal{P}^T\mathbf{x}$, elle est orthogonale à $\mathbf{W}$. La première partie est calculée à partir de :

$$\begin{aligned}\Big(\mathbf{I}-\mathcal{P}^T\Big)\mathbf{x} &= \Big(\mathbf{AW}(\mathbf{W}^T\mathbf{AW})^{-1}\mathbf{W}^T\Big)^T\mathbf{x}\\ &= \mathbf{W}\Big(\mathbf{W}^T\mathbf{AW}\Big)^{-1}\mathbf{W}^T\mathbf{Ax}\\ &= \mathbf{W}\Big(\mathbf{W}^T\mathbf{AW}\Big)^{-1}\mathbf{W}^T\mathbf{b}.\end{aligned}$$

Pour calculer la deuxième partie $\mathcal{P}^T\mathbf{x}$, nous utilisons la propriété $\mathbf{A}\mathcal{P}^T\mathbf{x} = \mathcal{P}\mathbf{Ax} = \mathcal{P}\mathbf{b}$ (voir Prop. 3.1), et nous résolvons alors le système

$$\mathcal{P}\mathbf{Ax} = \mathcal{P}\mathbf{b}. \tag{3.7}$$

Kaasschieter [40] note qu'un système semi-défini positif peut être résolu par la méthode du gradient conjugué tant que le second membre est consistant (c-à-d. $\mathbf{b} \in \mathsf{Im}(\mathbf{A})$).

C'est le cas pour le système (3.7), où le même opérateur de projection est appliqué des deux côtés du système non singulier. En outre, puisque le noyau du système n'intervient jamais dans les itérations, les valeurs propres correspondantes (égales à zéro) n'influencent pas la convergence.

Nous nous référons à [51] pour une description détaillée. Cette technique est appelée gradient conjugué appliqué au système projeté (Proj-CG), et est résumée dans l'Algorithme 3.3.

Algorithme 3.3 : Proj-CG

**Entrée :** Une base orthonormale $\mathbf{W}$ du sous-espace invariant associé aux plus petites valeurs propres de $\mathbf{A}$. Cette base est calculée par l'Algorithme 2.2 (Tchebycheff-PSF)

**Début**

1. $\mathcal{P} = \mathbf{I} - \mathbf{AW}\Big(\mathbf{W}^T\mathbf{AW}\Big)^{-1}\mathbf{W}^T$.
2. $\mathbf{x}^{(\mathsf{Part1})} = \mathbf{W}\Big(\mathbf{W}^T\mathbf{AW}\Big)^{-1}\mathbf{W}^T\mathbf{b}$.
3. $\mathbf{x}^{(\mathsf{Part2})} = \mathsf{CG}\ \Big(\mathcal{P}\mathbf{A}, \mathcal{P}\mathbf{b}, \mathbf{x}^{(0)}, \mathrm{tol}\Big)$.
4. $\mathbf{x} = \mathbf{x}^{(\mathsf{Part1})} + \mathcal{P}^T\mathbf{x}^{(\mathsf{Part2})}$.

**Fin**

Nous rappelons les propriétés principales de la projection $\mathcal{P}$ (3.6), définie à partir de la base $\mathbf{W}$ calculée par l'Algorithme 2.2 (Tchebycheff-PSF), qui a servi pour construire l'algorithme Proj-CG.

**Proposition 3.1.** *L'opérateur $\mathcal{P}$ a les propriétés suivantes :*

1. $\mathcal{P}^2 = \mathcal{P}$,
2. $\mathbf{A}\mathcal{P}^T = (\mathcal{P}\mathbf{A})^T = \mathcal{P}\mathbf{A}$,
3. $\mathcal{P}\mathbf{A}\mathcal{P}^T = \mathcal{P}\mathbf{A}$,
4. *La matrice $\mathcal{P}\mathbf{A}$ est symétrique semi-définie positive.*
5. $\mathcal{P}\mathbf{A}\mathbf{W} = 0$.

*Démonstration.*

1. $$\begin{aligned}\mathcal{P}^2 &= \mathbf{I} - 2\,\mathbf{A}\mathbf{W}\Big(\mathbf{W}^T\mathbf{A}\mathbf{W}\Big)^{-1}\mathbf{W}^T \\ &\quad + \mathbf{A}\mathbf{W}\Big(\mathbf{W}^T\mathbf{A}\mathbf{W}\Big)^{-1}\mathbf{W}^T\mathbf{A}\mathbf{W}\Big(\mathbf{W}^T\mathbf{A}\mathbf{W}\Big)^{-1}\mathbf{W}^T \\ &= \mathbf{I} - \mathbf{A}\mathbf{W}(\mathbf{W}^T\mathbf{A}\mathbf{W})^{-1}\mathbf{W}^T \\ &= \mathcal{P}.\end{aligned}$$

2. $$\begin{aligned}(\mathcal{P}\mathbf{A})^T &= \Big(\mathbf{A} - \mathbf{A}\mathbf{W}\big(\mathbf{W}^T\mathbf{A}\mathbf{W}\big)^{-1}\mathbf{W}^T\mathbf{A}\Big)^T \\ &= \mathbf{A} - \mathbf{A}\mathbf{W}\Big(\mathbf{W}^T\mathbf{A}\mathbf{W}\Big)^{-1}\mathbf{W}^T\mathbf{A} \\ &= \mathcal{P}\mathbf{A} \quad \big(\text{car } \mathbf{A}^T = \mathbf{A}\big).\end{aligned}$$

Donc la matrice $\mathcal{P}\mathbf{A}$ est symétrique et $\mathcal{P}\mathbf{A} = (\mathcal{P}\mathbf{A})^T = \mathbf{A}^T\mathcal{P}^T = \mathbf{A}\mathcal{P}^T$.

3. $\mathcal{P}\mathbf{A} = \mathbf{A}\mathcal{P}^T \Longrightarrow \mathcal{P}\mathbf{A}\mathcal{P}^T = \mathcal{P}^2\mathbf{A} = \mathcal{P}\mathbf{A}$.

4. Par hypothèse, $\mathbf{z}^T\mathbf{A}\mathbf{z} \geq 0$ pour tout $\mathbf{z} \in \mathbb{R}^n$. En particulier, $0 \leq (\mathcal{P}^T\mathbf{z})^T\mathbf{A}(\mathcal{P}^T\mathbf{z}) = \mathbf{z}^T\mathcal{P}\mathbf{A}\mathcal{P}^T\mathbf{z}$ et donc la matrice $\mathcal{P}\mathbf{A}\mathcal{P} = \mathcal{P}\mathbf{A}$ est semi-définie positive.

5. $\mathcal{P}\mathbf{A}\mathbf{W} = \mathbf{A}\mathbf{W} - \mathbf{A}\mathbf{W}\Big(\mathbf{W}^T\mathbf{A}\mathbf{W}\Big)^{-1}\mathbf{W}^T\mathbf{A}\mathbf{W} = \mathbf{A}\mathbf{W} - \mathbf{A}\mathbf{W} = 0.$ □

### 3.2.4 Correction Spectrale de Rang Faible

Dans ce paragraphe, nous considérons l'approche proposée dans [10] qui essaye de "*gommer*" l'effet néfaste des plus petites valeurs propres de la matrice considérée. En pratique, cette technique consiste à construire le préconditionneur

$$\mathbf{M} = \mathbf{I} + \mathbf{W}\Big(\mathbf{W}^T\mathbf{A}\mathbf{W}\Big)^{-1}\mathbf{W}^T \tag{3.8}$$

dont l'objectif est de décaler vers 1 les valeurs propres de la matrice $\mathbf{A}$ associées à la base $\mathbf{W}$.

Il est alors possible d'introduire ce préconditionneur spectral $\mathbf{M}$ dans le gradient conjugué préconditonné (PCG), qui bénéficiera de la réduction substantielle du conditionnement du système préconditionné $\mathbf{MA}$. Cette approche, résumée dans l'Algorithme 3.4, est appelée "*Spectral Low Rank Update*" (SLRU) [10].

Algorithme 3.4 : SLRU

**Entrée :** Une base orthonormale $\mathbf{W}$ du sous-espace invariant associé aux plus petites valeurs propres de $\mathbf{A}$. Cette base est calculée par l'Algorithme 2.2 (Tchebycheff-PSF)

**Début**

1. $\mathbf{M} = \mathbf{I} + \mathbf{W}\left(\mathbf{W}^T\mathbf{A}\mathbf{W}\right)^{-1}\mathbf{W}^T$.
2. $\mathbf{x} = \mathsf{PCG}\left(\mathbf{A}, \mathbf{b}, \mathbf{x}^{(0)}, \mathbf{M}, \mathrm{tol}\right)$.

**Fin**

L'effet induit par ce préconditionneur consiste en une correction spectrale de rang faible, qui permet de translater de 1 la position des plus petites valeurs propres en module de la matrice du système de départ. Les principaux composants dans la construction de cette technique de préconditionnement sont rappelés dans la proposition suivante. Soit $\mathbf{W}$ l'ensemble des vecteurs propres associés aux valeurs propres $\lambda_i$ telles que $\lambda_i \leq \mu$, où $\mu \in ]0, \lambda_{\max}(\mathbf{A})[$ est la variable de coupure décrite au chapitre précédent.

**Proposition 3.2.** Si $\mathbf{A}$ est symétrique définie positive, alors $\mathbf{A}_c = \mathbf{W}^T\mathbf{A}\mathbf{W}$ est symétrique définie positive. La matrice définie par $\mathbf{M} = \mathbf{I} + \mathbf{M}_c$, où $\mathbf{M}_c = \mathbf{W}\mathbf{A}_c^{-1}\mathbf{W}^T$, est aussi symétrique définie positive et $\mathbf{M}\mathbf{A}$ est similaire à une matrice dont les valeurs propres sont

$$\begin{cases} \eta_i = & \lambda_i & \text{si} \quad \lambda_i > \mu, \\ \eta_i = & \lambda_i + 1 & \text{si} \quad \lambda_i \leq \mu. \end{cases}$$

*Démonstration.* Par construction la matrice $\mathbf{A}_c$ est symétrique, reste à montrer qu'elle est définie positive. $\mathbf{W}$ est une matrice de taille $n \times k$. Soit $\mathbf{z} \in \mathbb{R}^k \backslash \{0\}$, $\mathbf{z}^T\mathbf{A}_c\mathbf{z} = \mathbf{z}^T\mathbf{W}^T\mathbf{A}\mathbf{W}\mathbf{z} = (\mathbf{W}\mathbf{z})^T\mathbf{A}(\mathbf{W}\mathbf{z}) = ||\mathbf{W}\mathbf{z}||_{\mathbf{A}} > 0$.
En effet, $\mathbf{A}$ est symétrique définie positive et $\mathbf{W}\mathbf{z} \neq 0$ (car $\mathbf{W}$ est de rang plein). Par conséquent, la matrice $\mathbf{A}_c$ est symétrique définie positive et donc inversible.

Soit $\mathbf{x} \in \mathbb{R}^n \backslash \{0\}$, $\mathbf{x}^T\mathbf{M}_c\mathbf{x} = (\mathbf{W}^T\mathbf{x})^T\mathbf{A}_c^{-1}(\mathbf{W}^T\mathbf{x}) = ||(\mathbf{W}^T\mathbf{x})||_{\mathbf{A}_c^{-1}} \geq 0$ car $\mathbf{A}_c$ est symétrique définie positive. Alors, la matrice $\mathbf{M}_c$ est semi-définie positive et $\mathbf{M} = \mathbf{I} + \mathbf{M}_c$ est symétrique définie positive.

Soit $\mathbf{V} = (\mathbf{W}, \mathbf{Z})$, où les colonnes de $\mathbf{Z}$ sont constituées de l'ensemble des $(n-k)$ vecteurs propres associés aux valeurs propres $\lambda_i > \mu$. Soit $\mathbf{\Lambda}_1 = \mathsf{diag}(\lambda_i)$ telle que $\lambda_i \leq \mu$ et $\mathbf{\Lambda}_2 = \mathsf{diag}(\lambda_i)$ telle que $\lambda_i > \mu$, les deux matrices diagonales constituées des valeurs propres associées aux colonnes de $\mathbf{W}$ et $\mathbf{Z}$ respectivement. Nous avons

$$\begin{aligned}\mathbf{MAW} &= \mathbf{AW} + \mathbf{M}_c\mathbf{AW} \\ &= \mathbf{W}\mathbf{\Lambda}_1 + \mathbf{W}\Big(\mathbf{W}^T\mathbf{AW}\Big)^{-1}\Big(\mathbf{W}^T\mathbf{AW}\Big) \\ &= \mathbf{W}(\mathbf{\Lambda}_1 + \mathbf{I}_k)\end{aligned}$$

où $\mathbf{I}_k$ est la matrice identitée de taille $(k \times k)$ et

$$\begin{aligned}\mathbf{MAZ} &= \mathbf{Z}\mathbf{\Lambda}_2 + \mathbf{W}\Big(\mathbf{W}^T\mathbf{AW}\Big)^{-1}\mathbf{W}^T\mathbf{AZ} \\ &= \mathbf{Z}\mathbf{\Lambda}_2 + \mathbf{W}\Big(\mathbf{W}^T\mathbf{AW}\Big)^{-1}\mathbf{W}^T\mathbf{Z}\mathbf{\Lambda}_2 \\ &= \mathbf{Z}\mathbf{\Lambda}_2 \quad \text{puisque} \quad \mathbf{W}^T\mathbf{Z} = 0.\end{aligned}$$

Donc

$$\mathbf{MAV} = [\mathbf{W}, \mathbf{Z}] \begin{bmatrix} (\mathbf{\Lambda}_1 + \mathbf{I}_k) & 0 \\ 0 & \mathbf{\Lambda}_2 \end{bmatrix}.$$

Par conséquent, la matrice $\mathbf{MA}$ est semblable à $\begin{bmatrix} (\mathbf{\Lambda}_1 + \mathbf{I}_k) & 0 \\ 0 & \mathbf{\Lambda}_2 \end{bmatrix}$, et les plus petites valeurs propres sont décalées vers 1. □

### 3.2.5 Combinaison de la Projection Oblique avec l'Itération de Tchebycheff

Dans ce paragraphe, nous décrivons comment nous pouvons exploiter le sous-espace invariant $[\mathbf{W}]$ associé aux plus petites valeurs propres de $\mathbf{A}$ pour résoudre le système linéaire (1.1). L'idée est d'établir une projection oblique du résidu initial ($\mathbf{r}^{(0)} = \mathbf{b} - \mathbf{Ax}^{(0)}$) dans ce sous-espace afin d'obtenir les composantes propres de la solution correspondant aux plus petites valeurs propres :

$$\begin{aligned}\mathbf{r}^{(\text{proj})} &= \mathbf{r}^{(0)} - \mathbf{AW}\Big(\mathbf{W}^T\mathbf{AW}\Big)^{-1}\mathbf{W}^T\mathbf{r}^{(0)}, \\ \text{avec} \quad \mathbf{x}^{(\text{proj})} &= \mathbf{x}^{(0)} + \mathbf{W}\Big(\mathbf{W}^T\mathbf{AW}\Big)^{-1}\mathbf{W}^T\mathbf{r}^{(0)}.\end{aligned} \tag{3.9}$$

Pour calculer la partie restante du vecteur de solution $\mathbf{A}\widetilde{\mathbf{x}} = \mathbf{r}^{(\text{proj})}$, nous pouvons alors employer l'algorithme classique de Tchebycheff sur l'intervalle $[\mu, \lambda_{\max}(\mathbf{A})]$. Cette technique est appelée Init-Tchebycheff [28, 29, § 3.1], et est très similaire à Init-CG introduit au § 3.2.1, à la différence près que c'est une itération de Tchebycheff qui remplace l'itération du gradient conjugué.

Notons que, l'ordre dans la juxtaposition de la projection oblique et l'itération de Tchebycheff n'a pas d'importance quand la base orthogonale $\mathbf{W}$ est calculée avec une très grande exactitude, ou quand $\mathbf{W}$ coïncide avec $\mathbf{U}_1$ ($\mathbf{U}_1$ est la matrice dont les colonnes sont les vecteurs propres correspondant aux plus petits vecteurs propres). Cependant, puisque $[\mathbf{W}]$ n'est qu'une approximation du sous-espace invariant associé aux plus petites valeurs propres, il est préférable de commencer par l'itération de Tchebycheff suivi par la projection oblique. En effet, ceci aide à augmenter l'exactitude de la projection oblique en "*minimisant*" l'influence des composantes propres

Algorithme 3.5 : Init-Tchebycheff

**Entrée :** Une base orthonormale $\mathbf{W}$ du sous-espace invariant associé aux plus petites valeurs propres de $\mathbf{A}$. Cette base est calculée par l'Algorithme 2.2 (Tchebycheff-PSF)

**Début**

**Itération de Tchebycheff sur** $[\mu, \lambda_{\max}(\mathbf{A})]$

1. Calcul de $\mathbf{x}^{(m+1)}$ et $\mathbf{r}^{(m+1)}$ avec l'Algorithme 2.1

**Projection Oblique sur** $[\mathbf{W}]$

2. $\mathbf{e}^{(\text{proj})} = \mathbf{W}\left(\mathbf{W}^T\mathbf{A}\mathbf{W}\right)^{-1}\mathbf{W}^T\,\mathbf{r}^{(m+1)}$
3. $\mathbf{x} = \mathbf{x}^{(m+1)} + \mathbf{e}^{(\text{proj})}$

**Fin**

dans le complémentaire orthogonal de $\mathbf{U}_1$, relativement à celles relatives à $\mathbf{U}_1$, dans les produits scalaires $\mathbf{W}^T\mathbf{r}$ dans l'équation (3.9). Cet ordre, dans la juxtaposition, s'avère en particulier très intéressant quand le niveau de filtrage $\varepsilon$ utilisé pour calculer la base $\mathbf{W}$ n'est pas proche de la précision machine.

La première étape de l'Algorithme 3.5 (Init-Tchebycheff), correspond à l'application des filtres polynômiaux de Tchebycheff vus au Chapitre 2. Quant à la deuxième étape, elle consiste à résoudre l'équation d'erreur associée à $\mathbf{x}^{(m)}$ dans l'espace engendré par $\mathbf{W}$, et la dernière étape réalise la mise à jour de la solution. Le nombre d'itérations de Tchebycheff pour atteindre un niveau de filtrage $\varepsilon$ donné est directement lié au taux de convergence des polynômes de Tchebycheff sur l'intervalle $[\mu, \lambda_{\max}]$ (voir, par exemple, [30, 36]) qui dépend seulement du rapport $\lambda_{\max}/\mu$.

### 3.2.6 Cycle à deux grilles Algébrique

La méthode multigrille a été inventée au milieu des années 1960. C'est seulement au milieu des années 1970 qu'elle a commencé à être mise en pratique de façon efficace. À l'origine, elle ne s'appliquait qu'à la résolution de systèmes linéaires (1.1) provenant de la discrétisation d'équations aux dérivées partielles elliptiques, du type équation de diffusion. Depuis 20 ans, une abondante littérature est parue sur le sujet, aussi bien sur la théorie que sur les applications, particulièrement dans le domaine de la mécanique des fluides : équations qui changent de type pour des écoulements potentiels transsoniques, équations qui changent de type pour les calculs transitoires. Le lecteur pourra consulter [35].

Le noyau des algorithmes multigrilles est un procédé à deux grilles ap-

pliqué récursivement. Un cycle à deux grilles peut être décrit brièvement comme suit. Sur la grille fine, quelques itérations de pré-lissage sont appliquées pour atténuer les hautes fréquences de l'erreur qui correspondent aux composantes propres de l'erreur dans l'espace invariant associé aux plus grandes valeurs propres de $\mathbf{A}$. Le résidu est alors projeté sur la grille grossière où les basses fréquences, qui sont les composantes liées aux plus petites valeurs propres, peuvent être capturées et l'équation d'erreur grossière est résolue. L'erreur grossière est interpolée de nouveau sur la grille fine pour corriger la solution approchée que nous avions obtenue dans l'étape de pré-lissage sur la grille fine. En conclusion, si les nouveaux itérés ne sont pas assez précis, ce cycle à deux grilles est appliqué itérativement :

$$\begin{aligned} \mathbf{x}_1 &= \mathbf{S}(\mathbf{x}, \mathbf{b}, \xi) \\ \mathbf{x}_2 &= \mathbf{x}_1 + \mathbf{W}\mathbf{A}_c^{-1}\mathbf{W}^T(\mathbf{b} - \mathbf{A}\mathbf{x}_1) \\ \mathbf{x} &= \mathbf{S}(\mathbf{x}_2, \mathbf{b}, \xi) \end{aligned} \tag{3.10}$$

où $\mathbf{S}(\mathbf{x}, \mathbf{b}, \xi)$ est le lisseur exécutant $\xi$ itérations sur $\mathbf{x}$ pour résoudre (1.1). Le cycle à deux grilles algébrique est résumé dans l'Algorithme 3.6.

Algorithme 3.6 : Cycle à Deux grilles générique

**Entrée :** Une base orthonormale $\mathbf{W}$ du sous-espace invariant associé aux plus petites de $\mathbf{A}$. Cette base est calculée par l'Algorithme 2.2 (Tchebycheff-PSF)

**Début**

1. $\mathbf{x}^{(1)} = 0$
2. **Pour** k=1, *iter* **Faire :**

   **Pré-lissage : atténuer les hautes fréquences de l'erreur**

   i. $\mathbf{x}_1^{(k)} = \mathbf{S}(\mathbf{x}^{(k)}, \mathbf{b}, \xi)$.

   **Correction de la grille grossière**

   ii. $\mathbf{x}_2^{(k)} = \mathbf{x}_1^{(k)} + \mathbf{W}\mathbf{A}_c^{-1}\mathbf{W}^T(\mathbf{b} - \mathbf{A}\mathbf{x}_1^{(k)})$.

   **Post-lissage : atténuer les hautes fréquences de l'erreur**

   iii. $\mathbf{x}^{(k+1)} = \mathbf{S}(\mathbf{x}_2^{(k)}, \mathbf{b}, \xi)$.
3. **FinPour**
4. $\mathbf{x} = \mathbf{x}^{(iter)}$

**Fin**

Dans cette section, nous définissons explicitement l'espace associé à la grille grossière en calculant les vecteurs propres $\mathbf{W}$ liés aux plus petites

valeurs propres de $\mathbf{A}$. Pour des problèmes symétriques, l'opérateur de restriction de la grille fine à la grille grossière est $\mathbf{W}^T$ et l'opérateur de prolongement est sa transposée, pourvu que les colonnes de $\mathbf{W}$ forment un système de vecteurs orthonormés.

La matrice associée au problème de l'erreur grossière est défini par la formule de Galerkin $\mathbf{A}_c = \mathbf{W}^T \mathbf{A} \mathbf{W}$. Dans la suite, nous comparerons l'utilisation de différents lisseurs (Tchebycheff, Richardson relaxé, gradient conjugué) dans cet algorithme. On pourra se reporter à [11], où cette approche a aussi été exploitée pour concevoir des préconditionneurs, et où la transformation spectrale liée à cette combinaison est analysée.

## 3.3 Essais Numériques

Dans ce paragraphe, nous allons présenter quelques résultats numériques qui comparent les différentes approches en termes d'efficacité numérique pour la résolution des systèmes linéaires (1.1). Nous discuterons également leur complexité algorithmique ainsi que leur sensibilité à l'exactitude de l'information spectrale. Le calcul approximatif de la base orthonormale $\mathbf{W}$, est effectué par l'algorithme de la factorisation spectrale partielle (Tchebycheff-PSF) précédemment décrit au Chapitre 2. Nous évaluerons aussi l'impact du changement du niveau de filtrage $\varepsilon$ dans le calcul de la base $\mathbf{W}$ (en fonction de $\varepsilon$) sur le comportement de toutes ces techniques de résolution. Nous introduisons, pour cela, certains problèmes tests provenant de la collection de matrices creuses Harwell-Boeing [22], ainsi que d'autres exemples issus de la discrétisation par éléments finis de problèmes d'Équations aux Dérivées Partielles (EDP). Ces derniers seront réutilisés par la suite dans le Chapitre 4 pour tester la robustesse de notre méthode du gradient conjugué combinée avec les filtres de Tchebycheff comme préconditionneurs (ChebFilterCG). Nous considérons quatre différentes matrices creuses carrées, BCSSTK14 et BCSSTK15 de la collection Harwell-Boeing, EDP1 et EDP2 provenant des problèmes d'Équations aux Dérivés Partielles. Pour chacune de ces quatre matrices, nous indiquerons leur référence, leur taille, et le nombre d'éléments non nuls qu'elles comportent.

Les matrices EDP1 et EDP2 sont extraites de la discrétisation par éléments finis P1, en utilisant **pdetool**© sous **Matlab**©, de problèmes aux dérivés partielles elliptiques, du type équation de diffusion :

$$\begin{cases} -div\Big(\Lambda(x) \cdot \nabla u\Big) &= f \quad \text{in} \quad \Omega \\ u_{|\partial\Omega} &= 0, \end{cases}$$

où $\Omega \subset \mathbb{R}^2$ est une région en forme L décrite dans la figure 3.1. Les différences principales entre EDP1 et EDP2 résident dans le choix de $\Lambda(x)$ et dans la taille de la discrétisation.

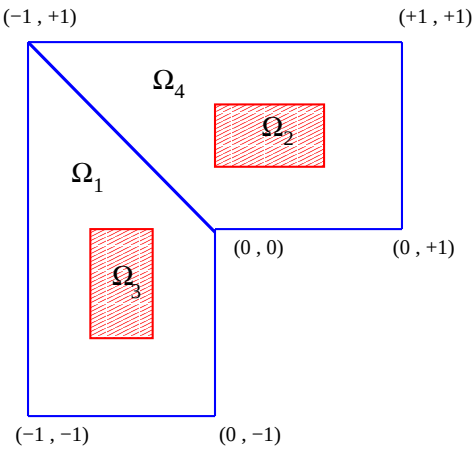



FIG. 3.1 – *La géométrie du domaine* $\Omega$.

Dans le problème test EDP1, $f = 10$ et la fonction $\Lambda(x) \in L^\infty(\Omega)$ prend différentes valeurs scalaires dans chaque domaine :

$$\Lambda(x) = \begin{cases} 1 & x \in \Omega_1 \cup \Omega_4 \\ 10^6 & x \in \Omega_2 \\ 10^4 & x \in \Omega_3. \end{cases}$$

L'ordre du système linéaire résultant (1.1) est $n = 7\,969$ et le nombre de ses éléments non nuls est $\mathsf{nnz}(\mathbf{A}) = 55\,131$.

Dans EDP2, $f = 200$ et le problème (EDP) incorpore de l'hétérogénéité et de l'anisotropie. La matrice $\Lambda(x)$ prend les valeurs suivantes :

$$\Lambda(x) = \begin{cases} \begin{bmatrix} 1 & 4\,\lambda_1 \\ 4\,\lambda_1 & \lambda_1 \end{bmatrix} & \text{si } x \in \Omega_1 \quad \text{et} \quad \lambda_2\,\mathbf{I}_2 \quad \text{si } x \in \Omega_2, \\ \begin{bmatrix} \lambda_1 & -2\,\lambda_1 \\ -2\,\lambda_1 & 1 \end{bmatrix} & \text{si } x \in \Omega_4 \quad \text{et} \quad \lambda_3\,\mathbf{I}_2 \quad \text{si } x \in \Omega_3. \end{cases}$$

où $\lambda_1 = 6 \cdot 10^{-2}$, $\lambda_2 = 1 \cdot 10^6$, $\lambda_3 = 1 \cdot 10^2$ et $\mathbf{I}_2$ la matrice identitée de taille $(2 \times 2)$. L'ordre du système linéaire résultant (1.1) est $n = 161\,313$ et le nombre de ses éléments non nuls est $\mathsf{nnz}(\mathbf{A}) = 1\,125\,897$.

BCSSTK14 et BCSSTK15 sont deux matrices symétriques définies positives. Elles proviennent d'analyses statiques en calcul de structure, et font partie du groupe BCSSTRUC2 de la collection Harwell-Boeing. L'ordre de la matrice BCSSTK14 est $n = 1\,806$ et le nombre de ses éléments non nuls est $\mathsf{nnz}(\mathbf{A}) = 63\,454$, tandis que l'ordre de BCSSTK15 est $n = 3\,948$ et le nombre de ses éléments non nuls est $\mathsf{nnz}(\mathbf{A}) = 117\,816$.

Toutes les informations (référence, taille, nombre d'éléments non nuls ($\mathsf{nnz}$ ($\mathbf{A}$))) pour nos problèmes tests sont récapitulées dans la table 3.1.

| Nom | Taille | nnz(**A**) | description |
|---|---|---|---|
| BCSSTK14 | 1 806 | 63 454 | analyse statique en calcul de structure du groupe BCSSTRUC2, Omni Coliseum, Atlanta |
| BCSSTK15 | 3 948 | 117 816 | analyse statique en calcul de structure du groupe BCSSTRUC2 |
| EDP1 | 7 969 | 55 131 | équation de diffusion discrétisée par éléments finis dans une région L-shape |
| EDP2 | 161 313 | 1 125 897 | équation de diffusion anisotropique discrétisée par éléments finis dans une région L-shape |

TAB. 3.1 – *Ensemble de matrices test.*

Pour toutes les expériences numériques rapportées dans ce paragraphe, un premier niveau de préconditionnement $\mathbf{M}_1 = \mathbf{L}\mathbf{L}^T$ est construit en utilisant la factorisation incomplète de Cholesky IC($t$) de $\mathbf{A}$, où le paramètre $t$ contrôle le niveau de remplissage. Le but de ce préconditionnement préliminaire du système linéaire est de regrouper en partie les valeurs propres, afin que le sous-espace invariant associé aux plus petites valeurs propres ne soit pas de dimension trop grande.

Soit $\widehat{\mathbf{b}} = \mathbf{L}^{-1}\mathbf{b}$ et $\widehat{\mathbf{A}} = \mathbf{L}^{-1}\mathbf{A}\mathbf{L}^{-T}$ la matrice préconditionnée symétriquement avec IC($t$). La matrice $\widehat{\mathbf{A}}$ est symétrique définie positive et similaire à $\mathbf{M}_1^{-1}\mathbf{A}$. Dans la table 3.2, nous montrons l'effet de la factorisation incomplète de Cholesky (IC) sur la distribution des valeurs propres de la matrice préconditionnée $\widehat{\mathbf{A}}$. Nous notons d'abord le conditionnement élevé de la matrice originale BCSSTK14, de l'ordre de $10^{10}$, et celui de la matrice préconditionnée réduit à l'ordre de $10^2$. Comme mentionné précédemment, un spectre bien regroupé de la matrice préconditionnée avec un conditionnement bien réduit est une propriété souhaitable pour une convergence rapide des solveurs de Krylov et rend abordable les techniques de résolution considérées dans cette thèse, puisque seulement quelques composantes propres doivent être manipulées par une approche complémentaire.

| Matrice | Matrice Non Préconditionnée | | Matrice Préconditionnée | | | | Nombre de valeur propres au dessous de $\mu = \lambda_{\max}/10$ |
|---|---|---|---|---|---|---|---|
| | $\lambda_{\min}$ | $\lambda_{\max}$ | Seuil IC(t) | nnz(**L**) | $\lambda_{\min}$ | $\lambda_{\max}$ | |
| BCSSTK14 | 1 | $1.19 \cdot 10^{10}$ | $10^{-2}$ | 21, 197 | $1.41 \cdot 10^{-2}$ | 3.13 | 23 |
| BCSSTK15 | 1 | $6.53 \cdot 10^{9}$ | $10^{-3}$ | 109, 860 | $5.35 \cdot 10^{-3}$ | 1.77 | 4 |
| EDP1 | $2.92 \cdot 10^{-9}$ | 2.17 | $10^{-2}$ | 64, 566 | $1.79 \cdot 10^{-7}$ | 1.11 | 3 |
| EDP2 | $7.69 \cdot 10^{-5}$ | $1.00 \cdot 10^{7}$ | 0 | 643, 605 | $6.78 \cdot 10^{-10}$ | 2.03 | 16$^\star$ |

TAB. 3.2 – *Distribution de valeurs propres.*

$^\star$ Dans ce cas, nous prenons $\mu = \lambda_{\max}/2000$.

Le critère d'arrêt utilisé dans nos essais numériques est la norme relative du résidu $||\widehat{\mathbf{b}} - \widehat{\mathbf{A}}\mathbf{x}^{(i)}||_2/||\widehat{\mathbf{b}}||_2$. Dans tous ces tests, nous avons contrôlé cette norme relative du résidu jusqu'à la précision machine, quand c'est possible, pour illustrer et étudier le comportement numérique complet des méthodes. Tous les algorithmes ont été implémentés en **Matlab©** utilisant le format double précision. L'itéré initial est toujours $\mathbf{x}^{(0)} = 0$ et le second membre $\mathbf{b}$ est choisi de sorte que la solution $\mathbf{x}$ du système linéaire initial soit le vecteur $\mathbf{e} = [1, 1, \ldots, 1]$ et que $\widehat{\mathbf{b}} = \mathbf{L}^{-1}\mathbf{b}$. La valeur de coupure $\mu$ dans les filtres polynômiaux de Tchebycheff a été fixée à $\lambda_{\max}(\widehat{\mathbf{A}})/10$ quand cela est approprié, c-à-d. quand le nombre de valeurs propres de l'intervalle $]0, \mu]$ est raisonnablement petit. En particulier, dans le cas du problème de plus grande taille EDP2, nous avons pris comme premier niveau de préconditionnement $\mathbf{M}_1$ Cholesky Incomplet sans remplissage IC(0), et déplacé le paramètre $\mu$ plus près de zéro ($\mu = \lambda_{\max}(\widehat{\mathbf{A}})/2\,000$) pour incorporer seulement un petit ensemble de valeurs propres au deuxième niveau de préconditionnement.

Toutes les figures tracent la norme relative du résidu en fonction du nombre d'itérations (qui est égal au nombre de produits matrice-vecteur). En plus, nous dénoterons par "Classical CG" le gradient conjugué préconditionné par Cholesky Incomplet IC($t$).

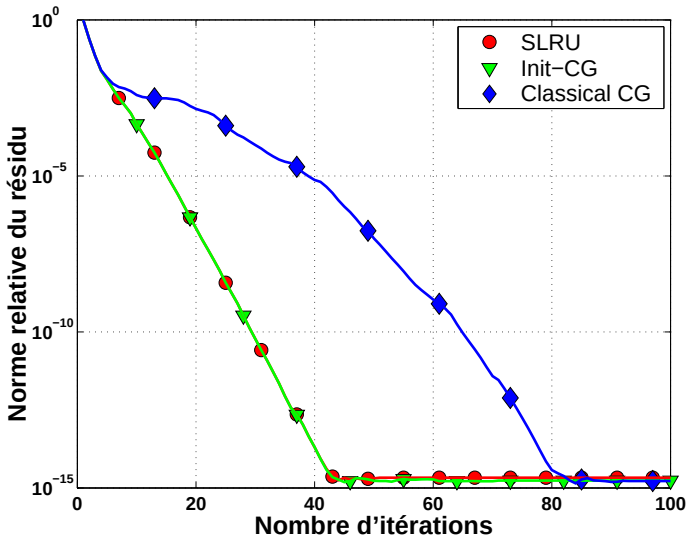
### 3.3.1 Comparaison des Différentes Techniques

Dans ce paragraphe, nous comparons la convergence des divers algorithmes décrits au § 3.2. Comme nous le verrons dans ce qui suit, les résultats numériques montrent que le comportement des différents algorithmes est le même. Ceci est particulièrement vrai quand la base approchée $\mathbf{W}(\varepsilon)$ est obtenue avec un niveau de filtrage $\varepsilon$ proche de la précision machine. Dans ce cas, le comportement numérique observé correspond à celui que nous obtiendrions en général avec un système linéaire où la plus petite valeur propre serait proche de la valeur de coupure $\mu$ (utilisée dans l'Algorithme 2.2 (Tchebycheff-PSF) pour extraire la base $\mathbf{W}$) et, par conséquent, avec un conditionnement réduit de l'ordre de $\lambda_{\max}(\mathbf{A})/\mu$.

La figure 3.2 montre, pour les problèmes test BCSSTK14 et EDP1, la convergence du gradient conjugué versus celle de Init-CG et SLRU obtenue avec une base $\mathbf{W}(\varepsilon)$ calculée avec un niveau du filtrage $\varepsilon$ égal à $10^{-16}$. La norme relative du résidu de la méthode du gradient conjugué standard tend à décroître très rapidement dans les premières itérations, puis décroît plus lentement dans une phase intermédiaire, là où des oscillations peuvent également apparaître, et stagne finalement. Par opposition à ceci, Init-CG et SLRU présentent des courbes de convergence qui coïncident presque totalement, sauf après stagnation.

Les algorithmes Def-CG et Proj-CG présentent aussi le même type de convergence, au moins dans la première partie de leurs historiques de conver-

*(a) Problème test BCSSTK14*




*(b) Problème test EDP1*

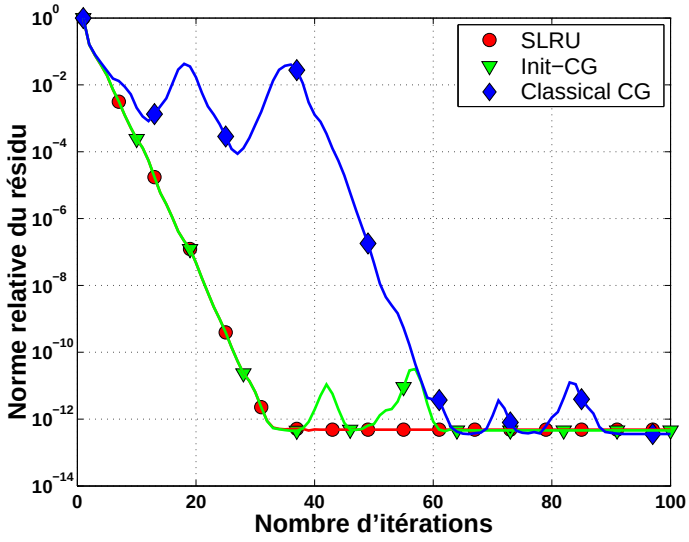
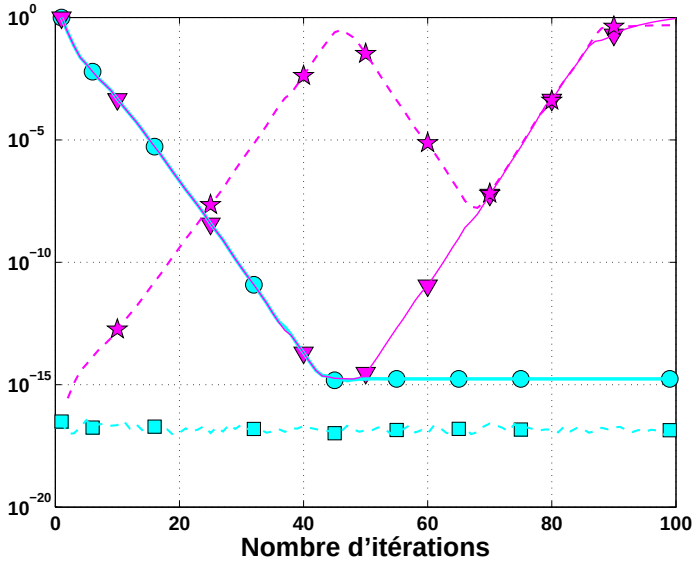
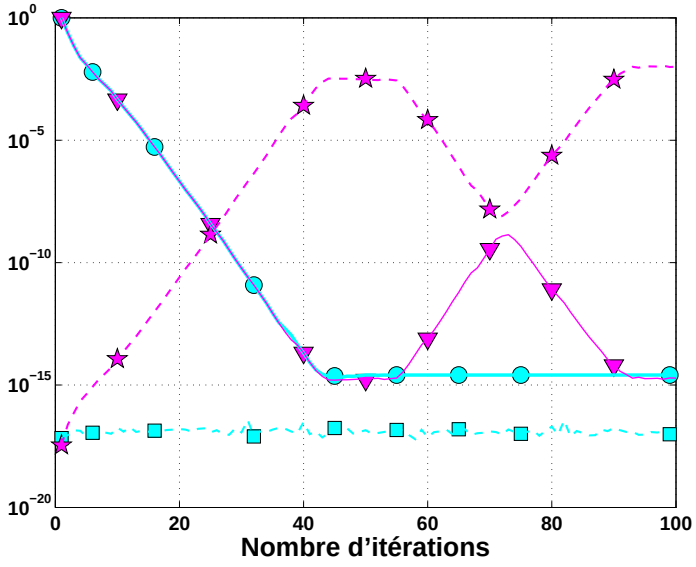



FIG. 3.2 – *Courbe de Convergence pour un niveau de filtrage $\varepsilon$ égal à* $10^{-16}$.

*(a) Courbe de Convergence de* Def-CG




*(b) Courbe de Convergence de* Proj-CG




★ ortho(k) sans correction de $\mathbf{r}^{(k)}$ □ ortho(k) avec correction de $\mathbf{r}^{(k)}$
▽ backward error sans correction de $\mathbf{r}^{(k)}$ ○ backward error avec correction de $\mathbf{r}^{(k)}$

FIG. 3.3 – *Orthogonalité de* $\mathbf{r}^{(k)}$ *par rapport à la base* $\mathbf{W}(\varepsilon)$ *où le niveau de filtrage* $\varepsilon$ *est égal à* $10^{-16}$ *et Courbes de convergence avec et sans correction de* $\mathbf{r}^{(k)}$ *sur le problème test BCSSTK14*

gence comme illustré par les résultats dans la figure 3.3. Cependant, nous observons que la convergence devient très instable numériquement et, pour certains systèmes, les courbes de convergence remontent même fortement : l'algorithme ne converge pas pour un grand nombre d'itérations (voir [77]). Théoriquement, les résidus $\mathbf{r}^{(k)}$ sont orthogonaux à $\mathbf{W}(\varepsilon)$. Cependant, en pratique, cette orthogonalité est perdue au cours des itérations. Ceci est illustré par la figure 3.3 qui montre, pour le problème test BCSSTK14, la courbe de la fonction suivante

$$\text{ortho(k)} = \max_{\text{j=1,2,...,p}} \left( \frac{\mathbf{w}^{(\text{j})^{\text{T}}} \mathbf{r}^{(\text{k})}}{||\mathbf{w}^{(\text{j})}||_2 ||\mathbf{r}^{(\text{k})}||_2} \right),$$

au cours des itérations dans les algorithmes Def-CG and Proj-CG. Les deux courbes de la fonction ortho(k) montrent que la perte d'orthogonalité est inversement proportionnelle à la courbe de convergence de la norme relative du résidu. Un remède pour récupérer l'orthogonalité est d'ajouter une étape de réorthogonalisation

$$\mathbf{r}^{(k)} = \mathbf{r}^{(k)} - \mathbf{W}\left(\mathbf{W}^T\mathbf{W}\right)^{-1}\mathbf{W}^T\mathbf{r}^{(k)} \tag{3.11}$$

tout de suite après avoir calculé $\mathbf{r}^{(k)}$ dans les algorithmes Def-CG et Proj-CG. Le coût de cette opération est $\mathcal{O}(p * n)$ par itération. On exécute à nouveau les deux algorithmes avec cette nouvelle correction, et on trace les courbes de convergence dans la figure 3.3, représentées par le sigle (○, et ligne pleine). Dans ce cas, les courbes de convergence de Def-CG et Proj-CG sont très similaires à celles des algorithmes Init-CG et SLRU comme dans la figure 3.2.

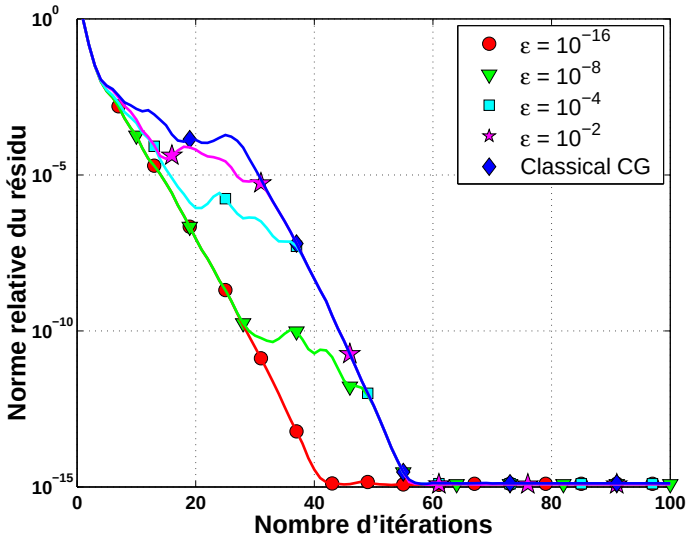
Pour conclure, quand la base approchée $\mathbf{W}(\varepsilon)$ est calculée avec une grande précision (c-à-d. avec un niveau de filtrage $\varepsilon$ très proche de la précision machine), Init-CG est l'algorithme de choix parce qu'il est moins cher, étant donné qu'il évite l'utilisation des projecteurs obliques à chaque itération.

### 3.3.2 Quand la base approchée W est moins précise

Ces différentes techniques de résolution peuvent cependant présenter un comportement (historique de convergence) très différent quand le calcul des vecteurs de la base approchée $\mathbf{W}(\varepsilon)$, est réalisé avec de plus grandes valeurs du niveau de filtrage $\varepsilon$.

La figure 3.4 montre, pour les problèmes tests BCSSTK15 et EDP1, l'impact du changement du niveau de filtrage $\varepsilon$ (quand on calcule la base $\mathbf{W}(\varepsilon)$) sur la convergence de l'Algorithme 3.1 (Init-CG). Quand le niveau de filtrage $\varepsilon$ est égal à $10^{-8}$, le phénomène des plateaux se produit encore, mais à des niveaux intermédiaires.

*(a) Le Problème test BCSSTK15*




*(b) Le Problème test EDP1*

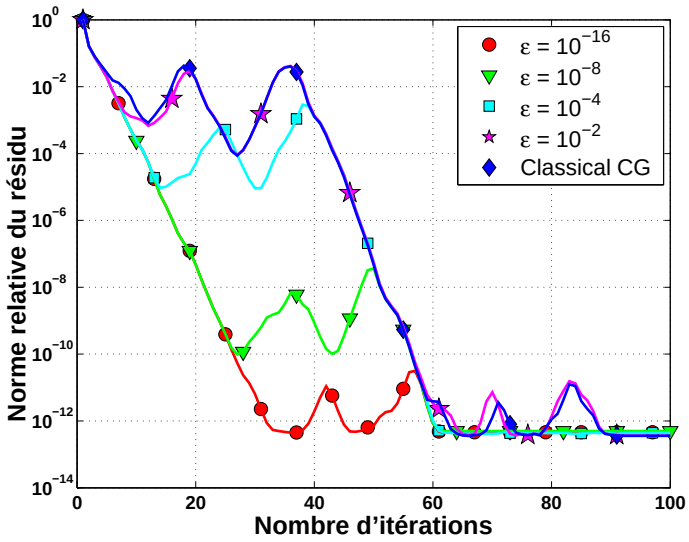



FIG. 3.4 – *Courbe de Convergence de* Init-CG *pour différentes valeurs du niveau de filtrage* $\varepsilon$.

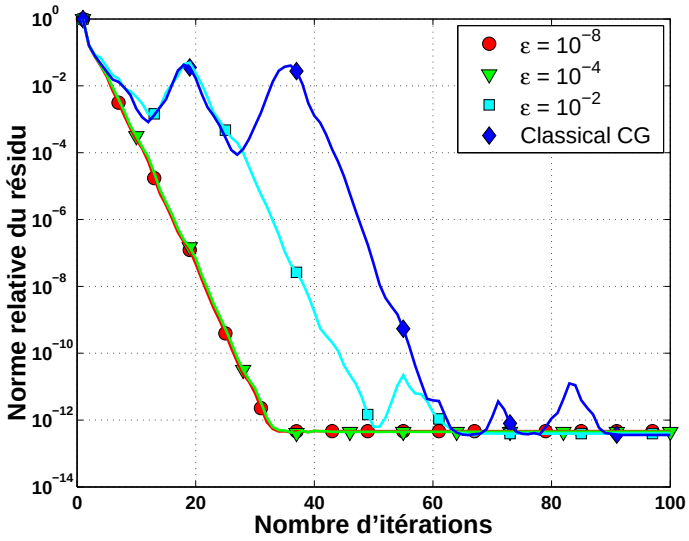
En effet, les courbes correspondantes pour l'Algorithme 3.1 (Init-CG) (gradient conjugué avec projection oblique initiale) se décomposent en deux parties différentes. La première partie correspond à une phase de convergence linéaire très semblable à celle observée dans la figure 3.2 lorsque la base $\mathbf{W}(\varepsilon)$ est calculée avec une très bonne précision. Dans la deuxième partie, la vitesse de convergence commence à être perturbée à un niveau intermédiaire de la norme relative du résidu, étroitement liée à la valeur du niveau de filtrage $\varepsilon$, et enfin, la courbe de convergence rejoint celle du gradient conjugué sur le système original.

Les raisons à cela sont que Init-CG fournit, au gradient conjugué, un vecteur résidu initial dont les plus grandes composantes propres sont dans le complémentaire orthogonal de la base $\mathbf{W}(\varepsilon)$, et dans l'espace de projection $[\mathbf{W}(\varepsilon)]$ ses composantes propres sont de l'ordre $\varepsilon$. Une fois que la norme du résidu est réduite au niveau du filtrage $\varepsilon$, le gradient conjugué se retrouve à nouveau avec une direction de Krylov dont toutes les composantes propres sont du même ordre de grandeur, et le comportement numérique devient alors identique à celui obtenu dans le cas d'un résidu initial non projeté et "*activant*" le spectre complet du système linéaire. Pour être efficace, il est clair que nous devrions alors arrêter la convergence de Init-CG quand la norme relative du résidu est inférieure au niveau de filtrage $\varepsilon$, avant d'atteindre le plateau intermédiaire. En outre, comme nous pouvons l'observer dans la figure 3.6 (a), quand la taille du système linéaire devient très importante, le phénomène des plateaux peut également apparître dans le comportement numérique de l'algorithme Init-CG , même avec une base très bien calculée $\mathbf{W}(10^{-16})$. Ceci est dû à la sensibilité numérique du projecteur oblique relativement à la taille du système linéaire [4].

Les courbes de convergence des algorithmes Def-CG et SLRU, représentées dans la figure 3.5 pour différents niveaux de filtrage $\varepsilon$, prouvent que ces deux algorithmes ont exactement le même comportement numérique dans tous les cas. Nous observons également que, si l'exactitude dans le calcul de la base approchée $\mathbf{W}(\varepsilon)$ est fortement détériorée (par exemple $\varepsilon = 10^{-2}$), alors la vitesse de la convergence peut changer légèrement. Par opposition à l'algorithme Init-CG, Def-CG et SLRU montrent cependant un comportement numérique très stable, semblable à celui observé dans les figures 3.2 et 3.3, même lorsque le niveau de filtrage $\varepsilon$ est dans l'intervalle $[10^{-8}, 10^{-4}]$ (voir la figure 3.5). C'est l'avantage d'établir à chaque itération du gradient conjugué une projection oblique avec la base de $\mathbf{W}(\varepsilon)$, ce qui permet de maintenir une séparation constante entre les composantes propres correspondant aux plus petites valeurs propres et les autres.

Dans la figure 3.6 (b), nous comparons le comportement numérique de Def-CG et de SLRU quand le niveau de filtrage $\varepsilon$ est grand. Nous pouvons remarquer que la convergence de Def-CG est un peu plus rapide que celle de l'algorithme SLRU. Sachant que la convergence de Def-CG et de Proj-CG

*(a) Courbe de convergence de* SLRU




*(b) Courbe de convergence de* Def-CG

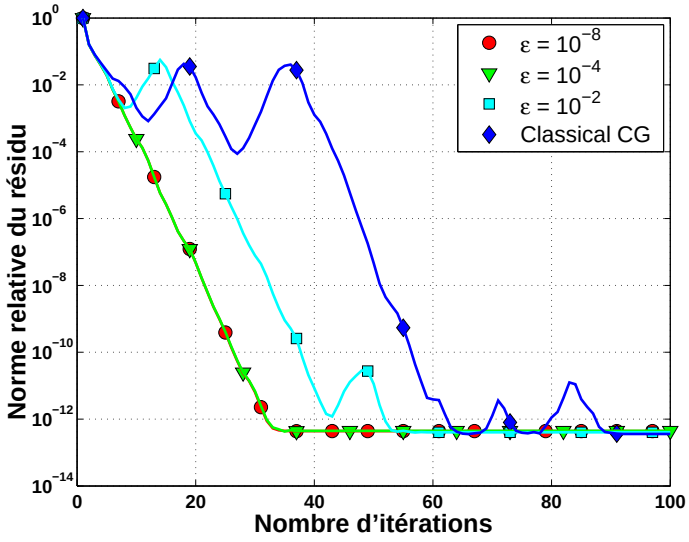
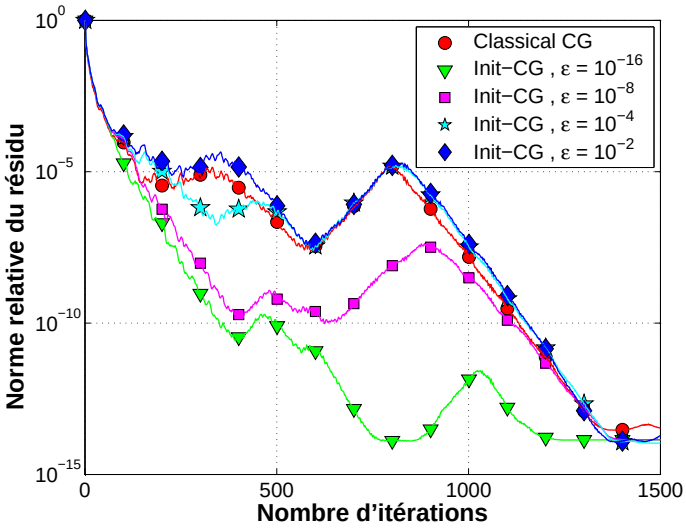



FIG. 3.5 – *Courbes de Convergence pour différents niveaux de filtrage $\varepsilon$ sur le problème test EDP1.*

*(a) Courbe de convergence de* Init-CG




*(b) Courbes de convergence de* SLRU *et* Def-CG

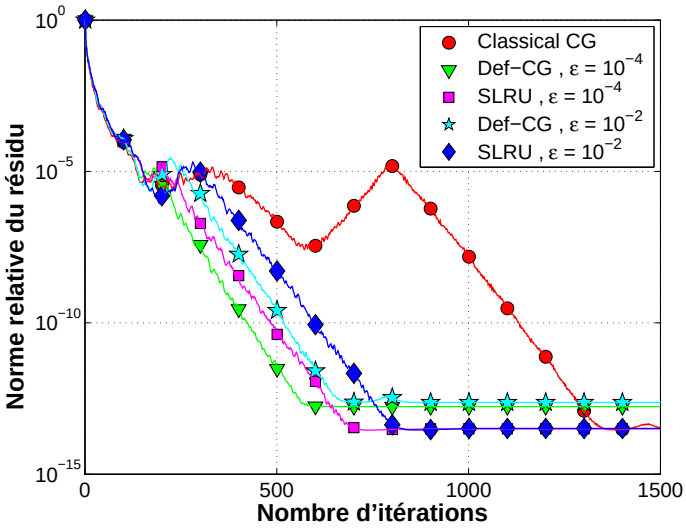



FIG. 3.6 – *Courbes de Convergence de* Init-CG, Def-CG *et* SLRU, *pour différentes valeurs du niveau du filtrage $\varepsilon$, sur le problème test EDP2.*

restent néanmoins très identiques, une explication possible de cette légère différence est que le conditionnement du système projeté est toujours inférieur à celui du système préconditionné par la correction spectrale de rang faible [51].

### 3.3.3 Combinaison de la Projection Oblique avec l'Itération de Tchebycheff

l'Algorithme 3.5 (Init-Tchebycheff) correspond fondamentalement à la juxtaposition de l'itération de Tchebycheff et d'une projection oblique, qui agissent indépendemment sur deux parties séparées du spectre de $\mathbf{A}$. À cet égard, cette technique peut être vue comme une méthode directe [4], si le sous-espace propre est calculé très exactement, ou comme un cycle à deux grilles, où le lisseur est défini par quelques étapes de Tchebycheff (voir § 3.2.5).

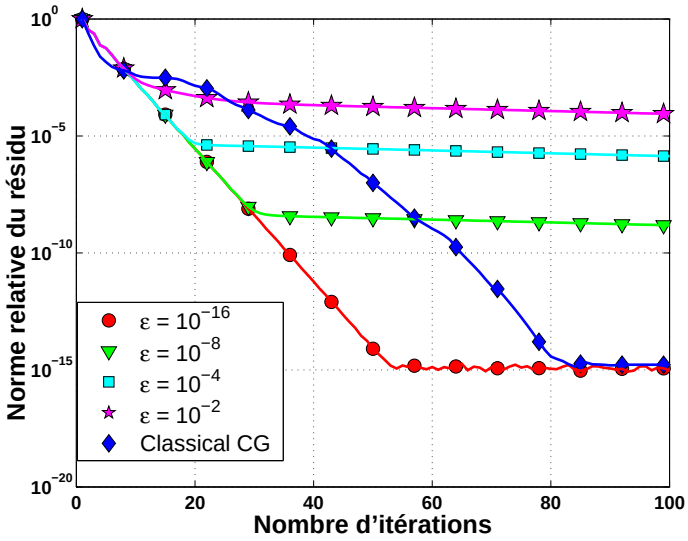
Dans l'intérêt de la comparaison avec les résultats obtenus pour les autres techniques de résolution de type Krylov, nous présentons, dans la figure 3.7, la norme relative du résidu calculée à chaque itération de Tchebycheff, et combinée directement avec la projection oblique finale. Naturellement, ce n'est pas de cette façon que l'Algorithme 3.5 (Init-Tchebycheff) est implémenté, puisque la projection oblique n'a besoin d'être appliquée qu'une seule fois, après la terminaison des itérations de Tchebycheff.

Dans la figure 3.7, nous illustrons l'effet du changement de niveau de filtrage $\varepsilon$ sur le comportement et la convergence de Init-Tchebycheff. Quand le niveau du filtrage $\varepsilon$ est égal à $10^{-16}$, Init-Tchebycheff se comporte comme les autres algorithmes, mais avec une petite différence sur le nombre d'itérations. C'est simplement dû au fait que les taux de convergence dans la méthode semi-itérative de Tchebycheff et dans la méthode du gradient conjugué ne sont pas les mêmes en général.

Cependant, pour de plus grandes valeurs du niveau du filtrage $\varepsilon$, nous pouvons observer que la courbe de convergence de l'algorithme Init-Tchebycheff est perturbée, avec une stagnation de la norme relative du résidu très proche du niveau du filtrage $\varepsilon$. Ceci peut s'expliquer par le fait que la base $\mathbf{W}(\varepsilon)$ a des composantes propres de l'ordre de $\varepsilon$ dans l'intervalle $[\mu, \lambda_{\max}]$. Cette information spectrale est incorporée de manière explicite dans la projection oblique, appliquée à l'étape finale, et perturbe les composantes propres dans le résidu final à l'ordre de ce niveau de filtrage. Pour une analyse plus détaillée de ce phénomène, nous nous référons à [4].

Il est également intéressant de mentionner que cette combinaison de la projection oblique avec les itérations de Tchebycheff exige seulement des produits "matrice-vecteur", ainsi que quelques mises à jour de vecteur, mais aucun produit scalaire, par opposition aux autres techniques de résolution du type Krylov. Cette remarque a également de l'importance dans le contexte

*(a) Le Problème test BCSSTK14*




*(b) Le Problème test EDP1*

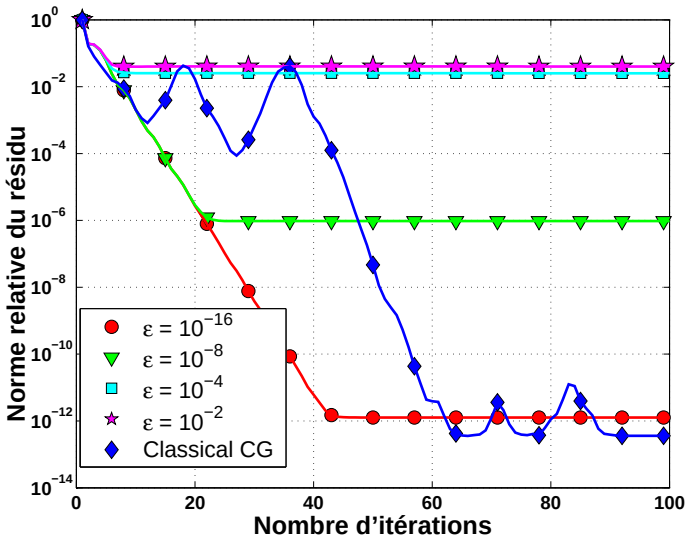



FIG. 3.7 – *Courbe de Convergence de* Init-Tchebycheff *pour différents niveaux de filtrage* $\varepsilon$.

du calcul parallèle, et en particulier dans les environnements à mémoire distribuée où le calcul des produits scalaires exige une attention particulière.

### 3.3.4 Méthode à deux grilles Algébrique

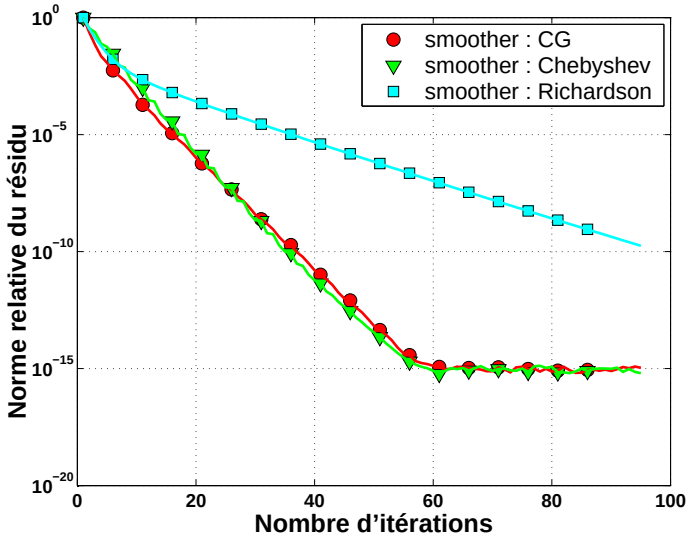
Une façon d'exploiter l'information spectrale pré-calculée est de définir un cycle à deux grilles en tant que solveur itératif stationnaire. Cette approche n'a de sens que lorsque le calcul de l'information spectrale n'est pas trop précis. Pour illustrer le comportement numérique de cet arrangement, nous exhibons dans la figure 3.8, la courbe de convergence d'un solveur algébrique à deux grilles en utilisant différents lisseurs. Plus précisément, pour ces expériences numériques, nous avons utilisé trois lisseurs différents : le gradient conjugué, l'itération de Tchebycheff, et la méthode de relaxation de Richardson.

Pour le facteur de relaxation de Richardson, nous prenons $\alpha_{opt} = 2/\big(\lambda_{\max} + \mu\big)$ afin de minimiser le rayon spectral de la matrice projetée (égal à $\max\big\{|1 - \alpha\,\lambda_{\max}|, |1 - \alpha\,\mu|\big\}$), en supposant que la correction sur la grille grossière supprime l'effet des valeurs propres dans l'intervalle $[\lambda_{\min}, \mu]$. Dans la figure 3.8, nous avons pris $\xi = 5$ comme pas de lissage (i.e. le nombre d'itérations dans le lisseur). Il apparaît que chacun de ces trois différents lisseurs exhibe un taux de convergence linéaire, et que Tchebycheff et le gradient conjugué ont les mêmes propriétés de lissage.

Pour être efficace, nous utilisons un nombre de pas de lissage $\xi$ aussi grand que possible afin de réduire le nombre de cycles, et par conséquent diminuer le nombre de projections obliques, tout en préservant le même taux de convergence. Dans la figure 3.9, nous montrons, pour les problèmes test BCSSTK14 and EDP1, la courbe de convergence du solveur algébrique à deux grilles quand le nombre de pas de lissage change. Le gradient conjugué est le lisseur considéré pour ces expériences. Nous observons que l'utilisation de grandes valeurs de $\xi$ peut retarder la convergence. Ces situations se produisent quand le lisseur a réussi à atténuer la plupart des hautes fréquences de l'erreur, et nécessite alors plus d'efforts pour réduire les basses fréquences. L'application d'une correction de grille grossière, à ce moment précis, est une manière efficace pour accélérer la convergence. Sur la base de cette observation, une heuristique pourrait être mise en oeuvre pour adapter automatiquement le nombre de pas de lissage.

L'algorithme Init-Tchebycheff peut également être vu comme une étape d'un cycle à deux grilles, où le lisseur correspond à l'itération de Tchebycheff. Dans la figure 3.10, nous comparons la courbe de convergence de Init-Tchebycheff, obtenue avec une base $\mathbf{W}(\varepsilon)$ calculée avec un niveau de filtrage $\varepsilon$ égal à $10^{-16}$, versus le solveur algébrique à deux grilles basé sur l'itération de Tchebycheff avec un nombre de pas de lissage $\xi = 5$, et une base $\mathbf{W}(\varepsilon)$ calculée avec un niveau $\varepsilon$ de $10^{-4}$.

*(a) Le problème test BCSSTK14*




*(b) Le problème test EDP1*

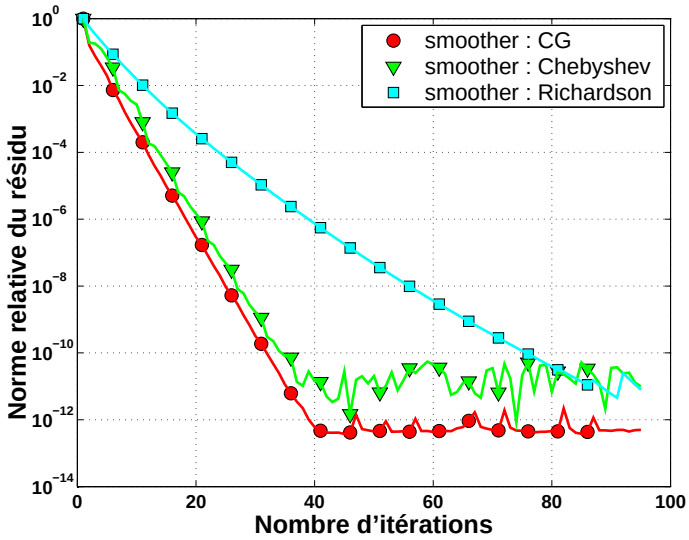
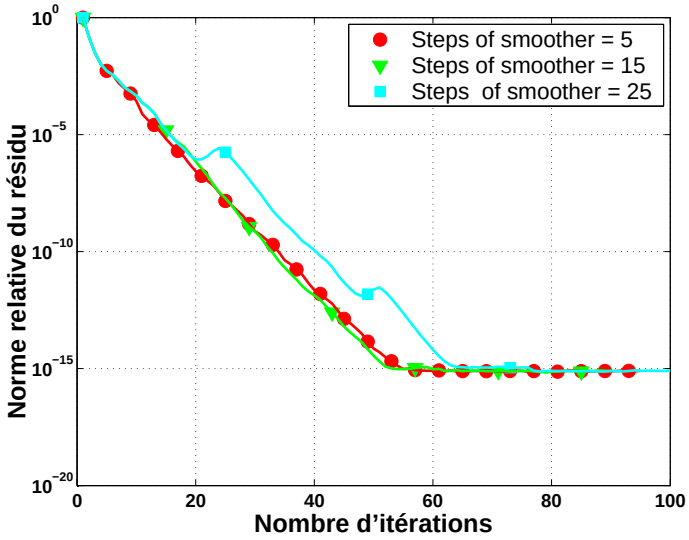



FIG. 3.8 – *Courbe de Convergence du solveur à deux grilles avec différents lisseurs, pour une base* $\mathbf{W}$ *initialement calculée avec un niveau de filtrage* $\varepsilon = 10^{-4}$.

*(a) Le Problème test BCSSTK14*




*(b) Le Problème test EDP1*

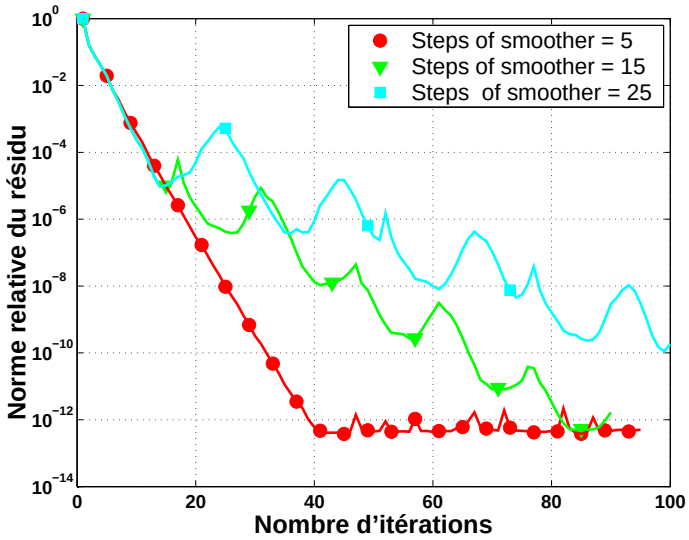
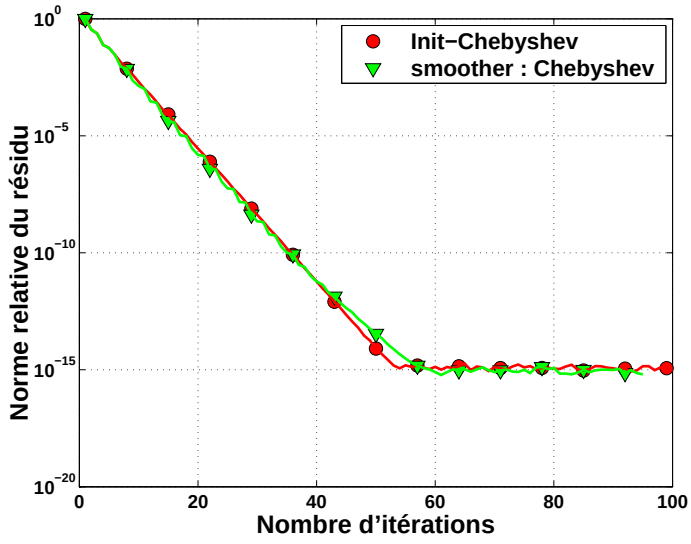



FIG. 3.9 – *Courbe de Convergence du cycle à deux grilles quand le nombre de pas de lissage du gradient conjugué change. Le niveau de filtrage de la base* $\mathbf{W}$ *est* $\varepsilon = 10^{-4}$.

*(a) Le Problème test BCSSTK14*




*(b) Le Problème test EDP1*

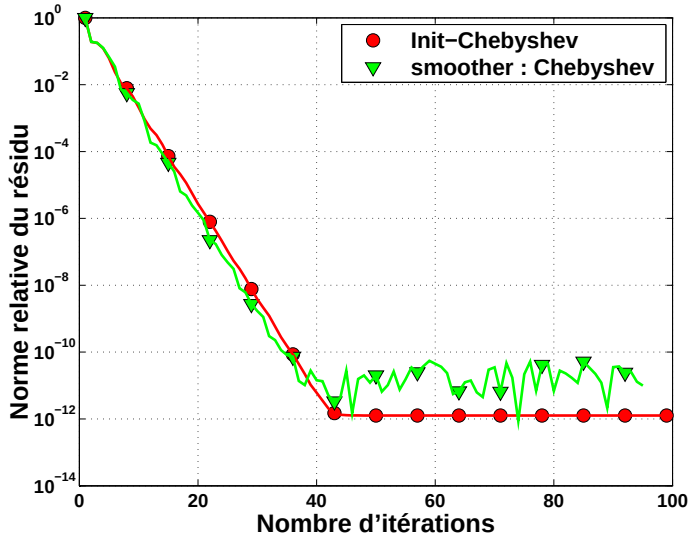



FIG. 3.10 – *Courbe de Convergence de* Init-Tchebycheff *avec une base* $\mathbf{W}(10^{-16})$ *versus le solveur à deux grilles avec Tchebycheff comme lisseur. Le pas de lissage est fixé à* $\xi = 5$*, et la base* $\mathbf{W}(\varepsilon)$ *est calculée avec un niveau de filtrage* $\varepsilon$ *égal à* $10^{-4}$*.*

Nous avions mentionné au § 3.3.3 que, quand la base $\mathbf{W}$ n'était pas calculée avec une très grande précision, la courbe de convergence de Init-Tchebycheff pouvait être perturbée, avec une stagnation de la norme relative du résidu à un niveau proche du niveau de filtrage $\varepsilon$. Le solveur algébrique à deux grilles, par contre, avec Tchebycheff comme lisseur, montre la même courbe de convergence que celle obtenue avec Init-Tchebycheff et une base $\mathbf{W}(\varepsilon)$ où $\varepsilon = 10^{-16}$. Ceci illustre l'avantage d'employer un arrangement du type cycle à deux grilles, notamment quand l'information spectrale n'est pas trop précise.

Dans le cas de nos essais numériques, l'algorithme à deux grilles algébrique exploitant le gradient conjugué comme lisseur, et avec une valeur appropriée de $\xi$ (c-à-d. pas trop grande), exhibe une courbe de convergence très semblable à celle obtenue avec SLRU quand $\varepsilon$ est assez petit (c-à-d. inférieur ou égal à $10^{-4}$). Par conséquent, le cycle à deux grilles devance SLRU en termes d'opérations flottantes, puisqu'il réalise moins de projections obliques au total que l'algorithme SLRU. Cette observation n'est pas tout le temps vraie, en particulier quand la base $\mathbf{W}(\varepsilon)$ est mal approchée (c-à-d. avec $\varepsilon = 10^{-2}$, dans nos expériences). Dans cette situation, la méthode algébrique à deux grilles ne converge plus, quel que soit le lisseur considéré, parce que la correction sur la grille grossière corrompt systématiquement l'itéré calculé par le lisseur. Par opposition à cela, nous rappelons que SLRU, Def-CG et Proj-CG arrivent toujours à tirer profit de l'espace propre, y compris dans ce cas là (voir la figure 3.5).

Enfin, il faut noter que le cycle à deux grilles avec une méthode stationnaire comme lisseur peut également être utilisé pour définir un préconditionneur pour le gradient conjugué, et nous référons à [11] pour plus de détails sur cette approche.

## 3.4 Considérations Pratiques

Dans cette section, nous analysons le coût des différents algorithmes introduits précédemment. Nous supposons que $p \ll n$, $p$ étant la dimension de la base approchée $\mathbf{W}$ du sous-espace invariant associé aux plus petites valeurs propres, et nous négligerons donc tous les termes ne contenant pas $n$. Tous les coûts seront évalués en nombre d'opérations flottantes. Le coût d'un produit matrice-vecteur avec $\mathbf{A}$ est donné par :

$$\mathcal{C}_{\mathbf{A}} \approx 2\,\mathsf{nnz}\,(\mathbf{A}) - n \tag{3.12}$$

où $\mathsf{nnz}\,(\mathbf{A})$ est le nombre d'éléments non nuls que la matrice $\mathbf{A}$ comporte.

Les différents algorithmes comparés dans ce chapitre réalisent tous une projection oblique du vecteur résidu initial $\mathbf{r}^{(0)}$ dans le sous-espace engendré par $\mathbf{W}$, requérant le même opérateur $\mathbf{W}\mathbf{A}_c^{-1}\mathbf{W}^T\mathbf{r}^{(0)}$, où $\mathbf{A}_c = \mathbf{W}^T\mathbf{A}\mathbf{W}$ est

une matrice dense d'ordre $p$. Le coût du calcul de $\mathbf{A}_c$ est :

$$\mathcal{C}_{\mathbf{A}_c} \approx n\,p^2 + 2\,\mathsf{nnz}\,(\mathbf{A})\,p - n\,p. \tag{3.13}$$

Nous notons que $\mathbf{A}_c$ n'est factorisée qu'une seule fois, au début de chaque algorithme, et ses facteurs seront utilisés pour n'importe quelle résolution avec $\mathbf{A}_c$.

Outre les coûts associés aux matrices $\mathbf{A}$ et $\mathbf{W}$, il faut aussi considérer les deux produits scalaires (`_DOT`) et les trois mises à jour de vecteur (`_AXPY`) réalisées à chaque itération du gradient conjugué. Le coût de ces opérations est donné par :

$$\mathcal{C}_{\texttt{\_AXPY}} \approx 2\,n \text{ et } \mathcal{C}_{\texttt{\_DOT}} \approx 2\,n. \tag{3.14}$$

L'algorithme Def-CG (voir § 3.2.2) (resp. Proj-CG (voir § 3.2.3)), calcule $\mathcal{Q}\,\mathbf{r}^{(k)}$ (resp. $\mathcal{P}\,\mathbf{r}^{(k)}$) (voir les formules (3.5) et (3.6)) à chaque itération. Cela peut être effectué grâce à des opérations de type BLAS2 pour $\mathbf{z}^{(k)} = (\mathbf{AW})^T\mathbf{r}^{(k)}$ et $\mathbf{r}^{(k)} - \mathbf{W}\mathbf{A}_c^{-1}\mathbf{z}^{(k)}$. Si nous considérons que les matrices $\mathbf{AW}$ et $\mathbf{A}_c$ sont déjà calculées et stockées au début de l'algorithme, le coût de ces opérations est le même que celui de l'application du préconditionneur SLRU, $\mathbf{M}_{\text{SLRU}}$, qui implante une mise à jour de rang $p$ de la matrice identitée de la forme $\mathbf{I} + \mathbf{W}\mathbf{A}_c^{-1}\mathbf{W}^T$. Le coût total de ces opérations, de type BLAS2, est donné par :

$$\mathcal{C}_{\text{Proj}} \approx 4\,(p+1)\,n. \tag{3.15}$$

Init-Tchebycheff ne fait pas partie des techniques de Krylov précédemment décrites. Il correspond à la juxtaposition de l'itération de Tchebycheff, qui requiert trois mises à jour de vecteur, un produit matrice-vecteur avec $\mathbf{A}$, et une projection oblique à la fin. Pour faciliter la comparaison, nous mettrons le coût de cette dernière projection oblique au début de l'algorithme.

La table 3.3 récapitule, pour chaque algorithme, le coût en opérations flottantes, avec un coût d'initialisation et un coût par itération.

| | Le coût en opérations flottantes | |
|---|---|---|
| | Initialisation | À chaque itération |
| Init-CG | $\mathcal{C}_{\mathbf{A}_c} + \mathcal{C}_{\text{Proj}} + \mathcal{C}_{\mathbf{A}} + 0$ | $\mathcal{C}_{\mathbf{A}} + 0 + 3\mathcal{C}_{\texttt{\_AXPY}} + 2\mathcal{C}_{\texttt{\_DOT}}$ |
| Def-CG | $\mathcal{C}_{\mathbf{A}_c} + \mathcal{C}_{\text{Proj}} + \mathcal{C}_{\mathbf{A}} + \mathcal{C}_{\text{Proj}}$ | $\mathcal{C}_{\mathbf{A}} + \mathcal{C}_{\text{Proj}} + 3\mathcal{C}_{\texttt{\_AXPY}} + 2\mathcal{C}_{\texttt{\_DOT}}$ |
| Proj-CG | $\mathcal{C}_{\mathbf{A}_c} + \mathcal{C}_{\text{Proj}} + \mathcal{C}_{\mathbf{A}} + 0$ | $\mathcal{C}_{\mathbf{A}} + \mathcal{C}_{\text{Proj}} + 3\mathcal{C}_{\texttt{\_AXPY}} + 2\mathcal{C}_{\texttt{\_DOT}}$ |
| SLRU | $\mathcal{C}_{\mathbf{A}_c} + \mathcal{C}_{\text{Proj}} + 0 + 0$ | $\mathcal{C}_{\mathbf{A}} + \mathcal{C}_{\text{Proj}} + 3\mathcal{C}_{\texttt{\_AXPY}} + 2\mathcal{C}_{\texttt{\_DOT}}$ |
| Init-Tchebycheff | $\mathcal{C}_{\mathbf{A}_c} + \mathcal{C}_{\text{Proj}} + \mathcal{C}_{\mathbf{A}} + 0$ | $\mathcal{C}_{\mathbf{A}} + 0 + 3\mathcal{C}_{\texttt{\_AXPY}} + 0$ |

TAB. 3.3 – *Coût en opérations flottantes des différentes méthodes.*

Une des différences principales entre ces algorithmes est que Def-CG, Proj-CG, et SLRU, exploitent l'opérateur de projection à chaque itération, tandis

que Init-CG ne l'emploie qu'une seule fois, au début (pour calculer l'itéré initial $\mathbf{x}^{(0)}$). L'algorithme à deux grilles est entre les deux, en ce sens qu'il effectue une projection oblique pour la correction de la grille grossière tous les $\xi$ pas de lissage (le lisseur pouvant être le gradient conjugué, Tchebycheff ou Richardson). À cet égard, Init-CG et Init-Tchebycheff sont les algorithmes les moins chers, puisqu'ils évitent l'utilisation des projecteurs obliques à chaque itération. Cependant, ces méthodes peuvent souffrir d'une mauvaise convergence quand le calcul de base $\mathbf{W}$ n'est pas précis, à l'opposé de Def-CG, Proj-CG, et SLRU qui restent stables numériquement.

Un dernier commentaire sur la mémoire utilisée par ces cinq techniques de résolution (Init-CG, Init-Tchebycheff, Def-CG, Proj-CG, et SLRU). Tous ces algorithmes requièrent la même quantité de stockage supplémentaire, de l'ordre de $2\,n\,p$, pour conserver $\mathbf{W}$ (la base approchée du sous-espace invariant associé aux plus petites valeurs propres de $\mathbf{A}$) ainsi que $\mathbf{AW}$.

### 3.4.1 Comparaison des Différentes Techniques en Terme d'Opérations Flottantes

Comme mentionné au début du § 3.3, un premier niveau de préconditionnement $\mathbf{M}_1$ peut également être employé pour mieux regrouper le spectre de la matrice d'itération. Le coût d'une multiplication de $\mathbf{M}_1$ par un vecteur est représenté par $\mathcal{C}_{\mathbf{M}_1}$. Puisque, dans nos expériences numériques, $\mathbf{M}_1$ est construit à l'aide d'une factorisation incomplète de Cholesky, $\mathcal{C}_{\mathbf{M}_1}$ peut être estimé comme

$$\mathcal{C}_{\mathbf{M}_1} \approx 4\,\mathsf{nnz}\,(\mathbf{L}) - 2\,n \tag{3.16}$$

où $\mathsf{nnz}\,(\mathbf{L})$ est le nombre d'éléments non nuls dans la matrice $\mathbf{L}$ issue de la factorisation incomplète de Cholesky. Le coût de ce premier niveau de préconditionnement $\mathbf{M}_1$ doit être incorporé dans les coûts en opérations flottantes détaillés dans la table 3.3. Ceci se résume à ajouter simplement $\mathcal{C}_{\mathbf{M}_1}$ à $\mathcal{C}_{\text{Proj}}$ (4.15) partout où il apparaît, ou explicitement à chaque itération pour Init-CG et Init-Tchebycheff.

Pour plus de simplicité dans la comparaison des coûts, nous exploitons une base $\mathbf{W}(\varepsilon)$ obtenue avec un bon niveau de filtrage $\varepsilon = 10^{-16}$, afin d'observer et comparer directement le comportement des algorithmes pour un intervalle étendu de la norme relative du résidu, sans "*rupture*" éventuelle de la pente de la convergence à cause d'un manque potentiel de précision dans le sous-espace invariant $[\mathbf{W}(\varepsilon)]$, ainsi que nous avons pu l'observer dans les sections précédentes.

Dans les figures 3.11 et 3.12, nous traçons la norme relative du résidu en fonction du nombre d'opérations flottantes. L'écart entre le gradient conjugué et les autres algorithmes, au début ces courbes, dénote le coût initial additionnel nécessaire pour incorporer la base approchée $\mathbf{W}(\varepsilon)$ et construire le

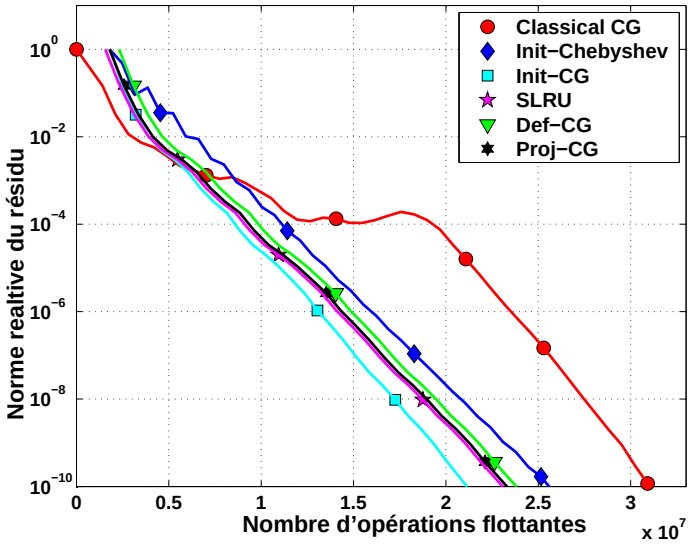
projecteur oblique associé. Selon la dimension de la base approchée $\mathbf{W}(\varepsilon)$, cet écart est plus ou moins grand. Par exemple, cette dimension est d'ordre 3–4 dans la figure 3.11 ($a$) et la figure 3.12 ($a$), et d'ordre 23 dans la figure 3.12 ($b$). Nous pouvons également observer les différences entre les pentes des diverses courbes de convergence, qui sont plus marquées en particulier entre Init-CG ou Init-Tchebycheff et les trois autres (Def-CG, Proj-CG et SLRU). En effet, Init-CG et Init-Tchebycheff exploitent la projection oblique seulement au début, alors que les autres le font à chaque itération.

Quand la base approchée $\mathbf{W}(\varepsilon)$ est de dimension petite – ce qui est d'ailleurs souhaitable dans le cadre de l'utilisation de la factorisation spectrale partielle Tchebycheff-PSF (voir Chapitre 2) –, il est clair que le surcoût lors de l'initialisation n'est pas extrêmement important, et peut même être compensé en très peu d'itérations. Par contre, quand la base est trop grande, comme nous pouvons l'observer pour les algorithmes Def-CG, Proj-CG et SLRU dans la figure 3.12 ($b$), où la dimension de $\mathbf{W}(\varepsilon)$ est égale à 23, le coût de l'utilisation du projecteur oblique est très important, pas seulement au début mais aussi dans les itérations, et les gains en termes de vitesse de convergence peuvent ne pas être assez bons pour compenser ce surcoût. Dans ce cas là, seules des techniques comme Init-CG ou Init-Tchebycheff peuvent être compétitives en termes de nombre d'opérations, puisqu'elles n'effectuent qu'une seule projection oblique.

Pour plus de détails, nous mentionnons aussi que, dans le cas du problème test BCSSTK14 dans la figure 3.12, le gradient conjugué préconditionné nécessite 65 itérations pour atteindre $10^{-10}$ dans la norme relative du résidu, alors que les autres techniques de Krylov requièrent 51 itérations, pour la même norme du résidu, et lorsque la valeur de coupure $\mu = \lambda_{\max}/40$ et la base approchée $\mathbf{W}(\varepsilon)$ est de dimension 3, et 30 itérations quand $\mu = \lambda_{\max}/10$ et $\mathbf{W}(\varepsilon)$ est de dimension 23. Pour le même problème test, l'algorithme Init-Tchebycheff converge au bout de 74 itérations dans le premier cas, et 36 itérations dans l'autre, toujours pour atteindre la même norme relative de l'erreur. Il y a un certain compromis à trouver, en fonction du problème, entre le coût directement lié à la taille de la base approchée $\mathbf{W}(\varepsilon)$ et le conditionnement réduit résultant de la matrice projetée ou préconditionnée.

Comme mentionné au Chapitre 2, le paramètre $\mu$ divise le spectre de la matrice en deux sous-ensembles et fixe la dimension $p$ du sous-espace invariant calculé par la factorisation spectrale partielle Tchebycheff-PSF. Ceci dépend bien évidemment de la distribution actuelle de valeurs propres de la matrice $\widehat{\mathbf{A}}$, qui est spécifique à chaque problème. Une conséquence directe du choix de $\mu$ peut être observée en particulier dans le comportement de l'algorithme Init-Tchebycheff, qui est explicitement lié au conditionnement réduit $\lambda_{\max}/\mu$, par opposition aux autres techniques de Krylov où la convergence dépend intimement de la distribution complète des valeurs propres.

*(a) Le Problème test BCSSTK15*




*(b) Le Problème test EDP1*

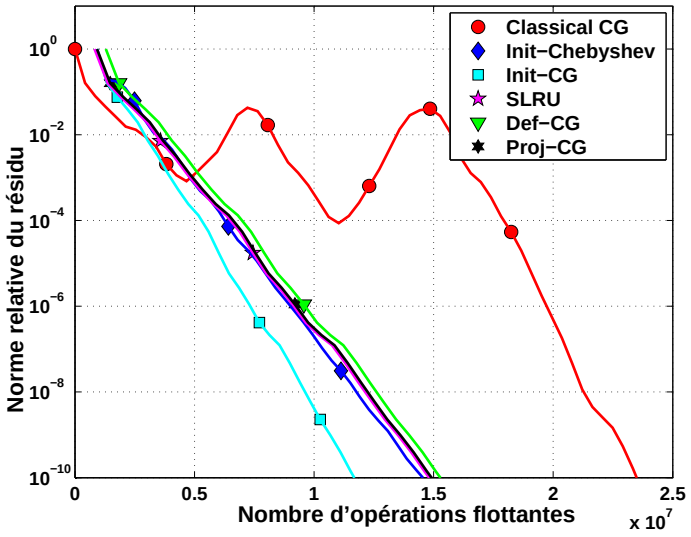
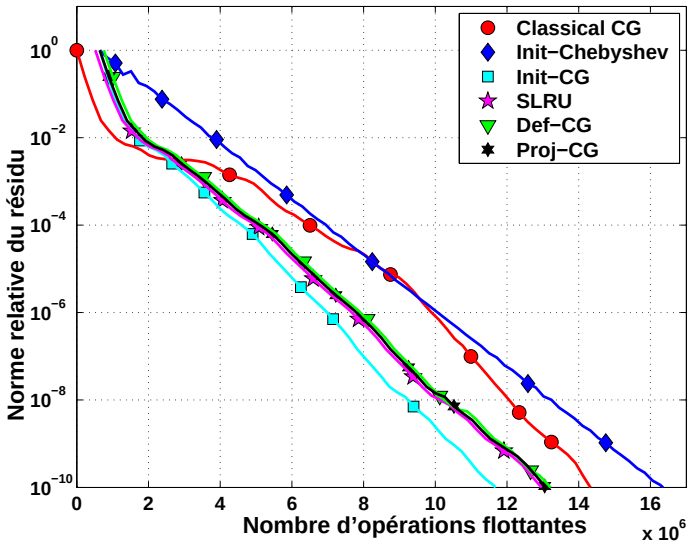



FIG. 3.11 – *Coût des différentes méthodes en opérations flottantes, pour une base* $\mathbf{W}$ *calculée avec un niveau de filtrage* $\varepsilon$ *égal à* $10^{-16}$.

*(a) Valeur de coupure* $\mu = \lambda_{\max}/40$




*(b) Valeur de coupure* $\mu = \lambda_{\max}/10$

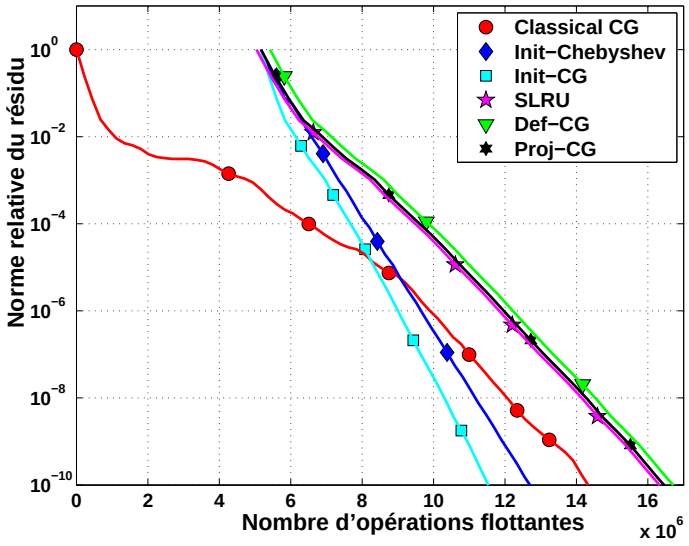



FIG. 3.12 – *Coût des différentes méthodes en opérations flottantes, dans le cas du problème test BCSSTK14, pour diverses valeurs de coupure* $\mu$ *et une base* $\mathbf{W}$ *calculée dans chaque cas avec un niveau de filtrage* $\varepsilon$ *égal à* $10^{-16}$.

Par exemple, dans la figure 3.12, nous pouvons observer que Init-Tchebycheff n'est pas un bon candidat – en dépit de la petite taille de la base approchée $\mathbf{W}(\varepsilon)$ – quand le paramètre $\mu$ est placé à $\lambda_{\max}/40$ (et la base $\mathbf{W}(\varepsilon)$ est de dimension 3), par opposition au cas où $\mu = \lambda_{\max}/10$ (avec une base $\mathbf{W}(\varepsilon)$ de dimension 23).

Pour finir, les différentes courbes de convergence prouvent que les trois techniques Def-CG, Proj-CG et SLRU ont exactement le même comportement. Une analyse fine basée sur les valeurs données dans la table 3.3 peut aider à conclure que SLRU est un peu moins coûteux que les deux autres. Cependant, si le calcul de la base approchée $\mathbf{W}(\varepsilon)$ est très précis (indépendamment de la dimension de la base $\mathbf{W}(\varepsilon)$), Init-CG reste l'algorithme de choix par rapport à ces trois techniques, et par rapport aussi à Init-Tchebycheff qui sera pénalisé par une vitesse de convergence plus lente.

### 3.4.2 Gains Potentiels

Le but de ce paragraphe est de comparer, en terme de coût, les gains potentiels qu'il est possible de réaliser avec les diverses techniques de résolution introduites au § 3.2. Nous considérons maintenant le cas de la multirésolution, à savoir que nous voudrions résoudre une séquence de systèmes linéaires avec la même matrice mais avec différents second membres calculés séquentiellement. Nous illustrons comment les gains obtenus à chaque résolution linéaire, peuvent réduire considérablement le coût total de la multirésolution. Pour le pré-calcul de la base orthogonale approchée $\mathbf{W}(\varepsilon)$, nous avons utilisé l'Algorithme 2.2 (Tchebycheff-PSF), décrit dans le Chapitre 2.

Les tables 3.4 et 3.5, illustrent, dans le cas des problèmes test BCSSTK15, EDP1 et EDP2, les gains potentiels résultant de l'utilisation des divers algorithmes. Le critère d'arrêt pour ces tests, portant sur le résidu relatif, est donné par :

$$||\widehat{\mathbf{b}} - \widehat{\mathbf{A}}\mathbf{x}^{(i)}||_2/||\widehat{\mathbf{b}}||_2 \leq 10^{-10}.$$

Pour chaque niveau de filtrage $\varepsilon$, nous indiquons, dans ces tables, le nombre d'opérations flottantes en `Mflop` (ou en `Gflop` pour le problème test EDP2) et les gains potentiels (Gain) en % . Pour atteindre $10^{-10}$ dans la norme relative du résidu, le gradient conjugué nécessite 57 itérations dans le cas du problème test EDP1, avec un coût de 23.76 `Mflop`, 45 itérations dans le cas du problème test BCSSTK15, avec un coût de 31.62 `Mflop`, et, dans le cas du problème anisotrope EDP2, 1 060 itérations avec un coût de 6.31 `Gflop`. En particulier, si nous considérons le problème test EDP1 avec une base calculée avec un niveau de filtrage $\varepsilon = 10^{-16}$, Init-CG converge en 27 itérations avec un coût de 11.50 `Mflop`. Ceci rapporte une réduction de 47%, comparé au gradient conjugué sans cette information spectrale et un gain de 52%. Le coût du pré-calcul de la base $\mathbf{W}$ sera amorti en quelques résolutions consécutives avec Init-CG. Cet aspect sera en fait analysé en détail au chapitre suivant,

où nous introduirons aussi une autre technique pour le pré-calcul de la base orthogonale approchée $\mathbf{W}(\varepsilon)$, alternative à l'Algorithme 2.2 (Tchebycheff-PSF), mais permettant de résoudre en même temps un des systèmes linéaires de la séquence.

Quand le calcul de la base approchée $\mathbf{W}(\varepsilon)$ est très précis (c-à-d. pour un niveau de filtrage $\varepsilon = 10^{-16}$), le surcoût du pré-calcul de la base est contrebalancé en quelques résolutions consécutives, et ceci avec n'importe quel solveur itératif. Comme mentionné précédemment, Init-CG semble être l'algorithme de choix, car c'est celui permettant les meilleurs gains.

Init-Tchebycheff est un algorithme un peu particulier, puisqu'il ne fait pas partie des techniques du type Krylov. Cependant, quand le niveau de filtrage $\varepsilon$ est égal à $10^{-16}$, il se comporte comme les autres algorithmes, mais avec une légère différence sur le nombre d'itérations et d'opérations flottantes. Ceci est dû à la différence entre le taux de convergence de l'itération de Tchebycheff et celui du gradient conjugué. Par contre, quand l'information spectrale n'est pas extrêmement précise, c-à-d. quand le niveau de filtrage $\varepsilon$ est dans l'intervalle $[10^{-8}, 10^{-4}]$, le fait d'utiliser la projection oblique une seule fois au début est insuffisant pour permettre à l'itération de Tchebycheff de réduire l'erreur suffisamment.

Le cycle à deux grilles, proposé au § 3.2.6, peut être une alternative pour tirer bénéfice de l'information spectrale disponible, malgré sa faible qualité numérique, sans avoir pour autant à réaliser une projection oblique à chaque itération, comme dans les algorithmes Def-CG, Proj-CG , ou SLRU. La table 3.5 indique le coût de cette approche, avec l'itération de Tchebycheff comme lisseur. Nous changeons également le nombre de pas de lissage dans chaque cycle, selon la qualité numérique de la base $\mathbf{W}(\varepsilon)$, afin de réduire au minimum le nombre de corrections de grille grossière (c-à-d. le nombre de projections obliques) versus le nombre total d'itérations.

Les divers résultats numériques montrent que, lorsqu'une grande exactitude est exigée dans la résolution de plusieurs systèmes linéaires en séquence, une base pré-calculée avec un niveau de filtrage $\varepsilon$ supérieur à $10^{-2}$ ne devrait pas être employée. La meilleure stratégie pour aborder cette situation est de calculer une base avec un niveau de filtrage précis, $\varepsilon \in [10^{-16}, 10^{-8}[$, et d'exécuter l'algorithme Init-CG ou le cycle à deux grilles avec le gradient conjugué comme lisseur. Nous notons également que, si la base $\mathbf{W}(\varepsilon)$ est suffisamment précise (par exemple $\varepsilon = 10^{-4}$), l'utilisation des trois autres techniques de Krylov (Def-CG, Proj-CG et SLRU) reste en général très efficace, et que le cycle à deux grilles peut être une bonne alternative.

| | Le Problème test EDP1<br>Le gradient conjugué nécessite 57 itérations avec un coût de 23.76 Mflop | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Init-CG | | Def-CG | | Proj-CG | | SLRU | | Init-Tchebycheff | |
| $\varepsilon$ | Mflop | Gain % | Mflop | Gain % | Mflop | Gain % | Mflop | Gain % | Mflop | Gain % |
| $10^{-16}$ | **11.50** | **52** % | **14.70** | **39** % | **14.57** | **39** % | **14.47** | **39** % | **14.20** | **40** % |
| $10^{-8}$ | **11.50** | **52** % | **14.70** | **39** % | **14.57** | **39** % | **14.47** | **39** % | — | — |
| $10^{-4}$ | 24.23 | -2 % | **14.70** | **39** % | **14.57** | **39** % | **14.47** | **39** % | — | — |
| $10^{-2}$ | 24.23 | -2 % | **19.77** | **17** % | **20.64** | **13** % | 24.40 | -3 % | — | — |
| | Le Problème test BCSSTK15<br>Le gradient conjugué nécessite 45 itérations avec un coût de 31.62 Mflop | | | | | | | | | |
| | Init-CG | | Def-CG | | Proj-CG | | SLRU | | Init-Tchebycheff | |
| $\varepsilon$ | Mflop | Gain % | Mflop | Gain % | Mflop | Gain % | Mflop | Gain % | Mflop | Gain % |
| $10^{-16}$ | **20.41** | **35** % | **22.27** | **29** % | **22.19** | **29** % | **21.96** | **30** % | **24.78** | **21** % |
| $10^{-8}$ | **20.41** | **35** % | **22.27** | **29** % | **22.19** | **29** % | **21.96** | **30** % | **30.96** | **2** % |
| $10^{-4}$ | 31.66 | -0.1 % | **22.27** | **29** % | **22.19** | **29** % | **21.96** | **30** % | — | — |
| $10^{-2}$ | 31.66 | -0.1 % | **24.62** | **22** % | **24.54** | **22** % | **24.31** | **23** % | — | — |

| | Le Problème test EDP2<br>Le gradient conjugué nécessite 1060 itérations avec un coût de 6.31 Gflop | | | | | |
|---|---|---|---|---|---|---|
| | Init-CG | | Def-CG | | SLRU | |
| $\varepsilon$ | Gflop | Gain % | Gflop | Gain % | Gflop | Gain % |
| $10^{-16}$ | **1.70** | **73** % | **4.80** | **23** % | **4.86** | **23** % |
| $10^{-8}$ | **2.04**$^{\star}$ | **67** % | **4.95** | **22** % | **5.32** | **15** % |
| $10^{-4}$ | 6.42 | -1 % | **6.05** | **4** % | 7.06 | -11 % |
| $10^{-2}$ | 6.42 | -1 % | 7.63 | -21 % | 9.02 | -43 % |

TAB. 3.4 – *Coût des différentes approches pour différents niveaux de filtrage $\varepsilon$. Les itérations du gradient conjugué sont stoppées quand la norme relative du résidu est inférieure à $10^{-10}$.*

$^{\star}$ Dans ce cas, nous utilisons l'algorithme algébrique à deux grilles avec le gradient conjugué comme lisseur et une seule correction de la grille grossière.

| | Cycle à deux grilles avec Tchebycheff comme lisseur | | | | | |
|---|---|---|---|---|---|---|
| | Problème test EDP1 | | | Problème test BCSSTK15 | | |
| $\varepsilon$ | $\xi$ | Mflop | Gain % | $\xi$ | Mflop | Gain % |
| $10^{-8}$ | 15 | **15.56** | **34** % | 20 | **25.52** | **19** % |
| $10^{-4}$ | 5 | **17.03** | **28** % | 15 | **27.25** | **14** % |

TAB. 3.5 – *Coût, en opérations flottantes, du cycle à deux grilles avec Tchebycheff comme lisseur.*

## 3.5 Conclusions

Dans ce chapitre, nous avons comparé diverses méthodes de résolution exploitant une certaine information spectrale partielle extraite de la matrice **A**. L'idée générale, commune à toutes ces techniques, est d'employer la base orthonormale **W** du sous-espace invariant associé aux plus petites valeurs propres de **A**, en vue d'établir une projection sur le complémentaire orthogonal de **W**. Pour le pré-calcul de cette base, nous avons employé la factorisation spectrale partielle (Tchebycheff-PSF) [4, 28, 29], basée sur les filtres polynômiaux de Tchebycheff et le processus de Lanczos, et présentée au Chapitre 2.

Parmi les méthodes de résolution, nous avons considéré la version du gradient conjugué déflaté [77], qui consiste à augmenter l'espace de Krylov, généré dans les itérations du gradient conjugué, avec la base **W**, et à projeter, à chaque itération, les vecteurs de Krylov sur le complémentaire orthogonal de cette base **W**. Une alternative consiste à appliquer l'algorithme du gradient conjugué au système projeté [51]. La troisième approche est basée sur un préconditionnement spectral [10], qui vise à translater de 1 la position des plus petites valeurs propres de la matrice du système. Ces trois techniques de Krylov ont permis une convergence rapide dans le gradient conjugué, avec des taux de convergence linéaires, très identiques, et pouvant être obtenus même quand le calcul de la base approchée **W** n'est pas trop précis.

Nous avons également expérimenté trois autres approches, à savoir l'algorithme du gradient conjugué classique et les méthodes itératives de Tchebycheff et de Richardson, les trois ont été combinées avec un itéré initial obtenu par une projection oblique du résidu de départ dans le complémentaire orthogonal de **W**. Nous avons observé que, quand le calcul de la base approchée **W** est précis, ces techniques présentent également le même comportement numérique, avec des taux de convergence linéaires, mais qui diffèrent légèrement, puisque leurs propriétés numériques intrinsèques sont différentes.

Dans tous les cas, les taux de convergence résultants, sont directement liés au paramètre de coupure $\mu$, qui correspond à la plus petite valeur propre de $\mathbf{A}$ qui n'est pas incluse dans l'ensemble de valeurs propres approchées par la base $\mathbf{W}$. Plus précisément, ces taux de convergence correspondent à ceux obtenus avec des matrices ayant un conditionnement réduit égal à $\lambda_{\max}(\mathbf{A})/\mu$. Enfin, en dépit de leurs taux de convergence linéaires relativement plus élevés, les techniques itératives de Tchebycheff et de Richardson présentent l'avantage additionnel de n'employer aucun produit scalaire dans leurs itérations. Ceci est un aspect très important pour les implémentations parallèles dans les environnements à mémoire distribuée.

Quand la base calculée $\mathbf{W}$ est invariante avec une grande précision, il est suffisant de décomposer la solution en deux parties, la première s'obtenant simplement par projection oblique du second membre dans l'espace engendré par $\mathbf{W}$. Les techniques du type Init-CG et Init-Tchebycheff sont alors les algorithmes de choix, parce qu'elles évitent l'utilisation des projecteurs obliques à chaque itération, ce qui les rend beaucoup moins chères. Par contre, quand le calcul de la base $\mathbf{W}$ n'est pas trop précis, l'utilisation d'une projection oblique une seule fois, au début, ne suffit pas pour assurer un taux de convergence linéaire. En particulier, quand la base approchée est trop pauvre, aucune amélioration n'est observée. Par opposition à ceci, les trois premières techniques de type Krylov, qui calculent à chaque itération une projection oblique avec la base $\mathbf{W}$, n'ont pas ce problème de mauvaise convergence, et montrent un comportement numérique très stable dans la plupart des cas. Dans des situations intermédiaires, c-à-d. quand la base pré-calculée est suffisamment précise, nous pouvons aussi considérer un cycle à deux grilles, pour tirer profit de l'information spectrale sans établir une correction de grille grossière (projection oblique) à chaque itération.

Les coûts en opérations flottantes analysés dans § 3.4.2, ont également montré que, lorsque le spectre de la matrice d'itération est très bien regroupé et que la base $\mathbf{W}$ est de petite dimension, les gains peuvent être plutôt intéressants, et le coût de pré-calcul de cette base peut être rapidement compensé dans le cadre de la résolution d'une séquence de systèmes linéaires avec la même matrice mais différents second membres.

Dans le chapitre suivant nous étudierons la combinaison du gradient conjugué avec les filtres polynômiaux de Tchebycheff appliqués à une partie du spectre de la matrice, comme préconditionneurs. Cette approche vise à mettre le gradient conjugué dans une meilleure situation de convergence. L'idée est de construire directement une base de Krylov de dimension petite, riche en informations relatives aux vecteurs propres associés aux plus petites valeurs [32].

# Chapitre 4

# Approche Hybride Combinant les Filtres de Tchebycheff et le Gradient Conjugué pour la Résolution de Systèmes Linéaires à Second Membres Multiples

L'un des solveurs itératifs les plus puissants pour résoudre les systèmes linéaires symétriques définis positifs est l'algorithme du gradient conjugué de Hestenes et de Stiefel [37], plus particulièrement quand il est combiné avec un certain préconditionnement [16]. De nombreux problèmes de calcul scientifique réclament la résolution de plusieurs systèmes linéaires impliquant la même matrice mais des second membres différents. Ici nous proposons une nouvelle approche basée sur une combinaison de la méthode du gradient conjugué avec des filtres polynomiaux de Tchebycheff comme préconditionneurs. Notre technique consiste à appliquer ces filtres polynomiaux à une partie seulement du spectre de la matrice de coefficient, en ciblant quelques propriétés spécifiques de convergence de la méthode du gradient conjugué. Nous montrerons que notre préconditionneur met un grand nombre de valeurs propres proches de 1 et ne dégrade pas la distribution des plus petites valeurs propres. Ceci permet au Gradient Conjugué, ainsi préconditionné, de construire directement une base de Krylov de dimension petite, très riche en ce qui concerne les plus petites valeurs propres et les vecteurs propres associés.

Une conséquence directe est que cette information spectrale peut alors être exploitée de manière très simple, avec un petit effort supplémentaire, pour une multirésolution linéaire, c-à-d. la résolution d'une séquence de systèmes linéaires avec la même matrice mais différents second membres fournis séquentiellement. Nous illustrerons avec quelques essais numériques, l'efficacité de cette approche.

## 4.1 Introduction

L'algorithme du gradient conjugué préconditionné est l'une des techniques les plus puissantes pour résoudre des systèmes linéaires de la forme (1.1), où $\mathbf{A}$ est une matrice creuse, symétrique définie positive de $\mathbb{R}^{n\times n}$ [16]. L'utilisation de préconditionneurs polynomiaux est une solution attrayante, déjà considérée par plusieurs auteurs [39, 72]. Cela permet de transformer le spectre de la matrice préconditionnée, avec une distribution des valeurs propres plus favorable à la méthode du gradient conjugué, soit en réduisant fortement le conditionnement du système préconditionné, soit en regroupant l'ensemble des valeurs propres dans un nombre très restreint d'intervalles.

Pour l'itération de Tchebycheff, par exemple, l'évaluation de $\lambda_{\min}(\mathbf{A})$ et $\lambda_{\max}(\mathbf{A})$, la plus petite et la plus grande valeur propre, est indispensable. Une borne supérieure peut souvent être obtenue par des techniques très simples, telles que les disques de Gershgörin [72]. Par contre, il est plus difficile d'estimer la plus petite valeur propre. Saad [72] a proposé une technique de préconditionnement polynomial qui exige seulement une borne supérieure pour la plus grande valeur propre, alors que la borne $\lambda_{\min}(\mathbf{A}) = 0$ est employée comme plus petite valeur propre de $\mathbf{A}$. Sa technique est basée sur les polynômes de moindres carrés associés à la famille de fonctions-poids de Jacobi [82]. Une autre manière d'éviter le calcul de $\lambda_{\min}(\mathbf{A})$ et de $\lambda_{\max}(\mathbf{A})$ est d'employer directement le polynôme construit par le gradient conjugué lui-même comme préconditionneur. Cette approche a été étudiée par O'Leary [55]. L'inconvénient de cette technique est que le système préconditionné n'a pas la garantie d'être défini positif. Golub et Kent [30] ont proposé une technique basée sur des moments modifiés permettant d'estimer les valeurs propres extrêmes du système linéaire. Ashby et al. [6] ont observé, sur une variété d'exemples numériques, que l'efficacité des préconditionneurs polynomiaux de Tchebycheff et de moindres carrés dépend de la distribution des valeurs propres de la matrice $\mathbf{A}$.

Ici, nous proposons une approche différente, qui vise à placer le gradient conjugué dans un mode particulier, "*en mode amas*", où le conditionnement du système linéaire n'est pas tellement réduit mais son spectre en grande partie regroupé autour de 1. Notre préconditionneur, appelé ChebFilter, consiste à appliquer les filtres polynomiaux de Tchebycheff à une partie seulement du spectre de la matrice d'itération, afin de décaler un grand nombre de va-

leurs propres du système linéaire près de 1, sans dégrader la distribution des plus petites valeurs propres. Avec cette technique de préconditionnement, le gradient conjugué exhibe toujours une convergence avec un certain nombre de plateaux, sans pouvoir entrer dans un mode linéaire directement. Ces plateaux, qui correspondent à la combinaison du mauvais conditionnement avec des amas de valeurs propres aux extrêmes, sont cependant considérablement réduits en taille (en comparaison de ce qui peut être observé sans cette technique de préconditionnement). Le Gradient Conjugué, ainsi préconditionné, est alors à même de construire directement une base de Krylov de dimension petite, très riche en ce qui concerne les plus petites valeurs propres et les vecteurs propres associés. Cette caractéristique est fondamentale, car elle permet alors d'exploiter ces espaces de Krylov de manière très simple, sans effort supplémentaire pour trier ou sélectionner l'information spectrale qu'ils contiennent, afin de résoudre en particulier une séquence de systèmes linéaires avec la même matrice mais différents second membres fournis séquentiellement.

Une des motivations de l'ensemble de ce travail a été, en effet, de proposer une alternative à l'utilisation des méthodes directes pour la résolution d'une séquence de systèmes linéaires, avec une même matrice (symétrique définie positive mais mal conditionnée) et différents second membres. Pour des second membres simultanés, les méthodes de Krylov par blocs [53] fournissent une réponse appropriée. Pour une séquence de second membres qui ne changent que très légèrement, une idée toute simple est d'employer la dernière solution comme vecteur de départ pour la résolution suivante dans cette séquence. Dans le cas où seulement un second membre est disponible à la fois, la méthode de Fischer [27], le gradient conjugué déflaté (deflated CG) [77], ou la méthode hybride de Simoncini et al. [80] peuvent être considérées. La méthode de Fischer recherche, dans l'ordre, une solution dans l'espace engendré par les directions de Krylov construites lors du calcul de la solution précédente. Ceci est utile quand les vecteurs de solution sont corrélés. Dans la méthode du gradient conjugué déflaté, seules les premières directions de Krylov précédemment construites sont conservées et employés pour mettre à jour le sous-espace approximativement invariant exploité pour accélérer la résolution des divers systèmes. Cette approche est relativement efficace en termes de calcul et d'occupation mémoire, mais la convergence vers un sous-espace invariant est lente et la réduction du nombre des itérations reste relativement modeste. La méthode hybride de Simoncini et al. [80] est très efficace seulement quand les second membres partagent la même information spectrale.

Dans ce chapitre, l'idée principale est de résoudre le premier système avec la méthode du gradient conjugué combiné avec les filtres de Tchebycheff comme préconditionneurs, et exploiter ensuite la base de Krylov résultante, générée lors de cette première résolution, pour projeter le vecteur résidu

initial avant d'exécuter l'algorithme du gradient conjugué classique pour la résolution des systèmes suivants. Nous pouvons également exploiter cette base de Krylov dans des techniques de déflation ainsi que des algorithmes à deux-Grilles (voir Chapitre 3).

Comme déjà mentionné, cette approche diffère des autres car elle permet de construire une base de Krylov qui contient de manière concentrée toute l'information spectrale associée aux plus petites valeurs propres, et ne nécessite pas de traitement ou de tri a posteriori, puisque cette base de Krylov est de taille extrêmement réduite relativement à la dimension de l'espace invariant associé aux valeurs propres petites dans la matrice $\mathbf{A}$. L'information contenue dans cette base de Krylov est également assez indépendante du second membre du système linéaire utilisé pour la construire. En effet, les filtres polynomiaux de Tchebycheff agissent uniformément sur les composantes propres, indépendamment des vecteurs résidus. Par opposition à cela, la méthode du gradient conjugué détermine à chaque itération le polynôme optimal associé à ces composantes propres, afin minimiser l'erreur en norme $\mathbf{A}$.

Ce chapitre s'articule comme suit. Nous consacrerons le paragraphe 4.2 à la description de notre préconditionneur, ainsi qu'à la discussion de certaines de ses propriétés. Au § 4.3, nous illustrerons le comportement et l'efficacité numérique de notre approche sur un ensemble de problèmes modèles provenant de la discrétisation par éléments finis des problèmes d'Équations aux Dérivées Partielles (EDP), du type équation de diffusion. Au § 4.5, nous analyserons en détail la complexité algorithmique de notre technique et nous étudierons comment peut-elle être employée efficacement pour résoudre des systèmes linéaires avec la même matrice mais à second membres multiples. En conclusion, nous rappellerons les résultats nouveaux et donnerons une vue des perspectives de ce travail au § 4.6.

## 4.2 Filtres Polynomiaux de Tchebycheff comme Préconditionneurs

Ce paragraphe est dédié à la description de la combinaison du gradient conjugué avec les filtres polynomiaux de Tchebycheff comme préconditionneurs, et que nous appellerons ChebFilterCG. Pour décrire ceci en détails, nous introduisons d'abord la décomposition propre de la matrice $\mathbf{A}$ :

$$\mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T = \mathbf{U}_1\mathbf{\Lambda}_1\mathbf{U}_1^T + \mathbf{U}_2\mathbf{\Lambda}_2\mathbf{U}_2^T$$

où le spectre de $\mathbf{A}$ est divisé en deux parties, $\mathbf{\Lambda}_1$ matrice diagonale formée des valeurs propres $\lambda_i$ de $\mathbf{A}$ inférieures au paramètre $\mu \in ]0, \lambda_{\max}[$, $\mathbf{U}_1$ matrice rectangulaire dont les colonnes correspondent aux vecteurs propres

orthonormalisés, $\mathbf{\Lambda}_2$ et $\mathbf{U}_2$ étant les matrices complémentaires correspondantes. Soit $\mathbf{x}^{(0)} \in \mathbb{R}^n$ un itéré initial pour la solution de (1.1), et soit $\mathbf{r}^{(0)} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(0)}$ le vecteur résidu associé. Nous introduisons alors le vecteur filtré

$$\mathbf{w}_f = \mathcal{F}_m(\mathbf{A})\,\mathbf{r}^{(0)} = \mathbf{U}_1\mathcal{F}_m(\mathbf{\Lambda}_1)\mathbf{U}_1^T\,\mathbf{r}^{(0)} + \mathbf{U}_2\mathcal{F}_m(\mathbf{\Lambda}_2)\mathbf{U}_2^T\,\mathbf{r}^{(0)}, \tag{4.1}$$

où $\mathcal{F}_m$ est le polynôme de degré $m$ donné par

$$\mathcal{F}_m(\lambda) = \frac{T_m(\Theta_\mu(\lambda))}{T_m(\Theta_\mu(0))}, \tag{4.2}$$

où $T_m$ est le polynôme de Tchebycheff de degré $m$, et $\Theta_\mu$ la similitude qui transforme l'intervalle $[\mu, \lambda_{\max}]$ en $[-1, 1]$ (avec $\Theta_\mu(\mu) = 1$ et $\Theta_\mu(\lambda_{\max}) = -1$).

Pour des valeurs données de $\mu$, $\lambda_{\max}(\mathbf{A})$ et $\varepsilon$, nous pouvons fixer le degré $m$ de $T_m$ tel que $1/|T_m(\Theta_\mu(0))| < \varepsilon$ et par conséquent $||\mathcal{F}_m(\lambda)||_\infty < \varepsilon$ sur $[\mu, \lambda_{\max}]$. En utilisant la formule (4.1), on peut écrire

$$||\mathbf{U}_2^T\,\mathbf{w}_f||_2 \leq ||\mathcal{F}_m(\mathbf{\Lambda}_2)||_2\,||\mathbf{U}_2^T\,\mathbf{r}^{(0)}||_2 \leq \varepsilon\,||\mathbf{U}_2^T\,\mathbf{r}^{(0)}||_2\,. \tag{4.3}$$

L'équation (4.3) montre explicitement comment le filtre matriciel de Tchebycheff en $\mathbf{A}$ appliqué au vecteur résidu $\mathbf{r}^{(0)}$ pourra atténuer les composantes propres associés à toutes les valeurs propres de l'intervalle $[\mu, \lambda_{\max}(\mathbf{A})]$ au niveau $\varepsilon$ relativement aux autres. Le nombre d'itérations de Tchebycheff permettant de réduire ces composantes propres à un certain niveau du filtrage $\varepsilon$, est directement lié au taux de convergence de l'itération de Tchebycheff sur l'intervalle $[\mu, \lambda_{\max}]$ (voir, e.g., [30], [36, page 47]), qui dépend seulement du rapport $\lambda_{\max}/\mu$.

Notre préconditionneur polynomial est résumé dans l'Algorithme 4.1 (appelé ChebFilter).

Cet algorithme correspond à l'application du polynôme matriciel de Tchebycheff en $\mathbf{A}$, réduisant les composantes propres de $\mathbf{r}^{(0)}$ associées à toutes les valeurs propres de l'intervalle $[\mu, \lambda_{\max}(\mathbf{A})]$ au niveau du filtrage $\varepsilon$, pour calculer le vecteur résidu $\mathbf{w}_f = \mathcal{F}_{m+1}(\mathbf{A})\,\mathbf{r}^{(0)}$ et l'approximation correspondante $\mathbf{x}_f$ telle que $\mathbf{b} - \mathbf{A}\mathbf{x}_f = \mathbf{r}_f$. Les étapes 6.ii et 6.iii de l'Algorithme 4.1 sont liées par la relation suivante :

$$\begin{aligned} \mathbf{w}_f = \mathbf{w}^{(m+1)} = \mathcal{F}_{m+1}(\mathbf{A})\,\mathbf{r}^{(0)} &= \frac{T_{m+1}(\Theta_\mu(\mathbf{A}))\,\mathbf{r}^{(0)}}{T_{m+1}(d_\mu)} \\ &= \frac{1}{\sigma_{m+1}}\,T_{m+1}\Big(d_\mu\,\mathbf{I} - \alpha_\mu\,\mathbf{A}\Big)\,\mathbf{r}^{(0)} \end{aligned}$$

où nous notons $d_\mu = \dfrac{\lambda_{\max} + \mu}{\lambda_{\max} - \mu}$, $\alpha_\mu = \dfrac{2}{\lambda_{\max} - \mu}$ et $\sigma_m = T_m(d_\mu)$ pour tout $m \geq 0$. L'étape 6.iv peut être également remplacée par la relation

$$\mathbf{w}_f = \mathbf{w}^{(m+1)} = 2\,\frac{\sigma_m}{\sigma_{m+1}}\Big(d_\mu\mathbf{w}^{(m)} - \alpha_\mu\mathbf{A}\mathbf{w}^{(m)}\Big) - \frac{\sigma_{m-1}}{\sigma_{m+1}}\,\mathbf{w}^{(m-1)}, \tag{4.4}$$

**Algorithme 4.1 : ChebFilter**

$[\mathbf{w}_f, \mathbf{x}_f] = \mathsf{ChebFilter}\ \big(\mathbf{A}, \mathbf{b}, \mu, \lambda_{\max}(\mathbf{A}), \mathbf{x}^{(0)}, \varepsilon\big)$

**Début**

1. $\alpha_\mu = \dfrac{2}{\lambda_{\max}(\mathbf{A}) - \mu}$ et $d_\mu = \dfrac{\lambda_{\max}(\mathbf{A}) + \mu}{\lambda_{\max}(\mathbf{A}) - \mu}$
2. Choix de $\mathbf{x}^{(0)}$
3. $\mathbf{x}_f = \mathbf{x}^{(0)}$, et $\mathbf{w}_f = \mathbf{b} - \mathbf{A}\mathbf{x}_f$
4. $\mathbf{y} = \mathbf{x}_f,\ m = 1,\ \sigma_0 = 1$ et $\sigma_1 = d_\mu$
5. $\mathbf{x}_f = \mathbf{x}_f + \dfrac{\alpha_\mu}{d_\mu}\,\mathbf{w}_f$ et $\mathbf{w}_f = \mathbf{b} - \mathbf{A}\mathbf{x}_f$
6. **Tant que** $1/\sigma_m \geq \varepsilon$ **Faire**
   - i. $\sigma_{m+1} = 2\,d_\mu\,\sigma_m - \sigma_{m-1}$
   - ii. $\mathbf{p} = 2\,\dfrac{\sigma_m}{\sigma_{m+1}}\Big(d_\mu \mathbf{x}_f + \alpha_\mu \mathbf{w}_f\Big) - \dfrac{\sigma_{m-1}}{\sigma_{m+1}}\,\mathbf{y}$
   - iii. $\mathbf{y} = \mathbf{x}_f$ et $\mathbf{x}_f = \mathbf{p}$
   - iv. $\mathbf{w}_f = \mathbf{b} - \mathbf{A}\mathbf{x}_f$
   - v. $m = m + 1$
7. **FinTantque**

**Fin**

qui correspond à la récurrence à trois termes définissant l'itération de Tchebycheff calculant $\mathbf{w}^{(m+1)}$ en fonction de $\mathbf{w}^{(m)}$ et $\mathbf{w}^{(m-1)}$ pour tout $m \geq 1$, où $\mathbf{w}^{(0)}$ est le vecteur $\mathbf{r}^{(0)}$, et où $\mathbf{w}^{(1)}$ est égal à $\big(\mathbf{I} - \frac{\alpha_\mu}{d_\mu}\mathbf{A}\big)\mathbf{r}^{(0)}$.

Soit $\mathbf{x}^{(i)}$ l'itéré à l'étape $i$ du gradient conjugué préconditionné et $\mathbf{r}^{(i)} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(i)} = \mathbf{A}(\mathbf{x} - \mathbf{x}^{(i)})$ le résidu associé. L'application du préconditionneur ChebFilter consiste à résoudre approximativement le système linéaire $\mathbf{A}\mathbf{z}^{(i)} = \mathbf{r}^{(i)}$ tel que le résidu final est égal à $\mathbf{r}^{(i)} - \mathbf{A}\mathbf{z}^{(i)} = \mathcal{F}_m(\mathbf{A})\,\mathbf{r}^{(i)}$. Ceci peut également être formulé comme suit :

$$\mathbf{z}^{(i)} = \mathbf{A}^{-1}\big(\mathbf{I} - \mathcal{F}_m(\mathbf{A})\big)\mathbf{r}^{(i)} = \mathbf{M}^{-1}\mathbf{r}^{(i)}. \tag{4.5}$$

Nous mentionnons que notre préconditionneur polynomial effectue un nombre fixe d'étapes pour des valeurs données de $\lambda_{\max}(\mathbf{A})$, $\mu$ et $\varepsilon$, indépendamment du second membre.

À l'étape $i$, le gradient conjugué préconditionné est caractérisé par :

$$\begin{cases} \mathbf{x}^{(i)} \in & \{\mathbf{x}^{(0)}\} + \mathcal{K}_i(\mathbf{M}^{-1}\mathbf{A}, \mathbf{z}^{(0)}) \quad \text{avec} \quad \mathbf{z}^{(0)} = \mathbf{M}^{-1}\mathbf{b} \\ \mathbf{r}^{(i)} \perp & \mathcal{K}_i(\mathbf{M}^{-1}\mathbf{A}, \mathbf{z}^{(0)}). \end{cases}$$

$\mathcal{K}_i(\mathbf{M}^{-1}\mathbf{A}, \mathbf{z}^{(0)})$ est l'espace de Krylov de dimension $i$.

$$\begin{aligned} \mathcal{K}_i\big(\mathbf{M}^{-1}\mathbf{A}, \mathbf{z}^{(0)}\big) &= \big[\mathbf{z}^{(0)}, \ldots, \big(\mathbf{M}^{-1}\mathbf{A}\big)^{i-1}\mathbf{z}^{(0)}\big] \\ &= \big[\mathbf{z}^{(0)}, \ldots, \big(\mathbf{I} - \mathcal{F}_m(\mathbf{A})\big)^{i-1}\mathbf{z}^{(0)}\big] \\ &= \big[\mathbf{z}^{(0)}, \ldots, \big(\mathcal{F}_m(\mathbf{A})\big)^{i-1}\mathbf{z}^{(0)}\big], \end{aligned} \tag{4.6}$$

étant donné que $\mathbf{A}^{-1}\mathcal{F}_m(\mathbf{A}) = \mathcal{F}_m(\mathbf{A})\mathbf{A}^{-1}$, et donc que $\mathbf{M}^{-1}\mathbf{A} = \mathbf{A}^{-1}\big(\mathbf{I} - \mathcal{F}_m(\mathbf{A})\big)\mathbf{A} = \big(\mathbf{I} - \mathcal{F}_m(\mathbf{A})\big)\mathbf{A}^{-1}\mathbf{A}$. L'équation (4.6) montre explicitement que les directions de recherche du gradient conjugué préconditionné par les filtres polynomiaux de Tchebycheff sont obtenues dans un espace de Krylov filtré.

Comme $\mathbf{A}$ est symétrique définie positive, alors $\mathbf{M}^{-1} = \mathbf{A}^{-1}\big(\mathbf{I} - \mathcal{F}_m(\mathbf{A})\big)$ l'est aussi. En effet, puisque $\mathbf{A}^{-1}$ et $\mathcal{F}_m(\mathbf{A})$ commutent, il est facile de voir que $\mathbf{M}^{-1}$ est symétrique. En utilisant la décomposition propre $\mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$, nous pouvons écrire $\mathbf{M}^{-1} = \mathbf{U}\mathbf{\Lambda}^{-1}\big(\mathbf{I} - \mathcal{F}_m(\mathbf{\Lambda})\big)\mathbf{U}^T$, où $\mathbf{\Lambda}^{-1}\big(\mathbf{I} - \mathcal{F}_m(\mathbf{\Lambda})\big)$ est une matrice diagonale définie positive puisque $\mathcal{F}_m(\lambda) \in [-\varepsilon, 1[$ pour tout $\lambda$ dans $]0, \lambda_{\max}]$. En plus, nous pouvons voir que les matrices $\mathbf{M}^{-1}\mathbf{A}$ et $\mathbf{A}\mathbf{M}^{-1}$ ont les mêmes valeurs propres, qui sont celles de la matrice $(\mathbf{I} - \mathcal{F}_m(\mathbf{A}))$. Par conséquent, les valeurs propres de $\mathbf{M}^{-1}\mathbf{A}$ sont données par $1 - \mathcal{F}_m(\lambda_i)$, avec $\lambda_i$, $i \in \{1, \ldots, n\}$ valeurs propres de $\mathbf{A}$.

Un aspect très important dans l'utilisation des filtres de Tchebycheff comme préconditionneurs est qu'ils agissent uniformément sur l'intervalle $[\mu, \lambda_{\max}(\mathbf{A})]$, indépendamment du second membre. Par opposition à cela, le gradient conjugué détermine à chaque itération le polynôme optimal pour minimiser l'erreur courante en norme $\mathbf{A}$, qui dépend du second membre.

Les étapes de la combinaison du gradient conjugué avec les filtres polynomiaux comme préconditionneurs sont résumées dans l'Algorithme 4.2, qu'on appellera ChebFilterCG.

Algorithme 4.2 : ChebFilterCG

**Début**

1. choix de $\mathbf{x}^{(0)}$ et $\mathbf{r}^{(0)} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(0)}$
2. **Pour** $m = 1, 2, \ldots$ **Faire**
   - i. $[\mathbf{y}_f, \mathbf{z}^{(m-1)}] = \mathsf{ChebFilter}\ \big(\mathbf{A}, \mathbf{r}^{(m-1)}, \mu, \lambda_{\max}(\mathbf{A}), \mathbf{y}, \varepsilon\big)$
   - ii. $\rho_{m-1} = {\mathbf{r}^{(m-1)}}^T \mathbf{z}^{(m-1)}$
   - iii. **Si** $m = 1$
     - $\mathbf{p}^{(1)} = \mathbf{z}^{(0)}$
     - **Sinon**
     - $\beta_{m-1} = \rho_{m-1}/\rho_{m-2}$
     - $\mathbf{p}^{(m)} = \mathbf{z}^{(m-1)} + \beta_{m-1}\mathbf{p}^{(m-1)}$
   - iv. **FinSi**
   - v. $\mathbf{q}^{(m)} = \mathbf{A}\mathbf{p}^{(m)}$
   - vi. $\alpha_m = \rho_{m-1}/({\mathbf{p}^{(m)}}^T \mathbf{q}^{(m)})$
   - vii. $\mathbf{x}^{(m)} = \mathbf{x}^{(m-1)} + \alpha_m \mathbf{p}^{(m)}$
   - viii. $\mathbf{r}^{(m)} = \mathbf{r}^{(m-1)} - \alpha_m \mathbf{q}^{(m)}$
   - ix. Vérifier la Convergence et Continuer si nécessaire
3. **FinPour**

**Fin**

### 4.2.1 Préconditionnement des Algorithmes ChebFilter et ChebFilterCG

La version préconditionnée de l'Algorithme 4.1 (ChebFilter) peut être obtenue de la manière suivante. On suppose que l'on veut résoudre le système préconditionné symétriquement

$$\mathbf{L}^{-1}\mathbf{A}\mathbf{L}^{-T}\mathbf{u} = \mathbf{L}^{-1}\mathbf{b}, \quad \mathbf{x} = \mathbf{L}^{-T}\mathbf{u}, \tag{4.7}$$

et on applique l'Algorithme 4.1 au système (4.7). En redéfinissant les variables de la façon suivante

$$\begin{array}{lcl} \mathbf{L}^{-T}\mathbf{u}_f & \rightarrow & \mathbf{x}_f \\ \mathbf{L}\,\mathbf{w}_f & \rightarrow & \mathbf{w}_f \end{array} \tag{4.8}$$

et en posant $\mathbf{M}_1 = \mathbf{L}\mathbf{L}^T$, on obtient l'Algorithme 4.3, et qu'on appellera PChebFilter.

**Algorithme 4.3 : PChebFilter**

$[\mathbf{w}_f, \mathbf{x}_f] =$ PChebFilter $\left(\mathbf{A}, \mathbf{b}, \mu, \lambda_{\max}, \mathbf{x}^{(0)}, \varepsilon, \mathbf{M}_1\right)$
$\lambda_{\max}$ correspond à la plus grande valeur propre du système préconditionné.

**Début**

1. $\alpha_\mu = \dfrac{2}{\lambda_{\max} - \mu}$ et $d_\mu = \dfrac{\lambda_{\max} + \mu}{\lambda_{\max} - \mu}$
2. Choix de $\mathbf{x}^{(0)}$
3. $\mathbf{x}_f = \mathbf{x}^{(0)}$, et $\mathbf{w}_f = \mathbf{M}_1^{-1}(\mathbf{b} - \mathbf{A}\mathbf{x}_f)$
4. $\mathbf{y} = \mathbf{x}_f,\ m = 1,\ \sigma_0 = 1$ et $\sigma_1 = d_\mu$
5. $\mathbf{x}_f = \mathbf{x}_f + \dfrac{\alpha_\mu}{d_\mu}\mathbf{w}_f$ et $\mathbf{w}_f = \mathbf{M}_1^{-1}(\mathbf{b} - \mathbf{A}\mathbf{x}_f)$
6. **Tant que** $1/\sigma_m \geq \varepsilon$ **Faire**
   - i. $\sigma_{m+1} = 2\,d_\mu\,\sigma_m - \sigma_{m-1}$
   - ii. $\mathbf{p} = 2\,\dfrac{\sigma_m}{\sigma_{m+1}}\left(d_\mu\mathbf{x}_f + \alpha_\mu\mathbf{w}_f\right) - \dfrac{\sigma_{m-1}}{\sigma_{m+1}}\,\mathbf{y}$
   - iii. $\mathbf{y} = \mathbf{x}_f$ et $\mathbf{x}_f = \mathbf{p}$
   - iv. $\mathbf{w}_f = \mathbf{M}_1^{-1}(\mathbf{b} - \mathbf{A}\mathbf{x}_f)$
   - v. $m = m + 1$
7. **FinTantque**

**Fin**

Quant à la version préconditionnée de l'Algorithme 4.2 (PChebFilterCG), elle peut être obtenue en remplaçant l'appel à ChebFilter par PChebFilter dans l'algorithme ChebFilterCG.

## 4.3 Essais numériques

Dans ce paragraphe, nous allons illustrer, avec quelques essais numériques, l'efficacité de notre préconditionneur polynomial. Nous considérons, pour cela, les deux problèmes test EDP1 et EDP2 déjà décrits au § 3.3. Ces deux problèmes sont extraits de la discrétisation par éléments finis, en utilisant **pdetool**© sous **Matlab**©, de problèmes aux dérivés partielles elliptiques.

Cependant, nous utiliserons trois types de premier niveau de préconditionnement : Jacobi classique $\mathbf{M}_1 = \text{diag}(\mathbf{A})$, la factorisation incomplète de Cholesky de $\mathbf{A}$ sans remplissage, et la factorisation incomplète de Cholesky IC($t$) de $\mathbf{A}$ où $t$, paramètre de seuil contrôlant le niveau de remplissage, est fixé égal à $10^{-2}$ ; $\mathsf{cholinc}(\mathbf{A}, 10^{-2})$ comme dans les notations **Matlab**©.
En utilisant la factorisation incomplète de Cholesky, nous calculons la matrice triangulaire inférieure $\mathbf{L}$ telle que $\mathbf{M}_1 = \mathbf{L}\mathbf{L}^T$. Le but de ce premier niveau de préconditionnement est de mieux regrouper le spectre de la matrice d'itération, ce qui est une situation favorable pour la technique de résolution que nous proposons ici.

| Préconditionneur Premier Niveau | $\kappa(\mathbf{M}_1^{-1}\mathbf{A})$ | $\lambda_{\min}$ | $\lambda_{\max}$ |
|---|---|---|---|
| $\mathbf{M}_1 = \mathbf{I}$ | $2.6 \cdot 10^9$ | $3.7 \cdot 10^{-3}$ | $9.6 \cdot 10^6$ |
| $\mathbf{M}_1 =$ Jacobi Classique ; $\mathbf{M}_1 = \text{diag}(\mathbf{A})$ | $6.8 \cdot 10^8$ | $3.1 \cdot 10^{-9}$ | 2.08 |
| $\mathbf{M}_1 = \text{IC}(0)$ ; sans Remplissage | $9.4 \cdot 10^7$ | $1.7 \cdot 10^{-8}$ | 1.6 |
| $\mathbf{M}_1 = \text{IC}(10^{-2})$ ; avec Remplissage | $6.2 \cdot 10^6$ | $1.8 \cdot 10^{-7}$ | 1.1 |

TAB. 4.1 – *Estimations de* $\kappa(\mathbf{M}_1^{-1}\mathbf{A})$, $\lambda_{\min}$, *et* $\lambda_{\max}$, *pour le problème test EDP1.*

Dans un premier temps, nous considérons le problème test EDP1. Dans la table 4.1, nous indiquons les différentes valeurs du conditionnement de la matrice $\mathbf{A}$, $\kappa(\mathbf{A})$, et de la matrice préconditionnée $\mathbf{M}_1^{-1}\mathbf{A}$, $\kappa(\mathbf{M}_1^{-1}\mathbf{A})$, pour Jacobi classique et Cholesky Incomplet sans remplissage IC(0) et avec remplissage IC($10^{-2}$). Nous notons le conditionnement élevé de la matrice originale $\mathbf{A}$, de l'ordre $10^9$, et celui de la matrice préconditionnée par Cholesky Incomplet IC($10^{-2}$), réduit d'un facteur $10^3$.

Dans la table 4.2, nous indiquons le nombre de valeurs propres dans l'intervalle $[\lambda_{\min}, \mu]$ pour le problème test EDP1, et ceci pour diverses du paramètre de coupure $\mu$ dans le préconditionneur. Nous pouvons observer que la matrice originale a des problèmes d'échelle dans ses entrées, et que le préconditionneur Jacobi classique permet déjà de regrouper substantiellement les valeurs propres, bien que le nombre du conditionnement demeure très grand.

| $\mu = \lambda_{\max}/\gamma$ | Préconditionneur Premier Niveau $\mathbf{M}_1$ | | | |
|---|---|---|---|---|
| $\gamma$ | Identité $\mathbf{I}$ | Jacobi Classique $\mathsf{diag}(\mathbf{A})$ | Cholesky Incomplet IC(0) | Cholesky Incomplet IC($10^{-2}$) |
| $10^9$ | 3 | | | |
| $10^8$ | 41 | | | |
| $10^7$ | >200 | | | |
| $10^3$ | | 3 | | |
| 500 | | 5 | | |
| 200 | | 18 | | |
| 100 | | 43 | 3 | |
| 50 | | 89 | 11 | |
| 20 | | >200 | 32 | |
| 10 | | | 68 | 3 |
| 5 | | | 157 | 9 |
| 2 | | | >200 | 40 |

TAB. 4.2 – *Nombre de valeurs propres dans l'intervalle* $[\lambda_{\min}, \mu]$ *pour différentes valeurs de coupure* $\mu$*. Le problème test considéré est EDP1.*

Dans les tables suivantes, le temps CPU ne sera pas affiché, puisque toutes nos expériences ont été effectuées sous **Matlab**©, pour illustrer le comportement et l'efficacité numérique de l'approche proposée.

### 4.3.1 Impact de la Valeur de Coupure $\mu$ et du Niveau de Filtrage $\varepsilon$

Le préconditionneur ChebFilter développé au § 4.2 dépend de deux paramètres, qui correspondent aux choix de la valeur de coupure, $\mu$, et du niveau du filtrage, $\varepsilon$.

Le premier paramètre $\mu$ divise le spectre de la matrice $\mathbf{A}$ en deux sous-ensembles et détermine le taux de convergence des étapes de Tchebycheff calculées à chaque itération du gradient conjugué pour réaliser le préconditionnement polynomial, puisque celui-ci définit l'intervalle d'atténuation $[\mu, \lambda_{\max}]$ où le polynôme de Tchebycheff converge uniformément vers 0.

Dans la table 4.3, nous considérons le problème test EDP1 décrit au § 3.3. L'itéré initial est $\mathbf{x}^{(0)} = 0$ et le second membre $\mathbf{b}$ est choisi de sorte que la solution $\mathbf{x}^\star$ du système linéaire initial soit le vecteur $\mathbf{e} = [1, 1, \ldots, 1]$. Dans cette table, nous montrons le nombre d'étapes de filtrage de Tchebycheff (`ChebIt`), le nombre d'itérations du gradient conjugué (`CGIt`), et le nombre total de produits matrice-vecteur (`ChebIt`×`CGIt`), ceci pour différentes valeurs du niveau du filtrage $\varepsilon$ et pour différentes valeurs de coupure $\mu$. Dans

| | Le problème test EDP1 préconditionné par Jacobi Classique<br>CG sans préconditionnement de Tchebycheff requiert `CGIt` = 485 itérations | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | $\mu = \lambda_{\max}/1000$ | | | $\mu = \lambda_{\max}/500$ | | | $\mu = \lambda_{\max}/100$ | | |
| Valeur de $\varepsilon$ | `ChebIt` | `CGIt` | `ChebIt`×`CGIt` | `ChebIt` | `CGIt` | `ChebIt`×`CGIt` | `ChebIt` | `CGIt` | `ChebIt`×`CGIt` |
| $10^{-16}$ | 594 | 4 | 2376 | 420 | 5 | 2100 | 188 | 1 | 2068 |
| $10^{-8}$ | 303 | 5 | 1515 | 214 | 6 | 1284 | 96 | 16 | 1536 |
| $10^{-4}$ | 157 | 7 | 1099 | 111 | 8 | 888 | 50 | 22 | 1100 |
| $10^{-2}$ | 84 | 11 | 924 | 60 | 12 | 720 | 27 | 30 | 810 |
| $10^{-1}$ | 48 | 17 | 816 | 34 | 19 | 646 | 15 | 42 | 630 |
| | Le problème test EDP1 préconditionné par Cholesky Incomplet sans remplissage IC(0)<br>CG sans préconditionnement de Tchebycheff requiert `CGIt` = 191 itérations | | | | | | | | |
| | $\mu = \lambda_{\max}/100$ | | | $\mu = \lambda_{\max}/50$ | | | $\mu = \lambda_{\max}/20$ | | |
| Valeur de $\varepsilon$ | `ChebIt` | `CGIt` | `ChebIt`×`CGIt` | `ChebIt` | `CGIt` | `ChebIt`×`CGIt` | `ChebIt` | `CGIt` | `ChebIt`×`CGIt` |
| $10^{-16}$ | 188 | 4 | 752 | 132 | 6 | 792 | 83 | 9 | 747 |
| $10^{-8}$ | 96 | 5 | 480 | 68 | 8 | 544 | 43 | 13 | 559 |
| $10^{-4}$ | 50 | 7 | 350 | 35 | 11 | 385 | 22 | 18 | 396 |
| $10^{-2}$ | 27 | 11 | 297 | 19 | 15 | 285 | 12 | 25 | 300 |
| $10^{-1}$ | 15 | 18 | 270 | 11 | 22 | 242 | 7 | 35 | 245 |
| | Le problème test EDP1 préconditionné par Cholesky Incomplet IC($10^{-2}$)<br>CG sans préconditionnement de Tchebycheff requiert `CGIt` = 54 itérations | | | | | | | | |
| | $\mu = \lambda_{\max}/20$ | | | $\mu = \lambda_{\max}/10$ | | | $\mu = \lambda_{\max}/5$ | | |
| Valeur de $\varepsilon$ | `ChebIt` | `CGIt` | `ChebIt`×`CGIt` | `ChebIt` | `CGIt` | `ChebIt`×`CGIt` | `ChebIt` | `CGIt` | `ChebIt`×`CGIt` |
| $10^{-16}$ | 83 | 4 | 332 | 58 | 4 | 232 | 39 | 5 | 195 |
| $10^{-8}$ | 43 | 4 | 172 | 30 | 5 | 150 | 20 | 7 | 140 |
| $10^{-4}$ | 22 | 6 | 132 | 16 | 7 | 102 | 11 | 9 | 99 |
| $10^{-2}$ | 12 | 10 | 120 | 9 | 10 | 90 | 6 | 13 | 78 |
| $10^{-1}$ | 7 | 15 | 105 | 5 | 16 | 80 | 4 | 17 | 68 |

TAB. 4.3 – *Comparaison du nombre d'étapes du filtrage de Tchebycheff, du nombre d'itérations du gradient conjugué, et du nombre total des produits matrice-vecteur pour différentes valeurs du niveau du filtrage $\varepsilon$ et valeurs de coupure $\mu$. Le problème test considéré est EDP1.*

- `CGIt` *correspond au nombre d'itérations du gradient conjugué .*
- `ChebIt` *correspond au nombre d'étapes du filtrage de Tchebycheff à chaque itération du gradient conjugué.*
- `ChebIt`×`CGIt` *correspond au nombre total des produits matrice-vecteur.*

ces essais numériques, le critère d'arrêt utilisé, portant sur l'erreur relative en norme $\mathbf{A}$, est donné par $||\mathbf{x}^\star - \mathbf{x}^{(k)}||_\mathbf{A} \leq 10^{-9}\ ||\mathbf{x}^\star||_\mathbf{A}$.

Nous pouvons observer les effets combinés de ces deux paramètres sur nos problèmes test. Le changement rapide du taux de convergence de Tchebycheff, pour des valeurs de coupure $\mu$ décroissantes, induit plus d'étapes de filtrage à chaque itération, et ceci ne peut être compensé que par une bonne réduction du nombre total d'itérations de la méthode du gradient conjugué préconditionné. À cet égard, un préconditionneur de premier niveau $\mathbf{M}_1$ est un point clé pour bien regrouper le spectre de la matrice d'itération $\mathbf{M}_1^{-1}\mathbf{A}$, et permettre éventuellement d'utiliser des valeurs de coupure $\mu$ raisonnables.

Le deuxième paramètre, le niveau du filtrage $\varepsilon$, influence directement le nombre d'étapes de Tchebycheff. Pour illustrer cela, nous indiquons dans la table 4.3 le nombre d'étapes de filtrage effectuées à chaque itération du gradient conjugué, pour différentes valeurs de $\varepsilon$.

Nous notons que, pour un niveau fixe du filtrage $\varepsilon$, le nombre total de produits matrice-vecteur (`ChebIt`×`CGIt`) ne change pas drastiquement quand le paramètre $\mu$ varie. Pour une valeur fixe de coupure $\mu$, si nous considérons le nombre total de produits matrice-vecteur (`ChebIt`×`CGIt`), il est clair que le gradient conjugué tout seul nécessite moins de produits matrice-vecteur que sa combinaison avec les filtres de Tchebycheff. Ceci est dû au fait que les polynômes de Tchebycheff ne sont pas optimaux en $||\cdot||_\mathbf{A}$, car ils agissent uniformément et indépendemment des composantes propres du vecteur résidu.

Si nous considérons une valeur du filtrage $\varepsilon$ proche de $10^{-1}$ pour minimiser l'effort dans chaque étape du filtrage de Tchebycheff, nous observons que le nombre total de produits matrice-vecteur réalisés n'est pas trop loin de celui réalisé par le gradient conjugué tout seul. Cependant, le résultat le plus important à souligner concerne la réduction substantielle de la dimension de la base de Krylov produite avec cette combinaison Tchebycheff–CG, qui peut être 3 fois plus petite dans le cas de IC($10^{-2}$), ou réduite par un facteur environ 6 dans le cas de IC(0), et 10 fois plus petite dans le cas de Jacobi classique en tant que préconditionneur premier niveau.

Nous verrons d'ailleurs que, si nous avons l'intention de réutiliser la base de Krylov filtrée pour accélérer d'autres résolutions, minimiser la taille de cette base est un facteur très important pour la réduction des coûts, et il est alors préférable d'utiliser des petites valeurs pour le niveau du filtrage $\varepsilon$. Ces différents aspects seront analysés plus en détail, en termes d'opérations flottantes, au § 4.5.

### 4.3.2 Pertinence de la Base de Krylov

Dans ce paragraphe, nous considérons le problème test EDP1. Nous évaluons la pertinence de l'information spectrale stockée dans le sous-espace de Krylov $[\mathbf{W}_k]$ obtenu après $k$ itérations du gradient conjugué préconditionné dans l'algorithme ChebFilterCG. Nous notons que, dans ce paragraphe, la base $\mathbf{W}_k$ est orthogonale. Elle est formée par les résidus et non par les directions de descentes. Nous effectuons une analyse spectrale de Ritz de $\mathbf{W}_k$, et nous étudions également les cosinus des angles principaux entre ce sous-espace de Krylov et le sous-espace invariant correspondant engendré par $\mathbf{U}_k$, associé aux $k$ plus petites valeurs propres de la matrice $\mathbf{A}$. La base propre $\mathbf{U}_k$ est calculée par ARPACK [44].

#### Analyse Spectrale de Ritz

Les valeurs de Ritz $\delta_i$, $1 \leq i \leq k$ sont les valeurs propres de la matrice de Rayleigh $\mathbf{W}_k^T \mathbf{A} \mathbf{W}_k$ où $\mathbf{W}_k$ est la matrice de $\mathbb{R}^{n \times k}$ dont les colonnes constituent l'ensemble des vecteurs de Krylov orthonormalisés.

| Matrice Préconditionnée par IC($10^{-2}$) | | | Matrice Préconditionnée par IC(0) | | |
|---|---|---|---|---|---|
| $\lambda_{\max} = 1.1$ | valeur de $\lvert\lambda_i - \delta_i\rvert/\lvert\lambda_i\rvert$ | | $\lambda_{\max} = 1.6$ | valeur de $\lvert\lambda_i - \delta_i\rvert/\lvert\lambda_i\rvert$ | |
| $\lambda_i$ | $\mu = \lambda_{\max}/10$ | $\mu = \lambda_{\max}/5$ | $\lambda_i$ | $\mu = \lambda_{\max}/50$ | $\mu = \lambda_{\max}/20$ |
| $1.79 \cdot 10^{-7}$ | $4.92 \cdot 10^{-10}$ | $3.40 \cdot 10^{-10}$ | $1.66 \cdot 10^{-8}$ | $8.87 \cdot 10^{-9}$ | $8.45 \cdot 10^{-9}$ |
| $1.80 \cdot 10^{-5}$ | $1.10 \cdot 10^{-11}$ | $1.85 \cdot 10^{-13}$ | $1.67 \cdot 10^{-6}$ | $2.74 \cdot 10^{-11}$ | $3.58 \cdot 10^{-11}$ |
| $6.89 \cdot 10^{-2}$ | $1.00 \cdot 10^{-15}$ | $2.61 \cdot 10^{-15}$ | $8.19 \cdot 10^{-3}$ | $2.07 \cdot 10^{-14}$ | $1.69 \cdot 10^{-14}$ |
| $1.65 \cdot 10^{-1}$ | $2.78 \cdot 10^{0}$ | $4.01 \cdot 10^{-5}$ | $1.64 \cdot 10^{-2}$ | $2.89 \cdot 10^{-4}$ | $2.36 \cdot 10^{-6}$ |
| $1.66 \cdot 10^{-1}$ | $3.33 \cdot 10^{0}$ | $1.48 \cdot 10^{-2}$ | $1.64 \cdot 10^{-2}$ | $9.55 \cdot 10^{-4}$ | $7.94 \cdot 10^{-9}$ |
| $1.67 \cdot 10^{-1}$ | $4.08 \cdot 10^{0}$ | $1.42 \cdot 10^{-2}$ | $2.09 \cdot 10^{-2}$ | $8.86 \cdot 10^{-7}$ | $1.08 \cdot 10^{-10}$ |
| $1.69 \cdot 10^{-1}$ | $8.77 \cdot 10^{0}$ | $8.77 \cdot 10^{-2}$ | $2.15 \cdot 10^{-2}$ | $3.08 \cdot 10^{-5}$ | $3.01 \cdot 10^{-9}$ |

TAB. 4.4 – *Résidu relatif correspondant aux plus petites valeurs propres $\lambda_i$, $1 \leq i \leq 7$, pour un niveau du filtrage $\varepsilon = 10^{-4}$ et pour différentes valeurs de coupure $\mu$. Le problème considéré est EDP1.*

Dans la table 4.4, nous indiquons le résidu relatif correspondant aux plus petites valeurs propres $\lambda_i$, $1 \leq i \leq 7$, ce qui indique le nombre de chiffres corrects pour chaque valeur propre approchée $\delta_i$, $1 \leq i \leq 7$, pour un niveau du filtrage $\varepsilon = 10^{-4}$ et pour différentes valeurs de coupure $\mu$. Par exemple, le gradient conjugué préconditionné par les filtres de Tchebycheff (ChebFilterCG) donne une bonne approximation des trois valeurs propres les plus petites (dix chiffres corrects) pour la matrice préconditionnée par IC($10^{-2}$) et $\mu = \lambda_{\max}/10$. Nous observons que, avec $\mu = \lambda_{\max}/10$, les quatre autres valeurs propres ne sont pas bien approchées. Ceci est simplement dû au fait que ces valeurs propres tombent directement dans l'intervalle $[\lambda_{\max}/10, \lambda_{\max}]$,

où le polynôme de Tchebycheff converge uniformément vers 0. Cette table prouve l'intérêt de l'information spectrale stockée dans le sous-espace de Krylov généré par ChebFilterCG, qui offre en effet une bonne approximation de la partie spectrale associée à l'intervalle $]0, \mu[$.

### Cosinus des Angles Principaux

La table 4.5 donne les cosinus des angles principaux entre les deux sous-espaces $[\mathbf{W}_k]$ et $[\mathbf{U}_k]$. $\mathbf{W}_k$ est une base orthonormale de l'espace de Krylov généré après $k$ itérations de l'algorithme ChebFilterCG. $\mathbf{U}_k$ est la matrice dont les colonnes constituent le système orthonormé des $k$ premiers vecteurs propres de la matrice préconditionnée $\mathbf{M}_1^{-1}\mathbf{A}$. L'index $k$ réfère au nombre d'itérations de l'algorithme ChebFilterCG. Le calcul est réalisé par la décomposition en valeurs singulières (SVD) [34], puisque les valeurs singulières de $\mathbf{W}_k^T\,\mathbf{U}_k$ correspondent aux cosinus des angles principaux entre les deux sous-espaces $[\mathbf{W}_k]$ et $[\mathbf{U}_k]$. Comme nous pouvons l'observer dans la table 4.5, ces angles principaux restent très proches de zéro. Nous déduisons de ce calcul de cosinus que les deux sous-espaces $[\mathbf{W}_k]$ et $[\mathbf{U}_k]$ sont quasiment presque colinéaires, et plus particulièrement pour les directions qui sont associées à l'intervalle $]0, \mu[$.

| SVD($\mathbf{W}_k^T\,\mathbf{U}_k$) | | | |
|---|---|---|---|
| Valeurs du niveau du filtrage $\varepsilon$ et la dimension correspondante $k$ de la base de Krylov $\mathbf{W}_k$ | | | |
| $\varepsilon = 10^{-16}$ et $(k = 4)$ | $\varepsilon = 10^{-4}$ et $(k = 7)$ | $\varepsilon = 10^{-1}$ et $(k = 16)$ | |
| 1.000E+00 | 1.000E+00 | 1.000E+00 | 9.999E-01 |
| 1.000E+00 | 1.000E+00 | 1.000E+00 | 9.994E-01 |
| 9.999E-01 | 9.999E-01 | 1.000E+00 | 9.731E-01 |
| 9.538E-01 | 9.815E-01 | 9.999E-01 | 9.580E-01 |
| | 9.725E-01 | 9.999E-01 | 9.389E-01 |
| | 9.586E-01 | 9.993E-01 | 8.682E-01 |
| | 8.998E-01 | 9.999E-01 | 8.192E-01 |
| | | 9.999E-01 | 6.219E-01 |

TAB. 4.5 – *Cosinus des angles principaux entre* $[\mathbf{W}_k]$ *et* $[\mathbf{U}_k]$, *le sous-espace associé aux k plus petites valeurs propres. La matrice est initialement préconditionnée par Cholesky Incomplet avec remplissage* $IC(10^{-2})$, *et la valeur de coupure dans l'algorithme* ChebFilterCG *a été fixée à* $\mu = \lambda_{\max}/10$. *Le problème test considéré est EDP1.*

## 4.4 Réutilisation du Sous-Espace de Krylov Filtré pour une Multirésolution

Dans ce paragraphe, notre principal intérêt est de résoudre une séquence de systèmes linéaires impliquant une même matrice mais des second membres différents :

$$\mathbf{A}\mathbf{x}_\ell = \mathbf{b}_\ell,\ \ell = 1, \ldots, s, \tag{4.9}$$

où $\mathbf{A}$ est une matrice symétrique définie positive de $\mathbb{R}^{n\times n}$, $\mathbf{x}_\ell$ et $\mathbf{b}_\ell$ sont des vecteurs de $\mathbb{R}^n$. L'idée principale est de résoudre le premier système $\mathbf{A}\mathbf{x}_1 = \mathbf{b}_1$ dans la séquence (4.9), par la méthode du gradient conjugué combiné avec les filtres de Tchebycheff comme préconditionneurs (ChebFilterCG), et d'exploiter la base de Krylov résultante pour accélérer les résolutions suivantes.

Une fois que cette base de Krylov $\mathbf{W}_k$ est obtenue, après $k$ itérations du gradient conjugué dans l'algorithme ChebFilterCG, nous pouvons la réutiliser dans l'Algorithme 3.1 (Init-CG), par exemple (voir [12, 13, 28, 29, 62, 73, 77]), pour la résolution de tout autre système dans la séquence (4.9). Nous insistons sur le fait que nous gardons la même base $\mathbf{W}_k$ pour toutes les résolutions ultérieures.

Comme précédemment décrit (§ 3.2.1 du Chapitre 3), le gradient conjugué avec projection initiale, Init-CG, établit une projection oblique du résidu initial dans $[\mathbf{W}_k]$ afin d'obtenir les composantes propres de la solution qui correspondent aux plus petites valeurs propres. La partie restante de la solution est calculée par la méthode du gradient conjugué classique.

Algorithme 4.4 :

**Début**

1. $[\mathbf{x}_1, \mathbf{W}] = \text{PChebFilterCG}\ (\mathbf{A}, \mathbf{b}_1, \mathbf{x}^{(0)}, \text{tol}_1, \mathbf{M}_1)$
2. $\mathbf{A}_c = \mathbf{W}^T\mathbf{A}\mathbf{W}$, est une matrice diagonale
3. **For** $\ell = 2, \ldots, s$ **Faire**
   i. $\mathbf{x}^{(0)} = \mathbf{W}\mathbf{A}_c^{-1}\mathbf{W}^T\mathbf{b}_\ell$
   ii. $\mathbf{x}_\ell = \text{PCG}(\mathbf{A}, \mathbf{b}_\ell, \mathbf{x}^{(0)}, \text{tol}_2, \mathbf{M}_1)$
4. **FinPour**

**Fin**

Du point de vue pratique, cela se résume à utiliser la méthode du gradient conjugué préconditionné par $\mathbf{M}_1$ pour résoudre $\mathbf{A}\mathbf{x}_\ell = \mathbf{b}_\ell$, $\ell \geq 2$ (cf. Algorithme 4.4, étape 3.ii), avec un vecteur de départ $\mathbf{x}_\ell^{(0)} = \mathbf{W}_k(\mathbf{W}_k^T\mathbf{A}\mathbf{W}_k)^{-1}\mathbf{W}_k^T\mathbf{b}_\ell$ où $\mathbf{W}_k$ est une matrice de $\mathbb{R}^{n\times k}$. La base de Krylov $\mathbf{W}_k$ est obtenue après $k$ itérations de l'algorithme ChebFilterCG en résolvant le premier système ($\ell = 1$). Elle est constituée par les vecteurs de descente du gradient conjugué

et non pas les vecteurs résidus. Dans ce cas, $\mathbf{W}_k$ est une base $\mathbf{A}$-orthogonale et $\mathbf{A}_c = \mathbf{W}_k^T \mathbf{A} \mathbf{W}_k$ est une matrice diagonale. Nous notons que nous ne projetons pas la matrice à chaque itération, mais que nous employons le gradient conjugué préconditionné par $\mathbf{M}_1$ (préconditionneur de premier niveau) avec la matrice originale, et un premier itéré $\mathbf{x}_\ell^{(0)}$ avec lequel nous espérons "*gommer*" les difficultés que le CG pourrait rencontrer, à savoir les plateaux qui peuvent être observés dans les courbes de convergence. Cette approche est résumée dans l'Algorithme 4.4.

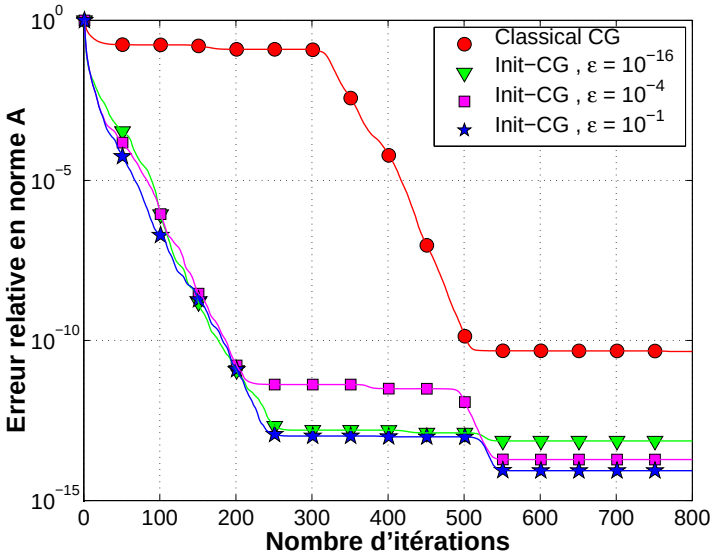
Pour illustrer cette stratégie, nous considérons le problème test EDP1. Un premier système $\mathbf{A}\mathbf{x}_1 = \mathbf{b}_1$ est d'abord résolu, puis un second $\mathbf{A}\mathbf{x}_2 = \mathbf{b}_2$. Pour le premier système, l'itéré initial est $\mathbf{x}_1^{(0)} = 0$ et le second membre $\mathbf{b}_1$ est choisi tel que la solution soit le vecteur $\mathbf{e} = [1, 1, \ldots, 1]$. Le critère d'arrêt pour ce premier système porte sur l'erreur relative en norme $\mathbf{A}$ : $||\mathbf{x}_1^\star - \mathbf{x}_1^{(k)}||_\mathbf{A} \leq 10^{-9} ||\mathbf{x}_1^\star||_\mathbf{A}$. Quand au deuxième système, le second membre $\mathbf{b}_2$ est choisi de sorte que la solution soit un vecteur de nombres aléatoires qui suivent la loi normale $\mathcal{N}(0, 1)$. Pour ces tests, nous avons également contrôlé l'erreur relative en norme $\mathbf{A}$ jusqu'à la précision machine, pour illustrer et étudier le comportement numérique complet de la méthode.

Les diverses figures tracent la décroissance de l'erreur en norme $\mathbf{A}$ en fonction du nombre d'itérations (qui est aussi égal au nombre de produits matrice-vecteur). En plus, nous dénoterons par "Classical CG" le gradient conjugué préconditionné par Jacobi classique ou Cholesky Incomplet, c'est à dire le premier niveau de préconditionnement seulement.

Les figures 4.1 et 4.2 montrent une amélioration significative, lorsqu'on réutilise le sous-espace de Krylov généré par l'algorithme ChebFilterCG, dans la résolution du deuxième système linéaire. Nous traçons l'erreur relative en norme $\mathbf{A}$, en fonction du nombre d'itérations, respectivement pour les problèmes test EDP1 préconditionné par Jacobi classique et par Cholesky Incomplet sans remplissage. Pour la comparaison, nous montrons également la courbe de convergence de l'erreur relative en norme $\mathbf{A}$ du gradient conjugué préconditionné mais sans projection du vecteur résidu initial "Classical CG".

La figure 4.1 montre les améliorations obtenues pour une base de Krylov générée par l'algorithme ChebFilterCG avec différents niveaux de filtrage $\varepsilon$. Nous observons que, pour une valeur de coupure fixée $\mu$, les courbes de convergence de Init-CG présentent un comportement numérique constant quel que soit le choix de ce niveau de filtrage $\varepsilon$, puisque toutes les courbes coïncident presque totalement, sauf après stagnation. Nous observons également que le nombre d'itérations est réduit par un facteur de 4 pour atteindre une erreur relative en norme $\mathbf{A}$ de $10^{-8}$, ce qui illustre la pertinence de l'information spectrale contenue dans l'espace de Krylov obtenu par ChebFilterCG.

*EDP1 préconditionné par Jacobi Classique, pour la valeur de coupure* $\mu = \lambda_{\max}/500$




*EDP1 préconditionné par Cholesky Incomplet sans remplissage, pour la valeur de coupure* $\mu = \lambda_{\max}/100$

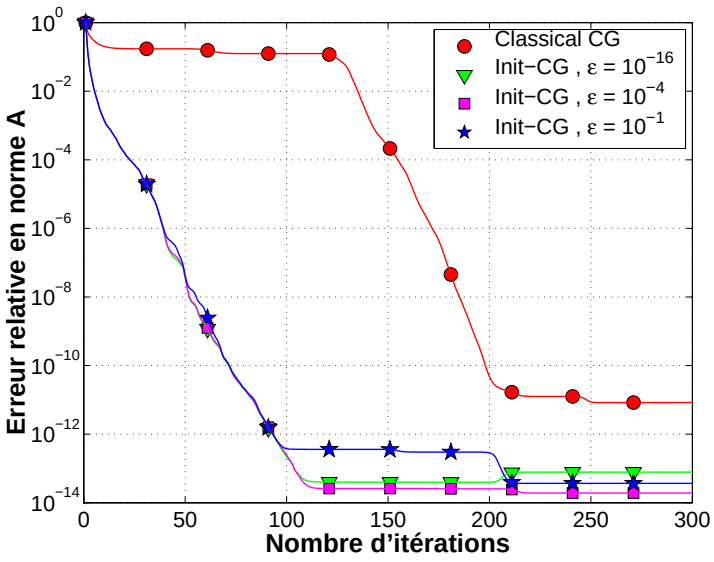
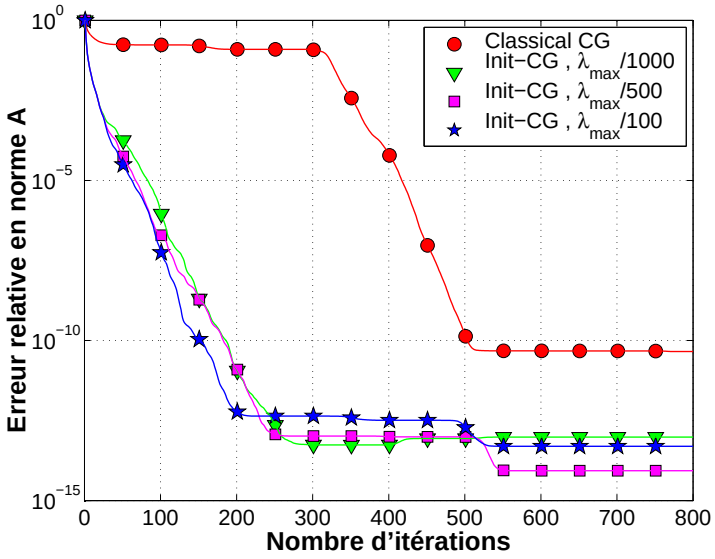



FIG. 4.1 – *Courbes de Convergence de* Init-CG *avec une base de Krylov générée par* ChebFilterCG*, pour une valeur de coupure* $\mu$ *fixée et différents niveaux du filtrage* $\varepsilon$*. Le problème test considéré est EDP1.*

*EDP1 préconditionné par Jacobi classique*




*EDP1 préconditionné par Cholesky Incomplet sans remplissage*

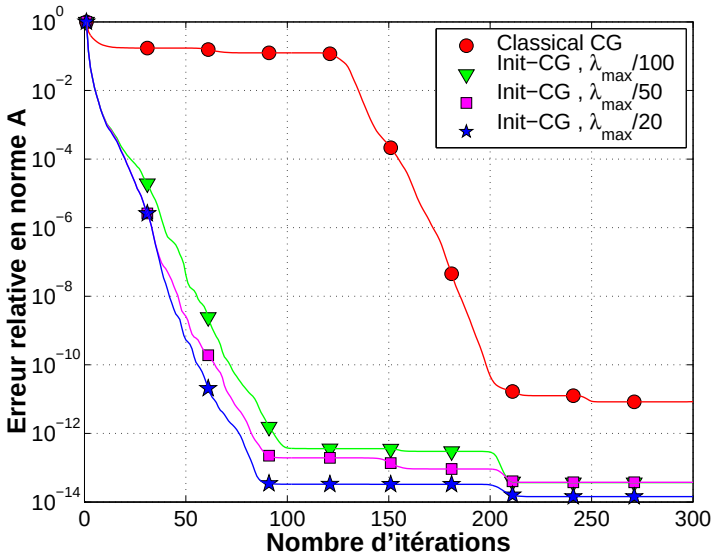



FIG. 4.2 – *Courbe de convergence de l'algorithme* Init-CG *avec la base de Krylov générée par* ChebFilterCG*, pour un niveau du filtrage* $\varepsilon = 10^{-1}$*, et pour différentes valeurs de coupure* $\mu$*. Le problème considéré est EDP1*

Une propriété importante du préconditionneur ChebFilter est que l'on n'a pas besoin d'un niveau du filtrage très proche de la précision machine pour que l'information spectrale contenue dans la base de Krylov soit efficacement réutilisable. Seule la dimension de cette base change quand $\varepsilon$ varie, mais pas le comportement de Init-CG lors de la résolution du second système.

Dans la figure 4.2, nous fixons cette fois le niveau du filtrage (par exemple, $\varepsilon = 10^{-1}$) et nous faisons varier la valeur de coupure $\mu$. Les courbes de convergence de Init-CG, pour ces différentes valeurs de $\mu$, montrent là encore un comportement numérique très similaire. Les légères variations de pente sont dues au fait que la variation du paramètre de coupure $\mu$ change le nombre des valeurs propres approchées dans l'espace de Krylov filtré correspondant. Bien que ces résultats soient limités à deux systèmes seulement, les gains obtenus peuvent être facilement étendus à de plus longues séquences de systèmes linéaires avec la même matrice mais avec différents second membres.

En conclusion, les différentes courbes prouvent que le niveau du filtrage $\varepsilon$ et la valeur de coupure $\mu$ n'ont pas un grand effet sur la qualité de la base de Krylov, puisque le comportement de convergence de l'algorithme Init-CG ne change pas beaucoup pour une base $\mathbf{W}$ obtenue avec différentes valeurs de ces paramètres. Cependant, ces paramètres agissent certainement sur la taille de la base résultante $\mathbf{W}$, qui demeure en tout cas, relativement petite, et très riche en ce qui concerne les plus petites valeurs propres et les vecteurs propres associés.

## 4.5 Considérations Pratiques

Dans ce paragraphe, tous les coûts seront évalués en opérations flottantes. Le coût d'un produit matrice-vecteur avec $\mathbf{A}$ est donné par :

$$\mathcal{C}_{\mathbf{A}} \approx 2\,\mathsf{nnz}\,(\mathbf{A}) - n \tag{4.10}$$

où $\mathsf{nnz}\,(\mathbf{A})$ est le nombre d'éléments non nuls que la matrice $\mathbf{A}$ comporte.

Dans l'algorithme ChebFilterCG, les filtres polynomiaux de Tchebycheff sont employés à chaque itération, de façon à atténuer uniformément toutes les composantes propres du résidu associées à toutes les valeurs propres de $[\mu, \lambda_{\max}]$ au dessous du niveau $\varepsilon$ relativement aux autres.

Chaque étape de filtrage de Tchebycheff implique un produit matrice×vecteur avec la matrice $\mathbf{A}$ et trois mises à jour de vecteur (`_AXPY`). Ce coût est donné par :

$$\mathcal{C}_{\texttt{ChebFilter}} = \mathcal{C}_{\mathbf{A}} + 3\,\mathcal{C}_{\texttt{\_AXPY}}, \text{ où } \mathcal{C}_{\texttt{\_AXPY}} \approx 2\,n. \tag{4.11}$$

Pour ce qui est de l'itération du Gradient Conjugué, en plus d'un produit matrice-vecteur avec $\mathbf{A}$, nous devons aussi considérer les coûts relatifs aux

deux produits scalaires (_DOT) et aux trois mises à jour de vecteur (_AXPY). Le coût des ces opérations est donné par :

$$\mathcal{C}_{\texttt{CG}} = \mathcal{C}_{\mathbf{A}} + 2\,\mathcal{C}_{\texttt{\_DOT}} + 3\,\mathcal{C}_{\texttt{\_AXPY}},\ \text{où}\ \ \mathcal{C}_{\texttt{\_DOT}} \approx 2\,n. \tag{4.12}$$

Par conséquent, le coût total de notre approche est de l'ordre de :

$$\mathcal{C}_{\texttt{ChebFilterCG}} = \Big(\mathcal{C}_{\texttt{CG}} + \texttt{ChebIt} \times \mathcal{C}_{\texttt{ChebFilter}}\Big) \times \texttt{CGIt} \tag{4.13}$$

où ChebIt est le nombre d'étapes du filtrage de Tchebycheff établies à chaque itération du gradient conjugué, et CGIt est le nombre d'itérations du gradient conjugué préconditionné.

### 4.5.1 Gains Potentiels et Amortissements

Dans ce paragraphe, nous analysons les gains pouvant être réalisés à l'aide de notre technique dans le cadre d'une multirésolution. Nous résolvons le premier système de la séquence (4.9) par ChebFilterCG et exploitons ensuite la base de Krylov résultante $\mathbf{W}$ dans l'algorithme Init-CG (voir § 4.4) pour les autres résolutions.

Un préconditionneur de premier niveau $\mathbf{M}_1$ est également utilisé pour bien regrouper le spectre de la matrice d'itération. Le coût d'une multiplication de $\mathbf{M}_1$ par un vecteur est représentée par $\mathcal{C}_{\mathbf{M}_1}$. Comme dans nos essais numériques $\mathbf{M}_1$ est construit par factorisation incomplète de Cholesky ou par Jacobi classique, et $\mathcal{C}_{\mathbf{M}_1}$ peut donc être estimé comme

$$\mathcal{C}_{\mathbf{M}_1} \approx 4\,\mathsf{nnz}\,(\mathbf{L}) - 2\,n \tag{4.14}$$

où $\mathsf{nnz}\,(\mathbf{L})$ est le nombre d'éléments non nuls que comporte la matrice $\mathbf{L}$ issue de Cholesky Incomplet ou de Jacobi classique, et $n$ est la taille du problème. Le coût de ce premier niveau de préconditionnement $\mathbf{M}_1$ doit être incorporé dans les coûts en opérations flottantes décrits au § 4.5. Ceci se résume à ajouter $\mathcal{C}_{\mathbf{M}_1}$ à chaque étape de Tchebycheff (dans ChebFilter), ainsi que du gradient conjugué (CG) ou (Init-CG) ( voir les formules (4.11), (4.12), (4.16)).

L'algorithme Init-CG calcule, au début de la méthode du gradient conjugué, une projection oblique du résidu initial $\mathbf{r}^{(0)}$ dans l'espace de Krylov $\mathsf{Vect}\{\mathbf{W}_k\}$ de dimension $k$, à savoir : $\mathbf{x}^{(0)} = \mathbf{W}_k\mathbf{A}_c^{-1}\mathbf{W}_k^T\mathbf{b}$ où $\mathbf{A}_c = \mathbf{W}_k^T\mathbf{A}\mathbf{W}_k$ est une matrice dense d'ordre $k$, $\mathbf{r}^{(0)} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(0)}$ et $\mathbf{p}^{(0)} = \mathbf{r}^{(0)}$.

Dans le cas où la base de Krylov $\mathbf{W}$ est constituée par les directions de descente $\mathbf{A}$-conjuguées construites dans l'algorithme du Gradient Conjugué, le calcul de $\mathbf{A}_c$ devient même trivial, en ce sens que $\mathbf{A}_c$ est diagonale et peut être construite à la volée au fur et à mesure des itérations. Nous avons effectivement considéré cela dans la mise en œuvre de cette approche et,

| | EDP1 préconditionnée par Jacobi Classique<br>CG sans préconditionnement de Tchebycheff requiert 485 itérations<br>avec un coût de 95.79 Mflop | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | $\mu = \lambda_{\max}/500$ | | | $\mu = \lambda_{\max}/100$ | | | $\mu = \lambda_{\max}/50$ | | |
| $\varepsilon$ | ChebIt | CGIt | Mflop | ChebIt | CGIt | Mflop | ChebIt | CGIt | Mflop |
| $10^{-16}$ | 420 | 5 | 349.68 | 188 | 11 | 345.56 | 132 | 16 | 353.85 |
| $10^{-8}$ | 214 | 6 | 214.39 | 96 | 16 | 258.21 | 68 | 23 | 264.25 |
| $10^{-4}$ | 111 | 8 | 149.03 | 50 | 22 | 187.00 | 35 | 31 | 186.29 |
| $10^{-2}$ | 60 | 12 | 121.92 | 27 | 30 | 140.43 | 19 | 43 | 144.17 |
| $10^{-1}$ | 34 | 19 | 111.03 | 15 | 42 | 112.92 | 11 | 59 | 119.44 |
| | EDP1 préconditionnée par Cholesky Incomplet sans remplissage<br>CG sans préconditionnement de Tchebycheff requiert 191 itérations<br>avec un coût de 55.52 Mflop | | | | | | | | |
| | $\mu = \lambda_{\max}/100$ | | | $\mu = \lambda_{\max}/50$ | | | $\mu = \lambda_{\max}/20$ | | |
| $\varepsilon$ | ChebIt | CGIt | Mflop | ChebIt | CGIt | Mflop | ChebIt | CGIt | Mflop |
| $10^{-16}$ | 188 | 4 | 196.97 | 132 | 6 | 207.97 | 83 | 9 | 197.13 |
| $10^{-8}$ | 96 | 5 | 126.44 | 68 | 8 | 143.98 | 43 | 13 | 149.36 |
| $10^{-4}$ | 50 | 7 | 93.17 | 35 | 11 | 103.46 | 22 | 18 | 108.37 |
| $10^{-2}$ | 27 | 11 | 80.54 | 19 | 15 | 78.58 | 12 | 25 | 85.42 |
| $10^{-1}$ | 15 | 18 | 75.56 | 11 | 22 | 69.44 | 7 | 35 | 74.02 |
| | EDP1 préconditionnée par Cholesky Incomplet avec remplissage $10^{-2}$<br>CG sans préconditionnement de Tchebycheff requiert 54 itérations<br>avec un coût de 22.91 Mflop | | | | | | | | |
| | $\mu = \lambda_{\max}/20$ | | | $\mu = \lambda_{\max}/10$ | | | $\mu = \lambda_{\max}/5$ | | |
| $\varepsilon$ | ChebIt | CGIt | Mflop | ChebIt | CGIt | Mflop | ChebIt | CGIt | Mflop |
| $10^{-16}$ | 83 | 4 | 131.99 | 58 | 4 | 92.74 | 39 | 5 | 78.65 |
| $10^{-8}$ | 43 | 4 | 69.20 | 30 | 5 | 60.99 | 20 | 7 | 57.91 |
| $10^{-4}$ | 22 | 6 | 54.35 | 16 | 7 | 46.92 | 11 | 9 | 42.67 |
| $10^{-2}$ | 12 | 10 | 51.34 | 9 | 10 | 39.56 | 6 | 13 | 36.13 |
| $10^{-1}$ | 7 | 15 | 47.57 | 5 | 16 | 38.18 | 4 | 17 | 33.89 |

TAB. 4.6 – *Coût de* ChebFilterCG *pour différents niveaux du filtrage $\varepsilon$ et pour différentes valeurs de coupure $\mu$. Le critère d'arrêt est satisfait lorsque l'erreur relative en norme $\mathbf{A}$ est inférieure à $10^{-9}$. Le problème considéré est EDP1.*

dans ces conditions, le coût du calcul de $\mathbf{A}_c$ peut être négligé. Avec $\mathbf{A}_c$ sous forme diagonale, le calcul de la projection oblique et de la mise à jour du vecteur résidu peut être effectué grâce à des opérations de type BLAS2. Le coût total de ces opérations est

$$\mathcal{C}_{\text{Proj}} \approx 4\,k\,n. \tag{4.15}$$

De plus, chaque itération nécessite un produit matrice-vecteur avec $\mathbf{A}$, deux produits scalaires (`_DOT`) et trois mises à jour de vecteur (`_AXPY`) (voir les formules (4.10) et (4.12)). Le coût total de l'algorithme Init-CG est donc

$$\mathcal{C}_{\texttt{InitCG}} = \underbrace{\left(\mathcal{C}_{\mathbf{A}} + \mathcal{C}_{\text{Proj}}\right)}_{\text{Initialisation}} + \underbrace{\left(\mathcal{C}_{\mathbf{A}} + 3\,\mathcal{C}_{\texttt{\_AXPY}} + 2\,\mathcal{C}_{\texttt{\_DOT}}\right)}_{\text{À chaque itération}} \times \texttt{Nit}, \tag{4.16}$$

où `Nit` est le nombre d'itérations total de Init-CG.

Dans la table 4.6, nous montrons, pour le problème test EDP1, le nombre d'étapes de filtrage de Tchebycheff (`ChebIt`) à chaque itération du CG, le nombre d'itérations du CG (`CGIt`), et le nombre d'opérations flottantes (en `Mflop`), et ceci pour différents niveaux de filtrage $\varepsilon$.

La table 4.7 illustre les gains réalisés dans Init-CG. Le critère d'arrêt utilisé est :

$$||\mathbf{x}^\star - \mathbf{x}^{(k)}|| \leq 10^{-9}||\mathbf{x}^\star||_{\mathbf{A}}. \tag{4.17}$$

Pour chaque niveau de filtrage $\varepsilon$, nous indiquons dans cette table la dimension $k$ de la base de Krylov résultant de la première résolution de la séquence (4.9) par ChebFilterCG (ce nombre correspond aussi à `CGIt` dans la table 4.6), le nombre d'itérations de Init-CG (`Nit`), et le nombre d'opérations flottantes (`Mflop`) pour atteindre le critère d'arrêt ci-dessus (4.17).

Nous calculons également le nombre minimal de second membres qu'il faudrait considérer, dans une séquence de systèmes linéaires, afin d'amortir le surcoût $\mathcal{C}_{\texttt{ChebFilterCG}}$ du calcul de la base de Krylov $\mathbf{W}$ générée par ChebFilterCG. Ceci est indiqué par le nombre de vecteurs d'amortissement (`Amor.Vec`). Pour calculer ce nombre, nous devons comparer le nombre d'opérations flottantes requises pour atteindre un certain niveau de l'erreur relative en norme $\mathbf{A}$ avec le gradient conjugué, et le nombre d'opérations flottantes pour atteindre le même niveau avec l'algorithme Init-CG. Il ne faut pas non plus oublier que l'utilisation de ChebFilterCG pour construire la base de Krylov $\mathbf{W}$ sert aussi à résoudre un système linéaire (le premier dans la séquence). Par conséquent, le nombre de vecteurs d'amortissement (`Amor.Vec`) est donné par :

$$\texttt{Amor.Vec} = \left\lceil \frac{\mathcal{C}_{\texttt{ChebFilterCG}} - \mathcal{C}_{\texttt{CG}}}{\mathcal{C}_{\texttt{CG}} - \mathcal{C}_{\texttt{InitCG}}} \right\rceil. \tag{4.18}$$

Par exemple, pour atteindre une erreur relative en norme $\mathbf{A}$ d'ordre $10^{-9}$, le gradient conjugué met 485 itérations dans le cas du problème test EDP1

| | EDP1 préconditionnée par Jacobi Classique<br>CG sans préconditionnement de Tchebycheff requiert 485 itérations<br>avec un coût de 95.79 Mflop | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $\mu = \lambda_{\max}/500$ | | | | $\mu = \lambda_{\max}/100$ | | | | $\mu = \lambda_{\max}/50$ | | | |
| $\varepsilon$ | k | Nit | Mflop | Amor. Vec | k | Nit | Mflop | Amor. Vec | k | Nit | Mflop | Amor. Vec |
| $10^{-16}$ | 5 | 152 | 30.19 | 4 | 11 | 160 | 31.96 | 4 | 16 | 153 | 30.74 | 5 |
| $10^{-8}$ | 6 | 152 | 30.22 | 2 | 16 | 142 | 28.56 | 3 | 23 | 131 | 26.61 | 3 |
| $10^{-4}$ | 8 | 163 | 32.46 | 1 | 22 | 130 | 26.38 | 2 | 31 | 140 | 28.58 | 2 |
| $10^{-2}$ | 12 | 159 | 31.79 | 1 | 30 | 127 | 26.04 | 1 | 43 | 126 | 26.26 | 1 |
| $10^{-1}$ | 19 | 159 | 32.02 | 1 | 42 | 126 | 26.23 | 1 | 59 | 126 | 26.76 | 1 |
| | EDP1 préconditionnée par Cholesky Incomplet sans remplissage<br>CG sans préconditionnement requiert 191 itérations<br>avec un coût de 55.52 Mflop | | | | | | | | | | | |
| | $\mu = \lambda_{\max}/100$ | | | | $\mu = \lambda_{\max}/50$ | | | | $\mu = \lambda_{\max}/20$ | | | |
| $\varepsilon$ | k | Nit | Mflop | Amor. Vec | k | Nit | Mflop | Amor. Vec | k | Nit | Mflop | Amor. Vec |
| $10^{-16}$ | 4 | 62 | 17.31 | 4 | 6 | 60 | 17.38 | 4 | 9 | 60 | 17.78 | 4 |
| $10^{-8}$ | 5 | 62 | 17.34 | 2 | 8 | 60 | 17.44 | 2 | 13 | 60 | 17.90 | 2 |
| $10^{-4}$ | 7 | 62 | 17.41 | 2 | 11 | 58 | 17.53 | 2 | 18 | 59 | 17.78 | 2 |
| $10^{-2}$ | 11 | 62 | 17.53 | 1 | 15 | 56 | 16.79 | 1 | 25 | 48 | 14.77 | 1 |
| $10^{-1}$ | 18 | 62 | 17.77 | 1 | 22 | 52 | 15.84 | 1 | 35 | 48 | 15.01 | 1 |
| | EDP1 préconditionnée par Cheolsky Incomplet avec remplissage $10^{-2}$<br>CG sans préconditionnement de Tchebycheff requiert 54 itérations<br>avec un coût de 22.91 Mflop | | | | | | | | | | | |
| | $\mu = \lambda_{\max}/20$ | | | | $\mu = \lambda_{\max}/10$ | | | | $\mu = \lambda_{\max}/5$ | | | |
| $\varepsilon$ | k | Nit | Mflop | Amor. Vec | k | Nit | Mflop | Amor. Vec | k | Nit | Mflop | Amor. Vec |
| $10^{-16}$ | 4 | 21 | 8.99 | 8 | 4 | 18 | 7.71 | 5 | 5 | 18 | 7.74 | 4 |
| $10^{-8}$ | 4 | 21 | 8.99 | 4 | 5 | 18 | 7.74 | 4 | 7 | 18 | 7.81 | 3 |
| $10^{-4}$ | 6 | 21 | 9.05 | 3 | 7 | 18 | 7.81 | 3 | 9 | 18 | 7.87 | 2 |
| $10^{-2}$ | 10 | 21 | 9.19 | 3 | 10 | 18 | 7.92 | 2 | 13 | 18 | 8.00 | 2 |
| $10^{-1}$ | 15 | 21 | 9.35 | 3 | 16 | 18 | 8.10 | 2 | 17 | 18 | 8.13 | 2 |

TAB. 4.7 – *Coûts de* Init-CG*, pour une base* $\mathbf{W}$ *calculée avec différents niveaux du filtrage* $\varepsilon$ *et valeurs de coupure* $\mu$*. Les itérations de l'algorithme* Init-CG *sont stoppées quand l'erreur relative en norme* $\mathbf{A}$ *est inférieure à* $10^{-9}$*. Le problème test considéré est EDP1.*

préconditionné par Jacobi Classique, avec un coût de 95.79 `Mflop` ; 191 itérations dans le cas où le problème EDP1 est préconditionné par Cholesky Incomplet sans remplissage, avec un coût de 55.52 `Mflop` ; et 54 itérations dans le cas où le problème test EDP1 est préconditionné par Cholesky Incomplet avec remplissage IC($10^{-2}$), avec un coût de 22.91 `Mflop`.

En particulier, si on considère le cas où la matrice est préconditionnée par Cholesky Incomplet avec remplissage IC($10^{-2}$), ainsi qu'un niveau du filtrage $\varepsilon = 10^{-4}$ et une valeur de coupure $\mu = \lambda_{\max}/10$, 46.92 `Mflop` sont nécessaires pour résoudre le premier système linéaire et calculer la base de Krylov $\mathbf{W}$ (voir Table 4.6). Quant à l'algorithme Init-CG, il converge en 18 itérations (voir Table 4.7), i.e. une réduction de 34% comparée à celle du gradient conjugué. Le surcoût de 46.92 `Mflop`, pour la construction de la base $\mathbf{W}$ avec ChebFilterCG, est remboursé après seulement trois résolutions consécutives (voir Table 4.7).

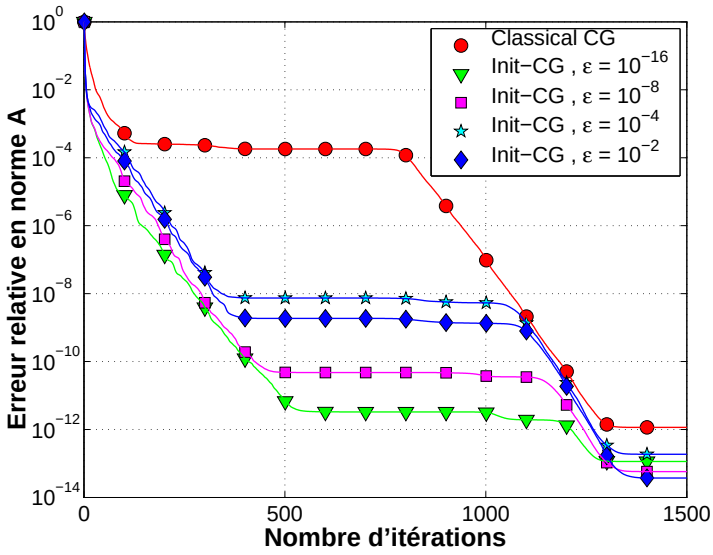
La table 4.7 montre également l'impact du changement du niveau du filtrage $\varepsilon$ et de la valeur de coupure $\mu$ sur le nombre de vecteurs d'amortissement. Pour un certain niveau du filtrage fixé, nous observons que le nombre de vecteurs d'amortissement (`Amor. Vec`) ne change pas drastiquement pour l'algorithme Init-CG. Dans tous les cas, l'amortissement du calcul de la base $\mathbf{W}$ est très rapide.

En conclusion, quand la séquence de systèmes linéaires avec la même matrice et différents second membres est très longue, la stratégie de choix est de résoudre le premier système à l'aide de ChebFilterCG, avec un niveau du filtrage $\varepsilon$ proche de $10^{-16}$ afin de minimiser la taille de la base de Krylov qui en résulte, et exploiter ensuite cette base de dimension réduite pour l'ensemble des autres résolutions. En effet, un niveau du filtrage proche de la précision machine permet d'obtenir d'une part, une base de Krylov de dimension petite, donc une occupation mémoire faible, et d'autre part, un coût optimal dans l'algorithme Init-CG (ou proche de l'optimal).

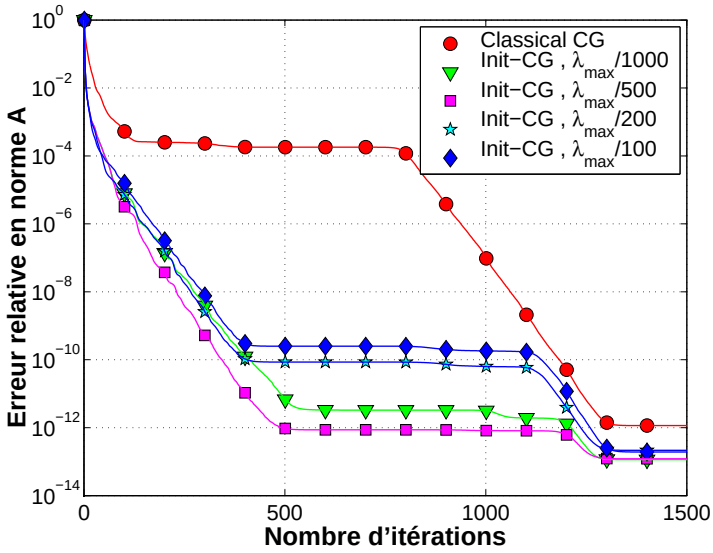
### 4.5.2 Cas du problème Anisotrope EDP2

Dans ce paragraphe, nous considérons le problème test EDP2 introduit au § 3.3. Dans le cas de ce problème test, le préconditionneur de premier niveau $\mathbf{M}_1$ utilisé est Cholesky Incomplet sans remplissage. $\mathbf{M}_1 = \mathbf{L}\mathbf{L}^T$, où $\mathsf{nnz}(\mathbf{L}) = 643\,605$.

Dans le haut de la figure 4.3, nous traçons l'erreur relative en norme $\mathbf{A}$ en fonction du nombre d'itérations pour l'algorithme Init-CG, avec différents niveaux du filtrage $\varepsilon$, et dans le bas de la figure c'est avec différentes valeurs de coupure $\mu$. Nous montrons également l'erreur relative en norme $\mathbf{A}$ du gradient conjugué sans projection initiale. Nous observons que le nombre d'itérations est réduit par un facteur 4 pour atteindre $10^{-8}$ dans l'erreur relative en norme $\mathbf{A}$. Ceci prouve l'efficacité de l'information stockée dans




*(a) Valeur de coupure* $\mu = \lambda_{\max}/1000$




*(b) Niveau du filtrage* $\varepsilon = 10^{-16}$

FIG. 4.3 – *Courbe de convergence du gradient conjugué sans projection initiale* (Classical CG) *et avec projection initiale* (Init-CG) *pour différents niveaux du filtrage* $\varepsilon$ *et valeurs de coupure* $\mu$*. Le problème test considéré est EDP2.*

l'espace de Krylov filtré.

Cependant, quand une grande précision est demandée pour résoudre plusieurs systèmes linéaires en séquence, le phénomène de plateaux peut apparaître dans le comportement numérique de l'algorithme Init-CG (voir figure 4.3). Ceci est dû à la sensibilité du projecteur oblique par rapport à la taille du problème [4]. Un remède pour réduire la taille de ces plateaux et conserver un mode de convergence linéaire, est d'employer le cycle à deux-Grilles précédemment décrit au § 3.2.6.

Dans la table 4.8, nous montrons le nombre d'étapes du filtrage de Tchebycheff (`ChebIt`) calculées à chaque itération du CG, le nombre d'itérations du gradient conjugué (`CGIt`), et le nombre d'opérations flottantes en `Gflop` et ceci avec différents niveaux du filtrage $\varepsilon$. Le coût de préconditionneur premier niveau $\mathbf{M}_1$ est également incorporé dans les opérations flottantes du gradient conjugué avec les filtres de Tchebycheff comme préconditionneurs.

| | CG sans préconditionnement de Tchebycheff requiert 1 060 itérations avec un coût de 6.31 `Gflop` | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | $\mu = \lambda_{\max}/1000$ | | | $\mu = \lambda_{\max}/500$ | | | $\mu = \lambda_{\max}/200$ | | |
| Valeur de $\varepsilon$ | `ChebIt` | `CGIt` | `Gflop` | `ChebIt` | `CGIt` | `Gflop` | `ChebIt` | `CGIt` | `Gflop` |
| $10^{-16}$ | 594 | 8 | 25.28 | 420 | 11 | 24.60 | 265 | 18 | 25.44 |
| $10^{-8}$ | 303 | 11 | 17.76 | 214 | 16 | 18.28 | 135 | 25 | 18.07 |
| $10^{-4}$ | 157 | 15 | 12.59 | 111 | 22 | 13.10 | 70 | 36 | 13.60 |
| $10^{-2}$ | 84 | 21 | 9.49 | 60 | 31 | 10.06 | 38 | 49 | 10.18 |

TAB. 4.8 – *Coût de* ChebFilterCG *pour différents niveaux du filtrage $\varepsilon$ et pour différentes valeurs de coupure $\mu$. Les itérations de* ChebFilterCG *sont stoppées quand l'erreur relative en norme* $\mathbf{A}$ *est inférieure à* $10^{-8}$*. Le problème test considéré est EDP2.*

Dans la table 4.9, nous illustrons les gains issus de l'utilisation de l'algorithme Init-CG. Les itérations sont stoppées quand l'erreur relative en norme $\mathbf{A}$ est inférieure à $10^{-8}$. Le méthode du gradient conjugué sans projection initiale du résidu effectue 1 060 itérations avec un coût de 6.31 `Gflop`. Nous indiquons également le nombre d'itérations du gradient conjugué (`Nit`), le nombre d'opérations flottantes (`Gflop`), et le nombre de vecteurs d'amortissement (`Amor. Vec`), et ceci pour chaque niveau de filtrage $\varepsilon$.

Nous tirons de ces résultats les mêmes conclusions générales qu'avec les cas d'essai de plus petite taille dans la section précédente. En particulier,

pour un niveau de filtrage $\varepsilon$ fixé, le nombre d'opérations flottantes (`Gflop`) ne change pas beaucoup quand le paramètre $\mu$ varie. Il est juste important de noter que, pour des systèmes linéaires de très grande taille, un cycle à deux grilles peut aussi être considéré pour assurer la robustesse numérique de cette approche (voir § 3.2.6).

| | CG sans préconditionnement de Tchebycheff requiert 1 060 itérations avec un coût de 6.31 `Gflop` | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | $\mu = \lambda_{\max}/1000$ | | | | $\mu = \lambda_{\max}/500$ | | | | $\mu = \lambda_{\max}/200$ | | | |
| $\varepsilon$ | `k` | `Nit` | `Gflop` | `Amor.Vec` | `k` | `Nit` | `Gflop` | `Amor.Vec` | `k` | `Nit` | `Gflop` | `Amor.Vec` |
| $10^{-16}$ | 8 | 274 | 1.64 | 5 | 11 | 230 | 1.38 | 4 | 18 | 265 | 1.59 | 5 |
| $10^{-8}$ | 11 | 291 | 1.74 | 3 | 16 | 246 | 1.48 | 3 | 25 | 252 | 1.55 | 3 |
| $10^{-4}$ | 15 | 343 | 2.06 | 2 | 22 | 259 | 1.56 | 2 | 36 | 312 | 1.88 | 2 |
| $10^{-2}$ | 21 | 320 | 1.92 | 1 | 31 | 324 | 1.96 | 1 | 49 | 341 | 2.07 | 1 |

TAB. 4.9 – *Coûts de* Init-CG*, pour une base* $\mathbf{W}$ *calculée avec différents niveaux du filtrage* $\varepsilon$ *et différentes valeurs de coupure* $\mu$*. Les itérations de* Init-CG *sont stoppées quand l'erreur relative en norme* $\mathbf{A}$ *est inférieure à* $10^{-8}$*. Le problème test considéré est EDP2.*

### 4.5.3 Filtrage Initial Additionnel dans l'Algorithme ChebFilterCG

Quand le paramètre $\mu$ est mal choisi, ou quand la matrice d'itération possède une mauvaise distribution de valeurs propres, nous observons parfois que la base de Krylov devient très grande en taille. Un remède pour réduire la dimension de cette base est d'appliquer un filtrage initial additionnel au vecteur de départ, avec un niveau de filtrage $\varepsilon_{\texttt{init}} = 10^{-16}$. Ceci va permettre au gradient conjugué de travailler directement dans le complémentaire orthogonal d'un grand nombre de vecteurs propres, c-à-d. ceux qui sont associés à toutes les valeurs propres de l'intervalle $[\mu_{\texttt{init}}, \lambda_{\max}]$, où $\mu_{\texttt{init}}$ est plus petit que la valeur de coupure $\mu$ qui sera exploitée pour la construction de la base de Krylov. L'intérêt de différencier ces deux valeurs de coupure et de filtrage associée, est de payer éventuellement un coût de préparation plus important pour bénéficier de propriétés numériques fortes dans le résidu initial, et de garder par contre un coût raisonnable dans le préconditionneur polynomial (qui lui est appliqué à chaque itération).

Dans la table 4.10, nous montrons, pour le problème test EDP2 préconditionné par Cholesky Incomplet sans remplissage, l'effet du rajout de ce filtrage initial du vecteur de départ sur la dimension de la base de Krylov générée. Dans cette table, nous indiquons le nombre d'étapes du filtrage de Tchebycheff effectuées à chaque itération du gradient conjugué (`ChebIt`), le nombre d'itérations du gradient conjugué (`CGIt`), et le nombre d'itérations du gradient conjugué avec le filtrage initiale additionnel ($\texttt{CGIt}_{\texttt{init}}$). Par exemple, dans le cas où le niveau du filtrage $\varepsilon = 10^{-2}$ et le paramètre $\mu = \lambda_{\max}/200$, la dimension de la base de Krylov est égale à 49. L'utilisation d'un filtrage initial additionnel sur le vecteur de départ, avec $\mu_{\texttt{init}} = \lambda_{\max}/1000$ permet de réduire la taille de cette base de Krylov à 26, ainsi que le coût total en opérations flottantes de l'algorithme ChebFilterCG.

Comme dernier commentaire, nous mentionnons que, pour une valeur fixe de $\mu$, avec un filtrage initial additionnel tel que $\varepsilon_{\texttt{init}} = 10^{-16}$ et $\mu_{\texttt{init}}$ est inférieure à $\mu$, nous obtenons une base de dimension petite même avec des niveaux du filtrage plus ou moins grands ($\varepsilon \in [10^{-4}, 10^{-1}]$). La qualité de cette base est la même que celle obtenue avec un niveau du filtrage $\varepsilon = 10^{-16}$ mais sans filtrage initial additionnel. Par exemple, quand le niveau du filtrage $\varepsilon$ égal à $10^{-16}$ et la valeur de coupure $\mu$ égale à $\lambda_{\max}/200$, la dimension de la base de Krylov est 18 et le coût est de l'ordre de 25.44 `Gflop`. Un filtrage initial additionnel du vecteur de départ avec $\mu_{\texttt{init}} = \lambda_{\max}/1000$, permet de réduire le coût total de ChebFilterCG à 10.70 `Gflop`, pour une base finale de Krylov de dimension 20 obtenue cette fois avec la même valeur de $\mu$ et un niveau du filtrage $\varepsilon = 10^{-4}$.

| | | | | $\mu_{\texttt{init}} = \lambda_{\max}/500$ $\varepsilon_{\texttt{init}} = 10^{-16}$ $\texttt{ChebIt}_{\texttt{init}} = 420$ | | $\mu_{\texttt{init}} = \lambda_{\max}/1000$ $\varepsilon_{\texttt{init}} = 10^{-16}$ $\texttt{ChebIt}_{\texttt{init}} = 594$ | |
|---|---|---|---|---|---|---|---|
| Valeur de $\varepsilon$ | `ChebIt` | `CGIt` | `Gflop` | $\texttt{CGIt}_{\texttt{init}}$ | `Gflop` | $\texttt{CGIt}_{\texttt{init}}$ | `Gflop` |
| $10^{-2}$ | 38 | 49 | 10.18 | 33 | 9.08 | 26 | 8.55 |
| $10^{-4}$ | 70 | 36 | 13.60 | 25 | 11.67 | 20 | 10.70 |
| $10^{-16}$ | 265 | 18 | 25.44 | | | | |

TAB. 4.10 – *Coût de* ChebFilterCG *pour différents niveaux du filtrage $\varepsilon$ et une valeur de coupure $\mu = \lambda_{\max}/200$. L'itéré initial est filtré avec $\varepsilon_{\texttt{init}}$ et $\mu_{\texttt{init}}$. Les itérations du gradient conjugué sont stoppées quand l'erreur relative en norme $\mathbf{A}$ est inférieure à $10^{-8}$. Le problème test considéré est EDP2.*

## 4.6 Conclusion

Dans ce chapitre, nous avons proposé une nouvelle approche basée sur une combinaison de la méthode du gradient conjugué avec des filtres polynomiaux de Tchebycheff comme préconditionneurs. Cette technique vise à mettre la méthode du gradient conjugué dans un mode particulier, où le conditionnement du système linéaire n'est pas tellement réduit, mais son spectre en grande partie regroupé autour de 1. Notre préconditionneur consiste à appliquer les filtres polynomiaux de Tchebycheff à une partie seulement du spectre de la matrice d'itération, et décaler la quasi totalité des valeurs propres du système linéaire près de 1, sans dégrader pour autant la distribution des plus petites valeurs propres. La méthode résultant de cette combinaison Tchebycheff–Krylov se distingue par sa capacité à construire une base de Krylov de taille petite, très riche en ce qui concerne les vecteurs propres associés aux plus petites valeurs propres. Cette base de Krylov peut être réutilisée dans un deuxième niveau de préconditionnement pour une multirésolution, c'est-à-dire une résolution d'une séquence de systèmes linéaires avec la même matrice mais avec différents second membres. La technique de préconditionnement à base de polynômes de Tchebycheff requiert principalement la multiplication d'un vecteur par la matrice initiale et quelques mises à jour de vecteur, mais très peu de produits scalaires. Cette remarque est très importante dans le contexte du calcul parallèle, et en particulier dans les environnements à mémoire distribuée où le calcul des produits scalaires exigent une attention particulière. Enfin, nos algorithmes ont été testés sur des problèmes d'Équations aux Dérivées Partielles (EDP). Ces tests confirment tous les résultats théoriques escomptés. Il nous reste enfin à étudier cette approche dans le cadre d'une multirésolution issue d'un processus de linéarisation et construire un préconditionneur adaptatif comme dans le chapitre suivant, basé sur l'information des sous-espaces de Krylov produite aux étapes précédentes dans l'itération non-linéaire.

# Chapitre 5

# Compléments, Conclusions et Perspectives

Dans ce chapitre d'ouverture, nous examinons tout d'abord certains cas pathologiques pour la méthode du Chapitre 4 et proposons des remèdes, comme par exemple l'utilisation du gradient conjugué par blocs. Nous introduisons également une nouvelle technique adaptative de préconditionnement dans le cadre d'une multirésolution issue d'un processus de linéarisation. Notre préconditionneur est basé sur l'information des sous-espaces de Krylov produite aux étapes précédentes dans l'itération non-linéaire. Notre approche sera validée sur des problèmes d'équations aux dérivées partielles non-linéaires en mécanique des fluides (Navier–Stokes) et sur des méthodes de décomposition de domaine pour la résolution de problèmes elliptiques.

## 5.1 Cas Pathologique et Accélération par le Gradient Conjugué par Blocs

Dans ce paragraphe, nous nous sommes principalement intéressés aux problèmes induisant une très mauvaise convergence du gradient conjugué.

Nous considérons le cas de la matrice test décrite dans le Chapitre 2 au § 2.4, pour laquelle nous nous sommes particulièrement intéressés au type de convergence relativement étrange qu'elle induit pour le gradient conjugué classique. La courbe de l'erreur, en fonction des itérations, comporte un certain nombre de plateaux (cf. figure 5.1). La précision de la solution obtenue est relativement moyenne, après 470 itérations, la valeur du critère d'arrêt n'étant que de $10^{-7}$, sachant que la taille du système linéaire est 137.

La figure 5.2 montre le spectre de la matrice $\mathbf{A}$ obtenue sur ce problème test. Nous pouvons tout d'abord remarquer le nombre de conditionnement très grand de cette matrice, qui est de l'ordre de $2.38 \cdot 10^{13}$, étant donné

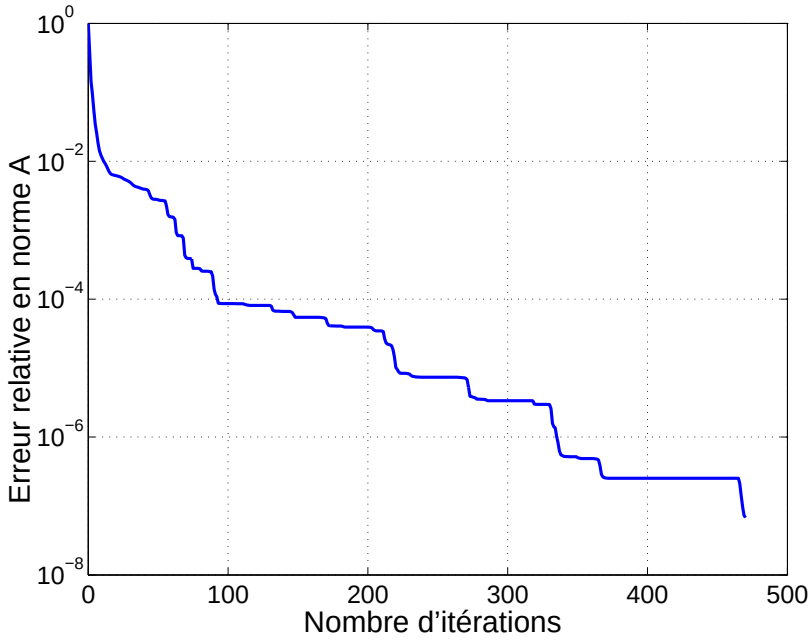
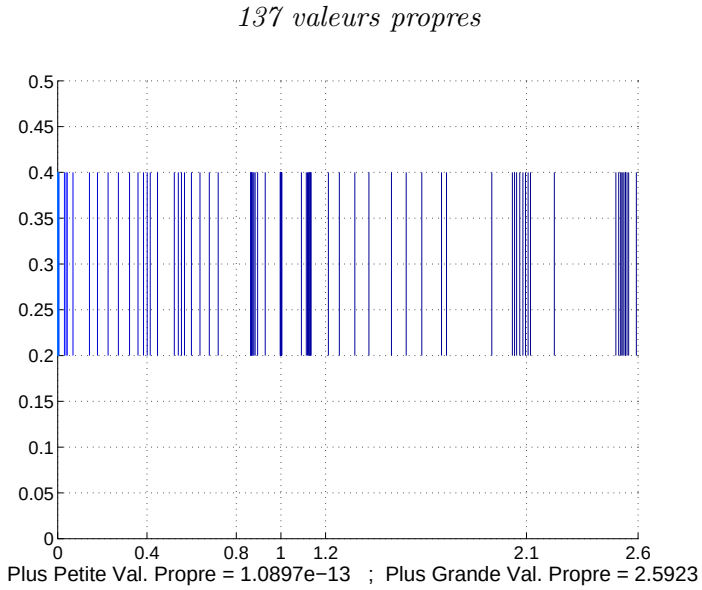



FIG. 5.1 – *Courbe de Convergence du gradient conjugué classique.*

que la plus petite des valeurs propres est $1.08 \cdot 10^{-13}$, et que la plus grande vaut 2.59. De plus, nous pouvons observer la forte concentration des valeurs propres autour de la valeur 1, sachant qu'il n'y a que 48 valeurs propres en dehors de l'intervalle $[1-0.5\cdot10^{-2}, 1+0.5\cdot10^{-2}]$. Malgré cela, nous voyons que l'algorithme du gradient conjugué nécessite bien plus de 48 itérations, sans atteindre pour autant un régime de convergence linéaire. Il faut aussi noter les divers amas de valeurs propres, avec en particulier un amas de 12 valeurs propres à l'extrémité gauche. Le comportement erratique de l'algorithme du gradient conjugué est dû à ces amas de valeurs propres aux extrêmes combinés au très mauvais conditionnement de la matrice d'itération. Ce mauvais comportement du gradient conjugué, en présence d'amas de valeurs propres, a déjà été observé et analysé dans [64, 86].

L'utilisation des filtres de Tchebycheff en tant que préconditionneurs peut améliorer quelque peu la situation, comme illustré dans la figure 5.3. Pour ces tests, le niveau de filtrage $\varepsilon$ a été fixé à $10^{-16}$, et nous faisons varier la valeur de coupure $\mu$. Les courbes de convergence de l'algorithme ChebFilterCG pour ces différentes valeurs de $\mu$ se décomposent en deux parties différentes. Une première partie correspond à une convergence linéaire semblable et presque indépendante du paramètre $\mu$. Quant à la deuxième partie, elle correspond à une phase de convergence comportant un certain



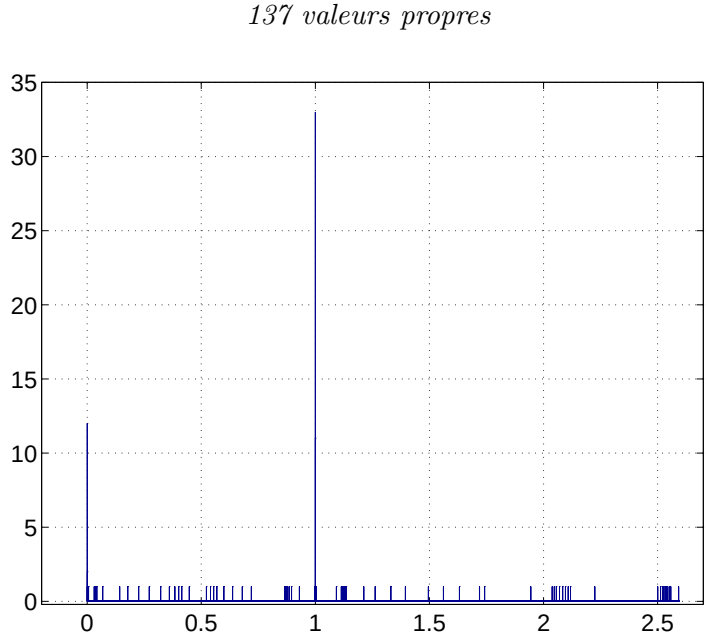




FIG. 5.2 – *Distribution des valeurs propres de la matrice* **A** *La plus petite valeur propre* $\lambda_{\min} = 1.08 \cdot 10^{-13}$ *et la plus grande valeur propre* $\lambda_{\max} = 2.59$. *Le spectre est calculé par* ARPACK.

nombre de plateaux, avec une amélioration moyenne de la précision. En général, l'algorithme ChebFilterCG converge plus rapidement que le gradient conjugué classique. Avec une valeur de coupure $\mu = \lambda_{\max}/100$ et un niveau de filtrage $\varepsilon = 10^{-16}$, par exemple, ChebFilterCG diminue d'un facteur 12 le nombre d'itérations comparé au gradient conjugué classique, pour une mesure de l'erreur (au sens de la norme $\mathbf{A}$) inférieure à $10^{-6}$. Cependant, pour des niveaux de précision inférieurs, on voit apparaître à nouveau des plateaux de taille plus ou moins grande indiquant que, dans cet exemple "*pathologique*", l'algorithme ChebFilterCG souffre des mêmes maux que ceux que le gradient conjuguée classique peut rencontrer.

En effet, malgré le filtrage implicite de l'information spectrale dans l'intervalle $[\mu, \lambda_{\max}]$, le spectre de $\mathbf{A}$ exhibe un certain nombre d'amas de valeurs propres dans l'intervalle $]0, \mu]$, et un conditionnement réduit $(\mu/\lambda_{\min})$ toujours très élevé (en tout cas pour des valeurs raisonnables de $\mu$).

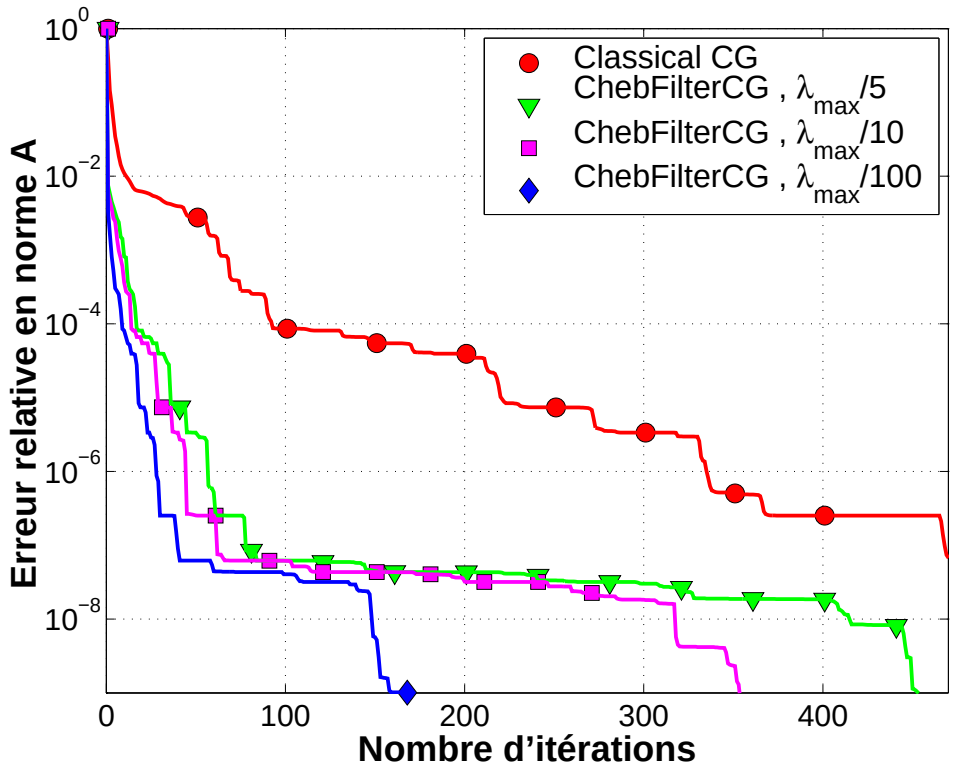



FIG. 5.3 – *Courbe de Convergence du gradient conjugué classique et de* ChebFilterCG *pour différentes valeurs de coupure $\mu$. Le niveau de filtrage $\varepsilon$ est fixé à $10^{-16}$.*

Dans ce qui suit, nous utiliserons ce même problème test pour proposer et valider expérimentalement des alternatives ne présentant pas un comportement erratique aussi marqué.

Par opposition à l'algorithme classique du gradient conjugué, l'algorithme du gradient conjugué par blocs permet l'identification de valeurs propres multiples et, de ce fait, présente certaines dispositions pour mieux se comporter en présence d'amas de valeurs propres. Une bonne description de ce type de technique de Krylov par blocs, ainsi qu'une étude détaillée de leurs propriétés de convergence, ont été réalisées par O'Leary [53] et Arioli et al. [2]. Nous considérons également la combinaison du gradient conjugué par blocs avec les filtres polynomiaux de Tchebycheff comme préconditionneurs, et que l'on dénotera par ChebFilterBlockCG.

Soit $s$ la taille de bloc dans l'algorithme du gradient conjugué par blocs, et soit $\mathbf{B}$ la matrice rectangulaire de $\mathbb{R}^{n\times s}$ correspondant au bloc de vecteurs du second membre. $\mathbf{B}$ représente en fait un ensemble de $s$ seconds membres, et l'accélération par le gradient conjugué par blocs peut être considérée comme un algorithme permettant de résoudre simultanément les $s$ systèmes linéaires suivants,

$$\mathbf{A}\mathbf{x}_\ell = \mathbf{b}_\ell \ , \quad \ell = 1, \ldots, s$$

où chacun des vecteurs $\mathbf{b}_\ell$ et $\mathbf{x}_\ell$, $\ell = 1, \ldots, s$, correspond à la $\ell^{\text{ème}}$ colonne de la matrice $\mathbf{B}$, et $\mathbf{X}$ respectivement. Pour plus de détails, nous référons à [68, § IX].

Dans les expériences numériques qui suivent, le bloc de départ $\mathbf{X}^{(0)}$ est pris égal à $\mathbf{0}$. Nous avons aussi considéré le fait de ne disposer que d'un seul second membre $\mathbf{b}$ à la fois, et nous avons donc généré les $s-1$ autres colonnes de $\mathbf{B}$ de manière aléatoire. Le critère d'arrêt utilisé dans ces essais numériques est l'erreur relative en norme $\mathbf{A}$. Cette mesure de l'erreur n'est calculée que pour le système linéaire considéré (associé au premier vecteur colonne de la matrice $\mathbf{B}$), et ne fait pas intervenir l'erreur commise sur les $s-1$ autres systèmes générés aléatoirement.

La figure 5.4 montre, pour notre problème test, la convergence de l'algorithme Classical BlockCG (gradient conjugué par blocs) obtenue pour différentes tailles de bloc $s$. Nous observons que, quelle que soit la taille de bloc (à part pour $s = 4$, qui ne semble pas suffisamment élevée), le comportement de Classical BlockCG est le même. Les améliorations apportées par la technique d'accélération par blocs, peuvent se voir facilement sur les courbes de convergences de l'algorithme Classical BlockCG, pour la taille de bloc de 8, par exemple. Dans ce cas, une valeur de l'erreur relative en norme $\mathbf{A}$ inférieure à $10^{-9}$ est obtenue en 18 itérations, ce qui équivaut, en nombre de produits matrice-vecteur effectués, à $18 \times 8 = 146$ itérations du gradient conjugué classique. Or nous avons vu précédemment (cf. figure 5.1), que l'algorithme du gradient conjugué classique nécessite 470 itérations, avant

d'atteindre $10^{-7}$, et n'arrive pas à atteindre la précision $10^{-9}$.

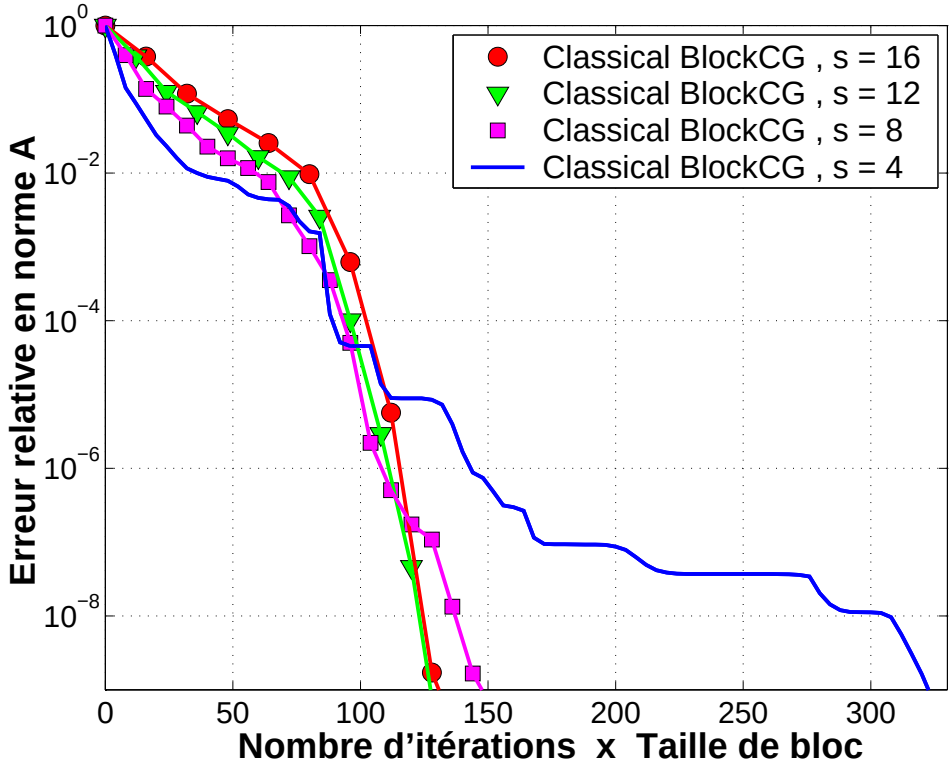



FIG. 5.4 – *Comportement du gradient conjugué par blocs* Classical BlockCG *pour différentes tailles de bloc* $s$.

Le comportement numérique de ChebFilterBlockCG est illustré dans la figure 5.5. Sont représentées les courbes de l'erreur relative en norme $\mathbf{A}$, pour différentes valeurs de coupure $\mu$, un niveau de filtrage $\varepsilon$ fixé à $10^{-16}$, et une la taille de bloc $s = 8$. Nous montrons également les courbes de convergence du gradient conjugué par blocs pour la même taille de bloc $s = 8$. Nous pouvons remarquer que, quelle que soit la valeur du paramètre de coupure $\mu$, le comportement de ChebFilterBlockCG est presque le même. Les améliorations apportées par cette technique peuvent se voir facilement, avec, pour la valeur de coupure $\mu = \lambda_{\max}/10$ par exemple, une réduction de la dimension de l'espace de Krylov généré d'un facteur de 3.5 environ, pour une précision finale en norme $\mathbf{A}$ de $10^{-9}$.

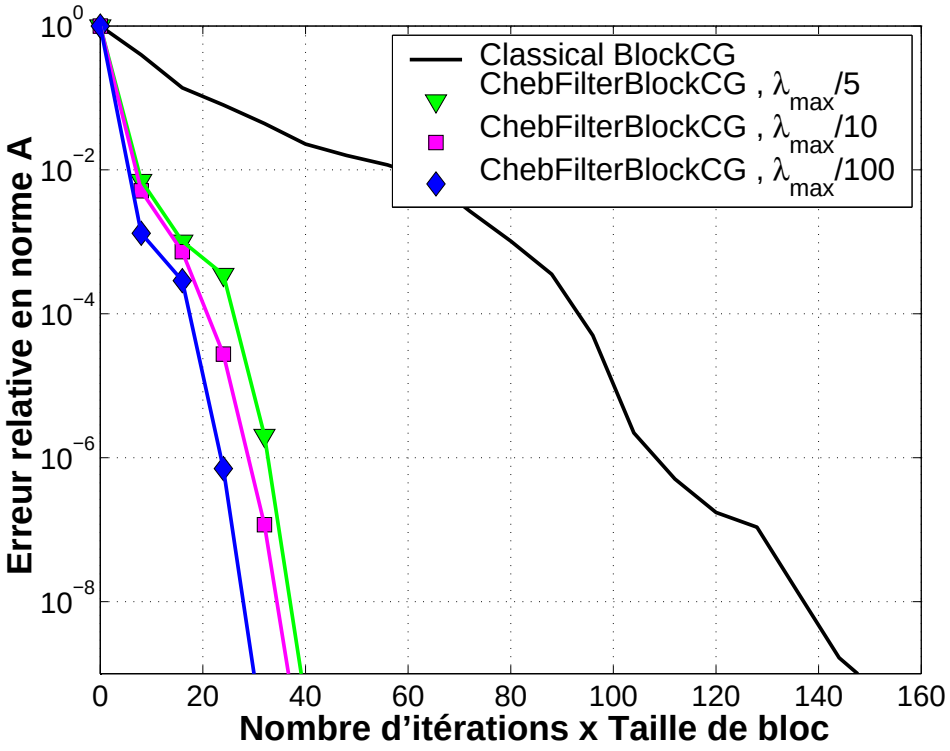



FIG. 5.5 – *Comportement du gradient conjugué par blocs* Classical BlockCG *et de* ChebFilterBlockCG *pour différentes valeurs de coupure* $\mu$. *La taille du bloc* $s$ *est égal à 8 et le niveau de filtrage* $\varepsilon$ *dans* ChebFilterBlockCG *est égal à* $10^{-16}$.

Dans la figure 5.6, nous fixons cette fois la taille de bloc $s$ à la valeur 8, et le paramètre de coupure à $\lambda_{\max}/100$, et nous faisons varier le niveau de filtrage $\varepsilon$. Les courbes de convergence montrent que l'algorithme ChebFilterBlockCG conserve le même type de comportement numérique dans tous les cas, indépendamment du choix du niveau de filtrage $\varepsilon$, Bien évidemment, nous observons une influence directe du niveau de filtrage $\varepsilon$ sur la dimension de l'espace de Krylov résultant. Comme déjà discuté au Chapitre 4, la stratégie pour minimiser la taille de cet espace de Krylov est de mettre le poids sur le filtrage de Tchebycheff, en considérant des valeurs de $\varepsilon$ plus petites que $10^{-8}$ (la racine carré de la précision machine).

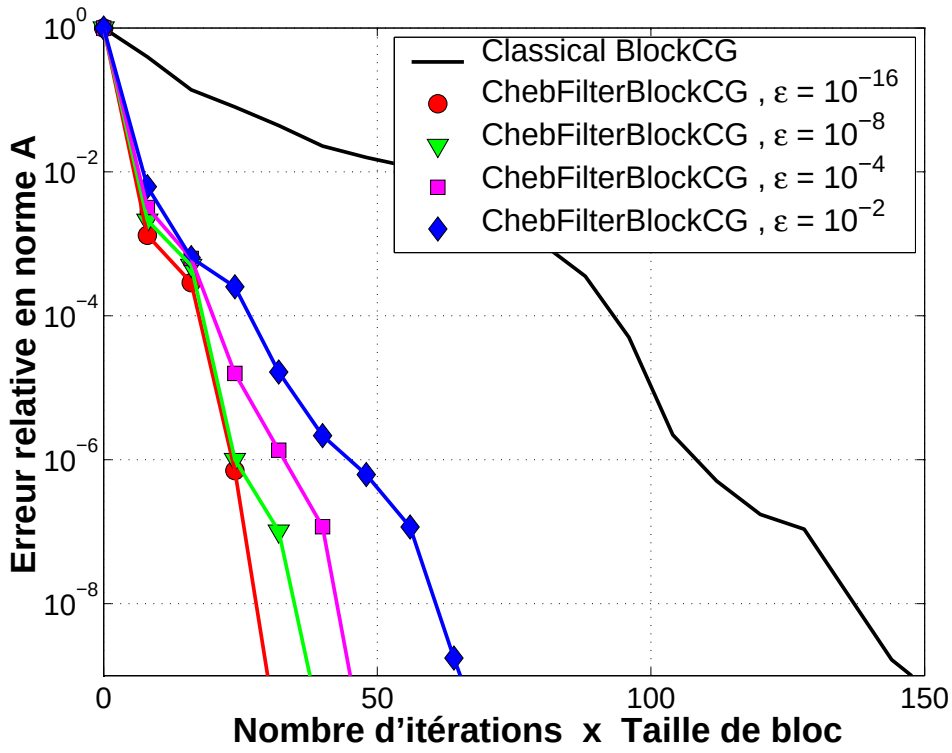



FIG. 5.6 – *Comportement du* ChebFilterBlockCG *pour différents niveaux de filtrage $\varepsilon$. La valeur de coupure $\mu$ est fixée à $\lambda_{\max}/100$ et la taille du bloc $s$ est égale à 8.*

Dans la figure 5.7, nous montrons les courbes de convergence de ChebFilterBlockCG obtenues pour différentes tailles de bloc $s$. Nous fixons le paramètre de coupure $\mu$ à $\lambda_{\max}/100$, et le niveau de filtrage $\varepsilon$ à $10^{-16}$. Nous observons que, quelle que soit la taille de bloc, le comportement de ChebFilterBlockCG est sensiblement le même.

La remarque fondamentale que l'on peut retirer de l'ensemble de ces résultats est qu'il est possible, en combinant des techniques de Krylov par blocs avec les filtres de Tchebycheff, d'obtenir, même dans les cas extrêmes, une base de Krylov de taille raisonnable. Par exemple, dans le cas de ce problème test, nous obtenons une base de Krylov filtrée de dimension 32, avec pour paramètres $\mu = \lambda_{\max}/100$, $\varepsilon = 10^{-16}$, et $s = 8$ dans ChebFilterBlockCG, et un critère d'arrêt en norme $\mathbf{A}$ à $10^{-9}$. Cette dimension n'est en effet pas trop éloignée du nombre total de valeurs propres de $\mathbf{A}$ contenues dans l'intervalle $]0, \mu]$, qui est égal 19 dans le cas présent. Dans le paragraphe

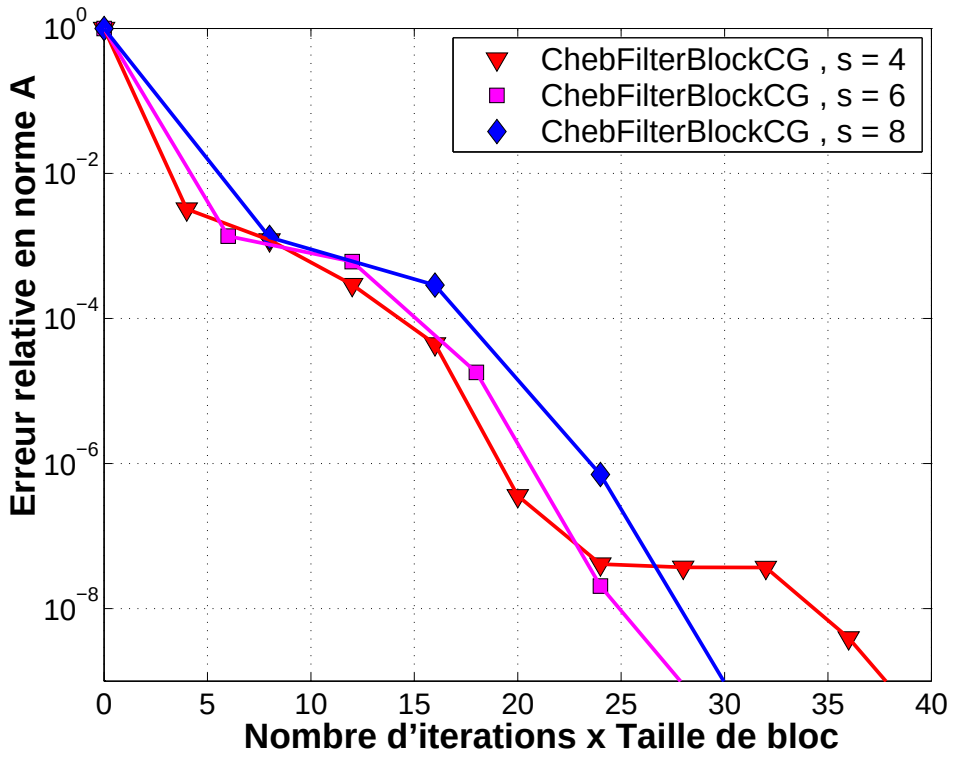



FIG. 5.7 – *Comportement du* ChebFilterBlockCG *pour différentes taille du bloc $s$. La valeur de coupure $\mu$ est fixée à $\lambda_{\max}/100$ et le niveau de filtrage $\varepsilon$ est égal à $10^{-16}$.*

suivant, nous discuterons succinctement de la réutilisation de cette base de Krylov dans le cadre d'une multirésolution.

Nous nous sommes intéressés, comme précédemment, à résoudre une séquence de système linéaires impliquant une même matrice mais des second-membres différents (cf. (4.9)). L'idée principale est de résoudre le premier système de la séquence (4.9) par la méthode du gradient conjugué par blocs combiné avec les filtres polynomiaux de Tchebycheff (ChebFilterBlockCG), et exploiter ensuite la base de Krylov résultante dans les techniques vues au Chapitre 3, telles que Init-CG, Def-CG, Proj-CG, etc, pour la résolution des autres systèmes dans cette séquence.

Pour illustrer cette stratégie, nous considérons toujours le cas pathologique décrit précédemment (cf. Chapitre 2 au § 2.4). Un premier système $\mathbf{A}\mathbf{x}_1 = \mathbf{b}_1$ est d'abord résolu, puis un second $\mathbf{A}\mathbf{x}_2 = \mathbf{b}_2$. Pour le premier système, on utilisera l'algorithme (ChebFilterBlockCG), avec pour paramètres

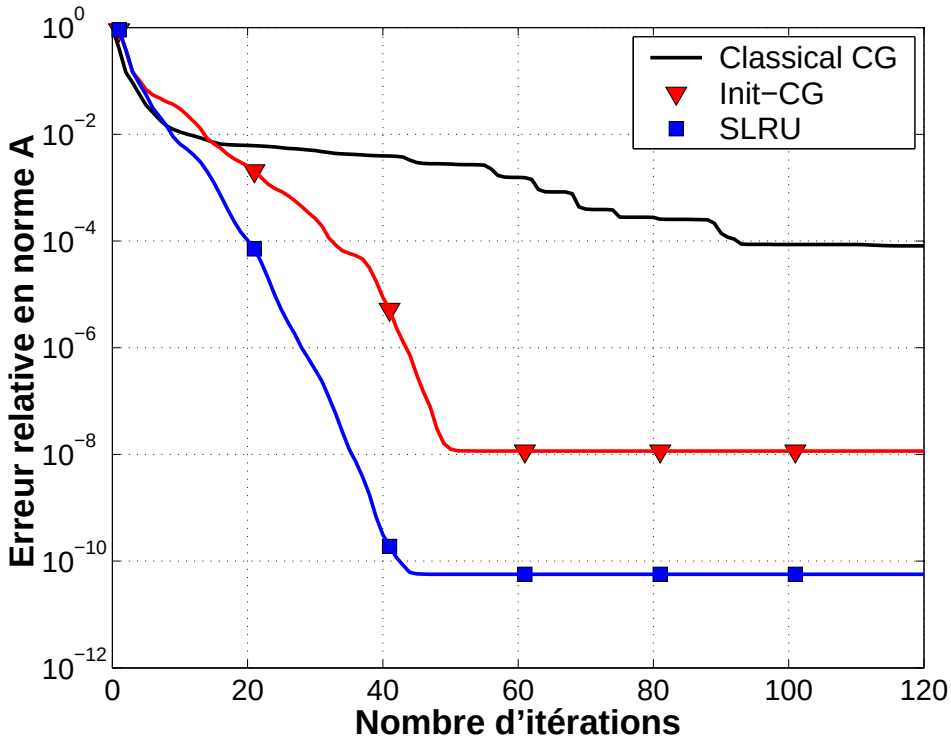



FIG. 5.8 – *Courbe de Convergence pour une base de Krylov filtrée de dimension 32 avec pour paramètres* $\mu = \lambda_{\max}/100$, $\varepsilon = 10^{-16}$, *et* $s = 8$ *dans* ChebFilterBlockCG, *et un critère d'arrêt en norme* $\mathbf{A}$ *à* $10^{-9}$.

$\mu = \lambda_{\max}/100$, $\varepsilon = 10^{-16}$, $s = 8$, et un critère d'arrêt en norme $\mathbf{A}$ à $10^{-9}$. Quant au second système, le second membre $\mathbf{b}_2$ est choisi de sorte que la solution soit un vecteur avec des nombres aléatoires qui suivent la loi normale $\mathcal{N}(0, 1)$. Pour ces tests, nous considérons l'Algorithme 3.1 (Init-CG) et Algorithme 3.4 (SLRU) et nous contrôlons l'erreur relative en norme $\mathbf{A}$ jusqu'à 120 itérations sachant que la taille du système est égale à 137.

La figure 5.8, montre une amélioration très significative lorsqu'on réutilise le sous-espace de Krylov généré par ChebFilterBlockCG, dans le cadre d'une multirésolution. Nous traçons la courbe de convergence de l'erreur relative en norme $\mathbf{A}$ en fonction du nombre d'itérations, respectivement pour le gradient conjugué classique (Classical CG), le gradient conjugué avec projection initiale (Init-CG), et l'algorithme SLRU (cf. Algorithme 3.1 et 3.4). La figure 5.8, montre également que la base de Krylov générée par ChebFilterBlockCG est très efficace et permet à la méthode du gradient conjugué avec projection initiale (Init-CG) de converger cette fois, sachant qu'on a un

mauvais conditionnement réduit dans l'intervalle $]0, \mu[$ qui reste de l'ordre de $\mu/\lambda_{\min} = 2.82 \cdot 10^{11}$. Nous pouvons remarquer aussi que la convergence de SLRU est plus rapide que celle de Init-CG. C'est l'avantage d'établir à chaque itération du gradient conjugué une projection oblique, avec la base de Krylov résultant de la première résolution avec ChebFilterBlockCG, pour maintenir une séparation constante entre les composantes propres correspondant aux plus petites valeurs propres et les autres.

Cependant, avec une taille de bloc $s$ non appropriée, ChebFilterBlockCG peut parfois fournir une base de Krylov avec un conditionnement très élevé. Ceci entache le projecteur oblique d'erreurs numériques et peut conduire Init-CG à reproduire les plateaux dans sa courbe de convergence. Pour remédier à cela, on peut toujours recourir au cycle à deux grilles (cf. 3.2.6) ou employer des méthodes plus robustes numériquement à savoir SLRU ou Def-CG (cf. Chapitre 3).

## 5.2 Préconditionnement Adaptatif pour des Problèmes d'Équations Non-Linéaires

Dans ce paragraphe, nous décrivons et analysons une nouvelle technique de préconditionnement adaptatif dans le cadre d'une multirésolution issue d'un processus de linéarisation. Notre préconditionneur est basé sur l'information des sous-espaces de Krylov produite aux étapes précédentes dans l'itération non-linéaire. En particulier, nous utilisons une approche adaptative proposée par Baglama et al. [7] pour GMRES avec redémarrage, qui permet d'enrichir les préconditionneurs existants à l'aide de l'information provenant des précédentes étapes de l'itération non-linéaire. Notre approche a été validée sur des problèmes d'équations aux dérivées partielles non-linéaires en mécanique des fluides (Navier–Stokes) et sur des méthodes de décomposition de domaine pour la résolution de problèmes elliptiques.

### 5.2.1 Description du Problème

Nous nous intéressons à la résolution d'une suite de systèmes linéaires de la forme

$$\mathbf{A}_\ell \mathbf{x}_\ell = \mathbf{b}_\ell, \tag{5.1}$$

où $\mathbf{A}_\ell$ est une matrice creuse et de grande taille. Ce type de problèmes apparaît habituellement lors de la linéarisation d'un système d'équations non-linéaires de la forme

$$\mathcal{F}(\mathbf{x}) = 0, \tag{5.2}$$

où $\mathcal{F} : \mathbb{R}^n \to \mathbb{R}^n$ et $\mathbf{x} \in \mathbb{R}^n$. Par exemple, dans le cas d'une linéarisation par Newton de (5.2), $\mathbf{A}_\ell$ est une approximation (éventuellement précondition-

née) de la matrice Jacobienne, alors que $\mathbf{x}_\ell$ est la nouvelle direction et $\mathbf{b}_\ell$ est l'opposé du résidu.

Dans ce qui suit, nous supposons que :
(i) $\mathbf{A}_\ell$ sont des matrices préconditionnées avec un spectre bien regroupé ;
(ii) $\mathbf{A}_{\ell+1} = \mathbf{A}_\ell + \mathcal{E}_\ell$ où $\|\mathcal{E}_\ell\| \to 0$ quand $\ell \to \infty$ ;
(iii) nous adoptons GMRES pour résoudre (5.1).

Le choix d'une méthode itérative est essentiel dans la conception d'un préconditionneur. En particulier, notre approche ne constitue pas une première tentative pour recycler l'information spectrale des sous-espaces de Krylov engendrés par GMRES, afin d'accélérer la convergence. Deux aspects sont récurrents : l'accélération de la convergence et le préconditionnement dans le contexte de (i) GMRES avec redémarrage [7, 8, 9, 18, 47, 49, 50], et, plus récemment, (ii) le cas de systèmes linéaires à second membres multiples [10, 21]. Pour les matrices symétriques définies positives, une approche similaire figure dans [62, 73, 89]. On peut aussi noter l'application de ces idées à des séquences de systèmes linéaires du type (5.1) [27, 60].

Un aspect notable, dans la quête d'un préconditionneur adaptatif, est l'idée du sous-espace invariant. Selon les cas, on peut déflater (ou décaler vers 1) les valeurs propres de la matrice du système associée à ce sous-espace. La distribution des valeurs propres qui résulte de ce traitement peut favoriser une convergence plus rapide. Dans cet esprit, diverses techniques de déflation et d'algorithmes de "*recherche d'invariance*" ont été conçus. Cependant, pour des applications réelles, la construction de sous-espaces invariants peut se révéler assez coûteuse.

Notre approche, basée sur [7], est plus simple, mais atteint l'un des objectifs de ces techniques "*orientées invariance*" – à savoir décaler les valeurs propres vers 1. Comme nous le montrons dans le paragraphe suivant, cet objectif est atteint directement (i.e. sans tri supplémentaire de l'information spectrale), étant donnée l'information dans GMRES disponible sous la forme d'une matrice de Hessenberg et de la base de Arnoldi. De plus, notre approche préserve le caractère défini-positif.

Le reste de ce chapitre est structuré de la façon suivante. Au paragraphe suivant nous introduisons notre préconditionneur et discutons certaines de ses propriétés. Nous décrivons également une technique incrémentale de préconditionnement pour la résolution de (5.1). En particulier, nous indiquons comment l'utilisation de préconditionneurs par blocs dans GMRES peut conduire à des implantations efficaces d'un de point vue coût et performance pour la construction de nos préconditionneurs adaptatifs. Enfin, pour illustrer l'efficacité de notre technique, nous utilisons deux problèmes génériques : décomposition de domaine pour des problèmes non-linéaires elliptiques et les équations de Navier–Stokes incompressibles en régime stationnaire.

### 5.2.2 Préconditionneurs Adaptatifs

Soit $\mathbf{A} \in \mathbb{R}^{n\times n}$ et $\mathbf{V}^T\mathbf{AV} = \mathbf{H}$ la décomposition de Hessenberg de $\mathbf{A}$, où $\mathbf{V} \in \mathbb{R}^{n\times n}$ est une matrice orthonormale de colonnes $\mathbf{v}_i$, $1 \leq i \leq n$. Soit $\mathbf{V} = [\mathbf{V}_k \ \mathbf{V}_{n-k}]$ une partition de l'espace engendré par les colonnes de $\mathbf{V}$ telle que $\mathbf{V}_k^T\mathbf{AV}_k = \mathbf{H}_k \in \mathbb{R}^{k\times k}$ soit une matrice de Hessenberg non-singulière, irréductible (i.e., avec des valeurs sous-diagonales non nulles).
En écrivant $h = \mathbf{H}_{k+1,k}$, la matrice $\mathbf{A}$ satisfait les relations suivantes :

$$\begin{aligned} \mathbf{AV}_k &= \mathbf{V}_k\mathbf{H}_k + h\mathbf{v}_{k+1}\mathbf{e}_k^T \quad \text{(cf. Proposition 1.2)} \\ \text{et} & \\ \mathbf{V}^T\mathbf{AV} &= \begin{pmatrix} \mathbf{H}_k & \mathbf{V}_k^T\mathbf{AV}_{n-k} \\ h\mathbf{e}_1\mathbf{e}_k{}^T & \mathbf{V}_{n-k}^T\mathbf{AV}_{n-k} \end{pmatrix} = \begin{pmatrix} \mathbf{H}_k & \mathbf{F} \\ h\mathbf{E} & \mathbf{G} \end{pmatrix}, \end{aligned}$$

où, $\mathbf{e}_1$ et $\mathbf{e}_k$ sont respectivement le $1^{\text{er}}$ et le $k^{\text{ème}}$ vecteur de la base canonique de $\mathbb{R}^n$.
Définissons maintenant l'opérateur

$$\mathbf{M}_\alpha = \mathbf{I}_n - \mathbf{V}_k\mathbf{V}_k^T + \mathbf{V}_k\mathbf{H}_k\mathbf{V}_k^T + \alpha\mathbf{v}_{k+1}\mathbf{v}_k^T, \quad \alpha \in \mathbb{R} \tag{5.3}$$

d'inverse

$$\mathbf{M}_\alpha^{-1} = \mathbf{I}_n - \mathbf{V}_k\mathbf{V}_k^T + \Big(\mathbf{V}_k - \alpha\mathbf{v}_{k+1}\mathbf{e}_k^T\Big)\mathbf{H}_k^{-1}\mathbf{V}_k^T. \tag{5.4}$$

Notre technique de préconditionnement est basée sur le résultat suivant :

**Proposition 5.1.** *Soit $\mathbf{A} \in \mathbb{R}^{n\times n}$ et $\mathbf{V}^T\mathbf{AV} = \mathbf{H}$ sa décomposition de Hessenberg et soit $\mathbf{M}_\alpha$ défini comme en (5.3). Pour tout $\alpha \neq h$, $\lambda = 1$ est une valeur propre de $\mathbf{AM}_\alpha^{-1}$ de multiplicité $k-1$. Si $\alpha = h$, alors*

$$\Lambda(\mathbf{AM}_h^{-1}) = \{1\} \cup \Lambda(\mathbf{S}(h))$$

*où $\lambda = 1$ est de multiplicité $k$ et*

$$\mathbf{S}(h) = \mathbf{V}_{n-k}^T\mathbf{AV}_{n-k} - h\mathbf{e}_1\mathbf{e}_k{}^T\mathbf{H}_k^{-1}\mathbf{V}_k^T\mathbf{AV}_{n-k}.$$

*Enfin, si $\mathbf{A}$ est définie positive, alors la matrice $\mathbf{AM}_\alpha^{-1}$ est définie positive pour $h - \alpha$ suffisamment petit.*

*Démonstration.* Comme

$$\mathbf{V}^T\mathbf{AV} = \begin{pmatrix} \mathbf{H}_k & \mathbf{F} \\ h\mathbf{E} & \mathbf{G} \end{pmatrix} \quad \text{et} \quad \mathbf{V}^T\mathbf{M}_\alpha\mathbf{V} = \begin{pmatrix} \mathbf{H}_k & \mathbf{0} \\ \alpha\mathbf{E} & \mathbf{I}_{n-k} \end{pmatrix},$$

les valeurs propres de $\mathbf{AM}_\alpha^{-1}$ satisfont à :

$$\begin{aligned} \mathbf{H}_k\mathbf{x} + \mathbf{Fy} &= \lambda\mathbf{H}_k\mathbf{x}, \\ h\mathbf{Ex} + \mathbf{Gy} &= \lambda\alpha\mathbf{Ex} + \lambda\mathbf{y}. \end{aligned}$$

Ainsi, si $\alpha \neq h$, les vecteurs de la forme, $(\mathbf{e}_i^T, 0)^T$ pour $1 \leq i \leq k-1$, sont des vecteurs propres de $\mathbf{A}\mathbf{M}_\alpha^{-1}$ correspondants à la valeur propre $\lambda = 1$. Si $\alpha = h$, $(\mathbf{e}_k^T, 0)^T$ est aussi un vecteur propre associé à $\lambda = 1$, tandis que pour $\lambda \neq 1$, l'élimination de $\mathbf{x}$ dans la première équation conduit à l'équation sur les valeurs propres $\mathbf{S}(h)\mathbf{y} = (\mathbf{G} - h\mathbf{E}\mathbf{H}_k^{-1}\mathbf{F})\mathbf{y} = \lambda\mathbf{y}$.

Supposons maintenant que $\mathbf{A}$ est définie positive. Alors le complément de Schur $\mathbf{S}(h)$ est défini positif. Pour le vérifier, considérons la décomposition réelle de Schur $\mathbf{A} = \mathbf{Q}^T\mathbf{T}\mathbf{Q}$ où $\mathbf{Q}$ est une matrice unitaire ; alors pour tout $\mathbf{x} \in \mathbb{R}^n \backslash \{0\}$, $0 < \mathbf{x}^T\mathbf{Q}^T\mathbf{T}\mathbf{Q}\mathbf{x} = \mathbf{z}^T\mathbf{A}^{-1}\mathbf{z}$, avec $\mathbf{z} = \mathbf{Q}^T\mathbf{T}\mathbf{Q}\mathbf{x}$. Puisque l'inverse de $\mathbf{A}$ est de la forme

$$\mathbf{A}^{-1} = \begin{pmatrix} * & * \\ * & \mathbf{S}(h)^{-1} \end{pmatrix}$$

le choix $\mathbf{z} = (0, \mathbf{w}^T)^T$ conduit à $0 < \mathbf{z}^T\mathbf{A}^{-1}\mathbf{z} = \mathbf{w}^T\mathbf{S}(h)^{-1}\mathbf{w}$ et de là, $\mathbf{S}(h)$ est défini positive.

Maintenant, les valeurs propres non unitaires $\lambda$ de $\mathbf{A}\mathbf{M}_\alpha^{-1}$ vérifient

$$\Big(\mathbf{S}(h) + (h-\alpha)\frac{\lambda}{\lambda - 1}\mathbf{E}\mathbf{H}_k^{-1}\mathbf{F}\Big)\mathbf{y} = \lambda\mathbf{y}$$

et la conclusion vient de la défini-positivité de $\mathbf{S}(h)$ et de la théorie de la perturbation ([43]). □

*Remarque* 5.1. On peut constater que notre résultat est différent, et en un sens plus général, que celui de Baglama et al. [7] qui ont étudié le même système préconditionné sous l'hypothèse que $\mathbf{V}$ est une base du sous-espace invariant de $\mathbf{A}$. Sous cette hypothèse, le préconditionneur $\mathbf{M}$ laisse inchangé $n-k$ valeurs de propres de $\mathbf{A}$ et décale les $k$ autres vers 1. (cf. [7]).

*Remarque* 5.2. La matrice dans le dernier problème aux valeurs propres de la démonstration est une perturbation de rang-un particulière de $\mathbf{G} = \mathbf{V}_{n-k}^T\mathbf{A}\mathbf{V}_{n-k}$ de la forme $\mathbf{e}_1\mathbf{z}^T$, ne modifiant donc que la première ligne de $\mathbf{G}$. Comme corollaire immédiat, une application du théorème de Gerschgörin-Hadamard montre que l'union des disques contenant les valeurs propres reste pratiquement inchangée, au pire un des disques se retrouve avec un centre et un rayon différents.

Soit maintenant $\mathbf{A}_\varepsilon = \mathbf{A} + \mathcal{E}$ une perturbation de $\mathbf{A}$, avec $\|\mathcal{E}\| \leq \varepsilon$. Le $\varepsilon$-pseudospectre [15] de la matrice $\mathbf{A}$ est donné par

$$\Lambda_\varepsilon(\mathbf{A}) = \{\mathbf{z} \in \mathbb{C} : z \in \Lambda(\mathbf{A} + \mathcal{E}), \text{pour certains } \mathcal{E} \text{ avec } \|\mathcal{E}\| \leq \varepsilon\}, \quad (5.5)$$

dont la longueur du contour $\mathcal{L}$ "*rentre*" dans la borne standard sur la convergence de GMRES [83]

$$\|\mathbf{r}^{(k)}\| \leq (2\pi\varepsilon)^{-1}\mathcal{L}(\partial\Lambda_\varepsilon(\mathbf{A})) \min_{p_k(0)=1} \max_{\mathbf{z}\in\Lambda_\varepsilon(\mathbf{A})} |p_k(\mathbf{z})| \, \|\mathbf{r}^{(0)}\|, \quad (5.6)$$

Le résultat suivant décrit pourquoi $\mathbf{M}$ constitue un bon préconditionneur de $\mathbf{A}_\varepsilon$.

**Proposition 5.2.** *Soit $\mathbf{A}_\varepsilon$ et $\mathbf{M}$ définies comme ci-dessus. Alors, $\delta$-pseudospectre de $\mathbf{A}_\varepsilon\mathbf{M}^{-1}$ satisfait*

$$\Lambda_\delta(\mathbf{A}_\varepsilon\mathbf{M}^{-1}) \subseteq \Lambda_{\widetilde{\varepsilon}+\delta}(\mathbf{A}\mathbf{M}^{-1})$$

*pour tout $\delta \geq 0$, où $\widetilde{\varepsilon} = \varepsilon/\sigma_{\min}(\widehat{\mathbf{H}}_{k+1})$, et $\widehat{\mathbf{H}}_{k+1} \in \mathbb{R}^{k+1\times k+1}$ est la matrice de Hessenberg*

$$\widehat{\mathbf{H}}_{k+1} = \begin{pmatrix} \mathbf{H}_k & 0 \\ \alpha\mathbf{e}_k^T & 1 \end{pmatrix}.$$

*Démonstration.* Soit $\mathbf{F} \in \mathbb{R}^{n\times n}$ tel que $\|\mathbf{F}\| \leq \delta$. Puisque $\Lambda(\mathbf{A}_\varepsilon\mathbf{M}^{-1}+\mathbf{F}) = \Lambda(\mathbf{A}\mathbf{M}^{-1}+\mathcal{E}\mathbf{M}^{-1}+\mathbf{F})$ avec $\|\mathcal{E}\mathbf{M}^{-1}+\mathbf{F}\| \leq \varepsilon/\sigma_{\min}(\widehat{\mathbf{H}}_{k+1}) + \delta$, le résultat se déduit de la définition (5.5). □

On peut donc utiliser (5.6) pour lier les résidus de GMRES $\mathbf{r}_\varepsilon^{(k)}$ correspondant à la matrice $\mathbf{A}_\varepsilon\mathbf{M}^{-1}$ à la tolérance *tol* utilisée pour la résolution du problème avec la matrice $\mathbf{A}\mathbf{M}^{-1}$. En supposant

$$\|\mathbf{r}^{(k)}\|/\|\mathbf{r}^{(0)}\| \leq (2\pi(\widetilde{\varepsilon}+\delta))^{-1}\mathcal{L}(\partial\Lambda_{\widetilde{\varepsilon}+\delta}(\mathbf{A}\mathbf{M}^{-1}))\min_{p_k(0)=1}\max_{\mathbf{z}\in\Lambda_{\widetilde{\varepsilon}+\delta}(\mathbf{A}\mathbf{M}^{-1})}|p_k(\mathbf{z})| \leq tol$$

on en déduit, avec la Proposition 5.2,

$$\begin{aligned}\|\mathbf{r}_\varepsilon^{(k)}\|/\|\mathbf{r}_\varepsilon^{(0)}\| &\leq (2\pi\delta)^{-1}\mathcal{L}(\partial\Lambda_\delta(\mathbf{A}_\varepsilon\mathbf{M}^{-1}))\min_{p_k(0)=1}\max_{\mathbf{z}\in\Lambda_\delta(\mathbf{A}_\varepsilon\mathbf{M}^{-1})}|p_k(\mathbf{z})| \\ &\leq (2\pi\delta)^{-1}\mathcal{L}(\partial\Lambda_{\widetilde{\varepsilon}+\delta}(\mathbf{A}\mathbf{M}^{-1}))\min_{p_k(0)=1}\max_{\mathbf{z}\in\Lambda_{\widetilde{\varepsilon}+\delta}(\mathbf{A}\mathbf{M}^{-1})}|p_k(\mathbf{z})| \\ &\leq (1+\widetilde{\varepsilon}/\delta)tol.\end{aligned}$$

De là, soit une petite valeur de $\varepsilon$ ou une grande valeur de $\delta$ assurent que l'utilisation du préconditionneur $\mathbf{M}$ conduira à une convergence avec approximativement le nombre d'étapes, pour une tolérance donnée, et ceci pour toute perturbation $\mathcal{E}$ de norme maximale $\varepsilon$. On note que les plus grandes valeurs de $\delta$ devraient correspondre à une grande tolérance de GMRES, ce qui est souvent utilisé dans les premières étapes des algorithmes de type Newton-Krylov [19].

L'application des résultats précédents à (5.2) est évidente : on résout $\mathbf{A}_1\mathbf{x}_1 = \mathbf{b}_1$ en utilisant GMRES, ce qui conduit à un préconditionneur $\mathbf{M}_{(1)}$ de la forme (5.4). On emploie alors GMRES pour résoudre le système avec la matrice préconditionnée $(\mathbf{A}_1+\mathcal{E}_1)\mathbf{M}_{(1)}^{-1} = \mathbf{A}_2\mathbf{M}_{(1)}^{-1}$ et on construit $\mathbf{M}_{(2)}$. Le préconditionneur général pour $\mathbf{A}_\ell\mathbf{x}_\ell = \mathbf{b}_\ell$ est alors donné par

$$\begin{cases} \mathbf{M}_\ell^{-1} &= \mathbf{M}_{\ell-1}^{-1}\mathbf{M}_{(\ell-1)}^{-1}, \quad \ell > 2, \\ \mathbf{M}_2 &= \mathbf{M}_{(1)}. \end{cases} \tag{5.7}$$

La définition ci-dessus amène à s'intéresser aux coûts de stockage et de calcul. Chacun des préconditionneurs $\mathbf{M}_\ell$ requiert le stockage et la multiplication par les bases $\mathbf{V}_\ell$ de dimension $k_\ell$. Bien que l'on puisse contrôler le choix de $k_\ell$, de trop petites valeurs conduisent en général à une détérioration de la performance. Pour éviter ce problème, notre approche consiste à travailler avec des matrices qui ont, éventuellement après permutation, une structure par bloc. À partir de quoi, une approche de type complément de Schur, couplée avec à un point de départ convenable, conduit à un algorithme de GMRES essentiellement sur le problème du complément de Schur, avec l'avantage que la base de Arnoldi qui en découle est de petite taille. Un exemple caractéristique concerne la classe des méthodes de décomposition de domaine qui conduisent naturellement à ce type de stratégie. Ci-dessous, nous étudions en détail un algorithme générique par bloc qui est à la fois efficace d'un point de vue coût de calcul et de stockage, tout en préservant les avantages de l'approche de préconditionnement précédemment décrite.

### Un Algorithme par Bloc Préconditionné à Droite

Soit $\mathbf{A} \in \mathbb{R}^{n\times n}$ avec la structure par bloc suivante

$$\mathbf{A} = \begin{pmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{pmatrix} \tag{5.8}$$

où $\mathbf{A}_{ij} \in \mathbb{R}^{n_i\times n_j}$, $i,j = 1,2$, avec $n_2 \ll n_1$ et tel que $\mathbf{A}_{11}$ soit inversible. De plus, nous supposons qu'il existe un algorithme efficace permettant d'approcher l'inverse de $\mathbf{A}_{11}$. C'est le cas pour les méthodes de décomposition de domaine, où par exemple $\mathbf{A}_{11}$ est une matrice bloc-diagonale, dont les blocs sont obtenus en discrétisant des problèmes d'EDP sur les sous-domaines, tandis que $\mathbf{A}_{22}$ est la matrice correspondant aux noeuds à la frontière. On rencontre aussi une telle structure dans de nombreuses applications de type point–selle, selon que l'on est dans le domaine des EDP (discrétisations mixtes par éléments finis), ou celui de l'optimisation (minimisation avec contraintes) ; en outre, dans ces applications en particulier, l'inverse de $\mathbf{A}_{11}$ peut souvent être calculé efficacement.

Considérons maintenant le système triangulaire par bloc

$$\mathbf{P}_{\mathsf{R}} = \begin{pmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ 0 & \widehat{\mathbf{S}} \end{pmatrix} \tag{5.9}$$

où $\widehat{\mathbf{S}}$ est une approximation du complément de Schur $\mathbf{S} = \mathbf{A}_{22} - \mathbf{A}_{12}\mathbf{A}_{11}^{-1}\mathbf{A}_{21}$. Puisque

$$\mathbf{A}\mathbf{P}_{\mathsf{R}}^{-1} = \begin{pmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{A}_{12}\mathbf{A}_{11}^{-1} & \mathbf{S}\widehat{\mathbf{S}}^{-1} \end{pmatrix},$$

un point initial de la forme

$$\mathbf{x}^{(0)} = \mathbf{P}_{\mathsf{R}}^{-1}\mathbf{b}$$

conduit à un résidu initial de la forme $\mathbf{r}^{(0)} = (\mathbf{0}, \widetilde{\mathbf{r}}^{(0)})^T$. Si on cherche $\mathbf{r}^{(k)}$ dans l'espace de Krylov $\mathcal{K}(\mathbf{A}\mathbf{P}_{\mathsf{R}}^{-1}, \mathbf{r}^{(0)}, k)$, alors tous les résidus auront la même forme : $\mathbf{r}^{(k)} = (\mathbf{0}, \widetilde{\mathbf{r}}^{(k)})^T$. De plus, comme l'espace de Krylov $\mathcal{K}(\mathbf{A}\mathbf{P}_{\mathsf{R}}^{-1}, \mathbf{r}^{(0)}, k) \equiv [\mathbf{r}^{(j)}],\ 0 \leq j \leq k-1$, la base de Arnoldi associée est de la forme

$$\mathbf{V}_k = \begin{pmatrix} \mathbf{0} \\ \widetilde{\mathbf{V}}_k \end{pmatrix}.$$

Ainsi, on a juste besoin de stocker et de travailler avec une base $\widetilde{\mathbf{V}}_k$ de longueur $n_2$. En particulier, pour des méthodes de décomposition de domaine $n_2 \sim O(n_1^{1/2})$ ; de là, le coût de GMRES préconditionné par le préconditionneur adaptatif $\mathbf{M}$ donné en (5.7) est $O(n)$ si

$$k\ell^2 \sim O(n^{1/2}),$$

où $k = \max_{1 \leq i \leq \ell} k_i$ est la taille maximale des bases de Arnoldi utilisées pour la construction de $\mathbf{M}$, et $\ell$ est le nombre d'étapes non-linéaires. En pratique, pour beaucoup d'applications non-linéaires, la valeur de $\ell$ est petite ($\ell \ll n^{1/2}$) tandis que le préconditionneur combiné $\mathbf{M}\mathbf{P}_{\mathsf{R}}$ garantira une petite valeur de $k$.

### 5.2.3 Essais Numériques

Pour illustrer l'efficacité de l'approche décrite au § 5.2.2, nous considérons deux applications issues de la discrétisation des problèmes d'Équations aux Dérivées Partielles (EDP) : méthode de décomposition de domaines pour la résolution du problème non-linéaire de réaction–diffusion avec convection, et un exemple issu des équations de Navier–Stokes. Ces deux problèmes sont discrétisés comme suit :

$$\mathbf{A}(\mathbf{x})\mathbf{x} = \Big(\nu\mathbf{L} + \mathbf{N}(\mathbf{x})\Big)\mathbf{x} = \mathbf{b}$$

où $\nu$ est un petit paramètre, $\mathbf{L}$ et $\mathbf{N}$ dénotent respectivement la partie linéaire et non-linéaire de la matrice $\mathbf{A}$. Dans les deux cas, nous employons deux méthodes de linéarisation : la méthode de Picard (point fixe) et la méthode de Newton. Ensuite, nous résolvons cette séquence de systèmes linéaires

$$\mathbf{A}_\ell \mathbf{x}_{\ell+1} = \Big(\mathbf{A}(\mathbf{x}_\ell) - \rho\mathbf{T}(\mathbf{x}_\ell)\Big)\mathbf{x}_{\ell+1} = \mathbf{b} - \rho\mathbf{T}(\mathbf{x}_\ell)\mathbf{x}_\ell = \mathbf{b}_\ell$$

où $\mathbf{T}(\mathbf{x}_\ell) = \mathbf{N}(\mathbf{x}_\ell) - \mathbf{N}'(\mathbf{x}_\ell)$, et $\mathbf{N}'$ dénote la matrice Jacobienne de $\mathbf{N}$ en $\mathbf{x}$. Les itérations de Picard et de Newton correspondent respectivement au choix de $\rho = 0$ et $\rho = 1$. Nous utilisons le critère d'arrêt suggéré par Dembo et al. dans [19] pour l'itération de Newton : à chaque étape non-linéaire $\ell$, les $k$ itérations de GMRES sont stoppées quand le vecteur résidu $\mathbf{r}^{(k)}$ satisfait

$$\|\mathbf{r}^{(k)}\| / \|\mathbf{b}_\ell - \mathbf{A}_\ell \mathbf{x}_\ell\| \leq c \, \|\mathbf{b}_\ell - \mathbf{A}_\ell \mathbf{x}_\ell\|^q, \tag{5.10}$$

où le choix de $c$ et de $q$ varie d'un problème à un autre.

**Problème Non-Linéaire de Réaction-Diffusion avec Convection**

Considérons la discrétisation par éléments finis de

$$\left\{ \begin{array}{rcll} -\nu\Delta\mathbf{u} + (\vec{\mathbf{b}}\cdot\nabla)\mathbf{u} + \mathbf{u}^2 & = & f & \text{dans } \Omega, \\ \mathbf{u} & = & 0 & \text{sur } \Gamma. \end{array} \right. \tag{5.11}$$

où $f = 1$, $\Omega = [-1,1]^2$ de frontière $\Gamma$ et $\vec{\mathbf{b}} = (2y(1-x^2), -2x(1-y^2))$. Ce problème sera résolu pour $\nu = 1,\ 1/10$ par la méthode de décomposition de domaines utilisant 4, 16 et 64 sous-domaines. L'itération non-linéaire sera stoppée quand le critère $\|\mathbf{b}_\ell - \mathbf{A}_\ell\mathbf{x}_\ell\|/\|\mathbf{b}_0 - \mathbf{A}_0\mathbf{x}_0\| \leq 10^{-6}$ est satisfait. Nous utilisons la relation (5.10) avec $c = 10^{-2}$ et $q = 4/5$, comme critère d'arrêt pour l'algorithme de GMRES. Pour l'itération non-linéaire, l'itéré initial est $\mathbf{x}_0 = 0$. La structure des matrices linéarisées $\mathbf{A}_\ell$ est donnée dans (5.8), où $\mathbf{A}_{11}$ est la matrice bloc-diagonale dont les blocs correspondent aux noeuds intérieurs pour chaque sous-domaine – ceci correspond à ce qu'on appelle "*non-overlapping Dirichlet-Dirichlet*". Comme le problème est non symétrique, nous employons un premier niveau de préconditionnement $\mathbf{P}_\ell$, la matrice triangulaire par blocs donnée dans (5.9), où nous choisissons l'approximation

$$\widehat{\mathbf{S}} = \mathbf{A}_{22} - \mathbf{A}_{21}\mathsf{diag}(\mathbf{A}_{11})^{-1}\mathbf{A}_{12},$$

afin de tester notre approche de préconditionnement. Dans ce cas, les valeurs propres du système préconditionné sont égales soit à 1, soit aux valeurs propres du complément de Schur préconditionné $\mathbf{S}\widehat{\mathbf{S}}^{-1}$. Pour des problèmes elliptiques, le conditionnement de $\mathbf{S}$ est de l'ordre de $h^{-1}H^{-1}$ où $h \sim n^{1/2}$ est la taille des mailles et $H$ est la taille moyenne des sous-domaines (cf. [67]). Ceci mène typiquement à une convergence qui dépend des mailles et domaines. Comme nous le verrons ci-dessous, le choix $\widehat{\mathbf{S}}$ semble enlever seulement la deuxième dépendance étant donnée notre stratégie itérative (5.10) pour le processus non-linéaire.

Tandis que divers algorithmes de décomposition de domaine enlèvent ce problème de dépendance pour certaines applications, leur conception est toujours particulière et demande plusieurs niveaux d'information pour atteindre une certaine optimalité. À cet égard, notre technique adaptative conduit à un préconditionneur "*black-box*" qui peut soit remplacer soit augmenter un préconditionneur de complément de Schur. En outre, comme démontré ci-dessous, notre préconditionneur adaptatif enlève le problème de dépendance des mailles et de domaines. Les résultats pour l'itération de Picard sont représentés dans les tables 5.1 et 5.3. Nous comparons les résultats obtenus avec un premier niveau de préconditionnement $\mathbf{A}_\ell\mathbf{P}_\ell^{-1}$ et avec les deux niveaux de préconditionnement $\mathbf{A}_\ell\mathbf{P}_\ell^{-1}\mathbf{M}_\ell^{-1}$ pour lesquels, on prendra $\alpha = 0$ et $\alpha = h$ (cf. (5.4)). La moyenne du nombre d'itérations par étape non-linéaire est également présentée, exclu la première étape de Picard. Ceci permettra

une comparaison plus claire et plus juste entre $\mathbf{P}_\ell$ et $\mathbf{M}_\ell$, puisque $\mathbf{M}_\ell = \mathbf{I}$ pour $\ell = 1$.

| $\nu = 1$ | | $\mathbf{A}_\ell\mathbf{P}_\ell^{-1}$ | $\mathbf{A}_\ell\mathbf{P}_\ell^{-1}\mathbf{M}_\ell^{-1}$ | |
|---|---|---|---|---|
| *#dom* | $n$ | # its | # its ($\alpha = 0$) | # its ($\alpha = h$) |
| 4 | 11,425 | 131/**24.2** | 41/**6.2** | 38/**5.6** |
| | 45,377 | 181/**33.4** | 54/**8.0** | 50/**7.2** |
| | 180,865 | 310/**48.5** | 80/**10.2** | 73/**9.0** |
| 16 | 11,585 | 191/**35.6** | 57/**8.8** | 52/**7.8** |
| | 45,953 | 272/**50.6** | 76/**11.4** | 72/**10.6** |
| | 183,041 | 467/**73.5** | 113/**14.5** | 107/**13.5** |
| 64 | 16,641 | 195/**35.4** | 53/**7.0** | 48/**6.0** |
| | 66,049 | 283/**51.4** | 72/**9.2** | 68/**8.4** |
| | 263,169 | 392/**71.0** | 97/**12.0** | 92/**11.0** |

TAB. 5.1 – *Nombre total/moyenne d'itérations de GMRES avec et sans préconditionnement adaptatif — La technique de linéarisation considérée est celle de Picard.*

| $\nu = 1$ | | $\alpha = 0$ | | | $\alpha = h$ | | |
|---|---|---|---|---|---|---|---|
| *#dom* | $n$ | $\mathbf{P}_\ell^{-1}\mathbf{v}$ | $\mathbf{A}_\ell\mathbf{v}$ | $\times nz(\mathbf{A}_\ell)$ | $\mathbf{P}_\ell^{-1}\mathbf{v}$ | $\mathbf{A}_\ell\mathbf{v}$ | $\times nz(\mathbf{A}_\ell)$ |
| 4 | 11,425 | 69% | 65% | 0.10 | 70% | 68% | 0.11 |
| | 45,377 | 70% | 68% | 0.06 | 72% | 71% | 0.07 |
| | 180,865 | 74% | 73% | 0.05 | 76% | 75% | 0.05 |
| 16 | 11,585 | 70% | 60% | 0.35 | 72% | 63% | 0.36 |
| | 45,953 | 72% | 66% | 0.24 | 74% | 67% | 0.24 |
| | 183,041 | 76% | 71% | 0.18 | 77% | 73% | 0.18 |
| 64 | 16,641 | 73% | 55% | 0.73 | 75% | 60% | 0.75 |
| | 66,049 | 75% | 63% | 0.50 | 76% | 65% | 0.51 |
| | 263,169 | 75% | 68% | 0.34 | 76% | 70% | 0.34 |

TAB. 5.2 – *Gains en (precvecs)* ($\mathbf{P}_\ell^{-1}\mathbf{v}$) *et (matvecs)* ($\mathbf{A}_\ell\mathbf{v}$) *et stockage additionnel* ($\times nz(\mathbf{A}_\ell)$) *— La technique de linéarisation considérée est celle de Picard.*

Dans l'intérêt de cette comparaison, nous montrons également les gains correspondants et le stockage additionnel dans les tables 5.2 et 5.4. En particulier, nous avons choisi de calculer le coût du préconditionneur adaptatif

seulement en nombre de produits matrice-vecteur (*matvecs*) et de l'ajouter au nombre total de produits matrice-vecteurs effectués. Les gains obtenus se résument dans la réduction relative des produits matrice-vecteur (*matvecs*) et des produits matrice-vecteur impliquant $\mathbf{P}_\ell^{-1}$ (*precvecs*). Le calcul du stockage additionnel se base aussi sur le stockage requis pour la matrice du système. Nous notons également que le coût ou le stockage du premier préconditionneur $\mathbf{P}_\ell$ n'a pas d'intérêt ici et n'a pas été évalué.

La performance de la technique adaptative peut être récapitulée comme suit :
(i) le problème de dépendance de mailles et des domaines est pratiquement enlevé ;
(ii) les gains augmentent avec la taille du problème ;
(iii) le stockage additionnel relatif décroît avec la taille du problème.
En effet, pour cette application l'amélioration en convergence est très remarquable, avec des gains entre 69–93% pour *precvecs* ($\mathbf{P}_\ell^{-1}\mathbf{v}$) et 55–90% pour *matvecs* ($\mathbf{A}_\ell\mathbf{v}$), tandis que le stockage additionnel est minimal (une fraction de stockage pour la matrice du système entre 0.1–0.3 pour la maille la plus fine).

| $\nu = 1/10$ | | $\mathbf{A}_\ell\mathbf{P}_\ell^{-1}$ | $\mathbf{A}_\ell\mathbf{P}_\ell^{-1}\mathbf{M}_\ell^{-1}$ | |
|---|---|---|---|---|
| $\#dom$ | $n$ | # its | # its ($\alpha = 0$) | # its ($\alpha = h$) |
| 4 | 11,425 | 543/**16.7** | 89/**2.5** | 79/**2.1** |
| | 45,377 | 775/**23.8** | 111/**3.03** | 89/**2.3** |
| | 180,865 | 1,099/**33.8** | 121/**3.2** | 102/**2.6** |
| 16 | 11,585 | 971/**29.7** | 118/**3.1** | 99/**2.4** |
| | 45,953 | 1,375/**42.1** | 144/**3.6** | 117/**2.8** |
| | 183,041 | 1,934/**59.3** | 173/**4.2** | 148/**3.5** |
| 64 | 16,641 | 1,104/**33.7** | 114/**2.8** | 101/**2.3** |
| | 66,049 | 1,589/**48.5** | 130/**2.9** | 119/**2.6** |
| | 263,169 | 2,266/**69.2** | 173/**3.8** | 145/**2.9** |

TAB. 5.3 – *Nombre Total/Moyenne d'itérations de GMRES avec et sans préconditionnement adaptatif — La méthode de linéarisation considérée est celle de Picard.*

Nous notons également que la différence de performance entre les cas $\nu = 1$ et $\nu = 1/10$ s'explique essentiellement par la convergence de l'itération de Picard. En effet, quand $\nu = 1$, le nombre d'itérations de Picard varie entre 6–7, tandis que pour $\nu = 1/10$ la convergence est plus lente et nécessite 33 itérations. Dans les deux cas, le nombre d'itérations de GMRES est similaire dans les premières étapes de Picard, avec une décroissance très rapide de ce

nombre d'itérations de GMRES pour atteindre une situation relativement stable, où seulement 2 à 4 itérations de GMRES sont requises par étape non-linéaire. Cependant, vu que le cas de $\nu = 1/10$ exige bien plus d'étapes non-linéaires, la moyenne du nombre d'itérations de GMRES par étape non-linéaire se retrouve donc être plus petite que celle du cas $\nu = 1$ (cf. table 5.1).

Pour le cas de l'itération de Newton, nous avons noté un comportement de performance très semblable, c'est pour cela nous l'avons pas détaillé. Cependant, nous avons utilisé cette technique de linéarisation (Newton) dans le cas d'un problème plus difficile, issue de l'équation de Navier–Stokes, dans le paragraphe suivant.

| $\nu = 1/10$ | | $\alpha = 0$ | | | $\alpha = h$ | | |
|---|---|---|---|---|---|---|---|
| $\#dom$ | $n$ | $\mathbf{P}_\ell^{-1}\mathbf{v}$ | $\mathbf{A}_\ell\mathbf{v}$ | $\times nz(\mathbf{A}_\ell)$ | $\mathbf{P}_\ell^{-1}\mathbf{v}$ | $\mathbf{A}_\ell\mathbf{v}$ | $\times nz(\mathbf{A}_\ell)$ |
| 4 | 11,425 | 84% | 80% | 0.25 | 85% | 81% | 0.31 |
| | 45,377 | 86% | 84% | 0.15 | 89% | 87% | 0.17 |
| | 180,865 | 89% | 88% | 0.08 | 91% | 90% | 0.09 |
| 16 | 11,585 | 88% | 78% | 0.84 | 90% | 81% | 0.94 |
| | 45,953 | 90% | 84% | 0.51 | 91% | 87% | 0.53 |
| | 183,041 | 91% | 88% | 0.31 | 92% | 90% | 0.32 |
| 64 | 16,641 | 90% | 73% | 1.73 | 91% | 74% | 2.05 |
| | 66,049 | 92% | 84% | 0.99 | 93% | 85% | 1.16 |
| | 263,169 | 92% | 87% | 0.66 | 93% | 90% | 0.68 |

TAB. 5.4 – *Gains en precvecs* ($\mathbf{P}_\ell^{-1}\mathbf{v}$) *et en* matvecs ($\mathbf{A}_\ell\mathbf{v}$) *et stockage additionnel* ($\times nz(\mathbf{A}_\ell)$) — *La méthode de linéarisation considérée est celle de Picard.*

**Problème de Navier–Stokes**

Comme deuxième application, nous considérons le cas des équations de Navier–Stokes incompressibles en régime stationnaire sur le domaine $\Omega$ fermé borné de $\mathbb{R}^d$, ($d = 2$ ou $d = 3$) de frontière $\Gamma$.

$$-\nu\Delta\mathbf{u} + (\mathbf{u}\cdot\nabla)\mathbf{u} + \nabla\mathbf{p} = f \quad \text{dans } \Omega, \tag{5.12a}$$

$$\nabla\cdot\mathbf{u} = 0 \quad \text{dans } \Omega, \tag{5.12b}$$

$$\mathbf{u} = 0 \quad \text{sur } \Gamma, \tag{5.12c}$$

où $\mathbf{u}$ désigne le vecteur vitesse, $\mathbf{p}$ la pression et $\nu$ le paramètre de viscosité. Le système non-linéaire résultant possède la structure point–selle suivante

(cf., e.g.,[66])

$$0 = F(\mathbf{x}) = F(\mathbf{u},\mathbf{p}) = \begin{pmatrix}\mathbf{f}\\ \mathbf{0}\end{pmatrix} - \begin{pmatrix}\mathbf{K}(\mathbf{u}) & \mathbf{B}^T\\ \mathbf{B} & \mathbf{0}\end{pmatrix}\begin{pmatrix}\mathbf{u}\\ \mathbf{p}\end{pmatrix} \tag{5.13}$$

où $\mathbf{K}(\mathbf{u}) = \nu\mathbf{L} + \mathbf{N}(\mathbf{u}) \in \mathbb{R}^{n_1\times n_1}$ est une matrice non symétrique et qui peut être indéfinie, et $\mathbf{B}^T \in \mathbb{R}^{n_1\times n_2}$ possède un noyau engendré par le vecteur constant. Nos deux linéarisations de (5.13) mènent à une séquence de systèmes de forme

$$\mathbf{A}_\ell\mathbf{x}_{\ell+1} = \begin{pmatrix}\mathbf{K}_\ell & \mathbf{B}^T\\ \mathbf{B} & \mathbf{0}\end{pmatrix}\begin{pmatrix}\mathbf{u}_{\ell+1}\\ \mathbf{p}_{\ell+1}\end{pmatrix} = \begin{pmatrix}\mathbf{f}\\ \mathbf{0}\end{pmatrix} - \begin{pmatrix}\rho\mathbf{T}(\mathbf{u}_\ell) & \mathbf{0}\\ \mathbf{0} & \mathbf{0}\end{pmatrix}\begin{pmatrix}\mathbf{u}_\ell\\ \mathbf{p}_\ell\end{pmatrix} = \mathbf{b}_\ell$$

où $\mathbf{K}_\ell = \mathbf{K}(\mathbf{u}_\ell) - \rho\mathbf{T}(\mathbf{u}_\ell)$ et $\mathbf{T}(\mathbf{u}_\ell) = \mathbf{N}(\mathbf{u}_\ell) - \mathbf{N}'(\mathbf{u}_\ell)$, avec $\mathbf{N}$ désignant la partie non-linéaire de $\mathbf{K}$. Comme premier niveau de préconditionnement, nous employons le préconditionneur par blocs donné par :

$$\mathbf{P}_\ell = \begin{pmatrix}\widehat{\mathbf{K}}_\ell & \mathbf{B}^T\\ \mathbf{0} & -\widehat{\mathbf{S}}\end{pmatrix} \tag{5.14}$$

où $\widehat{\mathbf{K}}_\ell$ approche $\mathbf{K}_\ell$ et $\widehat{\mathbf{S}}$ est l'opposé du complément de Schur $\mathbf{S} = \mathbf{B}\mathbf{K}_\ell^{-1}\mathbf{B}^T$. Nous utilisons, comme solveur interne, l'algorithme GMRES avec un préconditionnement Gauss-Seidel par blocs, pour résoudre les systèmes avec $\widehat{\mathbf{K}}_\ell$, et pour $\widehat{\mathbf{S}}$, nous employons le même choix que celui proposé dans [23], et donné par

$$\widehat{\mathbf{S}}^{-1} = \mathbf{M}_p^{-1}\mathbf{F}_p\mathbf{A}_p^{-1},$$

où $\mathbf{M}_p$, $\mathbf{F}_p$, $\mathbf{A}_p$ sont respectivement les matrices de masse, de convection-diffusion et du Laplacien, assemblées dans l'espace de pression (cf. [23] pour plus de détails). En particulier, il est montré dans [23] que pour un nombre de Reynolds petit, le système préconditionné a des valeurs propres majorées indépendamment de la taille du problème. Cependant, leur regroupement peut ne pas être optimal. Comme nous le montrons ci-dessous, notre deuxième niveau du préconditionnement améliorera fortement la convergence.

Nous présentons les résultats obtenus pour un problème test standard d'écoulement en cavité, pour trois valeurs du nombre de Reynolds : 10, 100, 1000. Le choix $c = 10^{-2}$ dans notre critère d'arrêt (5.10) n'affecte pas le nombre d'itérations non-linéaires si on prend $q = 3/4$, excepté pour un nombre de Reynolds $Re = 1000$ où nous prendrons alors $q = 1$. Nous stoppons les itérations de Newton quand $\|F(\mathbf{x}_\ell)\|/\|F(\mathbf{x}_0)\| \leq 10^{-10}$. L'itéré initial pour l'itération de Newton est égal à 0, excepté pour un écoulement à $Re = 1000$ où nous réutilisons la solution obtenue après trois itérations de Picard comme vecteur de départ. Comme précédemment décrit, au § 5.2.2, afin de réduire le stockage à chaque itération non-linéaire, l'itéré initial pour GMRES est $\mathbf{x} = \mathbf{P}_\ell^{-1}\mathbf{b}_\ell$, ce qui assure un vecteur résidu de type $\mathbf{r}^T =$

$(\mathbf{0}, \mathbf{r}_p^T)$. Les vecteurs de la base résultante de Arnoldi ont également la même forme, ce qui va nous permettre de stocker une base $\mathbf{V}$ de longueur $n_2$. Considérons la discrétisation P2–P1 standard, la relation entre $n_1$ et $n_2$ est que $n_1 = 8n_2$ en 2D, et $n_1 = 24n_2$ en 3D, auquel cas la réduction du stockage est très significative. D'ailleurs, le coût induit par l'application du préconditionneur adaptatif devient presque négligeable, comme le montre la table 5.5 ci-dessous. Nous notons que, en relâchant l'action de $\mathbf{P}^{-1}$ (par exemple, en ne réalisant qu'une inversion approximative de $\mathbf{K}_\ell$), la base exacte de Arnoldi ne sera pas creuse dans sa structure. Cependant, nous avons constaté que le fait de stocker uniquement la partie de longueur $n_2$ (le reste étant forcé à "$\mathbf{0}$") ne détériore pas la performance si le critère d'arrêt utilisé pour la résolution de systèmes linéaires avec $\mathbf{P}$ est du même ordre que la tolérance de résolution dans GMRES.

| *Re* | $n$ | $\mathbf{A}_\ell\mathbf{P}_\ell^{-1}$ | $\mathbf{A}_\ell\mathbf{P}_\ell^{-1}\mathbf{M}_\ell^{-1}$ |
|---|---|---|---|
| 10 | 4,771 | 113/**21.2** | 51/**8.8** |
| | 18,755 | 91/**21.0** | 49/**10.5** |
| | 74,371 | 98/**22.7** | 56/**12.2** |
| 100 | 4,771 | 227/**30.3** | 102/**12.4** |
| | 18,755 | 208/**32.5** | 103/**15.0** |
| | 74,371 | 174/**32.6** | 97/**17.2** |
| 1000 | 4,771 | 1,064/**105.0** | 408/**23.0** |
| | 18,755 | 952/**118.0** | 411/**27.8** |
| | 74,371 | 718/**118.2** | 375/**32.5** |

TAB. 5.5 – *Écoulement en cavité : Nombre Total/Moyenne d'itérations de GMRES avec et sans préconditionnement adaptatif.*

La table 5.5 montre le nombre d'itérations de GMRES quand seulement $\mathbf{P}_\ell$ est employé et quand le préconditionneur adaptatif (5.7) est additionnellement utilisé avec $\alpha = h$. Comme avant, nous évaluons le surcoût en *matvecs* impliquant la matrice du système $\mathbf{A}_\ell$, et les gains en *precvecs* et *matvecs*.

Les gains pour ce problème sont considérables. En effet, l'amélioration de la performance est due à la réduction significative des *precvecs* (entre 42–48% pour le plus grand problème), puisque dans ce cas-ci, les *precvecs* sont plus chers que les *matvecs*. C'est généralement le cas dans la pratique, à savoir qu'un préconditionnement efficace nécessite un certain coût, et donc la réduction substantielle du nombre de *precvecs* aura un impact d'autant plus important sur les coûts. Nous notons également que le préconditionneur additionnel préserve l'indépendance de la taille du problème réalisé par la première technique de préconditionnement. En conclusion, nous notons

| $Re$ | $n$ | $\mathbf{P}_\ell^{-1}\mathbf{v}$ | $\mathbf{A}_\ell\mathbf{v}$ | $\times nz(\mathbf{A}_\ell)$ |
|---|---|---|---|---|
| 10 | 4,771 | 55% | 45% | 0.21 |
| | 18,755 | 46% | 37% | 0.19 |
| | 74,371 | 42% | 31% | 0.20 |
| 100 | 4,771 | 55% | 37% | 0.44 |
| | 18,755 | 50% | 32% | 0.41 |
| | 74,371 | 44% | 25% | 0.38 |
| 1000 | 4,771 | 62% | 36% | 1.12 |
| | 18,755 | 57% | 32% | 1.02 |
| | 74,371 | 48% | 25% | 0.87 |

TAB. 5.6 – *Écoulement en cavité : Gains en precvecs* ($\mathbf{P}_\ell^{-1}\mathbf{v}$) *et en* matvecs ($\mathbf{A}_\ell\mathbf{v}$) *et stockage additionnel* ($\times nz(\mathbf{A}_\ell)$).

une augmentation en gains de *precvecs* avec l'augmentation du nombre de Reynolds ($Re$). Le stockage augmente également, bien que pour tout $Re$, le stockage supplémentaire exigé n'excède pas la quantité demandée pour la matrice elle-même. On pourrait aussi considérer une stratégie "*stockage-limite*" (si la mémoire est un aspect critique), bien que nous ayons constaté à cet égard que cela conduit en général à une détérioration substantielle de la performance.

## 5.3 Conclusions et Perspectives

Très brièvement, pour résumer l'ensemble de ce qui est abordé dans ce chapitre d'ouverture, l'objectif de ces compléments est de montrer toute la panoplie des approches possibles combinant filtrage spectral et techniques de Krylov, ainsi que d'élargir le champ des applications possibles en allant vers le non-linéaire en particulier.

Le point commun à toutes ces études réside dans la capacité à générer des espaces de Krylov de dimension réduite, contenant donc une information spectrale très concentrée et exploitable directement, sans post-traitement particulier, pour accélérer efficacement la convergence d'une suite de systèmes linéaires avec la même matrice mais à second membres multiples.

Nous avons vu notamment comment, dans le cas d'un complément de Schur issu d'un premier niveau de prétraitement/préconditionnement très efficace, au sein d'une itération non-linéaire, il est possible de "*profiter*" simplement de la faible dimension intrinsèque des espaces de Krylov, générés

dans GMRES, pour construire des préconditionneurs adaptatifs qui intègrent l'évolution de l'information spectrale au fur et à mesure des itérations non-linéaires, et permettent de conserver de bout en bout cette propriété essentielle, à savoir des espaces de Krylov de dimension très réduite.

L'intérêt de ces approches a été validé, dans les divers chapitres, sur certains problèmes test provenant de la collection de matrices creuses Harwell-Boeing, ainsi que d'autres exemples issus de la discrétisation par éléments finis de problèmes d'Équations aux Dérivées Partielles (EDP) du type équation de diffusion. Au vu des gains substantiels obtenus, il devient clair que ces techniques apparaissent assez prometteuses lors d'une multirésolution.

Nous avons vu également que la technique Tchebycheff-PSF démarre d'une génération aléatoire de vecteurs, et n'a pas pour vocation de résoudre un système linéaire comme dans l'algorithme ChebFilterCG qui a un avantage double : la résolution de systèmes linéaires et la génération d'une base de Krylov très riche et de dimension petite. Cependant, l'algorithme Tchebycheff-PSF n'est pas influencé de la même manière par les problèmes pathologiques. De manière plus générale, l'algorithme Tchebycheff-PSF est plus "*orienté*" vecteurs propres et valeurs propres, avec la capacité de produire en particulier des espaces de Krylov filtrés possédant une structure numérique très spécifique dans les composantes propres de ces vecteurs de Krylov. En particulier, cette technique peut être utilisée pour calculer un noyau, ou de faire du "*rank-revealing*". La contrepartie à cela est que Tchebycheff-PSF, pour converger, doit "*scanner*" l'ensemble du spectre compris entre zéro et la fréquence de coupure utilisée dans les filtres de Tchebycheff.

Par opposition, l'algorithme ChebFilterCG, qui n'exploite ces filtres que comme des préconditionneurs, a la capacité de générer des séquences de Krylov bien moins longues (dans les cas non pathologiques), relativement au nombre total de valeurs propres inférieures à la fréquence de coupure choisie. Bien évidemment, la base de Krylov résultante ne possède pas de structure numérique particulière, si ce n'est qu'elle reste très riche du point de vue de l'information spectrale relative aux plus petites valeurs propres (hors valeurs propres nulles). Enfin, et cela n'a pas été étudié ici car ne présentant pas d'intérêt majeur pour ce qui est de la simple résolution de systèmes linéaires, il est aussi possible, après obtention de la base de Krylov finale dans ChebFilterCG, de post-filtrer cette base pour lui donner une structure numérique particulière, comme dans Tchebycheff-PSF.

Un autre aspect non étudié dans ce travail, concerne l'intégration du noyau "*CG/Lanczos préconditionné par des filtres de Tchebycheff*", comme exploité dans ChebFilterCG, dans des algorithmes de calcul de valeurs propres, intégrant donc des techniques de redémarrage, de déflation, ... Les avantages que l'on pourrait pressentir résident là encore dans dans la capacité à générer des espaces de Krylov de dimension réduite, très riches du point de vue de l'information spectrale relative aux plus petites valeurs propres, et

ceci sans avoir à réaliser des "*inversions*" ou résolutions. Par contre, dans ce contexte, il faudra trouver des extensions satisfaisantes à notre travail pour le cas indéfini, notamment pour pouvoir prendre en considération la question du "*shift*", ou translation du spectre.

Il est clair, en effet, que les polynômes de Tchebycheff sont particulièrement appropriés au cas symétrique défini positif. Pour les cas symétriques indéfinis, et non symétriques, il faudrait en effet étudier les généralisations possibles de l'approche développée dans ce travail, en regardant en détail les propriétés intrinsèques des méthodes de Krylov habituellement exploitées dans ces cas là, et en intégrant des techniques polynomiales qui permettent de "*cibler*" efficacement les informations particulières assurant une accélération "*intelligente*" des ces techniques dans le cadre d'une multirésolution.

Enfin, dans le cas de la résolution de systèmes linéaires non symétriques, il est toujours possible d'utiliser des techniques dites de "*symétrisation*", telles que la méthode de Cimmino par blocs par exemple, qui transforme la matrice originale en une matrice symétrique définie positive dont les valeurs propres sont en général fortement regroupées dans un nombre relativement faible d'amas. Cette technique de transformation de la matrice d'itération n'améliore pas cependant de façon particulière le conditionnement, mais la réponse à cela est en partie donnée par la technique ChebFilterCG qui s'accomode plus "*naturellement*" de ces problèmes de conditionnement, et qui est d'autant plus intéressante dans le cadre d'une multirésolution. Ce genre de combinaison pourrait constituer une alternative intéressante à l'utilisation de méthodes directes, pour de très gros problèmes, issus par exemple de la discrétisation 3D de problèmes d'EDP.

Les autres perspectives qui se dégagent de ce travail sont la construction de préconditionneurs multiniveaux, combinant les filtres de Tchebycheff et la base de Krylov produite lors de la résolution du premier système avec ChebFilterCG. L'utilisation des polynômes de Tchebycheff offre en effet la possibilité de réduire fortement le nombre total de synchronisations "*fortes*", du type *produits scalaires*, et devrait permettre une implantation efficace dans un cadre de calcul parallèle, et en particulier dans des environnements à mémoire distribuée.

# Bibliographie


[1] M. Arioli, I.S. Duff, J. Noailles, and D. Ruiz. A block projection method for sparse matrices. *SIAM Journal on Scientific and Statistical Computing*, 13 :47–70, 1992.

[2] M. Arioli, I.S. Duff, D. Ruiz, and M. Sadkane. Block Lanczos techniques for accelerating the Block Cimmino method. *SIAM Journal on Scientific and Statistical Computing*, 16 :1478–1511, 1995.

[3] M. Arioli and D. Ruiz. Block conjugate gradient with subspace iteration for solving linear systems. In Svetozar D. Margenov and Panayot S. Vassilevski, editors, *Iterative Methods in Linear Algebra, II. Volume 3 in the IMACS Series in Computational and Applied Mathematics. Proceedings of The Second IMACS International Symposium on Iterative Methods in Linear Algebra.*, 1995.

[4] M. Arioli and D. Ruiz. A Chebyshev-based two-stage iterative method as an alternative to the direct solution of linear systems. Technical Report RAL-TR-2002-021, Rutherford Appleton Laboratory, Atlas Center, Didcot, Oxfordshire, OX11 0QX, England, 2002.

[5] W.E. Arnoldi. The principle of minimized iteration in the solution of the matrix eigenvalue problem. *Quarterly of Applied Mathematics*, 9 :17–29, 1951.

[6] S.F. Ashby, T.A. Manteuffel, and J.S. Otto. A comparison of adaptive Chebyshev and least squares polynomial preconditioning for Hermitian positive definite linear systems. *SIAM Journal on Scientific and Statistical Computing*, 13 :1–29, 1992.

[7] J. Baglama, D. Calvetti, G.H. Golub, and L. Reichel. Adaptively preconditioned GMRES algorithms. *SIAM Journal on Scientific Computing*, 20(1) :243–269, 1998.

[8] K. Burrage and J. Erhel. On the performance of various adaptive preconditioned GMRES strategies. *Numerical Linear Algebra with Applications*, 5 :101–121, 1998.

[9] K. Burrage, J. Erhel, B. Pohl, and A. Williams. A deflation technique for linear systems of equations. *SIAM Journal Scientific Computing*, 19(4) :1245–1260, 1998.

[10] B. Carpentieri, I.S. Duff, and L. Giraud. A class of spectral two-level preconditioners. *SIAM Journal on Scientific Computing*, 25 :749–765, 2003.

[11] B. Carpentieri, L. Giraud, and S. Gratton. Additive and multiplicative spectral two-level preconditioners for general linear systems. Technical Report TR/PA/04/38, CERFACS, Toulouse, France, 2004.

[12] T.F. Chan and M.K. Ng. Galerkin projection methods for solving multiple linear systems. *SIAM Journal on Scientific Computing*, 21(3) :836–850 (electronic), 1999.

[13] T.F Chan and W.L. Wan. Analysis of projection methods for solving linear systems with multiple right-hand sides. *SIAM Journal on Scientific Computing*, 18(6) :1698–1721, 1997.

[14] A. Chapman and Y. Saad. Deflated and augmented Krylov subspace techniques. *Numerical Linear Algebra with Applications*, 4(1) :43–66, 1997.

[15] F. Chatelin. *Valeurs propres de matrices*. Collection Mathématiques Appliquées pour la Maîtrise. [Collection of Applied Mathematics for the Master's Degree]. Masson, Paris, 1988.

[16] P. Concus, G.H. Golub, and D.P. O'Leary. A generalized conjugate gradient method for the numerical solution of elliptic partial differential equations. In *Sparse matrix computations (Proc. Sympos., Argonne Nat. Lab., Lemont, Ill., 1975)*, pages 309–332. Academic Press, New York, 1976.

[17] H. de Gersem and K. Hameyer. A deflated iterative solver for magnetostatic finite element models with large differences in permeability. *Eur. Phys. J. Appl. Phys.*, 13 :45–49, 2000.

[18] E. de Sturler. Truncation strategies for optimal Krylov subspace methods. *SIAM Journal on Numerical Analysis*, 36(3) :864–889, 1999.

[19] R.S. Dembo, S.C. Eisenstat, and T. Steihaug. Inexact Newton methods. *SIAM Journal on Numerical Analysis*, 19(2) :400–408, 1982.

[20] Z. Dostal. Conjugate gradient method with preconditioning by projector. *Intern. J. Computer Math.*, 23 :315–323, 1988.

[21] I.S. Duff, L. Giraud, J. Langou, and E. Martin. Using spectral low rank preconditioners for large electromagnetic calculations. *International Journal for Numerical Methods in Engineering*, 62(3) :416–434, 2005.

[22] I.S. Duff, R.G. Grimes, and J.G. Lewis. Sparse matrix test problems. *ACM Transactions on Mathematical Software*, 15 :1–14, 1989.

[23] H.C. Elman, D. Loghin, and A.J. Wathen. Preconditioning techniques for Newton's method for the incompressible Navier–Stokes equations. *BIT*, 43 :961–974, 2003.

[24] J. Erhel, K. Burrage, and B. Pohl. Restarted GMRES preconditioned by deflation. *Journal of Computational and Applied Mathematics*, 69 :303–318, 1996.

[25] J. Erhel and F. Guyomarc'h. An augmented conjugate gradient method for solving consecutive symmetric positive linear systems. *SIAM Journal on Matrix Analysis and Applications*, 21(4) :1279–1299, 2000.

[26] J. Erhel, N. Nassif, and B. Philippe. Calcul matriciel et systèmes linéaires. Cours de DEA Informatique et Modélisation, Beyrouth, Liban, 2004. Available from http ://www.irisa.fr/sage/jocelyne/pdf/beyrouth-dea-0204.pdf.

[27] P.F. Fischer. Projection techniques for iterative solution of $A\underline{x} = \underline{b}$ with successive right-hand sides. *Computer Methods in Applied Mechanics and Engineering*, 163(1-4) :193–204, 1998.

[28] L. Giraud, D. Ruiz, and A. Touhami. A comparative study of iterative solvers exploiting spectral information for spd systems. SIAM Journal on Scientific Computing, 27(5) :1760–1786, 2006.

[29] L. Giraud, D. Ruiz, and A. Touhami. Krylov based and polynomial iterative solvers combined with partial spectral factorization for SPD linear systems. In Michel Daydé, José M.L.M. Palma, Jack Dongarra, and Vicente Hernández, editors, *Lecture Notes in Computer Science, Vector and Parallel Processing, LNCS 3402*, pages 635–655. Springer–Verlag Berlin Heidelberg, 2005.

[30] G.H. Golub and M.D. Kent. Estimates of eigenvalues for iterative methods. *Mathematics of Computation*, 53 :249–263, 1989.

[31] G.H. Golub and G. Meurant. *Résolution numérique des grands systèmes linéaires*, volume 49 of *Collection de la Direction des Études et Recherches d'Électricité de France [Collection of the Department of Studies and Research of Électricité de France]*. Éditions Eyrolles, Paris, 1983.

[32] G.H. Golub, D. Ruiz, and A. Touhami. A hybrid approach combining Chebyshev filter and conjugate gradient for solving linear systems with multiple right-hand sides. Technical Report TR/TLSE/05/10, ENSEEIHT-IRIT, Toulouse, France, 2005. Also Technical Report SCCM-05-08 Stanford University. Submitted to SIAM Journal on Matrix Analysis and Applications.

[33] G.H. Golub and R. Underwood. The Block Lanczos method for computing eigenvalues. In *Mathematical Software Symposium, University of Wisconsin, Madison, 1977. Mathematical Software III : Proceedings of a Symposium Conducted by the Mathematics Research Center, the University of Wisconsin, Madison, March 28–30, 1977, J.R. Rice, ed., no.39 in Publication of the Mathematics Research Center, the Univer-*

*sity of Wisconsin, Madison, Academic Press, New York, pp. 361-377*, 1977.

[34] G.H. Golub and C.F. Van Loan. *Matrix Computations*. Johns Hopkins Studies in the Mathematical Sciences. The Johns Hopkins University Press, Baltimore, MD, USA, third edition, 1996.

[35] W. Hackbusch. *Multigrid methods and applications.* springer, 1985.

[36] L.A. Hageman and D.M. Young. *Applied Iterative Methods*. Academic Press, New York and London, 1981.

[37] M.R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *J. Res. Nat. Bur. Standards*, 49 :409–435, 1952.

[38] W. Jalby and B. Philippe. Stability analysis and improvement of the block Gram-Schmidt algorithm. *SIAM Journal on Scientific and Statistical Computing*, 12(5) :1058–1073, 1991.

[39] O.G. Johnson, C.A. Micchelli, and G. Paul. Polynomial preconditioners for conjugate gradient calculations. *SIAM Journal on Numerical Analysis*, 20 :362–376, 1983.

[40] E.F. Kaasschieter. Preconditioned conjugate gradient for solving singular systems. *Journal of Computational and Applied Mathematics*, 24 :265–275, 1988.

[41] W. Karush. An iterative method for finding characteristic vectors of a symmetric matrix. *Pacific Journal of Mathematics*, 1 :233–248, 1951.

[42] L.Yu. Kolotilina. Preconditioning of systems of linear algebraic equations by means of twofold deflation. I. Theory. (Russian) Zap. Nauchn. Sem. S.-Peterburg. Otdel. Mat. Inst. Steklov. (POMI) 229 (1995), Chisl. Metody i Voprosy Organ. Vychisl. 11, 95–152, 323 ; translation in J. Math. Sci. (New York) 89 (1998), no. 6, 1652–1689.

[43] P. Lancaster and M. Tismenetsky. *The Theory of Matrices*. Academic Press, 1985.

[44] R.B. Lehoucq, D.C. Sorensen, and C. Yang. *ARPACK User's guide : Solution of large-scale problem with implicitly restarted Arnoldi methods*. SIAM, Philadelphia, 1998.

[45] D. Loghin, D. Ruiz, and A. Touhami. Adaptive preconditioners for nonlinear systems of equations. *Journal of Computational and Applied Mathematics*, 189 :362–374, 2006.

[46] G. Meurant. *Computer solution of large linear systems*, volume 28 of *Studies in Mathematics and its Applications*. North-Holland Publishing Co., Amsterdam, 1999.

[47] R.B. Morgan. A restarted GMRES method augmented with eigenvectors. *SIAM Journal on Matrix Analysis and Applications*, 16 :1154–1171, 1995.

[48] R.B. Morgan. On restarting the Arnoldi method for large nonsymmetric eigenvalue problems. *Mathematics of Computation*, 65(215) :1213–1230, 1996.

[49] R.B. Morgan. Implicitly restarted GMRES and Arnoldi methods for nonsymmetric systems of equations. *SIAM Journal on Matrix Analysis and Applications*, 21 :1112–1135, 2000.

[50] R.B. Morgan. GMRES with deflated restarting. *SIAM Journal on Scientific Computing*, 24 :20–37, 2002.

[51] R. Nabben and C. Vuik. A comparison of Deflation and Coarse Grid Correction applied to porous media flow. *SIAM Journal on Numerical Analysis*, 42 :1631–1647, 2004.

[52] R. Nicolaides. Deflation of conjugate gradients with applications to boundary value problems. *SIAM Journal on Numerical Analysis*, 24 :355–365, 1987.

[53] D.P. O'Leary. The block conjugate gradient algorithm and related methods. *Linear Algebra and its Applications*, 29 :293–322, 1980.

[54] D.P. O'Leary. A generalized conjugate gradient algorithm for solving class of quadratic programming problems. *Linear Algebra and its Applications*, 34 :371–399, 1980.

[55] D.P. O'Leary. Yet another polynomial preconditioner for the conjugate gradient algorithm. *Linear Algebra and its Applications*, 154 :388–377, 1991.

[56] C.C. Paige. Practical use of the symmetric Lanczos process with re-orthogonalization. *Nordisk Tidskr. Informationsbehandling (BIT)*, 10 :183–195, 1970.

[57] C.C. Paige. *The computation of eigenvalues and eigenvectors of very large sparse matrices*. PhD thesis, University of London, 1971.

[58] C.C. Paige. Error analysis of the Lanczos algorithm for tridiagonalizing a symmetric matrix. *Journal of the Institute of Mathematics and its Applications*, 18(3) :341–349, 1976.

[59] C.C. Paige. Accuracy and effectiveness of the Lanczos algorithm for the symmetric eigenproblem. *Linear Algebra and its Applications*, 34 :235–258, 1980.

[60] M. Parks, E. de Sturler, G. Mackey, D. Johnson, and S. Mait. Recycling Krylov subspaces for sequences of linear systems. Technical Report UIUCDCS-R-2004-2421, University of Illinois, March 2004.

[61] B.N. Parlett. Canonical decomposition of Hessenberg matrices. *Mathematics of Computation*, 21 :223–277, 1967.

[62] B.N. Parlett. A new look at the Lanczos algorithm for solving symmetric systems of linear equations. *Linear Algebra and its Applications*, 29 :323–346, 1980.

[63] B.N. Parlett. *The Symmetric Eigenvalue Problems*. Prentice-Hall, Englewood Cliffs, NJ, 1980.

[64] B.N. Parlett. Misconvergence in the Lanczos algorithm. In *Reliable numerical computation*, Oxford Sci. Publ., pages 7–24. Oxford Univ. Press, New York, 1990.

[65] B.N. Parlett and D.S. Scott. The Lanczos algorithm with selective orthogonalization. *Mathematics of Computation*, 33(145) :217–238, 1979.

[66] A. Quarteroni and A. Valli. *Numerical Approximation of Partial Differential Equations*. Springer-Verlag, 1994.

[67] A. Quarteroni and A. Valli. *Domain Decomposition Methods for Partial Differential Equations*. Numerical Mathematics and Scientific Computation, Oxford University Press, 1999.

[68] D. Ruiz. *Solution of large sparse unsymmetric linear systems with a block iterative method in a multiprocessor environment*. PhD thesis, INPT, CERFACS, Toulouse, France, 1992.

[69] Y. Saad. On the rates of convergence of the Lanczos and the block Lanczos methods. *SIAM Journal on Numerical Analysis*, 17 :687–706, 1980.

[70] Y. Saad. Variations on Arnoldi's method for computing eigenelements of large unsymmetric matrices. *Linear Algebra and its Applications*, 34 :269–295, 1980.

[71] Y. Saad. Chebyshev acceleration techniques for solving nonsymmetric eigenvalue problems. *Mathematics of Computation*, 42(166) :567–588, 1984.

[72] Y. Saad. Practical use of polynomial preconditionings for the conjugate gradient method. *SIAM Journal on Scientific and Statistical Computing*, 6 :865–881, 1985.

[73] Y. Saad. On the Lanczos method for solving symmetric linear systems with several right-hand sides. *Mathematics of computations*, 178 :651–662, 1987.

[74] Y. Saad. *Numerical Methods for Large Eigenvalue Problems*. Manchester University Press, Manchester, UK, 1992.

[75] Y. Saad. *Iterative Methods for Sparse Linear Systems*. PWS Publishing Company, Boston, 1996.

[76] Y. Saad and M.H. Schultz. GMRES : a generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 7(3) :856–869, 1986.

[77] Y. Saad, M. Yeung, J. Erhel, and F. Guyomarc'h. A deflated version of the conjugate gradient algorithm. *SIAM Journal on Scientific Computing*, 21 :1909–1926, 2000.

[78] M. Sadkane. A block Arnoldi-Chebyshev method for computing the leading eigenpairs of large sparse unsymmetric matrices. *Numerische Mathematik*, 64(2) :181–193, 1993.

[79] H.D. Simon. Analysis of the symmetric Lanczos algorithm with reorthogonalization methods. *Linear Algebra and its Applications*, 61 :101–131, 1984.

[80] V. Simoncini and E. Gallopoulos. An iterative method for nonsymmetric systems with multiple right-hand sides. *SIAM Journal on Scientific Computing*, 16(4) :917–933, 1995.

[81] D.C. Sorensen. Implicit application of polynomial filters in a $k$-step Arnoldi method. *SIAM Journal on Matrix Analysis and Applications*, 13(1) :357–385, 1992.

[82] G. Szegö. *Orthogonal polynomials*. American Mathematical Society, Providence, R.I., 1975.

[83] L.N. Trefethen. *Approximation theory and numerical linear algebra*. In J. C. Mason and M.G. Cox, editors, Algorithms for approximation II, Chapman and Hall, London, 1990.

[84] R. Underwood. An iterative block Lanczos method for the solution of large sparse symmetric eigenproblems. PhD Thesis STAN-CS-75-496, Computer Science Department, Stanford, California, 1975.

[85] A. van der Sluis and H.A. van der Vorst. The rate of convergence of conjugate gradients. *Numerische Mathematik*, 48 :543–560, 1986.

[86] A. van der Sluis and H.A. van der Vorst. The convergence behavior of Ritz values in the presence of close eigenvalues. *Linear Algebra and its Applications*, 88/89 :651–694, 1987.

[87] H. van der Vorst. An iterative method for solving $f(A)x = b$ using Krylov subspace information obtained for the symmetric positive definite. *Journal of Computational and Applied Mathematics*, 18 :249–263, 1987.

[88] F. Vermolen and C. Vuik. The influence of deflation vectors at interfaces on the deflated conjugate gradient method. Report 01-13, Delft University of Technology, Department of Applied Mathematical Analysis, Delft, 2001.

[89] C. Vuik. Fast iterative solvers for the discretized incompressible Navier–Stokes equations. Technical Report 93–98, Defit University of Technology, 1993.

[90] H.F. Walker. Implementation of the GMRES method using Householder transformations. *SIAM Journal on Scientific and Statistical Computing*, 9 :152–163, 1988.

[91] I. Zavorin, D.P. O'Leary, and H.C. Elman. Complete stagnation of GMRES. *Linear Algebra and its Applications*, 367 :165–183, 2003.

# Liste des Publications

## Articles dans des Revues Internationales

[1] Daniel Loghin, Daniel Ruiz, and Ahmed Touhami. Adaptive Preconditioners for Nonlinear Systems of Equations. **J. Comp. Appl. Math. 189, pp. 362–374, 2006**.

[2] Frédéric Messine and Ahmed Touhami. A General Reliable Quadratic Form : an Extension of Affine Arithmetic. **Reliab. Comput. 12(3), pp. 171–192, 2006**.

[3] Luc Giraud, Daniel Ruiz, and Ahmed Touhami. A Comparative Study of Iterative Solvers Exploiting Spectral Information for SPD Systems. **SIAM J. Sci. Comput. 27(5), pp. 1760–1786, 2006**.

## Articles Soumis dans des Revues Internationales

[4] Gene H. Golub, Daniel Ruiz, and Ahmed Touhami. A Hybrid Approach Combining Chebyshev Filter and Conjugate Gradient for Solving Linear Systems with Multiple Right-Hand Sides. **Article soumis à to SIAM J. Matrix Anal. Appl.**, 2005.

## Conférences Internationales avec Actes Édités et Comité de Lecture

[5] Luc Giraud, Daniel Ruiz, and Ahmed Touhami. Combining Various Iterative Solvers with Partial Spectral Factorization for the Solution of SPD Linear Systems. In Radim Blaheta and Jirí Starý, editors, **IMET 2004, Iterative Methods, Preconditioning & Numerical PDEs**, pages 55–59. Institute of Geonics AS CR Ostrava, Czech Republic, 2004.

[6] Luc Giraud, Daniel Ruiz, and Ahmed Touhami. Krylov Based and Polynomial Iterative Solvers Combined with Partial Spectral Factorization for SPD Linear Systems. In Michel Daydé, José M.L.M. Palma, Jack

Dongarra, and Vicente Hernández, editors, **Lecture Notes in Computer Science, Vector and Parallel Processing, LNCS 3402**, pages 635–655. Springer–Verlag Berlin Heidelberg, 2005.

[7] Daniel Loghin, Daniel Ruiz, and Ahmed Touhami. Adaptive Preconditioners for Newton-Krylov Methods. In Radim Blaheta and Jirí Starý, editors, **IMET 2004, Iterative Methods, Preconditioning & Numerical PDEs**, pages 117–120. Institute of Geonics AS CR Ostrava, Czech Republic, 2004.

[8] Frédéric Messine and Ahmed Touhami. An Interval Branch-and-Bound Algorithm Based on the General Quadratic Form. In I. García, L.G. Casado, E.M.T. Hendrix, and B. Tóth, editors, **Proceedings of the International Workshop on Global Optimization (GO'05) Almería (Spain)**, pages 171–176, 2005.

## Conférences Internationales avec Comité de Sélection

[9] Luc Giraud, Daniel Ruiz, and Ahmed Touhami. Two-Phase Solver versus Two-Level Preconditioner for Solving Large Sparse and S.P.D Linear System. In **Sparse Days and Grid Computing. St. Girons, France. June 10–13**, 2003. Poster Session.

[10] Luc Giraud, Daniel Ruiz, and Ahmed Touhami. A Comparative Study of Iterative Solvers Exploiting Spectral Information for SPD Systems. In **Eight Copper Mountain Conference on Iterative Methods. Copper Mountain Resort, Colorado, USA. March 28 – April 2**, 2004.

[11] Gene H. Golub, Daniel Ruiz, and Ahmed Touhami. Chebyshev Polynomial Preconditioners as a Spectral Filtering Tool for the Conjugate Gradient Method. In **21st Biennial Conference on Numerical Analysis. Dundee, Scotland, U.K. 28 June – 1 July**, 2005.

[12] Daniel Loghin, Daniel Ruiz, and Ahmed Touhami. Adaptive Preconditioners for Nonlinear Systems of Equations. In **ICCAM 2004, Eleventh International Congress on Computational and Applied Mathematics. Leuven, Belgium. July 26–30**, 2004.

## Conférences Nationales avec Comité de Sélection

[13] Frédéric Messine and Ahmed Touhami. Méthodes Déterministes d'Optimisation Globale Basées sur l'Arithmétique Affine et Quadratique. In **11$^{ème}$ journées du groupe MODE de la SMAI. Pau, France. Mars 27–29**, 2003. Poster Session. Ce poster a reçu le Prix d'Éditeur.

## Rapports de Recherche

[14] Frédéric Messine and Ahmed Touhami. Exact and Rigorous Global Optimization Algorithm Based on Affine Arithmetic and its Extensions to Quadratic Forms. Rapport de Recherche RT/TLSE/03/01, ENSEEIHT-IRIT, Toulouse, France, 2003. Aussi Rapport de Recherche 0321, Laboratoire de Mathématiques Appliquées de Pau, CNRS-FRE 2570, France, 2003.

[15] Ahmed Touhami. Arithmétique Affine Robuste et ses Extensions aux Formes Quadratiques. Rapport de DEA, ENSEEIHT-IRIT, Toulouse, France, 2002.

**Résumé**. Le Contexte de ce travail est l'algèbre linéaire numérique. Plus précisément, on s'est intéressé à des préconditionnements pour les méthodes de Krylov, basés sur une connaissance de certains espaces propres. Ces techniques sont en particulier très utiles lorsque l'on résout une séquence de systèmes linéaires avec la même matrice mais différents second membres. L'information sur les espaces propres est extraite dans une phase d'initialisation, ou au cours de la résolution du premier système, et utilisée dans la résolution des systèmes suivants. L'approche développée dans cette thèse se base sur l'utilisation des filtres polynomiaux de Tchebycheff et sur la construction de préconditionneurs spectraux pour l'accélération des méthodes de Krylov.

**Mots-clés**. Polynômes de Tchebycheff, méthodes de Lanczos, méthode du gradient conjugué, filtrage, déflation, cycle à deux grilles, préconditionnement spectral, préconditionnement adaptatif, systèmes non linéaires, systèmes augmentés, cas pathologiques, méthodes de Krylov.

**Classification AMS**. 65F10, 65F15, 65F50, 65H10, 65N22

---

**Abstract**. The context of this work is numerical linear algebra. More precisely, we are interested in preconditioning Krylov techniques, with the knowledge of some eigenspaces. In particular, this can be very useful when solving a sequence of linear systems with the same matrix but different right-hand sides. The information about eigenspaces is extracted from an initialization phase, or during the solution of the first system, and is used in the solution of the following systems. The approach developed in this thesis is based on the use of the Chebyshev filtering polynomials and on the construction of spectral preconditioners to accelerate the convergence of Krylov methods.

**Key words**. Chebyshev polynnomials, Lanczos method, conjugate gradient method, filtering, deflation, two-grid schemes, spectral preconditioners, adaptive preconditioners, nonlinear systems, augmented systems, Krylov methods.

**AMS Classification**. 65F10, 65F15, 65F50, 65H10, 65N22

---

ENSEEIHT–IRIT, UMR CNRS 5505, 2 rue Camichel, F-31071 Toulouse