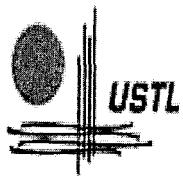


Numéro d'ordre : 3812

THÈSE

présentée à

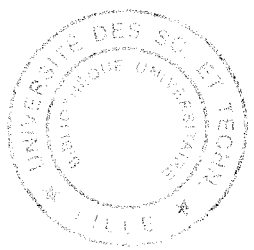
L'UNIVERSITÉ DES SCIENCES ET TECHNOLOGIES DE LILLE

pour obtenir le titre de

DOCTEUR EN MATHÉMATIQUES

par

Abdellatif TINZEFTE



Étude algorithmique et théorique de quelques méthodes de type Lanczos

soutenue le 21 juin 2006, devant la commission d'examen

Membres du Jury

Présidente	:	M. Redivo Zaglia	Professeur, Université de Padoue, Italie.
Rapporteurs	:	G. Meurant	Directeur de recherches, CEA, Bruyères-Le-Chatel.
		M. Redivo Zaglia	Professeur, Université de Padoue, Italie.
Examinateur	:	L. Reichel	Professeur, Kent State University, USA.
Directeurs de Thèse	:	C. Brezinski	Professeur, UFR maths, USTL, Lille.
		H. Sadok	Professeur, Université du Littoral, Calais.

Laboratoire Paul Painlevé - U.M.R.-C.N.R.S 8524, Batiment M3, USTL, 59655
Villeneuve d'Ascq Cedex, France.

A mes parents, mon frère, mes sœurs et à tous ceux qui ont contribué à la réalisation de ce travail

Cette thèse a été effectuée au sein du Laboratoire Paul Painlevé de Lille sous la direction du Professeur Claude Brezinski de l'Université des Sciences et Technologies de Lille et du Professeur Hassane Sadok de l'Université du Littoral. Qu'ils trouvent ici l'expression de ma reconnaissance, pour avoir accepté de diriger ma recherche, pour leurs conseils, pour le temps qu'ils m'ont consacré et pour avoir accepté d'examiner soigneusement ce travail.

Je tiens à exprimer ma reconnaissance à Khalid Jbilou Maître de Conférence à l'Université du Littoral, pour le temps qu'il m'a consacré, pour les discussions scientifiques que nous avons eues. Je le remercie de m'avoir fait bénéficier de son expérience dans la partie de la résolution de systèmes linéaires multiples.

Je suis très honoré de la présence dans le jury du professeur Michela Redivo Zaglia de l'Université de Padoue (Italie), ainsi que du professeur Gérard Meurant Directeur des recherches au CEA. Je les remercie d'avoir accepté d'être rapporteurs de cette thèse.

Mes remerciements vont également au professeur Lothar Reichel, Professeur de l'Université Kent State (USA), qui m'a fait l'honneur d'accepter d'être membre du jury.

Je remercie tous les membres de l'équipe AN-EDP du laboratoire Paul Painlevé de l'Université de Lille. Je tiens à porter une pensée particulière pour les thésards et ex-thésards du laboratoire, en particulier Grégory Boutry, Melissa Inglart, Anas Elfarouk, Delphine Jennequin, Youcef Mammeri et Christophe Roland.

Je voudrais remercier Caterina Calgaro et Jean-Paul Chehab pour leurs soutiens et les conseils qu'ils m'ont apportés.

Mon ami Wyssem a bien voulu lire soigneusement ce manuscrit. Qu'il trouve ici mes sincères remerciements.

Je n'oublie pas de saluer les personnes que j'ai rencontrées durant ces années. Je les remercie pour tous ces moments que j'ai appréciés.

Que mes amis Hicham Hansali, Denis Tekaya, Hicham Karimi, Youssef Kadiri, Nourdine Jaoiri... ne soient pas oubliés.

Finalement, je remercie tous les membres de ma famille, surtout mes parents, pour m'avoir gentiment accordé quelques années de travail pour ma thèse aux dépens des moments partagés.

Résumé

La méthode de Lanczos est l'une des méthodes itératives les plus utilisées pour la résolution des systèmes linéaires. Les polynômes orthogonaux formels permettent la mise en œuvre des différentes méthodes de type Lanczos. Cependant, des divisions par zéro ("*breakdown*") peuvent être rencontrées dans le calcul de ces polynômes par des relations de récurrence. Dans la première partie de cette thèse, nous avons effectué une étude détaillée de ce phénomène, le rapport entre les différentes situations de *breakdown* qui affectent les relations de récurrence a été établi. Cela nous a permis, d'une part, de donner une nouvelle implantation du processus de Lanczos par des récurrences à deux termes, d'autre part, d'appliquer le *look-ahead* à la méthode du Gradient Biconjugué. Les polynômes orthogonaux formels sont ensuite utilisés pour introduire le préconditionneur dans quelques méthodes de type Lanczos qui utilisent le *look-ahead* (MRZ-stab, HMRZ-stab, BIORES non générique, QMR et CSBCG).
Ensuite, nous nous sommes particulièrement intéressés à l'algorithme MRZ-stab qui est une variante de l'algorithme Lanczos/Orthodir dans laquelle nous appliquons le *look-ahead* pour éviter le *breakdown* et qui souffre de problème d'instabilité numérique. Une normalisation de ces vecteurs de direction par des produits scalaires nous a permis la mise en œuvre d'un algorithme qui évite les situations de dépassement de capacité qui sont très fréquentes dans l'algorithme MRZ-stab. Une adaptation de l'algorithme obtenu au cas symétrique non-défini positif est proposée.

A la fin de cette thèse, pour la résolution des systèmes linéaires multiples $AX = B$ avec $A \in \mathbb{R}^{N,N}$, $X, B \in \mathbb{R}^{N,s}$ et $s \ll N$, nous avons proposé une nouvelle approche basée sur une projection oblique par rapport aux sous-espaces de Krylov matriciels, nous avons défini la procédure du Lanczos global qui nous a permis de développer le processus de Lanczos global, ainsi que les méthodes de type Lanczos globales telles que le Gradient Biconjugué global (GL-BICG) et le BiCGSTAB global. Des situations de *breakdown* peuvent survenir dans ces algorithmes, pour résoudre ce problème, nous avons proposé des versions avec *look-ahead* de certaines méthodes de type Lanczos globales.

Mots clefs

Méthodes de Lanczos, polynômes orthogonaux formels, *breakdown*, *look-ahead*, schéma de Horner, préconditionneur, *overflow*, *underflow*, normalisation, sous-espace de Krylov matriciel, projection globale, méthodes de type Lanczos globales.

Abstract

The Lanczos method is one of the most famous iterative methods for solving linear systems. Formal orthogonal polynomials allows to implement several Lanczos-type methods. However, breakdowns can occur in the computation using recurrence relations. In the first part of this thesis, we study more precisely this phenomenon and we establish a generalization for all situations which lead to a breakdown. At first, this generalization gives a new implementation of the Lanczos process based on two-term recurrences. At last, we apply look-ahead to the Biconjugate Gradient method. Formal orthogonal polynomials are then used to introduce preconditioning in some Lanczos-type methods using look-ahead (MRZ-stab, HMRZ-stab, non-generic BIORES, QMR and CSBCG).
After that, we focus on the MRZ-stab algorithm, which is a variant of the Lanczos/Orthodir algorithm in which we apply look-ahead to avoid breakdowns. It is known that this variant has some numerical instabilities and we propose a normalisation of directional vectors by some scalar products which avoids the overflows generally present in the MRZ-stab method. An adaptation of this algorithm to the non-definite symmetric case is proposed.

The second part of this work deals with the resolution of multiple linear systems $AX = B$ with $A \in \mathbb{R}^{N,N}$, $X, B \in \mathbb{R}^{N,s}$ and $s \ll N$. We introduce a new approach based on an oblique projection onto matrix Krylov subspaces. We define a global Lanczos procedure, deduce the global Lanczos process and global Lanczos-type methods (global Biconjugate Gradient and global BiCGSTAB). Some breakdowns can occur in these algorithms and we propose versions with look-ahead of these global algorithms to avoid breakdowns.

key words

Lanczos methods, orthogonal polynomials, breakdown, look-ahead, Horner's rule, preconditioning, overflow, underflow, normalization, matrix Krylov subspace, global projection, global Lanczos-type methods.

Table des matières

Introduction générale

La résolution des systèmes linéaires est un sujet classique de l'algèbre linéaire qui intervient dans de nombreux domaines et qui reste néanmoins toujours d'actualité. La modélisation des problèmes que nous rencontrons en physique, en mécanique, en électrotechnique et d'une façon générale dans l'ingénierie, conduit, éventuellement après une discrétisation, à la résolution de systèmes d'équations linéaires de dimension finie. Citons par exemple la discrétisation par la méthode des différences finies ou par la méthode des éléments finis de certains problèmes d'équations aux dérivées partielles.

Soit à résoudre le système linéaire à N équations et N inconnues

$$Ax = b, \tag{1}$$

où $A \in \mathbb{R}^{N\times N}$, $b \in \mathbb{R}^N$ et $x \in \mathbb{R}^N$.
Pour une matrice A carrée régulière de dimension N, le calcul numérique d'une valeur approchée de la solution exacte $x = A^{-1}b$ se fait de deux manières différentes, l'une directe et l'autre itérative.

Les méthodes directes consistent à factoriser la matrice A en un produit de matrices faciles à inverser. En pratique, la stabilité numérique d'une telle factorisation a une importance capitale. Les deux factorisations LU avec pivotage et QR de la matrice A sont les plus utilisées en raison de leur stabilité numérique. Cependant, dans le cas d'une matrice A de grande dimension N, ces deux factorisations nécessitent un stockage de l'ordre de $\mathcal{O}(n^2)$ et un coût algorithmique élevé de l'ordre de $\mathcal{O}(n^3)$.

Les méthodes itératives consistent à générer des suites de vecteurs convergeant vers la solution du système (1). Contrairement aux méthodes directes, ces méthodes sont le plus souvent utilisées lorsque la matrice A est d'une grande dimension et surtout si A est creuse. La programmation de ces méthodes ne modifie pas la structure de A et nécessite, en général, un stockage et un coût algorithmique convenables. De plus, ces méthodes itératives sont les mieux adaptées au calcul parallèle. Une classe de ces méthodes itératives est basée sur la minimisation d'une certaine norme de l'erreur sur un sous-espace de Krylov. La

convergence des méthodes de cette classe vers une valeur approchée de la solution exacte est obtenue, en général, en un petit nombre d'itérations par rapport, bien sûr, à la taille de la matrice A. Parmi les éléments de cette classe, citons la méthode du gradient conjugué (CG : Conjugate Gradient method) due à Hestenes et Stiefel [44] qui est utilisée lorsque la matrice A est symétrique définie positive ainsi que les méthodes de type CG suivantes :

- MINRES (Minimum Residual method) et SYMMLQ (Symmetric LQ) dues à Paige et Saunders [56] qui supposent seulement que la matrice A est symétrique. Notons que la méthode MINRES est mathématiquement équivalente à la variante CR (Conjugate Residual) de la méthode CG sauf qu'elle résout un problème aux moindres carrés obtenu grâce au processus hermitien de Lanczos [50] pour minimiser le résidu.

- La méthode CG appliquée respectivement aux équations normales $AA^*y = b$ avec $x = A^*y$ et $A^*Ax = A^*b$ nous donne respectivement les méthodes CGNE due à Craig [24] et CGNR due à Hestenes et Stiefel [44]. Ces deux méthodes ne supposent aucune condition sur la matrice A et minimisent respectivement la norme euclidienne de l'erreur et du résidu. Cependant, il est connu que la matrice AA^* (ou A^*A) peu être mal conditionnée, ce qui entraîne une convergence lente de ces méthodes.

- LSQR (Least Squares QR), due à Paige et Saunders [57], est mathématiquement équivalente à CGNR. Mais LSQR est plus stable numériquement que CGNR.

- USYMLQ (Unsymmetric LQ) et USYMQR (Unsymmetric QR) sont dues à Saunders, Simon et Yip [68]. Ces deux méthodes ont une caractéristique particulière qui consiste à projeter la solution du système sur une somme de deux sous-espaces de Krylov. Notons cependant que ces deux méthodes présentent un problème de division par zéro auquel nous ne pouvons remédier que par un changement des vecteurs qui sont choisis initialement.

Il est important de noter que CGNE, CGNR, LSQR, USYMLQ, USYMQR sont utilisées pour une matrice A quelconque. En général, sans préconditionnement, ces méthodes ont une convergence lente, ce qui a poussé les chercheurs à développer d'autres méthodes plus compétitives. Nous en citons ci-dessous quelques unes.

- ORTHOMIN est due à Vinsome [78]. Les résultats théoriques sur la convergence de cette méthode ont été analysés par Axelsson [2] et les auteurs de [26, 27]. Cette méthode est aussi connue sous le nom de GCR (Generalized Conjugate Residual method) dans [27].

- Orthores et Orthodir sont dues à Young et Jea [81]. Orthomin, Orthores et Orthodir

différent seulement dans la façon de générer le résidu. Mais il s'avère que Orthomin est la plus stable numériquement.

- FOM (Full Orthogonalization Method), due à Saad [61], utilise le processus d'Arnoldi et la condition d'orthogonalité de Petrov-Galerkin pour se ramener, à chaque itération, à un système linéaire dont la matrice est de Hessenberg. Cette dernière peut être singulière, ce qui est considéré comme un handicap. Nous disons aussi que FOM souffre d'une division par zéro causée par la condition de Petrov-Galerkin.

- GMRES due à Saad et Shultz[63] minimise le résidu et utilise le processus d'Arnoldi, ce qui lui permet d'éviter les problèmes de divisions par zéro. GMRES consiste à se ramener, grâce au processus d'Arnoldi, à un problème de minimisation au sens des moindres carrés, qui se résout à l'aide d'une factorisation QR obtenue par rotations de Givens. GMRES utilise une orthogonalisation complète du processus d'Arnoldi. Il est bien connu qu'une telle orthogonalisation nécessite un coût algorithmique élevé. C'est pourquoi Wu et Saad ont utilisé la technique d'orthogonalisation incomplète [59], pour définir la méthode DQGmres [60]. Une autre technique souvent utilisée est celle du redémarrage.

- BICG (Biconjugate Gradient method), due à Lanczos [50] et rendue populaire par Fletcher [29], utilise le processus non hermitien de Lanczos et la condition de biorthogonalité de Petrov-Galerkin. BICG est une extension de la méthode du gradient conjugué au cas non- symétrique. Malheureusement, sa convergence peut être irrégulière en plus cette méthode souffre d'une division par zéro qui correspond à la singularité de la matrice de Lanczos. La méthode BICG souffre éventuellement d'une deuxième division par zéro qui est celle du processus de Lanczos. Cette dernière peut être évitée grâce aux stratégies de *look-ahead* [31, 58].

- QMR (Quasi Minimal Residual method) est due à Freund et Nachtigal [34]. QMR utilise le processus de Lanczos pour se ramener à un problème de "quasi-minimisation" qui évite une éventuelle division par zéro due à la singularité de la matrice de Lanczos. La division par zéro due au processus de Lanczos peut être évitée de la même façon que dans le BICG par les stratégies de $look - ahead$.

- CMRH (Changing Minimal Residual method based on the Hessenberg process) due à Sadok [67], utilise la technique de "quasi minimisation" du résidu comme pour la méthode QMR et le processus de Hessenberg

- CGM (Conjugate Gradient Multiple) est une classe de méthodes introduite simultanément par Brezinski [6] et Gutknecht [43]. Ce type de méthodes est aussi connu sous le nom de "Lanczos-type product methods" (LTPM). Les deux méthodes CGS due à Sonneveld [73] et BICGSTAB due à Van Der Vorst [77] sont de type CGM. Cette classe CGM est basée sur la méthode de Lanczos et donc ce type de méthode subit aussi des divisions par zéro.

Ici nous nous intéressons plus particulièrement à la méthode de Lanczos [50, 51], le résidu de cette méthode s'écrit sous la forme $r_k = b - Ax_k = P_k(A)r_0$, où P_k est un polynôme orthogonal formel par rapport à une fonctionnelle c bien déterminée. De ce fait, nous pouvons mettre en œuvre cette méthode en utilisant les différentes relations de récurrence satisfaites par la famille des polynômes orthogonaux formels et/ou celle de la famille adjacente [19]. L'existence de ces familles n'est pas toujours assurée, plus précisément, elle dépend de certains déterminants de Hankel ; lorsque ces derniers sont nuls, une division par zéro survient dans les algorithmes de type Lanczos. Cette situation est appelée "*true-breakdown*" ou *pivot-breakdown*. Un autre type de *breakdown* peut également survenir dans la méthode de Lanczos dû au fait que le polynôme P_k n'est pas exactement de degré k ce qui provoque l'impossibilité d'utilisation de certaines relations de récurrence. Il est connu sous le nom de "*ghost-breakdown*" ou *Lanczos-breakdown*.

Pour remédier à de telles situations, différentes procédures ont été définies récemment. Elles consistent soit à considérer uniquement les polynômes orthogonaux réguliers (ceux qui existent) qui peuvent être calculés et à chercher les relations de récurrence qu'ils vérifient (*look-ahead*), soit à calculer les polynômes orthogonaux réguliers en passant par d'autres polynômes intérmidaires (*look-around* et ALA), soit encore à modifier la méthode de Lanczos pour éviter ce genre de problème.

Un autre problème qui a attiré l'attention de nombreux auteurs ces dernières années concerne la résolution des systèmes d'équations linéaires à plusieurs seconds membres

$$AX = B, \qquad A \in I\!R^{N\times N} \quad \text{et} \quad B \in I\!R^{N\times s}, \quad \text{avec} \quad s \ll N.$$

Il survient dans plusieurs applications, notamment en chimie, en électromagnétisme, en mécanique des structures ...
Ainsi, différentes méthodes classiques de type Krylov ($s = 1$) ont été généralisées (BICG [29], QMR [34], GMRES [63]).

Dans le cas symétrique défini positif, la méthode du Gradient Conjugué par bloc (Bl-CG) [55] et ses variantes [54] sont très efficaces. Lorsque la matrice A est quelconque, le BL-BICG [55], le BL-GMRES [71, 79] et le BL-QMR [33] sont les généralisations par bloc

les plus connues. L'idée commune à ces méthodes est de construire une suite d'approximations matricielles $X_k = X_0 + Z_k$ où Z_k est une correction appartenant à l'espace de Krylov par bloc $\mathcal{K}_k(A, R_0)$ avec X_0 une approximation de départ, $R_0 = B - AX_0$ et $\mathcal{K}_k(A, R_0)$ l'espace vectoriel engendré par les colonnes des matrices $A^i R_0$ pour $i = 0, \ldots, k-1$. La correction Z_k est déterminée soit par une condition de (semi)minimisation, soit par une condition d'orthogonalité.
Notons que les méthodes par bloc, notamment la méthode de Lanczos par bloc [37, 38] et celle d'Arnoldi par bloc [65], sont aussi utilisées dans le calcul des valeurs propres pour les matrices de grande taille.

D'autres approches ont été également proposées pour résoudre les systèmes linéaires à plusieurs seconds membres. Citons les méthodes "Seed". Ces méthodes consistent à choisir un système linéaire particulier, à le résoudre avec une méthode de type Krylov et à projeter les résidus des autres systèmes sur l'espace de Krylov obtenu. Le processus est répété avec un autre système jusqu'à l'obtention de toutes les solutions. Cette technique a été utilisée dans [23, 72] pour le Gradient Conjugué et dans [71] pour la méthode du GMRES.

Dans cette thèse, deux thèmes essentiels sont abordés. Le premier concerne les méthodes du *look-ahead* pour éviter le problème du *breakdown* dans les méthodes de type Lanczos, le second est consacré à la méthode de Lanczos globale et ses variantes.

Dans le **chapitre 1**, nous étudions les polynômes orthogonaux formels utilisés dans la mise en œuvre de la méthode de Lanczos. Une étude détaillée du phénomène de *breakdown* qui risque de se produire dans les relations de récurrence utilisées dans le calcul de ces polynômes est établie. Nous donnons, suite à cette étude, un schéma représentant le rapport entre le *true* et le *ghost-breakdown*. Ces résultats permettent la généralisation, dans le cas d'un *breakdown*, de toutes les relations de récurrence en particulier celles dans lesquelles il se produit les deux types de *breakdown*. Une première application de ces résultats est proposée à travers une implantation du processus de Lanczos basée sur des relations de récurrence à deux termes dans lesquelles nous évitons le *true* et le *ghost-breakdown*. Ces relations de récurrence sont établies en utilisant un formalisme qui consiste à exprimer certains polynômes particuliers dans des bases formées par les polynômes orthogonaux réguliers et des polynômes complémentaires qui remplacent les polynômes orthogonaux qui n'existent pas.

Dans le **chapitre 2**, nous appliquons le *look-ahead* à la méthode BICG en utilisant les résultats, obtenus dans le premier chapitre, sur le rapport entre le "*true*" et le "*ghost-breakdown*". Dans la deuxième partie de ce chapitre, nous utilisons les polynômes orthogo-

naux formels pour introduire le préconditionneur dans quelques méthodes de type Lanczos qui utilisent le *look-ahead*, à savoir : La méthode MRZ-stab et HMRZ-stab, la méthode BIORES non générique, la méthode QMR avec *look-ahead* et la méthode CSBCG. Les exemples numériques présentés à la fin de ce chapitre permettent la comparaison de ces méthodes.

Dans le **chapitre 3**, nous proposons une normalisation de l'algorithme MRZ-stab. Elle consiste à remplacer les vecteurs de direction de MRZ-stab par des vecteurs normalisés par un produit scalaire ce qui permet de réduire le nombre de vecteurs et coefficients stockés. L'algorithme obtenu se comporte d'une manière plus stable numériquement et permet d'éviter les dépassements de capacité appelés $overflow$ et $underflow$ provoqués par les multiplications successives des matrices A et A^T par les vecteurs de directions. Des variantes de cet algorithme dans le cas symétrique non défini positif sont proposées. Nous montrons que dans le cas d'un *breakdown*, selon certains choix des vecteurs initiaux, le saut effectué ne peut pas dépasser deux. En tenant compte de ce résultat, ces vecteurs initiaux nous permettent de définir des versions de l'algorithme Lanczos/Orthodir dans lesquelles il ne se produit pas de *breakdown*.

Dans le **chapitre 4**, nous nous intéressons à la résolution des systèmes linéaires multiples. Nous introduisons la procédure du Lanczos globale qui consiste à projeter le résidu sur un sous-espace de Krylov matriciel. Ceci va nous permettre de définir le processus de Lanczos global ainsi que les méthodes de type Lanczos globales telles que le Gradient Biconjugué global (GL-BICG) et le BICGSTAB global. Des situations de *breakdown* peuvent survenir dans ces algorithmes. Pour résoudre ce problème, nous établissons le lien entre les méthodes globales et les polynômes orthogonaux formels. Ceci va nous permettre de proposer des versions avec *look-ahead* de certaines méthodes de type Lanczos globales.

Chapitre 1

Les polynômes orthogonaux et le problème du breakdown

Les polynômes orthogonaux formels sont à la base de plusieurs applications en analyse numérique. Ils interviennent implicitement dans la méthode de Lanczos [13, 15, 19, 31], dans les approximants de Padé [5, 75] ainsi que dans des méthodes d'extrapolation [8] pour accélérer la convergence des suites.
Dans ce travail, nous nous intéressons aux familles des polynômes orthogonaux formels adjacents utilisés dans la mise en œuvre de la méthode de Lanczos pour la tridiagonalisation d'une matrice et aussi pour la résolution des systèmes linéaires. Ces polynômes peuvent être calculés par des relations de récurrence dans lesquelles il risque de se produire des divisions par zéro. Cette situation est appelée "*breakdown*".
Le calcul de ces polynômes peut générer deux types de *breakdown*, que nous appelons *true* et *ghost-breakdown*. Ces deux problèmes sont dus à l'inexistence des polynômes orthogonaux calculés et dans certains cas à l'impossibilité de l'utilisation de certaines relations de récurrence. La technique du *look-ahead* apporte une solution à ce problème. Elle consiste à calculer les polynômes orthogonaux réguliers en sautant ceux qui n'existent pas. Cependant, cette technique a été appliquée à des relations de récurrence dans lesquelles il ne se produit qu'un seul type de *breakdown*. Par exemple, le *true-breakdown* a été complètement résolu par Brezinski, Redivo-Zaglia et Sadok [13, 15, 17], ainsi que par Draux [25] dans le cas des polynômes orthogonaux unitaires. Gutknecht lui, a proposé l'algorithme du BIORES non générique, qui est une version du BIORES dans laquelle on évite le *ghost-breakdown* [40].
Dans ce chapitre, nous allons montrer comment nous pouvons éviter à la fois le *ghost* et le *true-breakdown* lorsque ces derniers se produisent dans une relation de récurrence. Pour cela nous commençons par faire une étude détaillée des polynômes orthogonaux formels. Nous obtenons un schéma qui représente les polynômes orthogonaux formels réguliers

dans les différents cas de *breakdown* possibles ainsi que le rapport entre le *ghost* et le *true-breakdown*.
Les résultats obtenus vont nous permettre d'appliquer le *look-ahead* aux relations de récurrence dans lesquelles risque de se produire les deux types de *breakdown*. Nous allons prendre l'exemple des relations de récurrence utilisées dans la mise en œuvre du processus de tridiagonalisation de Lanczos par des récurrences a deux termes. La technique du *look-ahead* sera appliquée à travers un formalisme qui consiste tout simplement à exprimer certains polynômes particuliers dans des bases formées par les polynômes orthogonaux réguliers et complémentaires (les polynômes complémentaires seront définis dans le sous paragraphe 1.3.3).

Ce chapitre est organisé comme suit :

Au paragraphe 1.1 nous rappelons les polynômes orthogonaux formels utilisés dans la mise en œuvre de la méthode de Lanczos en donnant quelques-unes de leurs propriétés. Nous expliquons aussi le problème du *breakdown* qui peut survenir lors du calcul de ces polynômes par des relations de récurrence ; nous en distinguons deux types.

Au paragraphe 1.2 nous proposons un schéma représentant les différentes situations de *breakdown* possibles. En 1.2.1 nous commençons par étudier les deux familles de déterminants de Hankel responsables de l'existence des polynômes orthogonaux. En 1.2.2 nous établissons le rapport existant entre les sauts effectués par ces polynômes.

Au paragraphe 1.3 nous proposons un formalisme qui nous permet de développer les généralisations de toutes les relations de récurrence en évitant le *breakdown*. Nous commençons par appliquer ce formalisme aux relations de récurrence à trois termes pour retrouver le processus du Lanczos classique avec *look-ahead*. Ensuite, nous donnons une généralisation des relations de récurrence à deux termes qui nous permettra de mettre en œuvre le processus de Lanczos avec *look-ahead* par des récurrences à deux termes.

Le dernier paragraphe sera consacré aux résultats numériques.

1.1 Polynômes orthogonaux formels

1.1.1 Résultats préliminaires et définitions

Soit c la fonctionnelle linéaire définie sur l'espace des polynômes réels $\mathbb{R}[X]$ par

$$c(\xi^i) = c_i \qquad \text{pour } i = 0, 1, \ldots, \tag{1.1}$$

où les c_i sont des constantes réelles.
La famille des polynômes orthogonaux formels P_k par rapport à la fonctionnelle c est définie par

$$P_k \quad \text{de degré} \leq k$$

et

$$c(\xi^i P_k(\xi)) = 0 \qquad \text{pour } i = 0, \ldots, k-1. \tag{1.2}$$

Les conditions (1.2) sont appelées *conditions d'orthogonalité* et sont équivalentes à

$$c(p(\xi)P_k(\xi)) = 0$$

pour tout polynôme p de degré au plus égal à $k-1$.

Soit

$$P_k(\xi) = a_0 + a_1\xi + \cdots + a_k\xi^k.$$

Les conditions d'orthogonalité (1.2) appliquées au polynôme P_k nous donnent le système de k équations et $k+1$ inconnues suivant

$$a_0c_i + a_1c_{i+1} + \cdots + a_kc_{i+k} = 0, \qquad i = 0, \ldots, k-1. \tag{1.3}$$

Pour pouvoir déterminer P_k, nous rajoutons à ce système une $k+1^{ème}$ équation, appelée *condition de normalisation.* Ainsi, Le polynôme P_k existe si et seulement si la matrice du système obtenu est inversible.
Dans ce qui va suivre, nous donnons deux conditions de normalisation permettant de déterminer les deux types de polynômes orthogonaux formels utilisés dans la méthode de Lanczos.

- Nous notons par P_k le polynôme orthogonal formel normalisé par la condition

$$a_0 = P_k(0) = 1.$$

Ces polynômes interviennent naturellement dans la méthode de Lanczos. Nous allons voir, dans le deuxième chapitre, que ces polynômes permettent la mise en œuvre des différentes méthodes de type Lanczos.
À partir de cette condition de normalisation, le système (1.3) devient

$$\begin{pmatrix} 1 & 0 & \cdots & 0 \\ c_0 & c_1 & \cdots & c_k \\ \vdots & \vdots & & \vdots \\ c_{k-1} & c_k & \cdots & c_{2k-1} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_k \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

Alors, P_k peut s'écrire comme le rapport de déterminants suivant

$$P_k(\xi) = \begin{vmatrix} 1 & \xi & \cdots & \xi^k \\ c_0 & c_1 & \cdots & c_k \\ \vdots & \vdots & & \vdots \\ c_{k-1} & c_k & \cdots & c_{2k-1} \end{vmatrix} \Bigg/ \begin{vmatrix} c_1 & \cdots & c_k \\ \vdots & & \vdots \\ c_k & \cdots & c_{2k-1} \end{vmatrix}.$$

Lemme 1.1.1

Le polynôme P_k existe et est unique si et seulement si

$$H_k^{(1)} = \begin{vmatrix} c_1 & \cdots & c_k \\ \vdots & & \vdots \\ c_k & \cdots & c_{2k-1} \end{vmatrix} \neq 0.$$

De plus, P_k est de degré exactement k si et seulement si

$$H_k^{(0)} = \begin{vmatrix} c_0 & \cdots & c_{k-1} \\ \vdots & & \vdots \\ c_{k-1} & \cdots & c_{2k-2} \end{vmatrix} \neq 0.$$

• Notons $P_k^{(0)}$ le polynôme orthogonal formel par rapport à c dont le coefficient du terme de degré k est égal à 1 (polynôme orthogonal unitaire $a_k = 1$). Comme nous allons le voir dans la deuxième section de ce chapitre, les polynômes $P_k^{(0)}$ permettent la mise en œuvre de la méthode de tridiagonalisation de Lanczos.
Dans ce cas, le système (1.3) s'écrit

$$\begin{pmatrix} c_0 & \cdots & c_{k-1} & c_k \\ \vdots & & \vdots & \vdots \\ c_{k-1} & \cdots & c_{2k-2} & c_{2k-1} \\ 0 & \cdots & 0 & 1 \end{pmatrix} \begin{pmatrix} a_0 \\ \vdots \\ a_{k-1} \\ a_k \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix},$$

et nous avons

$$P_k^{(0)}(\xi) = \frac{\begin{vmatrix} c_0 & c_1 & \cdots & c_k \\ \vdots & \vdots & & \vdots \\ c_{k-1} & c_k & \cdots & c_{2k-1} \\ 1 & \xi & \cdots & \xi^k \end{vmatrix}}{H_k^{(0)}}.$$

Lemme 1.1.2

Le polynôme unitaire $P_k^{(0)}$ existe si et seulement si $H_k^{(0)} \neq 0$.

Definition 1.1.1
Le polynôme $P_k^{(0)}$ tel que $H_k^{(0)} \neq 0$ est appelé polynôme régulier. Sinon il est dit singulier.

Nous avons le Théorème suivant dû à Draux [25]. La démonstration proposée ici n'est pas la même que celle donnée dans [25] et les arguments que nous utilisons nous sont utiles pour la suite. Ce Théorème nous permet de trouver la taille des sauts existants entre deux polynômes réguliers $P_k^{(0)}$ successifs.

Théorème 1.1.1
Si $H_k^{(0)} \neq 0$, alors les propriétés suivantes sont équivalentes :
(i) $c(\xi^i P_k^{(0)}(\xi)) = 0 \quad pour \quad i = 0, \ldots, k+m-2 \quad et \quad c(\xi^{k+m-1} P_k^{(0)}(\xi)) \neq 0,$
(ii) $H_i^{(0)} = 0 \quad pour \quad i = k+1, \ldots, k+m-1 \quad et \quad H_{k+m}^{(0)} \neq 0.$

Preuve :

Nous avons $H_k^{(0)} \neq 0$, alors $P_k^{(0)}$ existe et vérifie la condition d'orthogonalité suivante

$$c(\xi^i P_k^{(0)}(\xi)) = 0 \quad \text{pour} \quad i = 0, \ldots, k-1. \tag{1.4}$$

D'autre part, si nous notons $M_i^{(0)} = \begin{pmatrix} c_0 & \cdots & c_{i-1} \\ \vdots & & \vdots \\ c_{i-1} & \cdots & c_{2i-2} \end{pmatrix}$, alors $det(M_i^{(0)}) = H_i^{(0)}$.
L'hypothèse $H_k^{(0)} \neq 0$ montre que la matrice $M_k^{(0)}$ est inversible.
Prenons

$$P_k^{(0)} = \lambda_0 + \lambda_1 \xi + \cdots + \lambda_{k-1} \xi^{k-1} + \xi^k.$$

$(i) \Rightarrow (ii)$
D'après (i), pour tout $k+1 \leq j \leq k+m-1$, nous avons

$$c(\xi^i P_k^{(0)}(\xi)) = 0 \quad \text{pour} \quad i = 0, \ldots, j-1. \tag{1.5}$$

Notant $\chi = (\lambda_0, \lambda_1, \ldots, \lambda_{k-1}, 1, 0 \ldots, 0)^T \in \mathbb{R}^j$, en tenant compte de (1.5) et du fait que

$$c(\xi^i P_k^{(0)}) = c_i \lambda_0 + \cdots + c_{i+k-1} \lambda_{k-1} + c_{k+i} \quad \text{pour tout} \quad i \in \mathbb{N}, \tag{1.6}$$

nous obtenons

$$M_j^{(0)} \chi = \begin{pmatrix} c_0\lambda_0 + \cdots + c_{k-1}\lambda_{k-1} + c_k \\ \vdots \\ c_{j-1}\lambda_0 + \cdots + c_{j+k-2}\lambda_{k-1} + c_{j+k-1} \end{pmatrix} = \begin{pmatrix} c(P_k^{(0)}) \\ \vdots \\ c(\xi^{j-1} P_k^{(0)}) \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix}.$$

Or, puisque $\chi \neq 0$, alors $M_j^{(0)}$ est une matrice singulière pour $j = k+1, \ldots, k+m-1$.. Nous en déduisons que

$$H_j^{(0)} = 0 \quad \text{pour} \quad j = k+1, \ldots, k+m-1.$$

Montrons maintenant que si en plus nous avons $c(\xi^{k+m-1} P_k^{(0)}(\xi)) \neq 0$ alors $H_{k+m}^{(0)} \neq 0$. Pour cela nous considérons la matrice $D_j = (d_{lp})$ de dimension $j \times j$ pour $j > k$, telle que

$$d_{ll} = 1 \quad \text{pour} \quad l = 1, \ldots, j,$$

$$d_{lp} = \lambda_{k+l-p} \qquad \text{pour} \quad p = k+1, \ldots, j \quad \text{et} \quad l = p-k, \ldots, p-1,$$

$$d_{lp} = 0 \qquad \text{ailleurs.}$$

En multipliant la matrice $M_{k+m}^{(0)}$ à droite par la matrice régulière D_{k+m}, nous obtenons

$$M_{k+m}^{(0)} D_{k+m} = \begin{pmatrix} c_0 & \cdots & c_{k-1} & \cdots & c_{k+m-1} \\ \vdots & & \vdots & & \vdots \\ c_{k-1} & \cdots & c_{2k-2} & \cdots & \vdots \\ \vdots & & \vdots & & \vdots \\ c_{k+m-1} & \cdots & c_{2k+m-2} & \cdots & c_{2k+2m-2} \end{pmatrix} \begin{pmatrix} 1 & & 0 & \lambda_0 & & 0 \\ & \ddots & & \vdots & \ddots & \\ & & 1 & \lambda_{k-1} & \ddots & \lambda_0 \\ & & & 1 & \ddots & \vdots \\ & & 0 & & \ddots & \lambda_{k-1} \\ & & & & & 1 \end{pmatrix}$$

$$= \left(\begin{array}{ccc|c} & M_k^{(0)} & & E_1 \\ \hline c_k & \cdots & c_{2k-1} & \\ \vdots & & \vdots & E_2 \\ c_{k+m-1} & \cdots & c_{2k+m-2} & \end{array}\right).$$

E_1 est une matrice de dimension $k \times m$ et E_2 une matrice de dimension $m \times m$ qui ont les formes suivantes

$$E_1 = \begin{pmatrix} c(\xi^0 P_k^{(0)}) & \cdots & c(\xi^{m-1} P_k^{(0)}) \\ \vdots & & \vdots \\ c(\xi^{k-1} P_k^{(0)}) & \cdots & c(\xi^{k+m-2} P_k^{(0)}) \end{pmatrix}, E_2 = \begin{pmatrix} c(\xi^k P_k^{(0)}) & \cdots & c(\xi^{k+m-1} P_k^{(0)}) \\ \vdots & \iddots & \vdots \\ c(\xi^{k+m-1} P_k^{(0)}) & \cdots & c(\xi^{k+2m-2} P_k^{(0)}) \end{pmatrix}.$$

Grâce à (1.5) et à l'hypothèse (i), nous obtenons

$$E_1 = 0 \quad \text{et} \quad E_2 = \begin{pmatrix} 0 & \cdots & 0 & c(\xi^{k+m-1} P_k^{(0)}) \\ \vdots & \iddots & \iddots & \vdots \\ 0 & \iddots & & \vdots \\ c(\xi^{k+m-1} P_k^{(0)}) & \cdots & \cdots & c(\xi^{k+2m-2} P_k^{(0)}) \end{pmatrix}.$$

Ainsi, le complément de Schur de $M_k^{(0)}$ dans $M_{k+m}^{(0)}D_{k+m}$, noté $([M_{k+m}^{(0)}D_{k+m}]/M_k^{(0)})$, est égal à la matrice E_2. Et nous avons

$$det(M_{k+m}^{(0)}D_{k+m}) = det(M_k^{(0)})\ det([M_{k+m}^{(0)}D_{k+m}]/M_k^{(0)}) = det(M_k^{(0)})\ det(E_2).$$

Puisque les matrices $M_k^{(0)}$ et D_{k+m} sont non-singulières, alors

$$H_{k+m}^{(0)} = 0 \iff c(\xi^{k+m-1}P_k^{(0)}(\xi)) = 0.$$

$(ii) \Rightarrow (i)$
Par définition, le polynôme $P_k^{(0)}$ vérifie la relation suivante

$$c(\xi^i P_k^{(0)}(\xi)) = 0 \quad \text{pour} \quad i = 0, \ldots, k-1.$$

Pour $j = k$, nous montrons que sous l'hypothèse (ii) (en particulier $H_{k+1}^{(0)} = 0$) nous avons $c(\xi^k P_k^{(0)}) = 0$.
En multipliant à droite la matrice $M_{k+1}^{(0)}$ par la matrice régulière D_{k+1}, nous obtenons

$$M_{k+1}^{(0)}D_{k+1} = \left(\begin{array}{c|c} M_k^{(0)} & 0 \\ \hline c_k \ \cdots \ c_{2k-1} & c(\xi^k P_k^{(0)}) \end{array}\right).$$

En tenant compte du fait que $H_k^{(0)} \neq 0$ et que $H_{k+1}^{(0)} = 0$, le calcul du déterminant de ces deux matrices nous permet d'avoir $c(\xi^k P_k^{(0)}) = 0$.
De la même manière, nous montrons que $c(\xi^j P_k^{(0)}) = 0$ pour $j = k+1, \ldots, m+k-2$ à partir de $c(\xi^i P_k^{(0)}) = 0$ pour $i = 0, \ldots, j-1$ et de l'hypothèse (ii)

□

Dans le Théorème qui va suivre, nous donnons un premier résultat qui nous permet de vérifier l'existence des polynômes P_k par l'intermédiaire des polynômes $P_k^{(0)}$.

Théorème 1.1.2
Si $H_k^{(0)} \neq 0$, alors les deux propriétés suivantes sont équivalentes
(i) $P_k^{(0)}(0) = 0$.
(ii) $H_k^{(1)} = 0$

Preuve :

Puisque $H_k^{(0)} \neq 0$, d'après sa définition, $P_k^{(0)}$ existe et vérifie

$$P_k^{(0)}(0) = (-1)^k \frac{H_k^{(1)}}{H_k^{(0)}}. \tag{1.7}$$

D'où le résultat.

□

A partir de la définition de P_k et $P_k^{(0)}$, nous avons le résultat suivant

Lemme 1.1.3
Si $H_k^{(0)} \neq 0$ *et* $H_k^{(1)} \neq 0$ *alors*

$$P_k(\xi) = \frac{P_k^{(0)}(\xi)}{P_k^{(0)}(0)}. \tag{1.8}$$

D'après les lemmes 1.1.1, 1.1.2 et 1.1.3 nous avons les remarques suivantes

Remarque 1.1.1

- *Si* P_k *existe, alors il est de degré exactement égal à* k *si et seulement si le polynôme* $P_k^{(0)}$ *existe.*
- *Les polynômes* P_k*, lorsqu'ils existent, peuvent être calculés à partir des polynômes* $P_k^{(0)}$.

Lemme 1.1.4
Si P_k *existe et est de degré* k*, alors les propriétés suivantes sont équivalentes :*
(i) $c(\xi^i P_k(\xi)) = 0 \quad pour \quad i = 0, \ldots, k+m-2,$
$c(\xi^{k+m-1} P_k(\xi)) \neq 0$
(ii) $H_i^{(0)} = 0 \quad pour \quad i = k+1, \ldots, k+m-1,$
$H_{k+m}^{(0)} \neq 0.$

Preuve :

Puisque P_k existe et est de degré égal à k, alors $H_k^{(0)} \neq 0$ et $P_k^{(0)}$ existe et vérifie $P_k^{(0)}(0) \neq 0$. D'après la relation (1.8) donnée dans Lemme 1.1.3, nous avons

$$P_k^{(0)}(\xi) = P_k^{(0)}(0) P_k(\xi).$$

Grâce au Théorème 1.1.1, nous pouvons montrer l'équivalence.

□

Dans ce Lemme, nous retrouvons les conditions d'existence des polynômes $P_k^{(0)}$ que nous pouvons donc déterminer à partir des relations d'orthogonalité vérifiées par les polynômes P_k. Ce Lemme, comme le Théorème 1.1.1, nous permet de détecter les sauts entre les polynômes orthogonaux formels $P_k^{(0)}$ qui existent. Toutefois, ces relations d'orthogonalité ne nous permettent pas de connaître le saut entre les polynômes réguliers P_k.

1.1.2 Relations de récurrence

L'une des propriétés les plus intéressantes des polynômes orthogonaux formels est le fait de pouvoir les calculer à partir de relations de récurrence [4, 7, 13, 25]. Par exemple, les polynômes orthogonaux P_k (ainsi que $P_k^{(0)}$) vérifient la relation de récurrence à trois termes suivante

$$P_{k+1}(\xi) = (\alpha_k \xi + \beta_k) P_k(\xi) + \gamma_k P_{k-1}(\xi), \tag{1.9}$$

avec $P_{-1}(\xi) = 0$ et $P_0(\xi) \neq 0$.
Grâce à cette relation, nous pouvons calculer récursivement la famille des polynômes $\{P_k\}$ (ainsi que $\{P_k^{(0)}\}$) lorsque ces derniers existent.
La condition de normalisation de ces polynômes nous fournit une relation supplémentaire nous permettant de calculer les coefficients α_k, β_k et γ_k. Nous donnons les expressions de ces coefficients en utilisant les deux conditions de normalisation établies précédemment.

1. Les polynômes P_k avec la condition de normalisation : $P_k(0) = 1 \quad \forall k$.
   En appliquant cette condition à la relation de récurrence (1.9), nous obtenons d'abord une premiere relation $\beta_k + \gamma_k = 1$.
   Ensuite, nous multiplions la relation (1.9) par un polynôme arbitraire U_i de degré exactement égal à i, et nous lui appliquons la fonctionnelle c.
   Les conditions d'orthogonalité (1.2) nous permettent d'avoir

   $$\begin{aligned} \text{pour} \quad i &= k-1 \qquad \alpha_k c(\xi U_{k-1} P_k) + \gamma_k c(U_{k-1} P_{k-1}) = 0, \\ \text{pour} \quad i &= k \qquad\quad \alpha_k c(\xi U_k P_k) + \beta_k c(U_k P_k) + \gamma_k c(U_k P_{k-1}) = 0. \end{aligned}$$

   Ainsi, nous obtenons un système de trois relations à trois inconnues. Son déterminant est donné par

   $$D_k = c(\xi U_{k-1} P_k)[c(U_k P_{k-1}) - c(U_k P_k)] + c(\xi U_k P_k) + c(U_{k-1} P_{k-1}),$$

   Si $D_k = 0$, alors il se produit un *breakdown* dans la relation de récurrence. Le choix de U_k est arbitraire, nous pouvons prendre par exemple $U_k = P_k$ ou bien $U_k = \xi^k$.
2. Les polynômes $P_k^{(0)}$ avec la condition de normalisation : $P_k^{(0)}$ polynômes unitaires.
   À partir de cette condition, nous avons $\alpha_k = 1$ et la relation de récurrence (1.9) devient

   $$P_{k+1}^{(0)}(\xi) = \xi P_k^{(0)}(\xi) + \beta_k P_k^{(0)}(\xi) + \gamma_k P_{k-1}^{(0)}(\xi).$$

   Nous pouvons obtenir les coefficients β_k et γ_k en procédant de la même manière que dans le premier cas de normalisation. Ainsi, nous obtenons

   $$\begin{aligned} \gamma_k &= -\frac{c(\xi U_{k-1} P_k^{(0)})}{c(U_{k-1} P_{k-1}^{(0)})}, \\ \beta_k &= -\frac{c(\xi U_k P_k^{(0)}) + \gamma_k c(U_k P_{k-1}^{(0)})}{c(U_k P_k^{(0)})}. \end{aligned}$$

Si $c(U_k P_k^{(0)}) = 0$ un *breakdown* se produit dans la relation de récurrence. Le choix $U_k = P_k^{(0)}$ nous permet de simplifier les calculs grâce aux conditions d'orthogonalité. Le polynôme P_{k+1} peut aussi être calculé par cette relation en divisant le polynôme $P_{k+1}^{(0)}$ par le coefficient $P_{k+1}^{(0)}(0)$. Ce dernier est obtenu par récurrence à partir de la relation

$$P_{k+1}^{(0)}(0) = \beta_k P_k^{(0)}(0) + \gamma_k P_{k-1}^{(0)}(0).$$

Il existe une autre procédure pour calculer récursivement les polynômes orthogonaux formels. Elle consiste à calculer simultanément la famille des polynômes orthogonaux formels par rapport à la fonctionnelle $c^{(1)}$ définie par

$$c^{(1)}(\xi^i) = c(\xi^{i+1}) = c_{i+1} \qquad i = 0, 1, \ldots$$

Nous notons $\{P_k^{(1)}\}$ la famille des polynômes orthogonaux formels unitaires par rapport à $c^{(1)}$. Ces polynômes vérifient les relations d'orthogonalité suivantes

$$c^{(1)}(\xi^i P_k^{(1)}(\xi)) = 0 \qquad \text{pour } i = 0, \ldots, k-1, \tag{1.10}$$

et peuvent s'écrire sous la forme

$$P_k^{(1)}(\xi) = \frac{\begin{vmatrix} c_1 & c_2 & \cdots & c_{k+1} \\ \vdots & \vdots & & \vdots \\ c_k & c_{k+1} & \cdots & c_{2k} \\ 1 & \xi & \cdots & \xi^k \end{vmatrix}}{H_k^{(1)}}.$$

Avant de donner un exemple de ces relations de récurrence, établissons d'abord quelques-unes des propriétés des polynômes $P_k^{(1)}$.

Lemme 1.1.5
Le polynôme $P_k^{(1)}$ existe sous la condition $H_k^{(1)} \neq 0$.

Definition 1.1.2
Le polynôme $P_k^{(1)}$ tel que $H_k^{(1)} \neq 0$ est appelé polynôme régulier. Sinon il est dit singulier.

Nous avons le Théorème suivant dont la démonstration est similaire à celle du Théorème 1.1.1.

Théorème 1.1.3
Si $H_k^{(1)} \neq 0$, alors les propriétés suivantes sont équivalentes :
(i) $c^{(1)}(\xi^i P_k^{(1)}(\xi)) = 0$ pour $i = 0, \ldots, k+m-2$, et $c^{(1)}(\xi^{k+m-1} P_k^{(1)}(\xi)) \neq 0$,
(ii) $H_i^{(1)} = 0$ pour $i = k+1, \ldots, k+m-1$, et $H_{k+m}^{(1)} \neq 0$.

Remarque 1.1.2

Le Théorème 1.1.3 permet de trouver la taille des sauts existants entre deux polynômes réguliers $P_k^{(1)}$ successifs. D'autre part, comme les polynômes P_k existent sous la condition $H_k^{(1)} \neq 0$, alors, ce Théorème nous donnent également le saut entre deux polynômes réguliers P_k successifs.

Les deux familles de polynômes $\{P_k\}$ et $\{P_k^{(1)}\}$ sont dites familles adjacentes de polynômes orthogonaux formels.
Il existe plusieurs relations de récurrence reliant ces deux familles de polynômes [4, 5, 7, 9, 25] ; par exemple, nous avons les relations

$$\begin{aligned} P_{k+1}(\xi) &= \alpha_k \xi P_k^{(1)}(\xi) + \beta_k P_k(\xi), \\ \text{et} & \\ P_{k+1}^{(1)}(\xi) &= \gamma_k P_{k+1}(\xi) + \delta_k P_k^{(1)}(\xi), \end{aligned} \tag{1.11}$$

avec $P_0(\xi) = 1$ et $P_0^{(1)}(\xi) = 1$.
Les conditions de normalisation des polynômes P_k et $P_k^{(1)}$ nous fournissent deux relations supplémentaires qui nous permettent de calculer les coefficients α_k, β_k, γ_k et δ_k.
En effet, soient $\{U_i\}$ et $\{V_i\}$ deux familles de polynômes arbitraires telles que U_i et V_i soient de degré exactement égal à i, $\forall i \in \mathbb{N}$.

- Sous la condition : $\forall k, \quad P_k(0) = 1$, nous obtenons $\beta_k = 1$.
  Ainsi, la première relation de (1.11) devient

  $$P_{k+1}(\xi) = \alpha_k \xi P_k^{(1)}(\xi) + P_k(\xi).$$

  Nous multiplions cette relation par U_k. Ensuite, nous lui appliquons la fonctionnelle c et grâce aux relations d'orthogonalité (1.2) et (1.10), nous obtenons

  $$\alpha_k = -\frac{c(U_k P_k)}{c(\xi U_k P_k^{(1)})} = -\frac{c(U_k P_k)}{c^{(1)}(U_k P_k^{(1)})}.$$

  Compte-tenu du fait que les polynômes $P_k^{(1)}$ sont unitaires par définition, et d'après les relations (1.11), α_k représente le coefficient de degré $k+1$ du polynôme P_{k+1}. Ainsi, nous avons

  $$\gamma_k = \frac{1}{\alpha_k} = -\frac{c^{(1)}(U_k P_k^{(1)})}{c(U_k P_k)}.$$

  Pour le calcul de δ_k, nous multiplions la deuxième équation de (1.11) par V_k et nous lui appliquons la fonctionnelle $c^{(1)}$ pour obtenir, grâce aux conditions d'orthogonalité (1.2) et (1.10), la relation suivante

  $$\delta_k = -\gamma_k \frac{c^{(1)}(V_k P_{k+1})}{c^{(1)}(V_k P_k^{(1)})} = \frac{c(\xi V_k P_{k+1})}{c(V_k P_k)}.$$

Nous remarquons que les deux relations proposées ici permettent de calculer P_{k+1} et $P_{k+1}^{(1)}$ sous les conditions $c(V_kP_k) \neq 0$ et $c^{(1)}(U_kP_k^{(1)}) \neq 0$. Dans le cas contraire, il se produit un *breakdown* dans les relations de récurrence.

- Les polynômes $P_k^{(0)}$ sont unitaires, alors $\alpha_k = \gamma_k = 1$.
  Dans ce cas, les deux relations (1.11) deviennent

$$\begin{aligned} P_{k+1}^{(0)}(\xi) &= \xi P_k^{(1)}(\xi) + \beta_k P_k^{(0)}(\xi), \\ P_{k+1}^{(1)}(\xi) &= P_{k+1}^{(0)}(\xi) + \delta_k P_k^{(1)}(\xi). \end{aligned}$$

  Les coefficients β_k et δ_k sont calculés de la même façon que dans le premier cas, ce qui nous donne

$$\beta_k = -\frac{c^{(1)}(U_kP_k^{(1)})}{c(U_kP_k^{(0)})} \qquad \text{et} \qquad \delta_k = -\frac{c(\xi V_kP_{k+1}^{(0)})}{c^{(1)}(V_kP_k^{(1)})}.$$

  Ici, le calcul des polynômes $P_{k+1}^{(0)}$ et $P_{k+1}^{(1)}$ dépend des conditions $c^{(1)}(V_kP_k^{(1)}) \neq 0$ et $c(U_kP_k^{(0)}) \neq 0$.
  Comme dans le cas des relations de récurrence à trois termes, nous remarquons que le polynôme P_{k+1} peut aussi être calculé en divisant $P_{k+1}^{(0)}$ par le coefficient $P_{k+1}^{(0)}(0)$ donné par la récurrence

$$P_{k+1}^{(0)}(0) = \beta_k P_k^{(0)}(0).$$

Nous venons de citer deux exemples de relations de récurrence permettant de calculer les polynômes orthogonaux formels. Dans ces relations, en général, nous faisons des divisions par des quantités qui peuvent être égales à zéro, ainsi nous risquons d'avoir des divisions par zéro (*breakdown*). Dans la sous-section suivante, nous effectuerons une étude détaillée de ce phénomène.

1.1.3 Problème du breakdown

Les polynômes orthogonaux formels peuvent être calculés par des relations de récurrence dans lesquelles il risque de se produire un problème de *breakdown*. Prenons l'exemple des relations (1.11) avec U_k et V_k deux polynômes arbitraires de degré exactement égal à k. Le calcul du polynôme P_{k+1} dépend de la condition $c^{(1)}(U_kP_k^{(1)}) \neq 0$. D'après le Théorème 1.1.3 nous avons $c^{(1)}(U_kP_k^{(1)}) = 0$ si et seulement si $H_{k+1}^{(1)} = 0$, et dans ce cas les deux polynômes P_{k+1} et $P_{k+1}^{(1)}$ n'existent pas.

Definition 1.1.3

Nous disons que dans une relation de récurrence nous avons un true-breakdown [13, 16] à la $k^{ème}$ itération si et seulement si $H_k^{(1)} = 0$.

D'autre part, l'existence des polynômes P_{k+1} et $P_{k+1}^{(1)}$ n'est pas suffisante pour pouvoir utiliser les relations (1.11) puisque nous avons aussi la condition $c(U_k P_k) \neq 0$ et $c(V_k P_k) \neq 0$. D'après le Lemme 1.1.4, nous avons $c(U_k P_k) = 0$ si et seulement si $H_{k+1}^{(0)} = 0$.

Definition 1.1.4

Dans une relation de récurrence, nous disons que nous avons un ghost-breakdown [13, 16] à la $k^{ème}$ itération si et seulement si $H_k^{(0)} = 0$.

Remarque 1.1.3

1. *Nous constatons ici que le ghost-breakdown n'est pas dû à l'inexistence des polynômes P_{k+1} et $P_{k+1}^{(1)}$ mais plutôt au fait que le polynôme P_{k+1} ne soit pas de degré égal à $k+1$ ($H_{k+1} = 0$), ce qui rend impossible l'utilisation des relations (1.11) puisque ces dernières exigent que le polynôme P_k soit de degré égal à k, $\forall k$.*
2. *Les relations de récurrence (1.11) nous permettent aussi de calculer les deux familles de polynômes $\{P_k^{(0)}\}$ et $\{P_k^{(1)}\}$. Le calcul du polynôme $P_{k+1}^{(0)}$ dépend de la condition $c(U_k P_k^{(0)}) \neq 0$. Dans le cas contraire nous avons $H_{k+1}^{(0)} = 0$. Il se produit ainsi un ghost-breakdown dans les relations de récurrence, ce qui est équivalent à l'inexistence du polynôme $P_{k+1}^{(0)}$.*
3. *Nous avons une autre condition liée au calcul du polynôme $P_{k+1}^{(1)}$: il s'agit de $c^{(1)}(V_k P_k^{(1)}) \neq 0$. Dans le cas où cette quantité est égale à zéro, ce qui est équivalent à $H_{k+1}^{(1)} = 0$, il se produit un true-breakdown dû à l'inexistence du polynôme $P_{k+1}^{(1)}$.*

Nous venons de définir les deux types de *breakdown* susceptibles de se produire dans le calcul des familles des polynômes orthogonaux formels P_k, $P_k^{(1)}$ et $P_k^{(0)}$.

- Le *true-breakdown*, dû à l'inexistence des polynômes P_k et $P_k^{(1)}$, $H_k^{(1)} = 0$.
- Le *ghost-breakdown*, dû à l'inexistence des polynômes $P_k^{(0)}$ et aussi au fait que les polynômes P_k ne sont pas de degré exactement égal à k, $H_k^{(0)} = 0$.

Le *ghost-breakdown* peut rendre impossible l'utilisation de certaines relations de récurrence malgré l'existence des polynômes orthogonaux que nous cherchons à calculer.

Il existe plusieurs stratégies pour résoudre le problème de *breakdown*.

- Nous savons qu'il est possible d'obtenir le polynôme orthogonal régulier suivant à partir de ceux qui le précèdent en résolvant un système régulier. Il suffit alors de sauter les polynômes qui n'existent pas et de calculer seulement ceux qui existent, cette technique est appelée *look-ahead* [9, 13, 15, 25, 40, 58].
- Le *breakdown* correspond à un bloc carré de polynômes adjacents presque identiques dans la table des polynômes orthogonaux formels. Il est possible de tourner autour

d'un tel bloc au lieu de sauter des polynômes. Cette stratégie s'appelle *look-around* [39].

- La procédure appelé ALA consiste à calculer les polynômes réguliers en passant par des polynômes biorthogonaux [3].

1.2 Propriétés des polynômes orthogonaux formels réguliers

Les déterminants de Hankel $H_k^{(0)}$ et $H_k^{(1)}$ ont un rapport direct avec le *ghost* et le *true-breakdown* qui peuvent se produire dans les relations de récurrence. Dans ce qui va suivre, nous commençons par étudier les deux familles des déterminants $\{H_k^{(0)}\}$ et $\{H_k^{(1)}\}$, en particulier le rapport entre ceux différents de zéro. Ensuite, nous donnons un schéma pour l'existence des polynômes P_k, $P_k^{(1)}$ et $P_k^{(0)}$, ce schéma représente le rapport entre le *ghost* et le *true-breakdown*.

1.2.1 lien entre les déterminants $\{H_k^{(0)}\}$ et $\{H_k^{(1)}\}$

Dans cette partie, nous donnons des propriétés vérifiées par les déterminants $\{H_k^{(0)}\}$ et $\{H_k^{(1)}\}$ qui illustrent les liens existants entre ces déterminants.

Lemme 1.2.1
Si $H_{k+1}^{(0)} \neq 0$ *et* $H_{k+1}^{(1)} = 0$ *alors*
(i) $H_k^{(1)} \neq 0$,
(ii) $P_{k+1}^{(0)}(\xi) = \xi P_k^{(1)}(\xi)$.

Preuve :

Nous avons $H_{k+1}^{(0)} \neq 0$. Alors, le polynôme $P_{k+1}^{(0)}$ existe et vérifie la relation d'orthogonalité suivante

$$c(\xi^i P_{k+1}^{(0)}) = 0 \qquad \text{pour} \quad i = 0, \ldots, k. \tag{1.12}$$

Or, puisque $H_{k+1}^{(1)} = 0$, d'après le Théorème 1.1.2 nous avons $P_{k+1}^{(0)}(0) = 0$ et nous pouvons donc écrire

$$P_{k+1}^{(0)}(\xi) = \xi \bar{P}_k(\xi)$$

où $\bar{P}_k$ est un polynôme unitaire de degré exactement k donné par

$$\bar{P}_k(\xi) = \lambda_0 + \lambda_1 \xi + \cdots + \lambda_{k-1}\xi^{k-1} + \xi^k.$$

La relation (1.12) devient

$$c(\xi^i P_{k+1}^{(0)}) = c(\xi^{i+1} \bar{P}_k) = c^{(1)}(\xi^i \bar{P}_k) = 0 \qquad \text{pour} \quad i = 0, \ldots, k. \tag{1.13}$$

$\Rightarrow (i)$
Notons L_{k+1} la matrice de dimension $(k+1)\times(k+1)$

$$L_{k+1} = \begin{pmatrix} 1 & 0 & \cdots & 0 & \lambda_0 \\ & \ddots & \ddots & \vdots & \vdots \\ & & \ddots & 0 & \vdots \\ & & & 1 & \lambda_{k-1} \\ & & & & 1 \end{pmatrix}.$$

En multipliant la matrice $M_{k+1}^{(0)}$ à droite par la matrice L_{k+1} et en utilisant (1.12), nous obtenons

$$M_{k+1}^{(0)} L_{k+1} = \begin{pmatrix} c_0 & \cdots & c_{k-1} & c(\bar{P}_k) \\ c_1 & \cdots & c_k & c^{(1)}(\bar{P}_k) \\ \vdots & & \vdots & \vdots \\ c_k & \cdots & c_{2k-1} & c^{(1)}(\xi^{k-1}\bar{P}_k) \end{pmatrix} = \left(\begin{array}{ccc|c} c_0 & \cdots & c_{k-1} & c(\bar{P}_k) \\ & M_k^{(1)} & & 0 \end{array}\right)$$

avec $M_k^{(1)} = \begin{pmatrix} c_1 & \cdots & c_k \\ \vdots & & \vdots \\ c_k & \cdots & c_{2k-1} \end{pmatrix}$.

Puisque les deux matrices $M_{k+1}^{(0)}$ et L_{k+1} sont régulières, par un simple calcul de déterminant nous en déduisons que $H_k^{(1)} = det(M_k^{(1)}) \neq 0$ et $c(\bar{P}_k) \neq 0$.

$\Rightarrow (ii)$
Par définition, $P_k^{(1)}$ est l'unique polynôme de degré au plus égal à k vérifiant la condition d'orthogonalité (1.10). Puisque $\bar{P}_k$ est de degré k et vérifie les relations (1.13) donc $\bar{P}_k = P_k^{(1)}$, d'où $P_{k+1}^{(0)}(\xi) = \xi P_k^{(1)}(\xi)$.

□

Théorème 1.2.1

Soient $H_k^{(0)} \neq 0$, $H_k^{(1)} \neq 0$ *et* $m > 1$.
Si $H_i^{(0)} = 0$ *pour* $i = k+1, \ldots, k+m-1$ *et* $H_{k+m}^{(0)} \neq 0$,
alors $H_j^{(1)} = 0$ *pour* $j = k+1, \ldots, k+m-2$ *et* $H_{k+m-1}^{(1)} \neq 0$.

Preuve :

Par hypothèse, d'après le Théorème 1.1.1, le polynôme $P_k^{(0)}$ existe et vérifie les relations suivantes

$$\begin{aligned} c(\xi^i P_k^{(0)}(\xi)) &= 0 \quad \text{pour} \quad i = 0, \ldots, k+m-2 \\ c(\xi^{k+m-1} P_k^{(0)}(\xi)) &\neq 0. \end{aligned}$$

Puisque $m > 1$, nous pouvons écrire ces relations sous la forme suivante

$$\begin{aligned} c(\xi^{i+1}P_k^{(0)}(\xi)) &= c^{(1)}(\xi^i P_k^{(0)}(\xi)) = 0 \quad \text{pour} \quad i = 0, \ldots, k+m-3 \quad (1.14) \\ c(\xi^{(k+m-2)+1}P_k^{(0)}(\xi)) &= c^{(1)}(\xi^{k+m-2}P_k^{(0)}(\xi)) \neq 0. \end{aligned}$$

Notons $P_k^{(0)}(\xi) = \lambda_0 + \lambda_1\xi + \cdots + \lambda_{k-1}\xi^{k-1} + \xi^k$.
Pour tout $k+1 \leq j \leq k+m-2$, nous multiplions la matrice $M_j^{(1)}$ par le vecteur

$$\chi^j = (\lambda_0, \ldots, \lambda_{k-1}, 1, 0, \ldots, 0)^T \in \mathbb{R}^j,$$

et nous obtenons

$$M_j^{(1)}\chi^j = \begin{pmatrix} c_1 & \cdots & c_k & c_{k+1} & \cdots & c_j \\ \vdots & & \vdots & \vdots & & \vdots \\ c_k & \cdots & c_{2k-1} & c_{2k} & \cdots & c_{k+j-1} \\ \vdots & & \vdots & \vdots & & \vdots \\ c_j & \cdots & c_{j+k-1} & c_{j+k} & \cdots & c_{2j-1} \end{pmatrix} \begin{pmatrix} \lambda_0 \\ \vdots \\ \lambda_{k-1} \\ 1 \\ \vdots \\ 0 \end{pmatrix} = \begin{pmatrix} c^{(1)}(P_k^{(0)}) \\ \vdots \\ \vdots \\ c^{(1)}(\xi^{j-1}P_k^{(0)}) \end{pmatrix}.$$

Grâce à (1.14), nous avons $M_j^{(1)}\chi^j = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix}$.

Puisque $\chi^j \neq 0$, la matrice $M_j^{(1)}$ est singulière, d'où

$$H_j^{(1)} = det(M_j^{(1)}) = 0 \quad \text{pour} \quad j = k+1, \ldots, k+m-2.$$

Considérant maintenant la matrice D_{k+m-1} déjà défini dans la preuve du Théorème 1.1.1, le produit $M_{k+m-1}^{(1)}D_{k+m-1}$ est donné par

$$M_{k+m-1}^{(1)}D_{k+m-1} = \left(\begin{array}{ccc|c} & M_k^{(1)} & & E \\ \hline c_{k+1} & \cdots & c_{2k} & \\ \vdots & & \vdots & F \\ c_{k+m-1} & \cdots & c_{2k+m-2} & \end{array}\right).$$

E est une matrice de dimension $k \times (m-1)$ et F une matrice de dimension $(m-1) \times (m-1)$, qui ont les formes suivantes

$$E = \begin{pmatrix} c^{(1)}(P_k^{(0)}) & \cdots & c^{(1)}(\xi^{m-2}P_k^{(0)}) \\ \vdots & & \vdots \\ c^{(1)}(\xi^{k-1}P_k^{(0)}) & \cdots & c^{(1)}(\xi^{k+m-3}P_k^{(0)}) \end{pmatrix}$$

$$F = \begin{pmatrix} c^{(1)}(\xi^k P_k^{(0)}) & \cdots & c^{(1)}(\xi^{k+m-2} P_k^{(0)}) \\ \vdots & & \vdots \\ c^{(1)}(\xi^{k+m-2} P_k^{(0)}) & \cdots & c^{(1)}(\xi^{k+2m-4} P_k^{(0)}) \end{pmatrix}.$$

Nous obtenons, grâce à (1.14)

$$E = 0 \quad \text{et} \quad F = \begin{pmatrix} 0 & \cdots & 0 & c^{(1)}(\xi^{k+m-2} P_k^{(0)}) \\ \vdots & \ddots & \iddots & \vdots \\ 0 & \iddots & & \vdots \\ c^{(1)}(\xi^{k+m-2} P_k^{(0)}) & \cdots & \cdots & c^{(1)}(\xi^{k+2m-4} P_k^{(0)}) \end{pmatrix}.$$

Par un simple calcul de déterminant, nous obtenons

$$\left|det(M_{k+m-1}^{(1)} D_{k+m-1})\right| = \left|det(M_k^{(1)})\right| \times \left|[c^{(1)}(\xi^{k+m-2} P_k^{(0)})]^{m-1}\right|,$$

avec $det(M_k^{(1)}) = H_k^{(1)} \neq 0$ et $det(D_{k+m-1}) \neq 0$.
Puisque $c^{(1)}(\xi^{k+m-2} P_k^{(0)}) \neq 0$ alors $det(M_{k+m-1}^{(1)}) = H_{k+m-1}^{(1)} \neq 0$.

□

Théorème 1.2.2

Soient $H_k^{(0)} \neq 0$ *et* $H_k^{(1)} = 0$.
Si $H_i^{(0)} = 0$ *pour* $i = k+1, \ldots, k+m-1$ *et* $H_{k+m}^{(0)} \neq 0$,
alors $H_{k-1}^{(1)} \neq 0$, $H_j^{(1)} = 0$ *pour* $j = k+1, \ldots, k+m-1$ *et* $H_{k+m}^{(1)} \neq 0$.

Preuve :

En supposant $H_k^{(0)} \neq 0$ et $H_k^{(1)} = 0$, grâce au Lemme 1.2.1, nous avons $H_{k-1}^{(1)} \neq 0$ et le polynôme $P_{k-1}^{(1)}$ existe et vérifie $P_k^{(0)}(\xi) = \xi P_{k-1}^{(1)}(\xi)$.
D'autre part, d'après le Théorème 1.1.1, le polynôme $P_k^{(0)}$ vérifie les relations suivantes

$$\begin{aligned} c(\xi^i P_k^{(0)}(\xi)) &= 0 \quad \text{pour} \quad i = 0, \ldots, k+m-2 \\ c(\xi^{k+m-1} P_k^{(0)}(\xi)) &\neq 0, \end{aligned}$$

donc, nous pouvons écrire

$$\begin{aligned} c(\xi^{i+1} P_{k-1}^{(1)}(\xi)) &= c^{(1)}(\xi^i P_{k-1}^{(1)}(\xi)) = 0 \quad \text{pour} \quad i = 0, \ldots, k+m-2 \\ c(\xi^{(k+m-1)+1} P_{k-1}^{(1)}) &= c^{(1)}(\xi^{k+m-1} P_{k-1}^{(1)}) \neq 0. \end{aligned}$$

Ainsi, en appliquant le Théorème 1.1.3 au polynôme $P_{k-1}^{(1)}$, nous obtenons

$$H_j^{(1)} = 0 \quad \text{pour} \quad j = k+1, \ldots, k+m-1 \quad \text{et} \quad H_{k+m}^{(1)} \neq 0.$$

□

1.2.2 Lien entre les sauts de $P_k^{(1)}$ et ceux de $P_k^{(0)}$

Les Théorèmes 1.2.1 et 1.2.2, établis dans la sous-section précédente, nous donnent le rapport entre les déterminants $H_k^{(0)}$ et $H_k^{(1)}$ différents de zéro. Cela nous donne le rapport entre l'existence des polynômes $P_k^{(0)}$ et celle des polynômes $P_k^{(1)}$ et P_k. Nous allons maintenant construire un schéma qui représente les polynômes réguliers $P_k^{(0)}$, $P_k^{(1)}$ et P_k ainsi que le lien existant entre les sauts effectués par ces polynômes.

Nous notons par $0 = n_0 < n_1 < n_2 < \cdots < n_k < \cdots$ les degrés des polynômes $P_{n_k}^{(1)}$ qui existent (i.e $H_{n_k}^{(1)} \neq 0$). Cette condition nous garantit l'existence des polynômes P_{n_k} sans qu'ils soient nécessairement de degré n_k.
Notons m_k la longueur du saut qui existe entre $P_{n_k}^{(1)}$ et $P_{n_{k+1}}^{(1)}$, i.e $n_{k+1} = n_k + m_k$.
D'après Le Théorème 1.1.1, m_k est défini par les conditions

$$\begin{aligned} c^{(1)}(\xi^i P_{n_k}^{(1)}) &= 0 \quad \text{pour} \quad i = 0, \ldots, n_k + m_k - 2, \\ &\neq 0 \quad \text{pour} \quad i = n_k + m_k - 1. \end{aligned} \tag{1.15}$$

Notons par $0 = \bar{n}_0 < \bar{n}_1 < \bar{n}_2 < \cdots < \bar{n}_l < \cdots$ les indices des polynômes unitaires réguliers $P_{\bar{n}_l}^{(0)}$ qui existent (i.e $H_{\bar{n}_l}^{(0)} \neq 0$). Ces polynômes vérifient les relations suivantes

$$\begin{aligned} c(\xi^i P_{\bar{n}_l}^{(0)}) &= 0 \quad \text{pour} \quad i = 0, \ldots, \bar{n}_l + \bar{m}_l - 2, \\ &\neq 0 \quad \text{pour} \quad i = \bar{n}_l + \bar{m}_l - 1. \end{aligned} \tag{1.16}$$

avec $\bar{m}_l$ le saut entre $P_{\bar{n}_l}^{(0)}$ et $P_{\bar{n}_{l+1}}^{(0)}$, $\bar{n}_{l+1} = \bar{n}_l + \bar{m}_l$.

Dans le cas où le polynôme P_{n_k} est de degré exactement égal à n_k, il existe $l \in \mathbb{N}$ tel que $n_k = \bar{n}_l$; dans ce cas $H_{n_k}^{(0)} = H_{\bar{n}_l}^{(0)} \neq 0$ et $H_{\bar{n}_l}^{(1)} = H_{n_k}^{(1)} \neq 0$.

Lemme 1.2.2
Soit P_{n_k} de degré exactement égal à n_k, i.e $\exists l \in \mathbb{N}$ tel que $n_k = \bar{n}_l$.
Si $H_{\bar{n}_{l+1}}^{(1)} = 0$ alors $H_{\bar{n}_{l+2}}^{(1)} \neq 0$.

Preuve :

P_{n_k} existe et est de degré exactement n_k ($\exists\ l \in \mathbb{N}$ tel que $n_k = \bar{n}_l$), donc le polynôme unitaire $P_{\bar{n}_l}^{(0)}$ existe.
Commençons par donner la relation de récurrence permettant de construire tous les polynômes unitaires réguliers $\{P_k^{(0)}\}$. Pour cela nous exprimons le polynôme $P_{\bar{n}_{l+2}}^{(0)} - \xi^{\bar{m}_{l+1}} P_{\bar{n}_{l+1}}^{(0)}$ de degré $\bar{n}_{l+1} + \bar{m}_{l+1} - 1$ dans la base suivante

$$\{P_{\bar{n}_0}^{(0)}, \xi P_{\bar{n}_0}^{(0)}, \ldots, \xi^{\bar{m}_0 - 1} P_{\bar{n}_0}^{(0)}, P_{\bar{n}_1}^{(0)}, \xi P_{\bar{n}_1}^{(0)}, \ldots, \xi^{\bar{m}_l - 1} P_{\bar{n}_l}^{(0)}, P_{\bar{n}_{l+1}}^{(0)}, \xi P_{\bar{n}_{l+1}}^{(0)}, \ldots, \xi^{\bar{m}_{l+1} - 1} P_{\bar{n}_{l+1}}^{(0)}\},$$

et nous obtenons alors la relation

$$\begin{aligned} P^{(0)}_{\bar{n}_{l+2}} &= \alpha^{(0)}_0 P^{(0)}_{\bar{n}_0} + \alpha^{(0)}_1 \xi P^{(0)}_{\bar{n}_0} + \cdots + \alpha^{(0)}_{\bar{m}_0-1} \xi^{\bar{m}_0-1} P^{(0)}_{\bar{n}_0} + \cdots \\ &\quad + \alpha^{(l)}_0 P^{(0)}_{\bar{n}_l} + \alpha^{(l)}_1 \xi P^{(0)}_{\bar{n}_l} + \cdots + \alpha^{(l)}_{\bar{m}_l-1} \xi^{\bar{m}_l-1} P^{(0)}_{\bar{n}_l} \\ &\quad + \alpha^{(l+1)}_0 P^{(0)}_{\bar{n}_{l+1}} + \alpha^{(l+1)}_1 \xi P^{(0)}_{\bar{n}_{l+1}} + \cdots + \alpha^{(l+1)}_{\bar{m}_{l+1}-1} \xi^{\bar{m}_{l+1}-1} P^{(0)}_{\bar{n}_{l+1}} + \xi^{\bar{m}_{l+1}} P^{(0)}_{\bar{n}_{l+1}}. \end{aligned}$$

En utilisant les relations d'orthogonalité (1.16), nous avons

$$\begin{aligned} \alpha^{(j)}_i &= 0 \quad \text{pour} \quad i = 0, \ldots, \bar{m}_j - 1 \quad \text{et} \quad j = 0, \ldots, l-1. \\ \alpha^{(l)}_s &= 0 \quad \text{pour} \quad s = 1, \ldots, \bar{m}_l - 1. \end{aligned}$$

Ce qui nous permet d'écrire

$$P^{(0)}_{\bar{n}_{l+2}} = \alpha^{(l)}_0 P^{(0)}_{\bar{n}_l} + \alpha^{(l+1)}_0 P^{(0)}_{\bar{n}_{l+1}} + \cdots + \alpha^{(l+1)}_{\bar{m}_{l+1}-1} \xi^{\bar{m}_{l+1}-1} P^{(0)}_{\bar{n}_{l+1}} + \xi^{\bar{m}_{l+1}} P^{(0)}_{\bar{n}_{l+1}}. \tag{1.17}$$

Supposons que $H^{(1)}_{\bar{n}_{l+1}} = 0$, il nous faut montrer que $H^{(1)}_{\bar{n}_{l+2}} \neq 0$.
Cela est équivalent, d'après le Théorème 1.1.2, à supposer que $P^{(0)}_{\bar{n}_{l+1}}(0) = 0$ et à montrer que $P^{(0)}_{\bar{n}_{l+2}}(0) \neq 0$.
Ainsi, à partir de (1.17), nous pouvons écrire

$$\begin{aligned} P^{(0)}_{\bar{n}_{l+2}}(0) &= \alpha^{(l)}_0 P^{(0)}_{\bar{n}_l}(0) + \alpha^{(l+1)}_0 P^{(0)}_{\bar{n}_{l+1}}(0) \\ &= \alpha^{(l)}_0 P^{(0)}_{\bar{n}_l}(0). \end{aligned}$$

Puisque P_{n_k} existe et est de degré exactement $n_k = \bar{n}_l$, nous avons $P^{(0)}_{\bar{n}_l}(0) \neq 0$.
D'autre part, grâce aux conditions d'orthogonalité (1.16), nous pouvons calculer les coefficients de (1.17), et en particulier $\alpha^{(l)}_0$.
En effet, en multipliant (1.17) par $\xi^{\bar{n}_l+\bar{m}_l-1}$ et en appliquant la fonctionnelle c, nous obtenons grâce à (1.16)

$$\alpha^{(l)}_0 = \frac{c(\xi^{\bar{m}_{l+1}+\bar{n}_l+\bar{m}_l-1} P^{(0)}_{\bar{n}_{l+1}})}{c(\xi^{\bar{n}_l+\bar{m}_l-1} P^{(0)}_{\bar{n}_l})} = \frac{c(\xi^{\bar{n}_{l+1}+\bar{m}_{l+1}-1} P^{(0)}_{\bar{n}_{l+1}})}{c(\xi^{\bar{n}_l+\bar{m}_l-1} P^{(0)}_{\bar{n}_l})}.$$

Puisque $\bar{m}_l$ et $\bar{m}_{l+1}$ sont déterminés par les conditions d'orthogonalité (1.16) appliquées respectivement aux polynômes $P^{(0)}_{\bar{n}_l}$ et $P^{(0)}_{\bar{n}_{l+1}}$, c'est à dire

$$\begin{aligned} c(\xi^i P^{(0)}_{\bar{n}_l}) &= 0 \quad \text{pour} \quad i = 0, \ldots, \bar{n}_l + \bar{m}_l - 2, \\ &\neq 0 \quad \text{pour} \quad i = \bar{n}_l + \bar{m}_l - 1, \end{aligned}$$

et

$$\begin{aligned} c(\xi^i P^{(0)}_{\bar{n}_{l+1}}) &= 0 \quad \text{pour} \quad i = 0, \ldots, \bar{n}_{l+1} + \bar{m}_{l+1} - 2, \\ &\neq 0 \quad \text{pour} \quad i = \bar{n}_{l+1} + \bar{m}_{l+1} - 1, \end{aligned}$$

nous en déduisons que $\alpha^{(l)}_0 \neq 0$, et donc $P^{(0)}_{\bar{n}_{l+2}}(0) \neq 0$. D'où le résultat.

□

Le résultat de ce Lemme montre que si, à une itération donnée nous avons $n_k = \bar{n}_l$, alors la longueur du saut entre les polynômes réguliers $P^{(1)}_{n_k}$ et $P^{(1)}_{n_{k+1}}$ ne peut pas dépasser la valeur $\bar{m}_l + \bar{m}_{l+1}$, i.e $m_k \leq \bar{m}_l + \bar{m}_{l+1}$.
En particulier lorsque nous n'avons pas de *ghost-breakdown*, c'est-à-dire $\forall k \in I\!N \ H^{(0)}_k \neq 0$, la longueur des sauts entre les polynômes réguliers $P^{(1)}_{n_k}$ ne dépasse pas la valeur $m_k = 2$. Ce résultat a permis de construire l'algorithme du CSBCG dû à Chan et Bank [21, 22].

Théorème 1.2.3
On suppose que P_{n_k} est de degré exactement n_k (i.e n_k est égal à un indice $\bar{n}_l$ tel que $P^{(0)}_{\bar{n}_l}$ existe), alors on a les propriétés suivantes :
(i) si $\bar{m}_l = 1$ et $m_k \neq 1$ alors $m_k = \bar{m}_{l+1} + 1$
(ii) si $\bar{m}_l \neq 1$ alors $m_k = \bar{m}_l - 1$. De plus

1. *si $m_{k+1} \neq 1$ alors $m_{k+1} = \bar{m}_{l+1} + 1$ et $m_k + m_{k+1} = \bar{m}_l + \bar{m}_{l+1}$.*
2. *si $m_{k+1} = 1$ alors $m_k + m_{k+1} = \bar{m}_l$.*

Preuve :

Les sauts m_k et $\bar{m}_l$ sont déterminés par les relations (1.15) et (1.16). Le polynôme P_{n_k} étant de degré exactement égal à $n_k = \bar{n}_l$, nous avons alors $H^{(0)}_{\bar{n}_l} = H^{(0)}_{n_k} \neq 0$ et $H^{(1)}_{\bar{n}_l} = H^{(1)}_{n_k} \neq 0$.

$\Rightarrow (ii)$
Dans le cas où $\bar{m}_l \neq 1$, grâce au Théorème 1.1.1, nous avons

$$H^{(0)}_i = 0 \quad \text{pour} \quad i = \bar{n}_l + 1, \ldots, \bar{n}_l + \bar{m}_l - 1 \quad \text{et} \quad H^{(0)}_{\bar{n}_l + \bar{m}_l} \neq 0,$$

et, en appliquant le Théorème 1.2.1, nous obtenons

$$H^{(1)}_i = 0 \quad \text{pour} \quad i = \bar{n}_l + 1, \ldots, \bar{n}_l + \bar{m}_l - 2 \quad \text{et} \quad H^{(1)}_{\bar{n}_l + \bar{m}_l - 1} \neq 0.$$

Comme $n_k = \bar{n}_l$, ces relations montrent, grâce au théorème 1.1.3, que $m_k = \bar{m}_l - 1$.

$\Rightarrow (ii.1)$
D'après (ii), nous avons $m_k = \bar{m}_l - 1$ alors $n_{k+1} = \bar{n}_{l+1} - 1$, donc $H^{(1)}_{n_{k+1}} = H^{(1)}_{\bar{n}_{l+1} - 1} \neq 0$. Par définition, $H^{(0)}_{\bar{n}_{l+1}} \neq 0$ et le saut $\bar{m}_{l+1}$ est déterminé par (1.16) appliquée au polynôme $P^{(0)}_{\bar{n}_{l+1}}$, ce qui est équivalent à

$$H^{(0)}_i = 0 \quad \text{pour} \quad i = \bar{n}_{l+1} + 1, \ldots, \bar{n}_{l+1} + \bar{m}_{l+1} - 1 \quad \text{et} \quad H^{(0)}_{\bar{n}_{l+1} + \bar{m}_{l+1}} \neq 0.$$

Par hypothèse, $m_{k+1} \neq 1$, donc $H^{(1)}_{n_{k+1}+1} = H^{(1)}_{\bar{n}_{l+1}} = 0$.
Alors, en appliquant le théorème 1.2.2, nous obtenons

$$H^{(1)}_i = 0 \quad \text{pour} \quad i = \bar{n}_{l+1}+1, \ldots, \bar{n}_{l+1}+\bar{m}_{l+1}-1 \quad \text{et} \quad H^{(1)}_{\bar{n}_{l+1}+\bar{m}_{l+1}} \neq 0,$$

et comme $n_{k+1} = \bar{n}_{l+1} - 1$, nous pouvons écrire

$$H^{(1)}_i = 0 \quad \text{pour} \quad i = n_{k+1}+2, \ldots, n_{k+1}+\bar{m}_{l+1} \quad \text{et} \quad H^{(1)}_{n_{k+1}+\bar{m}_{l+1}+1} \neq 0.$$

A partir du Théorème 1.1.3 et des relations (1.15), nous avons donc
$m_{k+1} = \bar{m}_{l+1} + 1$, d'où $m_k + m_{k+1} = \bar{m}_l + \bar{m}_{l+1}$.

$\Rightarrow (ii.2)$.
Évident.

$\Rightarrow (i)$
Dans le cas où $\bar{m}_l = 1$ et $m_k \neq 1$, nous avons

$$H^{(1)}_{n_k+1} = H^{(1)}_{\bar{n}_{l+1}} = 0 \quad \text{et} \quad H^{(0)}_{n_k+1} = H^{(0)}_{\bar{n}_{l+1}} \neq 0 \quad \text{avec} \quad \bar{n}_{l+1} = \bar{n}_l + 1 = n_k + 1.$$

Le saut $\bar{m}_{l+1}$ est déterminé par les relations (1.16) qui sont équivalentes à

$$H^{(0)}_i = 0 \quad \text{pour} \quad i = \bar{n}_{l+1}+1, \ldots, \bar{n}_{l+1}+\bar{m}_{l+1}-1 \quad \text{et} \quad H^{(0)}_{\bar{n}_{l+1}+\bar{m}_{l+1}} \neq 0.$$

Nous avons $H^{(1)}_{\bar{n}_{l+1}} = 0$, alors en appliquant le Théorème 1.2.2, nous obtenons

$$H^{(1)}_i = 0 \quad \text{pour} \quad i = \bar{n}_{l+1}+1, \ldots, \bar{n}_{l+1}+\bar{m}_{l+1}-1 \quad \text{et} \quad H^{(1)}_{\bar{n}_{l+1}+\bar{m}_{l+1}} \neq 0.$$

Comme $\bar{n}_{l+1} = n_k + 1$, alors nous pouvons écrire

$$H^{(1)}_i = 0 \quad \text{pour} \quad i = n_k+2, \ldots, n_k+\bar{m}_{l+1} \quad \text{et} \quad H^{(1)}_{n_k+\bar{m}_{l+1}+1} \neq 0,$$

et, à partir de ce résultat, nous avons donc $m_k = \bar{m}_{l+1} + 1$.

□

Ce Théorème nous donne le lien entre les sauts m_k et $\bar{m}_l$ effectués par les polynômes P_k, $P^{(1)}_k$ et $P^{(0)}_k$. Il représente, ainsi, les différents cas possibles d'existence des polynômes P_{n_k}, $P^{(1)}_{n_k}$ d'une part, et celle des polynômes $P^{(0)}_{n_k}$ d'autre part.
Nous retrouvons parmi ces résultats celui du Lemme 1.2.2.

Lemme 1.2.3
Soit P_{n_k} de degré n_k. Si $P_{n_{k+1}}$ n'est pas de degré exactement égal à n_{k+1} alors $P_{n_{k+1}} = P_{n_k}$.

Preuve :

Puisque $P_{n_{k+1}}$ n'est pas de degré n_{k+1}, alors $H^{(0)}_{n_{k+1}} = 0$.
De plus, à partir du Théorème 1.2.3, nous avons

$$H_i^{(0)} = 0 \quad \text{pour} \quad i = n_k + 1, \ldots, n_{k+1} - 1,$$

ce qui est équivalent, grâce au Lemme 1.1.4, aux relations suivantes

$$c(\xi^i P_{n_k}(\xi)) = 0 \quad \text{pour} \quad i = 0, \ldots, n_{k+1} - 1.$$

Or, par définition, comme $P_{n_{k+1}}$ est l'unique polynôme de degré au plus n_{k+1} vérifiant les relations

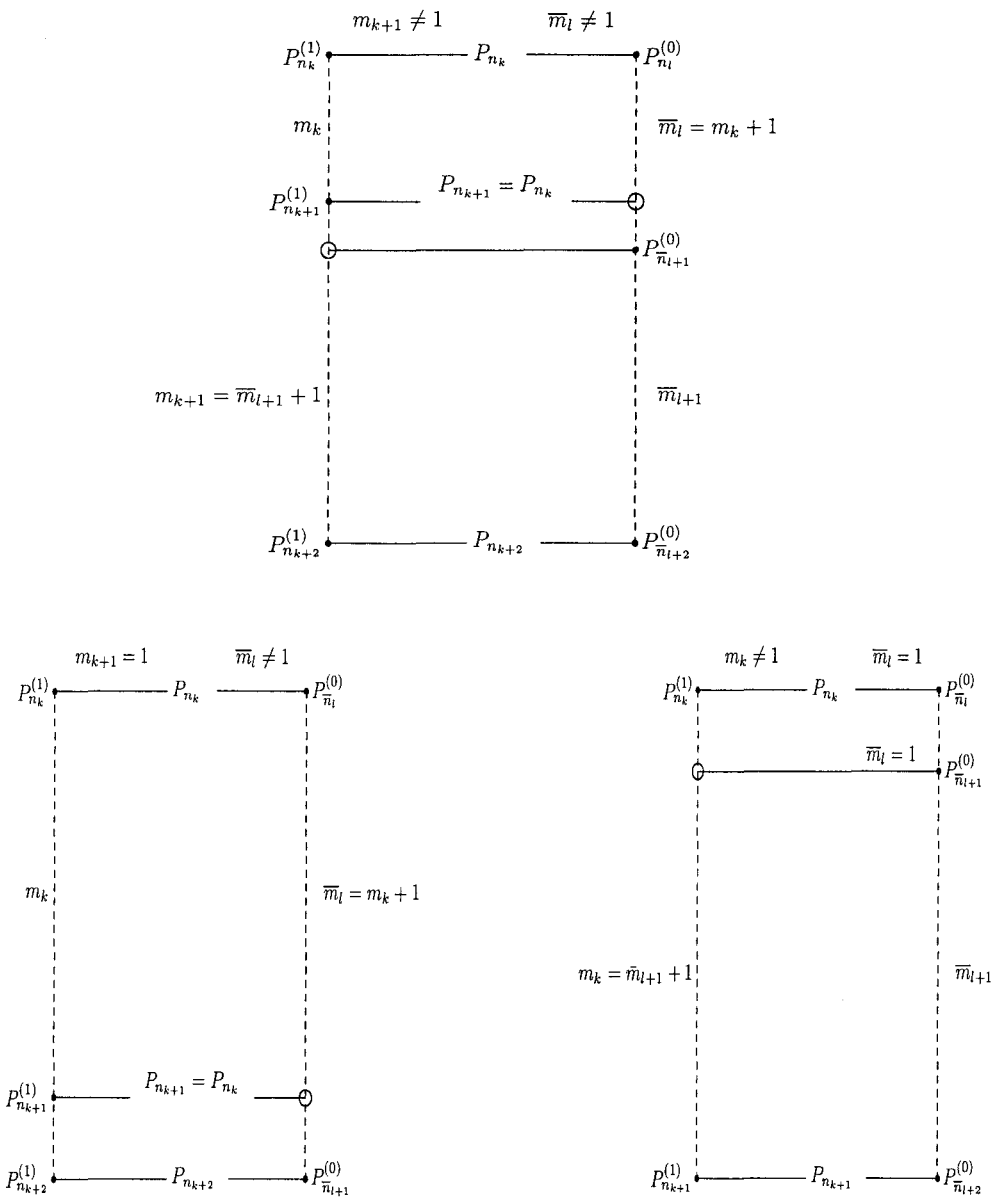
$$c(\xi^i P_{n_{k+1}}(\xi)) = 0 \quad \text{pour} \quad i = 0, \ldots, n_{k+1} - 1,$$

alors $P_{n_{k+1}} = P_{n_k}$.

□

Le Lemme 1.2.3 nous rappelle que le polynôme P_k, lorsqu'il existe, n'est pas toujours de degré k. L'existence de ce polynôme est liée à celle de $P_k^{(1)}$, contrairement au fait qu'il soit de degré k qui dépend de l'existence de $P_k^{(0)}$. Donc, établir le rapport entre l'existence des polynômes $P_k^{(0)}$ et $P_k^{(1)}$ nous permettra de prévoir celle des polynômes P_k ainsi que leur degré. Le Théorème 1.2.3 peut être interprété d'une manière polynômiale de façon à ce qu'il nous révèle le rapport entre les polynômes existants $P_k^{(0)}$ et $P_k^{(1)}$.
En effet, en supposant qu'à une itération donnée n_k nous ayons un polynôme P_{n_k} de degré exactement égal à n_k, ce qui correspond à l'existence d'un $l \in I\!N$ tel que $n_k = \bar{n}_l$, cela implique l'existence du polynôme $P_{n_k}^{(1)}$ ainsi que celle de $P_{n_k}^{(0)}$. Les relations (1.16) nous donnent le saut $\bar{m}_l$ qu'il peut y avoir entre $P_{\bar{n}_l}^{(0)}$ et $P_{\bar{n}_{l+1}}^{(0)}$.
Dans le cas où $\bar{m}_l = 1$, le polynôme $P_{\bar{n}_l+1}^{(0)}$ existe. Ici nous n'avons pas de *ghost-breakdown*, et si le saut m_k est égal à 1 alors il n'y a pas de *true-breakdown* non plus. Le calcul des polynômes orthogonaux à l'itération $n_k + 1$ peut se faire sans problème de *breakdown*.
Cependant, d'après le Théorème 1.2.3, nous pouvons avoir $\bar{m}_l = 1$ et $m_k \neq 1$, et nous avons donc un *true-breakdown* à l'itération $n_k + 1$. Dans ce cas le saut m_k est égal à $\bar{m}_{l+1} + \bar{m}_l$, ce qui se traduit par l'existence de $P_{\bar{n}_l+1}^{(0)} = P_{\bar{n}_{l+1}}^{(0)}$. En calculant le saut $\bar{m}_{l+1}$, nous obtenons à l'itération $\bar{n}_{l+2} = n_{k+1}$ l'existence des polynômes $P_{\bar{n}_{l+2}}^{(0)}$, $P_{n_{k+1}}^{(1)}$ et enfin $P_{n_{k+1}}$ qui est de degré exactement égal à $n_{k+1} = \bar{n}_{l+2}$.
Dans le cas où $\bar{m}_l \neq 1$ il se produit un *ghost-breakdown* à la $n_k + 1ème$ itération, et d'après le Théorème (1.2.3), nous avons $m_k = \bar{m}_l - 1$, ce qui implique que les polynômes $P_{n_{k+1}}^{(1)}$ et $P_{n_{k+1}}$ existent à l'itération $n_{k+1} = \bar{n}_{l+1} - 1$. Or, comme $P_{n_{k+1}}^{(0)}$ n'existe pas, alors,

FIG. 1.1 – Polynômes orthogonaux formels réguliers



d'après le Lemme 1.2.3, le polynôme $P_{n_{k+1}}$ n'est autre que le polynôme P_{n_k}. Ensuite, nous avons deux possibilités :
Soit $m_{k+1} = 1$, ce qui nous donne à l'itération $\bar{n}_{l+1} = n_{k+1} + 1 = n_{k+2}$ l'existence des polynômes $P^{(0)}_{n_{k+2}} = P^{(0)}_{\bar{n}_{l+1}}$, $P^{(1)}_{n_{k+2}}$ et $P_{n_{k+2}}$ de degré exactement égal à n_{k+2}.
Soit $m_{k+1} \neq 1$. Dans ce cas nous avons l'existence du polynôme $P^{(0)}_{\bar{n}_{l+1}} = P^{(0)}_{n_{k+1}+1}$, et d'après le Théorème 1.2.3, nous avons $\bar{m}_{l+1} = m_{k+1} - 1$. Donc à l'itération $n_{k+2} = \bar{n}_{l+2}$, nous avons l'existence des polynômes $P^{(0)}_{n_{k+2}} = P^{(0)}_{\bar{n}_{l+2}}$, $P^{(1)}_{n_{k+2}}$ et $P_{n_{k+2}}$ de degré exactement égal à n_{k+2}.

Il existe donc trois cas de *breakdown* possibles. Le schéma donné dans la figure 1.1 représente les polynômes $P^{(1)}_k$, $P^{(0)}_k$ et P_k ainsi que le rapport entre ces polynômes selon leur existence dans le cas d'un *breakdown*.

1.3 Application au processus de Lanczos

Le processus de Lanczos, proposé par Cornelius Lanczos [50] en 1950, est une méthode qui permet de transformer une matrice $A \in \mathbb{R}^{N \times N}$ en une matrice tridiagonale semblable en construisant, à partir de deux vecteurs initiaux, deux bases bi-orthogonales des deux sous-espaces de Krylov associés respectivement à A et A^T.
Dans cet algorithme, une division par zéro risque de se produire, ce qui cause un problème de *breakdown*. Pour remédier à cette situation, une première version de l'algorithme de Lanczos avec look-ahead a été proposée par Parlett, Taylor et Liu [58]. Cependant, cette procédure est assez compliquée et les détails de sa mise en œuvre ont été donnés seulement pour le cas où l'on aurait un saut égal à 2. C'est peut être la raison pour laquelle l'idée du *look-ahead* n'a pas suscité beaucoup d'intérêt jusqu'en 1989. Freund, Gutknecht et Nachtigal [31] ont donné une nouvelle approche de cet algorithme qui a servi à mettre en œuvre l'algorithme du QMR [35]. En 1992, Freund et Nachtigal [35] proposèrent une nouvelle mise en œuvre du processus de Lanczos basée sur des relations de récurrence à deux termes. L'algorithme obtenu étant plus stable que le premier à base de relations de récurrence à trois termes, a permis de rendre plus efficace l'algorithme du QMR.

Ici, dans un premier temps, nous rappelons le processus de tridiagonalisation de Lanczos et nous établissons sa connexion avec les polynômes orthogonaux formels. Ensuite, nous proposons de retrouver les relations de récurrence, utilisées dans l'algorithme de tridiagonalisation de Lanczos avec *look-ahead*, à travers un formalisme qui permet de généraliser toutes les relations de récurrence. Ce dernier consiste à exprimer certains polynômes particuliers dans des bases formées par les polynômes orthogonaux réguliers et des polynômes complémentaires qui remplacent les polynômes orthogonaux inexistants. Les deux bases du processus de Lanczos sont construites à partir de récurrences à trois termes. Or il a

été observé que, dans le cadre de l'arithmétique à précision finie, les itérations induites par des récurrences à trois termes sont moins robustes que celle issues de récurrences à deux termes. Brezinski et Redivo-Zaglia [12] ont notamment montré que, pour le calcul des valeurs propres, le processus de Lanczos classique et moins stable que le processus de Lanczos obtenu par des récurrences à deux termes. Cependant, dans ce dernier, il risque de se produire le *ghost* et le *true-breakdown*. Dans la deuxième partie de cette section, nous commençons par construire le processus de Lanczos à l'aide de relations de récurrence à deux termes. Ensuite, en utilisant les résultats obtenus dans la section précédente, nous établissons une généralisation de ces relations de récurrence que nous utilisons pour mettre en œuvre une nouvelle version du processus de Lanczos avec *look-ahead*.

1.3.1 L'algorithme classique de tridiagonalisation de Lanczos

Soit $A \in I\!R^{N\times N}$. Nous commençons par donner deux vecteurs initiaux v_0 et w_0. Le processus de Lanczos consiste à calculer deux suites finies de vecteurs (v_n) et (w_n), telles que, $\forall n$

$$\begin{aligned} vect\{v_0,\ldots,v_n\} &= \mathcal{K}_{n+1}(A,v_0),\\ vect\{w_0,\ldots,w_n\} &= \mathcal{K}_{n+1}(A^T,w_0), \end{aligned} \tag{1.18}$$

et

$$(w_j,v_i) = \begin{cases} \delta_i \neq 0 & i=j\\ 0 & i\neq j \end{cases} \qquad \text{pour} \quad i,j=0,1,\ldots,n. \tag{1.19}$$

Les vecteurs $\{v_i\}_{i=1}^n$ et $\{w_i\}_{i=1}^n$ sont calculés en utilisant les relations de récurrence à trois termes suivantes

$$\begin{aligned} v_{n+1} &= Av_n - \alpha_n v_n - \beta_n v_{n-1},\\ w_{n+1} &= A^T w_n - \alpha_n w_n - \beta_n w_{n-1}, \end{aligned} \tag{1.20}$$

avec

$$\alpha_n = \frac{(w_n,Av_n)}{\delta_n} \qquad \text{et} \qquad \beta_n = \frac{(w_{n-1},Av_n)}{\delta_{n-1}} = \frac{\delta_n}{\delta_{n-1}}. \tag{1.21}$$

Nous prenons dans la première itération $v_{-1}=0, \quad w_{-1}=0 \quad$ et $\quad \beta_{-1}=0$.

En pratique, pour des raisons de stabilité, la version normalisée du processus de Lanczos est utilisée afin d'éviter les problèmes de dépassement capacité connus sous les noms *overflow* et *underflow*. Elle consiste à calculer, dans les récurrences (1.20), les vecteurs v_n et w_n tels que $\|v_n\| = \|w_n\| = 1$ pour tout n. Ici, pour simplifier, nous allons employer les récurrences non normalisées (1.20), ces dernières peuvent s'écrire sous la forme matricielle

$$\begin{aligned} AV^{(n)} &= V^{(n)}H^{(n)} + [0 \quad \cdots \quad 0 \quad v_{n+1}],\\ A^TW^{(n)} &= W^{(n)}H^{(n)} + [0 \quad \cdots \quad 0 \quad w_{n+1}], \end{aligned} \tag{1.22}$$

avec

$$V^{(n)} := [v_0 \;\; v_1 \;\; \cdots \;\; v_n] \qquad \text{et} \qquad W^{(n)} := [w_0 \;\; w_1 \;\; \cdots \;\; w_n]$$

deux matrices dont les $i^{\text{ème}}$ colonnes sont respectivement v_{i-1} et w_{j-1}.

$$H^{(n)} := \begin{bmatrix} \alpha_0 & \beta_1 & 0 & \cdots & 0 \\ 1 & \alpha_1 & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \beta_n \\ 0 & \cdots & 0 & 1 & \alpha_n \end{bmatrix}$$

est la matrice tridiagonale de dimension $(n+1) \times (n+1)$ qui regroupe les coefficients des récurrences (1.20).
D'autre part, les conditions de bi-orthogonalité (1.19) peuvent s'écrire

$$W^{(n)^T} V^{(n)} = D^{(n)} := \begin{bmatrix} \delta_0 & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \delta_n \end{bmatrix}.$$

En considérant la matrice de dimension $(n+2) \times (n+1)$

$$H_e^{(n)} = \begin{bmatrix} H^{(n)} \\ 0 \cdots 0 \; 1 \end{bmatrix},$$

les relations (1.22) deviennent

$$\begin{aligned} AV^{(n)} &= V^{(n+1)} H_e^{(n)}, \\ A^T W^{(n)} &= W^{(n+1)} H_e^{(n)}. \end{aligned} \tag{1.23}$$

La terminaison régulière de cet algorithme s'effectue lorsque

$$v_L = 0 \quad \text{ou} \quad w_L = 0, \quad \text{avec} \quad L \leq N.$$

Dans ce cas, le sous-espace $\mathcal{K}_L(A, v_0)$ est invariant par rapport à A (resp. $\mathcal{K}_L(A^T, w_0)$ est invariant par rapport à A^T), et les vecteurs $\{v_i\}_{i=0}^{L-1}$ forment une base génératrice du sous-espace A-invariant $\mathcal{K}_L(A, v_0)$ de $\mathbb{R}^N$ (resp. les vecteurs $\{w_i\}_{i=0}^{L-1}$ forment une base génératrice du sous-espace A^T-invariant $\mathcal{K}_L(A^T, w_0)$).
Nous notons

$$L_v := dim\mathcal{K}_N(A, v_0) \qquad L_w := dim\mathcal{K}_N(A^T, w_0)$$

et

$$L^* = min\{L_v, L_w\}$$

Il y a une deuxième possibilité pour que l'algorithme soit interrompu ; c'est le cas où

$$(w_n, v_n) = 0 \qquad \text{avec} \quad v_n \neq 0 \quad \text{et} \quad w_n \neq 0.$$

Ici, l'algorithme s'arrête prématurément avant de calculer la base du sous-espace A-invariant (ou A^T- invariant). Par conséquent, nous ne pouvons pas calculer v_{n+1} et w_{n+1} en utilisant (1.20) car nous avons une division par zéro à la $(n+1)^{ème}$ itération, ce qui correspond au phénomène du *breakdown* dans l'algorithme de Lanczos appelé *serious-breakdown* par Wilkinson [80] et *Lanczos-breakdown* par Freund [34, 35]. Nous allons voir par la suite que cette situation est due au phénomène du *ghost-breakdown* qui se produit dans le calcul des polynômes orthogonaux formels.
Dans le cas où $(w_n, v_n) \neq 0$ mais est très proche de zéro, nous avons ce que nous appelons un *near-breakdown* qui provoque des problèmes d'instabilité numérique.

Nous remarquons que dans le cas symétrique $A = A^T$, nous n'avons pas de *breakdown* dans l'algorithme de Lanczos, puisque nous commençons avec $v_0 = w_0$, et à partir de (1.20) et (1.21) nous obtenons $w_n = v_n$ pour tout n. Donc

$$(w_n, v_n) = (v_n, v_n) = \|v_n\|^2 > 0 \quad \text{si} \quad v_n \neq 0.$$

Cela montre que le *serious-breakdown* (*ghost-breakdown*) ne peut pas se produire dans le processus symétrique de Lanczos.
D'autre part, seuls les vecteurs v_n sont calculés en utilisant la première relation (1.20).
Le processus de Lanczos est très utilisé dans le cas des matrices de grande taille. Grâce à ce processus nous pouvons calculer les valeurs propres de telles matrices. En effet, si $v_L = 0$ ou $w_L = 0$, alors nous en déduisons, d'après (1.22), que toute valeur propre de la matrice tridiagonale $H^{(L-1)}$ est aussi une valeur propre de la matrice A. Nous pouvons considérer, à chaque itération n, les valeurs propres de $H^{(n)}$ comme étant des approximations des valeurs propres de A, ces valeurs sont appelées valeurs de *Ritz*. De façon similaire, nous pouvons construire des approximations de la solution du système linéaire de grande taille $Ax = b$ en résolvant un nouveau système linéaire bien particulier dont $H^{(n)}$ est la matrice des coefficients. D'ailleurs, c'est ainsi que nous obtenons les méthode de Lanczos (BICG, Orthodir, Orthores, ...).

1.3.2 Connexion avec les polynômes orthogonaux formels

Pour mieux comprendre le problème du *breakdown* qui risque de se produire dans le processus de tridiagonalisation de Lanczos, nous établissons l'approche polynômiale de cet algorithme. Ainsi nous montrons que nous pouvons construire cet algorithme en utilisant les polynômes orthogonaux formels, lesquels vont nous être utiles par la suite pour, d'abord

mettre en œuvre l'algorithme de Lanczos avec *look-ahead* et ensuite construire le processus de Lanczos par des relations de récurrence à deux termes, et enfin éviter le *breakdown* dans ce dernier.

Soit

$$\Psi_n = \left\{ \psi(\xi) = a_0 + a_1\xi + a_2\xi^2 + \cdots + a_n\xi^n \quad / \quad a_i \in I\!R \right\}$$

l'espace vectoriel des polynômes réels de degré au plus égal à n. avec $n \leq L^*$
D'après la définition des sous-espace de Krylov $\mathcal{K}_{n+1}(A, v_0)$ et $\mathcal{K}_{n+1}(A^T, w_0)$, nous pouvons écrire

$$\mathcal{K}_{n+1}(A, v_0) = \{\psi(A)v_0 \quad /\psi \in \Psi_n\},$$
$$\mathcal{K}_{n+1}(A^T, w_0) = \{\psi(A^T)w_0 \quad /\psi \in \Psi_n\}.$$

En particulier, à partir de (1.20), nous avons

$$v_n = \psi_n(A)v_0 \qquad \text{et} \qquad w_n = \psi_n(A^T)w_0 \tag{1.24}$$

avec ψ_n un polynôme unique, de plus unitaire ($a_n = 1$).
Maintenant, si nous considérons la fonctionnelle c définie par

$$c(\xi^i) = (w_0, A^i v_0), \tag{1.25}$$

alors la relation d'orthogonalité (1.19) peut s'écrire sous la forme

$$c(\psi_n Q) = 0, \quad \forall\, Q \in \Psi_{n-1} \tag{1.26}$$

et

$$c(\psi_n^2) \neq 0. \tag{1.27}$$

Le polynôme ψ_n de degré exactement égal à n vérifiant (1.26) appartient à la famille des polynômes orthogonaux formels par rapport à la fonctionnelle c. Nous remarquons que ψ_n correspond au polynôme $P_n^{(0)}$ déjà défini auparavant.
Ces polynômes unitaires vérifient la relation de récurrence à trois termes suivante

$$P_{n+1}^{(0)}(\xi) = \xi P_n^{(0)}(\xi) - \alpha_n P_n^{(0)}(\xi) - \beta_n P_{n-1}^{(0)}(\xi), \tag{1.28}$$

avec $P_0^{(0)}(\xi) = 1$, $P_{-1}^{(0)}(\xi) = 0$ et $\beta_{-1} = 0$.
A partir de la relation (1.26), nous obtenons les expressions des coefficients de (1.28)

$$\alpha_n = \frac{c(\xi P_n^{(0)} P_n^{(0)})}{c(P_n^{(0)} P_n^{(0)})} \qquad \text{et} \qquad \beta_n = \frac{c(\xi P_n^{(0)} P_{n-1}^{(0)})}{c(P_{n-1}^{(0)} P_{n-1}^{(0)})} = \frac{c(P_n^{(0)} P_n^{(0)})}{c(P_{n-1}^{(0)} P_{n-1}^{(0)})}. \tag{1.29}$$

En utilisant (1.25), ces relations nous permettent de retrouver l'algorithme de Lanczos.

Dans le cas où $c(P_n^{(0)}P_n^{(0)}) = 0$, ce qui correspond à $(w_n, v_n) = 0$, nous ne pouvons pas calculer le polynôme $P_{n+1}^{(0)}$ puisque nous aurons une division par zéro dans le calcul des coefficients α_n et β_n. Nous avons ainsi un *ghost-breakdown* à la $(n+1)^{eme}$ itération.
Cette relation de récurrence correspond à la relation (1.9) étudiée dans la sous-section 1.1.2 ; ici nous choisissons $U_i = P_i^{(0)}$.

1.3.3 Le processus classique de Lanczos avec look-ahead

La procédure du *look-ahead* a été introduite dans la méthode classique de Lanczos par Freund et Nachtigal [34]. Elle permet d'éviter le problème du *ghost-breakdown* et de calculer les vecteurs réguliers de Lanczos en complétant les bases des sous-espace de Krylov par des vecteurs *complémentaires*. Les relations de récurrence utilisées dans cet algorithme ont été élaborées par Draux [25] et Gutknecht [40], elles représentent une généralisation sans *breakdown* des relations (1.28).
Ici, nous allons retrouver ces relations en utilisant un formalisme qui consiste à exprimer certains polynômes dans des bases contenant les polynômes orthogonaux réguliers et complétées par des polynômes particuliers.
Soient v_0 et w_0 deux vecteurs initiaux ; il existe un ensemble maximal d'indices $\bar{n}_j$

$$\{\bar{n}_0, \ldots, \bar{n}_J\} \subseteq \{1, 2, \ldots L^*\}, \quad \bar{n}_0 = 0 < \bar{n}_1 < \cdots < \bar{n}_J \leq L^*,$$

tel que pour tout $j = 0, \ldots, J$, il existe un polynôme *régulier* $P_{\bar{n}_j}^{(0)} \in \Psi_{n_j}$, de degré exactement égal à $\bar{n}_j$, avec $\{P_n^{(0)}\}$ la famille des polynômes orthogonaux formels par rapport à la fonctionnelle c définie par (1.25). En particulier, pour $\bar{n}_0 = 0$ le polynôme correspondant est $\bar{P}_0^{(0)}(\xi) = 1$.
Soit $\bar{m}_j = \bar{n}_{j+1} - \bar{n}_j$ le saut entre deux polynômes réguliers $P_{\bar{n}_j}^{(0)}$ et $P_{\bar{n}_{j+1}}^{(0)}$.

Nous appelons les vecteurs

$$v_{\bar{n}_j} = P_{\bar{n}_j}^{(0)}(A)v_0 \qquad \text{et} \qquad w_{\bar{n}_j} = P_{\bar{n}_j}^{(0)}(A^T)w_0$$

vecteurs *réguliers*. Le calcul de ces vecteurs est garanti par celui des polynômes réguliers $P_{\bar{n}_j}^{(0)}$. Afin d'obtenir les relations de récurrence permettant de calculer les polynômes $P_{\bar{n}_j}^{(0)}$, nous introduisons les polynômes *complémentaires*, appelés *intérieurs* par Freund [34] et *insuffisants* par Gutknecht [40], que nous pouvons calculer à partir de la relation suivante

$$P_{n+1}^{(0)}(\xi) = \xi P_n^{(0)}(\xi) - \zeta_n P_n^{(0)}(\xi) - \eta_n P_{n-1}^{(0)}(\xi) \tag{1.30}$$
$$\text{pour } n = \bar{n}_j, \ldots, \bar{n}_{j+1} - 2, \quad \forall j \in \{0, \ldots, J\}.$$

ζ_n et η_n sont choisis de façon arbitraire avec $\eta_{\bar{n}_j} = 0$.
Compte-tenu des conditions (1.16), le saut $\bar{m}_j$ peut être déterminé par les relations

$$\begin{aligned} c(P_n^{(0)} P_{\bar{n}_j}^{(0)}) &= 0 \qquad \text{pour } n = 0, \ldots, \bar{n}_{j+1} - 2 \\ c(P_{\bar{n}_{j+1}-1}^{(0)} P_{\bar{n}_j}^{(0)}) &\neq 0. \end{aligned} \tag{1.31}$$

Puisque $P_{\bar{n}_{j+1}}^{(0)} - \xi P_{\bar{n}_{j+1}-1}^{(0)}$ est un polynôme de degré $\bar{n}_{j+1}-1$, alors nous pouvons l'exprimer dans la base

$$\{P_{\bar{n}_0}^{(0)}, P_{\bar{n}_0+1}^{(0)}, \ldots, P_{\bar{n}_j}^{(0)}, P_{\bar{n}_j+1}^{(0)}, \ldots, P_{\bar{n}_{j+1}-2}^{(0)}, P_{\bar{n}_{j+1}-1}^{(0)}\},$$

ce qui nous permet d'écrire

$$\begin{aligned} P_{\bar{n}_{j+1}}^{(0)}(\xi) - \xi P_{\bar{n}_{j+1}-1}^{(0)}(\xi) = &- a_{\bar{n}_{j+1}-1} P_{\bar{n}_{j+1}-1}^{(0)}(\xi) - \cdots - a_{\bar{n}_j+1} P_{\bar{n}_{j+1}}^{(0)}(\xi) - a_{\bar{n}_j} P_{\bar{n}_j}^{(0)}(\xi) \\ &- a_{\bar{n}_j-1} P_{\bar{n}_j-1}^{(0)}(\xi) - \cdots - a_{\bar{n}_{j-1}+1} P_{\bar{n}_{j-1}+1}^{(0)}(\xi) - a_{\bar{n}_{j-1}} P_{\bar{n}_{j-1}}^{(0)}(\xi) \\ &\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ &- a_{\bar{n}_1-1} P_{\bar{n}_1-1}^{(0)}(\xi) - \cdots - a_{\bar{n}_0+1} P_{\bar{n}_0+1}^{(0)}(\xi) - a_{\bar{n}_0} P_{\bar{n}_0}^{(0)}(\xi). \end{aligned}$$

En utilisant les relations d'orthogonalité (1.31), nous obtenons

$$a_{\bar{n}_0} = a_{\bar{n}_0+1} = \cdots = a_{\bar{n}_{j-1}-1} = a_{\bar{n}_{j-1}+1} = \cdots = a_{\bar{n}_j-1} = 0.$$

Ainsi

$$\begin{aligned} P_{\bar{n}_{j+1}}^{(0)}(\xi) = \xi P_{\bar{n}_{j+1}-1}^{(0)}(\xi) &- a_{\bar{n}_{j+1}-1} P_{\bar{n}_{j+1}-1}^{(0)}(\xi) - \cdots - a_{\bar{n}_j+1} P_{\bar{n}_{j+1}}^{(0)}(\xi) \\ &- a_{\bar{n}_j} P_{\bar{n}_j}^{(0)}(\xi) - a_{\bar{n}_{j-1}} P_{\bar{n}_{j-1}}^{(0)}. \end{aligned} \tag{1.32}$$

Les coefficients $(a_i)_{i=\bar{n}_j}^{\bar{n}_{j+1}-1}$ et $a_{\bar{n}_{j-1}}$ peuvent être calculés en multipliant la relation (1.32) par $P_{\bar{n}_j+i}^{(0)}$ pour $i = 0, \ldots, \bar{m}_j - 1$ et $P_{\bar{n}_j-1}^{(0)}$ et en appliquant les orthogonalités (1.31). Nous obtenons le système triangulaire suivant

$$\begin{aligned} a_{\bar{n}_{j+1}-1} c(P_{\bar{n}_{j+1}-1}^{(0)} P_{\bar{n}_j}^{(0)}) &= c(\xi P_{\bar{n}_{j+1}-1}^{(0)} P_{\bar{n}_j}^{(0)}) \\ a_{\bar{n}_{j+1}-1} c(P_{\bar{n}_{j+1}-1}^{(0)} P_{\bar{n}_j+1}^{(0)}) + a_{\bar{n}_{j+1}-2} c(P_{\bar{n}_{j+1}-2}^{(0)} P_{\bar{n}_j+1}^{(0)}) &= c(\xi P_{\bar{n}_{j+1}-1}^{(0)} P_{\bar{n}_j+1}^{(0)}) \\ \cdots\cdots\cdots\cdots\cdots\cdots &\quad \cdots\cdots \\ \sum_{i=1}^{\bar{m}_j-1} a_{\bar{n}_j+i} c(P_{\bar{n}_j+i}^{(0)} P_{\bar{n}_{j+1}-1}^{(0)}) + a_{\bar{n}_j} c(P_{\bar{n}_j}^{(0)} P_{\bar{n}_{j+1}-1}^{(0)}) &= c(\xi P_{\bar{n}_{j+1}-1}^{(0)} P_{\bar{n}_{j+1}-1}^{(0)}), \end{aligned} \tag{1.33}$$

ainsi que

$$a_{\bar{n}_{j-1}} c(P_{\bar{n}_{j-1}}^{(0)} P_{\bar{n}_j-1}^{(0)}) = c(\xi P_{\bar{n}_{j+1}-1}^{(0)} P_{\bar{n}_j-1}^{(0)}). \tag{1.34}$$

À partir de tous ces résultats, nous pouvons maintenant mettre en œuvre l'algorithme complet du processus de Lanczos avec *look-ahead*. Les vecteurs réguliers $v_{\bar{n}_j}$ et $w_{\bar{n}_j}$ sont complétés par les vecteurs

$$v_{\bar{n}_j+i} = P_{\bar{n}_j+i}^{(0)}(A) v_0 \qquad \text{et} \qquad w_{\bar{n}_j+i} = P_{\bar{n}_j+i}^{(0)}(A^T) w_0 \qquad \text{pour tout } i = 1, \ldots, \bar{m}_j - 1.$$

D'après (1.30), ces vecteurs peuvent être calculés par la relation suivante

$$\begin{aligned} v_{n+1} &= Av_n - \zeta_n v_n - \eta_n v_{n-1}, \\ w_{n+1} &= A^T w_n - \zeta_n w_n - \eta_n w_{n-1}, \\ \text{pour } n &= \bar{n}_j, \ldots, \bar{n}_{j+1} - 2, \quad \forall j \in \{0, \ldots, J\}, \end{aligned} \tag{1.35}$$

avec ζ_n, η_n choisis de façon arbitraire et $\eta_{\bar{n}_j} = 0$.
En utilisant (1.25), les conditions d'orthogonalité (1.31) s'écrivent

$$\begin{aligned} (w_n, v_{\bar{n}_j}) &= 0 \qquad \text{pour } n = 0, \ldots, \bar{n}_{j+1} - 2, \\ (w_{\bar{n}_{j+1}-1} v_{\bar{n}_j}) &\neq 0. \end{aligned} \tag{1.36}$$

Nous notons, pour tout $j = 0, \ldots, J$,

$$V_j = [v_{\bar{n}_j} \;\; v_{\bar{n}_j+1} \;\; \ldots \;\; v_{\bar{n}_{j+1}-1}], \qquad W_j = [w_{\bar{n}_j} \;\; w_{\bar{n}_j+1} \;\; \ldots \;\; w_{\bar{n}_{j+1}-1}].$$

A partir des conditions (1.36) et en utilisant la relation de récurrence (1.35), nous pouvons montrer que

$$(w_{\bar{n}_{j+1}-1}, v_{\bar{n}_j}) = (w_{\bar{n}_{j+1}-2}, v_{\bar{n}_j+1}) = \cdots = (w_{\bar{n}_j}, v_{\bar{n}_{j+1}-1}) = d_j \neq 0,$$

ce qui prouve que la matrice $W_j^T V_j = D_j$ de dimension $\bar{m}_j$ est inversible et qu'elle est de la forme

$$D_j = \begin{bmatrix} 0 & \cdots & 0 & d_j \\ \vdots & \iddots & \iddots & * \\ 0 & \iddots & \iddots & \vdots \\ d_j & * & \cdots & * \end{bmatrix}.$$

Le système d'équations (1.33) et l'équation (1.34) deviennent

$$D_j \alpha_j = W_j^T A v_{\bar{n}_{j+1}-1} \qquad \text{avec} \quad \alpha_j = \begin{bmatrix} a_{\bar{n}_j} \\ a_{\bar{n}_j+2} \\ \vdots \\ a_{\bar{n}_{j+1}-1} \end{bmatrix}$$

et

$$\beta_j = a_{\bar{n}_{j-1}} = \frac{(w_{\bar{n}_{j+1}-1}, A v_{\bar{n}_j-1})}{(w_{\bar{n}_j-1}, v_{\bar{n}_{j-1}})} = \frac{(w_{\bar{n}_{j+1}-1}, v_{\bar{n}_j})}{(w_{\bar{n}_j-1}, v_{\bar{n}_{j-1}})} = \frac{d_j}{d_{j-1}}.$$

Enfin, pour calculer récursivement les vecteurs réguliers $v_{\bar{n}_j}$ et $w_{\bar{n}_j}$, nous utilisons la récurrence (1.32), ce qui donne

$$\begin{aligned} v_{\bar{n}_{j+1}} &= A v_{\bar{n}_{j+1}-1} - V_j \alpha_j - \beta_j v_{\bar{n}_{j-1}}, \\ w_{\bar{n}_{j+1}} &= A^T w_{\bar{n}_{j+1}-1} - W_j \alpha_j - \beta_j w_{\bar{n}_{j-1}}. \end{aligned} \tag{1.37}$$

Les relations ainsi établies définissent le processus de Lanczos avec *look-ahead*

Algorithme 1.3.1 : *Le processus de Lanczos avec look-ahead*

0) Initialisation :
Choisir $v_0,\ w_0 \in \mathbb{R}^N$ avec $v_0,\ w_0 \neq 0$;
$V_0 = v_0,\ W_0 = w_0,\ D_0 = W_0^T V_0$;
$n_0 = 0,\ l = 0,\ m_l = 0,\ v_{-1} = w_{-1} = 0$ et $\beta_0 = 0$;

Pour $n = 0, 1, \ldots, N$

1) Si $det(D_l) \neq 0$ aller à 2) sinon à 3) ;

2) (vecteur régulier) calculer

$$\begin{aligned} Si\ \ l \neq 0\ \ \text{alors} \quad \beta_l &= \frac{(D_l)_{m_l,1}}{(D_{l-1})_{m_{l-1},1}} \quad Fin(si); \\ \alpha_l &= D_l^{(-1)} (W_l)^T A v_n; \\ v_{n+1} &= A v_n - V_l \alpha_l - \beta_l v_{n_{l-1}}; \\ w_{n+1} &= A^T w_n - W_l \alpha_l - \beta_l w_{n_{l-1}}; \\ \text{prendre} \quad n_{l+1} &= n+1,\ l = l+1,\ m_l = 0,\ V_l = W_l = \varnothing \quad \text{et partir à 4)}; \end{aligned}$$

3) (vecteur complémentaire) calculer

$$\begin{aligned} v_{n+1} &= A v_n - \zeta_n v_n - \eta_n v_{n-1}; \\ w_{n+1} &= A^T w_n - \zeta_n w_n - \eta_n w_{n-1}; \end{aligned}$$

Prendre $m_l = m_l + 1$;

4) Si $v_{n+1} = 0$ ou $w_{n+1} = 0$, alors on arrête ;
sinon, on prend

$$V_l = [V_l \ \ v_{n+1}], \quad W_l = [W_l \ \ w_{n+1}], \quad D_l = W_l^T V_l;$$

Fin.

Les coefficients ζ_n et η_n sont choisis de façon arbitraire, en pratique nous prenons $\zeta_n = \eta_n = 1$. Le processus de Lanczos avec *look-ahead* préserve la forme matricielle (1.23), que l'on réécrit

$$\begin{aligned} A V^{(n)} &= V^{(n+1)} H_e^{(n)} \\ A^T W^{(n)} &= W^{(n+1)} H_e^{(n)}. \end{aligned} \tag{1.38}$$

$V^{(n)}$ et $W^{(n)}$ sont deux matrices de dimension $N \times (n+1)$ qui regroupent les vecteurs *réguliers* et *complémentaires* comme suit

$$V^{(n)} = [V_0\ V_1\ \cdots\ V_l] \qquad \text{et} \qquad W^{(n)} = [W_0\ W_1\ \cdots\ W_l], \tag{1.39}$$

tels que, pour tout $k = 0, \ldots, l$, nous ayons

$$V_k = \begin{cases} [v_{n_k}\ v_{n_k+1}\ \cdots\ v_{n_{k+1}-1}], & \text{si} \quad 0 \leq k < l, \\ [v_{n_l}\ v_{n_l+1}\ \cdots\ v_n], & \text{si} \quad k = l, \end{cases}$$

et

$$W_k = \begin{cases} [w_{n_k}\ w_{n_k+1}\ \cdots\ w_{n_{k+1}-1}], & \text{si} \quad 0 \leq k < l, \\ [w_{n_l}\ w_{n_l+1}\ \cdots\ w_n], & \text{si} \quad k = l, \end{cases}$$

avec $l = l(n)$ l'unique entier qui dénote l'indice du dernier vecteur *régulier* précédant v_n, c'est-à-dire $n_l \leq n < n_{l+1}$.
La matrice $H^{(n)}$ obtenue ici est une matrice tridiagonale par bloc de dimension $(n+1) \times (n+1)$. Cette matrice est de la forme suivante

$$H^{(n)} = \begin{bmatrix} \lambda_0 & \theta_1 & & \\ \gamma_1 & \ddots & \ddots & \\ & \ddots & \ddots & \theta_l \\ & & \gamma_l & \lambda_l \end{bmatrix},$$

où λ_j est une matrice de Hessenberg de dimension $\bar{m}_j \times \bar{m}_j$, θ_j et γ_j sont des matrices de rang 1 et respectivement de dimension $\bar{m}_{j-1} \times \bar{m}_j$ et $\bar{m}_j \times \bar{m}_{j-1}$

$$\lambda_j = \begin{bmatrix} \zeta_{n_j} & \eta_{n_j+1} & 0 & \cdots & 0 & * \\ 1 & \zeta_{n_j+1} & \eta_{n_j+2} & \ddots & \vdots & \vdots \\ & 1 & \zeta_{n_j+2} & \ddots & 0 & \vdots \\ & & \ddots & \ddots & \eta_{n_{j+1}-2} & * \\ & & & \ddots & \zeta_{n_{j+1}-2} & * \\ & & & & 1 & * \end{bmatrix},$$

$$\theta_j = \begin{bmatrix} 0 & \cdots & 0 & \beta_j \\ \vdots & & & 0 \\ \vdots & & & \vdots \\ 0 & \cdots & \cdots & 0 \end{bmatrix}, \qquad \gamma_j = \begin{bmatrix} 0 & \cdots & 0 & 1 \\ \vdots & & & 0 \\ \vdots & & & \vdots \\ 0 & \cdots & \cdots & 0 \end{bmatrix}.$$

Enfin, quand $n = n_{j+1} - 1$, la matrice de bi-orthogonalité $D^{(n)} = {W^{(n)}}^T V^{(n)}$ est diagonale par blocs

$$D^{(n)} = \begin{bmatrix} D_0 & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & D_l \end{bmatrix}.$$

En arithmétique exacte, nous obtenons $v_L = 0$ ou $w_L = 0$ pour $L \leq N$, ce qui n'est pas toujours vrai en arithmétique finie, puisque nous risquons d'avoir, à cause de l'accumulation des erreurs d'arrondi, $L > N$. C'est-à-dire que le dernier vecteur régulier d'indice $\bar{n}_J < N$ est différent de zéro ($v_{\bar{n}_J} \neq 0$) et $\bar{n}_J + \bar{m}_J > N$. Dans ce cas nous avons ce que l'on appelle l'*incurable-breakdown*. La seule solution face à ce problème est de recommencer le processus avec un nouveau choix des vecteurs initiaux v_0 et w_0.

1.3.4 Le processus de Lanczos par des récurrences à deux termes

Nous donnons ici une autre approche pour construire les vecteurs de Lanczos; elle consiste à rajouter aux vecteurs de Lanczos deux nouvelles bases des deux sous-espaces de Krylov $\mathcal{K}_n(A, v_0)$ et $\mathcal{K}_n(A^T, w_0)$. L'idée est de remplacer les récurrences à trois termes (1.28) par un couple de récurrences à deux termes, et enfin mettre en œuvre le nouveau processus de Lanczos à partir des nouvelles relations de récurrence.
En effet, soient les deux suites de vecteurs $\{v_j\}_{j=0}^n$ et $\{w_j\}_{j=0}^n$ données dans la sous-section 1.3.1. Nous considérons les deux ensembles de vecteurs

$$(p_0, \ldots, p_n) \qquad \text{et} \qquad (q_0, \ldots, q_n).$$

Ces deux familles de vecteurs sont construites par

$$p_n = P_n^{(1)}(A)v_0 \qquad \text{et} \qquad q_n = P_n^{(1)}(A^T)w_0. \tag{1.40}$$

$P_n^{(1)}$ est le polynôme orthogonal formel par rapport à la fonctionnelle $c^{(1)}$ définie par $c^{(1)}(\xi^i) = c(\xi^{i+1})$, avec c est donnée par (1.25).
Les relations (1.11), données dans la sous-section 1.1.2, constituent des récurrences à deux termes permettant de construire les deux familles de polynômes $P_n^{(0)}$ et $P_n^{(1)}$. Nous avons donc les relations

$$\begin{aligned} P_n^{(1)}(\xi) &= P_n^{(0)}(\xi) - \gamma_n P_{n-1}^{(1)}(\xi), \\ P_{n+1}^{(0)}(\xi) &= \xi P_n^{(1)}(\xi) - \lambda_{n+1} P_n^{(0)}(\xi). \end{aligned} \tag{1.41}$$

En multipliant ces deux relations respectivement par $P_{n-1}^{(1)}$ et $P_n^{(0)}$ et en appliquant les fonctionnelles c et $c^{(1)}$ nous obtenons, grâce aux conditions d'orthogonalité (1.2) et (1.10), les relations suivantes

$$\gamma_n = \frac{c(\xi P_{n-1}^{(1)} P_n^{(0)})}{c^{(1)}(P_{n-1}^{(1)} P_{n-1}^{(1)})}, \qquad \text{et} \qquad \lambda_{n+1} = \frac{c(\xi P_n^{(0)} P_n^{(1)})}{c(P_n^{(0)} P_n^{(0)})}.$$

Or, nous remarquons que

$$c(\xi P_{n-1}^{(1)} P_n^{(0)}) = c(P_n^{(0)} P_n^{(0)}) + \lambda_n c(P_{n-1}^{(0)} P_n^{(0)}) = c(P_n^{(0)} P_n^{(0)})$$

et

$$c(\xi P_n^{(0)} P_n^{(1)}) = c^{(1)}(P_n^{(1)} P_n^{(1)}) + \gamma_n c^{(1)}(P_{n-1}^{(1)} P_n^{(1)}) = c^{(1)}(P_n^{(1)} P_n^{(1)}).$$

Ainsi, nous obtenons

$$\gamma_n = \frac{c(P_n^{(0)} P_n^{(0)})}{c^{(1)}(P_{n-1}^{(1)} P_{n-1}^{(1)})} \qquad \text{et} \qquad \lambda_{n+1} = \frac{c(\xi P_n^{(1)} P_n^{(1)})}{c(P_n^{(0)} P_n^{(0)})}. \tag{1.42}$$

Ces relations peuvent être utilisées pour calculer les vecteurs p_n, q_n, v_n et w_n. Par conséquent, nous obtenons

$$\begin{aligned} p_n &= v_n - \gamma_n p_{n-1} \\ q_n &= w_n - \gamma_n q_{n-1} \end{aligned} \tag{1.43}$$

et

$$\begin{aligned} v_{n+1} &= A p_n - \lambda_{n+1} v_n \\ w_{n+1} &= A^T q_n - \lambda_{n+1} w_n \end{aligned} \tag{1.44}$$

avec les coefficients

$$\lambda_{n+1} = \frac{(q_n, A p_n)}{(w_n, v_n)} \qquad \text{et} \qquad \gamma_n = \frac{(w_n, v_n)}{(q_{n-1}, A p_{n-1})}. \tag{1.45}$$

Si nous considérons les matrices suivantes

$$P^{(n)} := [p_0 \quad p_1 \quad \dots \quad p_n] \qquad\qquad Q^{(n)} := [q_0 \quad q_1 \quad \dots \quad q_n],$$

alors les récurrences (1.43) et (1.44) qui construisent les vecteurs p_{n+1}, q_{n+1}, v_{n+1} et w_{n+1} peuvent s'écrire sous la forme matricielle

$$\begin{aligned} V^{(n)} &= P^{(n)} U_n, & A P^{(n)} &= V^{(n+1)} L_n, \\ W^{(n)} &= Q^{(n)} U_n, & A^T Q^{(n)} &= W^{(n+1)} L_n, \end{aligned} \tag{1.46}$$

avec U_n une matrice carrée bi-diagonale supérieure de dimension $(n+1)$ et L_n matrice bi-diagonale inférieure de dimension $(n+2) \times (n+1)$. Ces deux matrices ont la forme suivante

$$U_n := \begin{bmatrix} 1 & \gamma_1 & 0 & \cdots & 0 \\ 0 & 1 & \gamma_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ \vdots & & \ddots & \ddots & \gamma_n \\ 0 & \cdots & \cdots & 0 & 1 \end{bmatrix} \quad \text{et} \quad L_n := \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 1 & \lambda_2 & \ddots & \vdots \\ 0 & 1 & \ddots & 0 \\ \vdots & \ddots & \ddots & \lambda_{n+1} \\ 0 & \cdots & 0 & 1 \end{bmatrix}. \tag{1.47}$$

Nous remarquons qu'en éliminant la matrice $P^{(n)}$ dans (1.46), nous avons

$$AV^{(n)} = V^{(n+1)} L_n U_n. \tag{1.48}$$

En comparant ce résultat avec (1.23), nous obtenons

$$H_e^{(n)} = L_n U_n. \tag{1.49}$$

Donc, les deux matrices (1.47) définissent une factorisation de la matrice de Hessenberg tridiagonale $H_e^{(n)}$ générée par le processus de tridiagonalisation de Lanczos.
Nous pouvons récupérer la matrice tridiagonale de Lanczos $H^{(n)}$ à partir du produit $\bar{L}_n U_n$ avec $\bar{L}_n$ est la matrice bidiagonale de dimension $(n+1) \times (n+1)$ qui vérifie

$$L_n = \begin{bmatrix} & \bar{L}_n & & \\ 0 & \cdots & 0 & 1 \end{bmatrix}.$$

Les vecteurs $\{v_i\}_{i=0}^n$ et $\{w_i\}_{i=0}^n$ sont bi-orthogonaux, ils correspondent aux vecteurs calculés dans l'algorithme de tridiagonalisation de Lanczos, alors que les vecteurs $\{p_i\}_{i=0}^n$ et $\{q_i\}_{i=0}^n$ vérifient, à partir des conditions (1.10), les relations

$$(q_j, Ap_i) = \begin{cases} \epsilon_i \neq 0 & i = j \\ 0 & i \neq j \end{cases} \qquad \text{pour} \quad i, j = 0, 1, \ldots, n. \tag{1.50}$$

D'où

$$E^{(n)} = Q^{(n)} A P^{(n)} = \begin{bmatrix} \epsilon_1 & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \epsilon_n \end{bmatrix},$$

On dit que les vecteurs $\{p_i\}_{i=1}^n$ et $\{q_i\}_{i=1}^n$ sont A bi-orthogonaux.
A partir de toutes ces relations, nous obtenons l'algorithme suivant

Algorithme 1.3.2 *:Le processus de Lanczos par des relations de récurrence à deux termes*

1. Initialisation : Choisir $v_0 \in \mathbb{R}^N$ et $w_0 \in \mathbb{R}^N$ tels que $\delta_0 = (w_0, v_0) \neq 0$.

2. Pour $n = 0, 1, \ldots, N$:
   Si $n \neq 0$
   Calculer $\delta_n = (w_n, v_n)$, $\gamma_n = \dfrac{\delta_n}{\epsilon_{n-1}}$,
   $p_n = v_n - \gamma_n p_{n-1}$,
   $q_n = w_n - \gamma_n q_{n-1}$,
   Sinon $p_0 = v_0$,
   $q_0 = w_0$,
   Fin (si).

   Calculer $\epsilon_n = (q_n, Ap_n)$, $\lambda_{n+1} = \dfrac{\epsilon_n}{\delta_n}$,
   $v_{n+1} = Ap_n - \lambda_{n+1} v_n$,
   $w_{n+1} = A^T q_n - \lambda_{n+1} w_n$.
   Fin.

Nous remarquons que cet algorithme peut être considéré comme une version de l'algorithme du Gradient Biconjugé (BICG) [29] dont les vecteurs de direction et les résidus peuvent être obtenus à partir des suites $P^{(n)}$, $Q^{(n)}$, $V^{(n)}$ et $W^{(n)}$.
L'algorithme 1.3.2 s'arrête d'une façon régulière dès que

$$\|v_{n+1}\| = 0 \qquad \text{ou bien} \qquad \|w_{n+1}\| = 0.$$

En arithmétique finie, ce résultat est atteint avant la $N^{ème}$ itération, ce qui permet de construire les bases des sous-espaces invariants $\mathcal{K}_n(A, v_0)$ ou bien $\mathcal{K}_n(A^T, w_0)$, respectivement. Par contre, si $\delta_n = 0$ ou $\epsilon_n = 0$ alors l'algorithme s'arrête d'une façon prématurée à cause d'une division par zéro. Dans le cas où $\delta_n = 0$ nous avons un *ghost-breakdown*, c'est-à-dire le même type de breakdown que nous risquons de rencontrer dans le cas du processus classique de Lanczos. Nous notons que, comme dans l'algorithme du BICG, le processus de Lanczos par des récurrences à deux termes subit un deuxième type de *breakdown* dans le cas où $\epsilon_n = 0$; il s'agit du *true-breakdown*. Ce type de *breakdown* se traduit dans le cas du BICG par le fait que les $(n+1)^{\text{ème}}$ vecteurs d'itération ne sont pas définis par les conditions de Galerkin. Dans l'algorithme 1.3.2, la condition $\epsilon_n = 0$ est liée à la décompo-

sition (1.50) de la matrice tridiagonale $H_e^{(n)}$ de Lanczos. Ce type de breakdown est appelé aussi *pivot-breakdown* [21].

1.3.5 Le processus de Lanczos par des récurrences à deux termes avec look-ahead

Dans les relations de récurrence (1.41) nous risquons d'avoir le *true* et le *ghost-breakdown* à différentes itérations. Le Théorème (1.2.3) nous permet de prévoir les sauts à effectuer entre les polynômes réguliers $P_k^{(1)}$ et $P_k^{(0)}$. Nous avons trois situations possibles. Nous allons appliquer le *look-ahead*, dans chacune de ces situations, afin de donner les relations qui nous permettent de calculer les polynômes existant $P_k^{(1)}$ et $P_k^{(0)}$.

Pour cela, nous gardons les mêmes notations que celles données dans le paragraphe 1.2.2. Nous notons $\{n_0, \ldots, n_I\}$ les indices des polynômes réguliers $P_{n_k}^{(1)}$, tels que

$$n_0 = 0 < n_1 < \cdots < n_I \leq N$$

avec $m_k = n_{k+1} - n_k$ le saut entre deux polynômes réguliers $P_{n_k}^{(1)}$ et $P_{n_{k+1}}^{(1)}$,
et $\{\bar{n}_0, \ldots, \bar{n}_J\}$ les indices des polynômes réguliers $P_{\bar{n}_l}^{(0)}$, tels que

$$\bar{n}_0 = 0 < \bar{n}_1 < \cdots < \bar{n}_J \leq N$$

$\bar{m}_l = \bar{n}_{l+1} - \bar{n}_l$ représente le saut entre les deux polynômes réguliers $P_{\bar{n}_l}^{(0)}$ et $P_{\bar{n}_{l+1}}^{(0)}$.
J (resp I) représente le plus grand indice tel que $\|v_{\bar{n}_J}\| \neq 0$ et $\|w_{\bar{n}_J}\| \neq 0$ resp ($\|p_{n_I}\| \neq 0$ et $\|q_{n_I}\| \neq 0$).
Nous supposons qu'à l'itération n_k, nous avons les deux polynômes réguliers $P_{n_k}^{(1)}$ et $P_{n_k}^{(0)}$. Il existe alors un indice $l \in I\!N$ tel que $n_k = \bar{n}_l$ avec $H_{n_k}^{(0)} \neq 0$ et $H_{n_k}^{(1)} \neq 0$.

- *Premier cas* : $\bar{m}_l \neq 1$ et $m_{k+1} \neq 1$. Ce cas est présenté dans la figure 1.2.

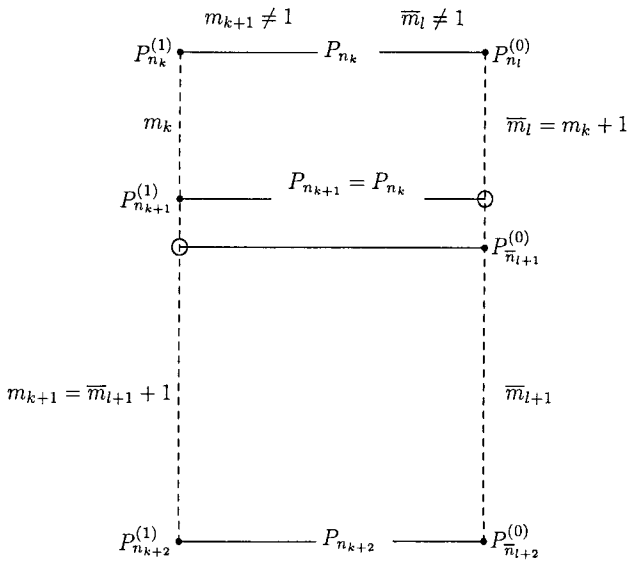
D'après le Théorème 1.2.3, m_k est égal à $\bar{m}_l - 1$. Nous avons donc

$$H_i^{(0)} = 0 \qquad \text{pour} \qquad i = n_k + 1, \ldots, n_{k+1},$$

c'est-à-dire que pour $i = n_k + 1, \ldots, n_{k+1}$ les polynômes $P_i^{(0)}$ n'existent pas.
Pour obtenir la récurrence qui nous permet de calculer le polynôme $P_{n_{k+1}}^{(1)}$, nous introduisons les polynômes *complémentaires* définis par

$$\begin{aligned} P_{n_k+i}^{(0)} &= \xi P_{n_k+i-1}^{(1)} - \mu_i^{(k)} P_{n_k+i-1}^{(0)} \qquad \text{pour} \quad 1 \leq i \leq m_k, \\ P_{n_k+i}^{(1)} &= P_{n_k+i}^{(0)} - \nu_i^{(k)} P_{n_k+i-1}^{(1)} \qquad \text{pour} \quad 1 \leq i \leq m_k - 1. \end{aligned} \tag{1.51}$$

FIG. 1.2 - Premier cas : $\overline{m}_l \neq 1$ et $m_{k+1} \neq 1$



Les polynômes *complémentaires* servent à remplacer les polynômes orthogonaux inexistants et ainsi compléter l'ensemble des polynômes réguliers. Il sont calculés par les relations (1.51) qui sont conçues de la même façon que les relations (1.41) avec des coefficients $\mu_i^{(k)}$ et $\nu_i^{(k)}$ arbitraires. En pratique, nous prenons $\mu_i^{(k)} = \nu_i^{(k)} = 1$.
Ces polynômes ne vérifient pas les conditions d'orthognalité (1.15) et (1.16), néanmoins ils vérifient, d'après (1.15) et (1.53), les orthogonalités suivantes

$$\begin{aligned}
&\text{pour tout} \quad j = 1, \ldots, m_k - 1 \\
&\qquad c^{(1)}(\xi^i P^{(1)}_{n_k+j}) = 0 \quad \text{pour} \quad i = 0, \ldots, n_k + m_k - 2 - j, \\
&\qquad\qquad \neq 0 \quad \text{pour} \quad i = n_k + m_k - 1 - j, \\
&\text{pour tout} \quad j = 1, \ldots, m_k \\
&\qquad c(\xi^i P^{(0)}_{n_k+j}) = 0 \quad \text{pour} \quad i = 0, \ldots, n_k + m_k - 1 - j \\
&\qquad\qquad \neq 0 \quad \text{pour} \quad i = n_k + m_k - j.
\end{aligned} \tag{1.52}$$

D'autre part, les conditions d'orthogonalité (1.16) peuvent s'écrire

$$\begin{aligned}
c(\xi^i P^{(0)}_{n_k}) &= 0 \quad \text{pour} \quad i = 0, \ldots, n_k + m_k - 1, \\
&\neq 0 \quad \text{pour} \quad i = n_k + m_k.
\end{aligned} \tag{1.53}$$

Il est évident que la famille

$$\{P^{(1)}_{n_0}, \xi P^{(1)}_{n_0}, \ldots, \xi^{m_0-1} P^{(1)}_{n_0}, P^{(1)}_{n_1}, \xi P^{(1)}_{n_1} \ldots, P^{(1)}_{n_{k-1}}, \ldots, \xi^{m_{k-1}-1} P^{(1)}_{n_{k-1}}, P^{(1)}_{n_k}, P^{(1)}_{n_k+1}, \ldots, P^{(1)}_{n_k+m_k-1}\}$$

forme une base de l'espace vectoriel des polynômes de degré au plus égal à $n_{k+1}-1$. Alors, nous pouvons exprimer le polynôme $P^{(1)}_{n_{k+1}} - P^{(0)}_{n_{k+1}}$ (de degré $n_{k+1}-1$) comme suit

$$\begin{aligned} P^{(1)}_{n_{k+1}} - P^{(0)}_{n_{k+1}} \;=\; & - \alpha^{(0)}_0 P^{(1)}_{n_0} - \alpha^{(0)}_1 \xi P^{(1)}_{n_0} - \cdots - \alpha^{(0)}_{m_0-1}\xi^{m_0-1} P^{(1)}_{n_0} \\ & - \alpha^{(1)}_0 P^{(1)}_{n_1} - \alpha^{(1)}_1 \xi P^{(1)}_{n_1} - \cdots - \alpha^{(1)}_{m_1-1}\xi^{m_1-1} P^{(1)}_{n_1} \\ & - \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ & - \alpha^{(k-1)}_0 P^{(1)}_{n_{k-1}} - \alpha^{(k-1)}_1 \xi P^{(1)}_{n_{k-1}} - \cdots - \alpha^{(k-1)}_{m_{k-1}-1}\xi^{m_{k-1}-1} P^{(1)}_{n_{k-1}} \\ & - \alpha^{(k)}_0 P^{(1)}_{n_k} - \alpha^{(k)}_1 P^{(1)}_{n_k+1} - \cdots - \alpha^{(k)}_{m_k-1} P^{(1)}_{n_{k+1}-1}. \end{aligned}$$

En utilisant les orthogonalités (1.15) et (1.52), nous obtenons

$$\alpha^{(j)}_i = 0 \quad \text{pour} \quad i = 0,\ldots,m_j-1 \quad ; \quad \alpha^{(k-1)}_s = 0 \quad \text{pour} \quad s = 1,\ldots,m_{k-1}-1$$
$$\text{et} \quad j = 0,\ldots,k-2$$

ce qui nous permet d'écrire

$$P^{(1)}_{n_{k+1}} = P^{(0)}_{n_{k+1}} - \alpha^{(k)}_{m_k-1} P^{(1)}_{n_{k+1}-1} - \cdots - \alpha^{(k)}_1 P^{(1)}_{n_k+1} - \alpha^{(k)}_0 P^{(1)}_{n_k} - \alpha^{(k-1)}_0 P^{(1)}_{n_{k-1}}. \qquad (1.54)$$

Pour le calcul des coefficients figurants dans cette relation, nous multiplions (1.54) par un polynôme arbitraire U_i de degré i. En lui appliquant la fonctionnelle $c^{(1)}$, nous obtenons

$$\begin{aligned} c^{(1)}(U_i P^{(1)}_{n_{k+1}}) = \; & c(\xi U_i P^{(0)}_{n_{k+1}}) - \alpha^{(k)}_{m_k-1} c^{(1)}(U_i P^{(1)}_{n_{k+1}-1}) - \cdots - \alpha^{(k)}_1 c^{(1)}(U_i P^{(1)}_{n_k+1}) \\ & - \alpha^{(k)}_0 c^{(1)}(U_i P^{(1)}_{n_k}) - \alpha^{(k-1)}_0 c^{(1)}(U_i P^{(1)}_{n_{k-1}}). \end{aligned}$$

Le choix de U_i intervient dans l'expression des coefficients calculés. Cela implique que ce choix peut avoir une grande influence sur l'algorithme obtenu. Ici, nous allons prendre $U_i = P^{(1)}_i$, ce qui va nous permettre par la suite de simplifier les calculs.
Nous commençons par $i = n_k - 1$ avec $U_{n_k-1} = P^{(1)}_{n_k-1}$. Cela nous donne, grâce aux conditions (1.15) et (1.52), l'équation suivante

$$c(\xi P^{(1)}_{n_k-1} P^{(0)}_{n_{k+1}}) = \alpha^{(k-1)}_0 c^{(1)}(P^{(1)}_{n_k-1} P^{(1)}_{n_{k-1}}).$$

Dans le cas où $m_{k-1} = 1$ nous avons $P^{(1)}_{n_k-1} = P^{(1)}_{n_{k-1}}$.
Par ailleurs, le polynôme $P^{(1)}_{n_k-1}$ correspond, dans le cas où $m_{k-1} \neq 1$, au polynôme *complémentaire* construit de façon similaire à celle de la relation (1.51) (plus précisément, la relation de calcul de $P^{(1)}_{n_k-1}$ sera définie par la suite dans la construction de $P^{(1)}_{n_{k+2}}$).
Puisque les polynômes $P^{(0)}_{n_k}$ et $\xi P^{(1)}_{n_k-1}$ sont unitaires de même degré, alors nous avons la relation

$$P^{(0)}_{n_k} = \xi P^{(1)}_{n_k-1} + \tilde{q}_{n_k-1},$$

avec $\tilde{q}_{n_k-1}$ un polynôme de degré au plus égal à $n_k - 1$.
Les relations (1.52) nous permettent d'avoir

$$c(P_{n_k}^{(0)} P_{n_{k+1}}^{(0)}) = c(\xi P_{n_k-1}^{(1)} P_{n_{k+1}}^{(0)}).$$

Ainsi

$$\alpha_0^{(k-1)} = \frac{c(P_{n_k}^{(0)} P_{n_{k+1}}^{(0)})}{c^{(1)}(P_{n_k-1}^{(1)} P_{n_{k-1}}^{(1)})}. \tag{1.55}$$

Nous remarquons, en appliquant les relations (1.15) à $P_{n_{k-1}}^{(1)}$, que $c^{(1)}(P_{n_k-1}^{(1)} P_{n_{k-1}}^{(1)}) \neq 0$.
Pour les autres coefficients, nous prenons pour $i = n_k, \ldots, n_{k+1} - 1$, $U_i = P_i^{(1)}$ (polynôme *régulier* ou *complémentaire*) et nous imposons les orthogonalités (1.52) pour obtenir le système triangulaire suivant

$$\begin{aligned}
c^{(1)}(P_{n_k}^{(1)} P_{n_{k+1}}^{(0)}) &= \alpha_{m_k-1}^{(k)} c^{(1)}(P_{n_k}^{(1)} P_{n_{k+1}-1}^{(1)}) \\
c^{(1)}(P_{n_k+1}^{(1)} P_{n_{k+1}}^{(0)}) &= \alpha_{m_k-2}^{(k)} c^{(1)}(P_{n_k+1}^{(1)} P_{n_{k+1}-2}^{(1)}) + \alpha_{m_k-1}^{(k)} c^{(1)}(P_{n_k+1}^{(1)} P_{n_{k+1}-1}^{(1)}) \\
\cdots\cdots\cdots\cdots &\quad \cdots \quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\
c^{(1)}(P_{n_{k+1}-1}^{(1)} P_{n_{k+1}}^{(0)}) &= \alpha_0^{(k)} c^{(1)}(P_{n_{k+1}-1}^{(1)} P_{n_k}^{(1)}) + \alpha_1^{(k)} c^{(1)}(P_{n_{k+1}-1}^{(1)} P_{n_k+1}^{(1)}) + \cdots \\
&\quad + \alpha_{m_k-2}^{(k)} c^{(1)}(P_{n_{k+1}-1}^{(1)} P_{n_{k+1}-2}^{(1)}) + \alpha_{m_k-1}^{(k)} c^{(1)}(P_{n_{k+1}-1}^{(1)} P_{n_{k+1}-1}^{(1)}).
\end{aligned} \tag{1.56}$$

D'après (1.51) et (1.52), nous avons

$$c^{(1)}(P_{n_k+i}^{(1)} P_{n_k+j}^{(1)}) = c^{(1)}(P_{n_k+f}^{(1)} P_{n_k+h}^{(1)}) \neq 0 \tag{1.57}$$

$\forall\ i,\ j,\ f$ et h dans $\mathbb{N}$ vérifiant $i + j = f + h = m_k - 1$.
Le système (1.56) étant triangulaire supérieur et à diagonale non nulle, il est donc non-singulier.
D'autre part, comme $m_k = \tilde{m}_l - 1$ ce qui implique que $n_{k+1} = \tilde{n}_{l+1} - 1$, alors le polynôme régulier $P_{\tilde{n}_{l+1}}^{(0)}$ existe à l'itération $n_{k+1} + 1$.
Et puisque $m_{k+1} \neq 1$, alors $H_{n_{k+1}+1}^{(1)} = 0$ et $H_{n_{k+1}+1}^{(0)} = H_{\tilde{n}_{l+1}}^{(0)} \neq 0$ ce qui nous donne, d'après Le Lemme 1.2.1, la relation

$$P_{\tilde{n}_{l+1}}^{(0)}(\xi) = P_{n_{k+1}+1}^{(0)}(\xi) = \xi P_{n_{k+1}}^{(1)}(\xi). \tag{1.58}$$

Nous remarquons qu'à l'itération n_{k+1} nous avons $H_{n_{k+1}}^{(1)} \neq 0$ et $H_{n_{k+1}}^{(0)} = 0$, donc le polynôme $P_{n_{k+1}}$ existe mais n'est pas de degré égal à n_{k+1}. D'après le Lemme (1.2.3) le polynôme $P_{n_{k+1}}$ est égal à P_{n_k}.
Les polynômes réguliers $P_{n_{k+1}}^{(1)}$ et $P_{\tilde{n}_{l+1}}^{(0)}$ sont donc calculés par les relations (1.54) et (1.58) sans problème de *breakdown*. Cependant, ces deux polynômes existent à deux itérations différentes. Donc les relations de récurrence que nous venons de donner ne sont pas suffisantes.

Par ailleurs, compte-tenu du résultat du Théorème 1.2.3, nous avons $m_{k+1} = \bar{m}_{l+1} + 1$, c'est-à-dire que $H^{(1)}_{n_{k+2}} \neq 0$ et $H^{(0)}_{n_{k+2}} \neq 0$ avec $n_{k+2} = \bar{n}_{l+2}$, ainsi les deux polynômes réguliers $P^{(1)}_{n_{k+2}}$ et $P^{(0)}_{\bar{n}_{l+2}}$ sont du même degré. Il est donc nécessaire d'établir les relations permettant de calculer ces deux polynômes.

Construisons d'abord les polynômes *complémentaires* suivants

$$\begin{aligned} P^{(0)}_{n_{k+1}+i} &= \xi P^{(1)}_{n_{k+1}+i-1} - \mu^{(k+1)}_{i-1} P^{(0)}_{n_{k+1}+i-1} \qquad \text{pour} \quad 2 \leq i \leq m_{k+1} - 1 \\ P^{(1)}_{n_{k+1}+i} &= P^{(0)}_{n_{k+1}+i} - \nu^{(k+1)}_{i} P^{(1)}_{n_{k+1}+i-1} \qquad \text{pour} \quad 1 \leq i \leq m_{k+1} - 1 \end{aligned} \tag{1.59}$$

avec $\mu^{(k+1)}_i$ et $\nu^{(k+1)}_i$ choisis de façon arbitraire.
Rappelons que $m_k = \bar{m}_l - 1$, $n_{k+1} = \bar{n}_{l+1} - 1$, $m_{k+1} = \bar{m}_{l+1} + 1$ et $n_{k+2} = \bar{n}_{l+2}$. Les conditions d'orthogonalité (1.15) et (1.16) nous donnent

$$\begin{aligned} &\text{pour tout} \quad j = 1, \ldots, m_{k+1} - 1 \\ &\qquad c^{(1)}(\xi^i P^{(1)}_{n_{k+1}+j}) = 0 \quad \text{pour} \quad i = 0, \ldots, n_{k+1} + m_{k+1} - 2 - j, \\ &\qquad\qquad \neq 0 \quad \text{pour} \quad i = n_{k+1} + m_{k+1} - 1 - j, \\ &\text{pour tout} \quad j = 2, \ldots, m_{k+1} - 1 \\ &\qquad c(\xi^i P^{(0)}_{n_{k+1}+j}) = 0 \quad \text{pour} \quad i = 0, \ldots, n_{k+1} + m_{k+1} - 1 - j, \\ &\qquad\qquad \neq 0 \quad \text{pour} \quad i = n_{k+1} + m_{k+1} - j. \end{aligned} \tag{1.60}$$

Considérons la base suivante

$$\{P^{(0)}_{\bar{n}_0}, \xi P^{(0)}_{\bar{n}_0}, \ldots, \xi^{\bar{m}_{l-1}-1} P^{(0)}_{\bar{n}_{l-1}}, P^{(0)}_{\bar{n}_l}, \xi P^{(0)}_{\bar{n}_l}, \ldots, \xi^{\bar{m}_l - 1} P^{(0)}_{\bar{n}_l}, P^{(0)}_{\bar{n}_{l+1}}, P^{(0)}_{n_{k+1}+2}, \ldots, P^{(0)}_{n_{k+2}-1}, \xi P^{(1)}_{n_{k+2}-1}\}.$$

En exprimant le polynôme $P^{(0)}_{\bar{n}_{l+2}}$ de degré $\bar{n}_{l+2} = n_{k+2}$ dans cette base, nous obtenons

$$\begin{aligned} P^{(0)}_{\bar{n}_{l+2}}(\xi) \;=\; & -\beta^{(0)}_0 P^{(0)}_{\bar{n}_0}(\xi) - \beta^{(0)}_1 \xi P^{(0)}_{\bar{n}_0}(\xi) - \cdots - \beta^{(0)}_{\bar{m}_0 - 1} \xi^{\bar{m}_0 - 1} P^{(0)}_{\bar{n}_0}(\xi) \\ & - \beta^{(1)}_0 P^{(0)}_{\bar{n}_1}(\xi) - \beta^{(1)}_1 \xi P^{(0)}_{\bar{n}_1}(\xi) - \cdots - \beta^{(1)}_{\bar{m}_1 - 1} \xi^{\bar{m}_1 - 1} P^{(0)}_{\bar{n}_1}(\xi) \\ & - \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ & - \beta^{(l)}_0 P^{(0)}_{\bar{n}_l}(\xi) - \beta^{(l)}_1 \xi P^{(0)}_{\bar{n}_l}(\xi) - \cdots - \beta^{(l)}_{\bar{m}_l - 1} \xi^{\bar{m}_l - 1} P^{(0)}_{\bar{n}_l}(\xi) \\ & - \beta^{(l+1)}_0 P^{(0)}_{\bar{n}_{l+1}}(\xi) - \beta^{(l+1)}_1 P^{(0)}_{\bar{n}_{l+1}+1}(\xi) - \cdots - \beta^{(l+1)}_{\bar{m}_{l+1} - 1} P^{(0)}_{\bar{n}_{l+2}-1}(\xi) + \xi P^{(1)}_{n_{k+2}-1}(\xi). \end{aligned}$$

Les orthogonalités (1.15), (1.16) et (1.60), nous permettent de montrer que

$$\beta^{(j)}_i = 0 \qquad \text{pour} \quad i = 0, \ldots, \bar{m}_j - 1 \qquad \text{et} \qquad \beta^{(l)}_i = 0 \quad \text{pour } i = 1, \ldots, \bar{m}_l - 1.$$
$$\text{et} \quad j = 0, \ldots, l - 1,$$

Ainsi, $P^{(0)}_{\bar{n}_{l+2}}$ vérifie la relation

$$\begin{aligned} P^{(0)}_{\bar{n}_{l+2}}(\xi) = \; & \xi P^{(1)}_{n_{k+2}-1}(\xi) - \beta^{(l+1)}_{\bar{m}_{l+1}-1} P^{(0)}_{\bar{n}_{l+2}-1}(\xi) - \cdots - \beta^{(l+1)}_{1} P^{(0)}_{\bar{n}_{l+1}+1}(\xi) \\ & - \beta^{(l+1)}_{0} P^{(0)}_{\bar{n}_{l+1}}(\xi) - \beta^{(l)}_{0} P^{(0)}_{\bar{n}_{l}}(\xi). \end{aligned} \tag{1.61}$$

Rappelons que nous avons $P^{(0)}_{\bar{n}_{l+1}} = P^{(0)}_{n_{k+1}+1}$ et que $P^{(0)}_{\bar{n}_{l}} = P^{(0)}_{n_k}$.
En multipliant la relation (1.61) par un polynôme U_i de degré i et en lui appliquant la fonctionnelle c, nous obtenons

$$\begin{aligned} c(U_i P^{(0)}_{\bar{n}_{l+2}}) = \; & c^{(1)}(U_i P^{(1)}_{n_{k+2}-1}) - \beta^{(l+1)}_{\bar{m}_{l+1}-1} c(U_i P^{(0)}_{\bar{n}_{l+2}-1}) - \cdots - \beta^{(l+1)}_{1} c(U_i P^{(0)}_{\bar{n}_{l+1}+1}) \\ & - \beta^{(l+1)}_{0} c(U_i P^{(0)}_{\bar{n}_{l+1}}) - \beta^{(l)}_{0} c(U_i P^{(0)}_{\bar{n}_{l}}). \end{aligned} \tag{1.62}$$

Si nous prenons $i = \bar{n}_{l+l} - 1 = n_{k+1}$ et $U_i = P^{(1)}_{n_{k+1}}$ alors, grâce à (1.60), la relation (1.62) devient

$$c^{(1)}(P^{(1)}_{n_{k+1}} P^{(1)}_{n_{k+2}-1}) = \beta^{(l)}_{0} c(P^{(1)}_{n_{k+1}} P^{(0)}_{\bar{n}_{l}}).$$

Puisque $P^{(1)}_{n_{k+1}}$ est de degré exactement égal à $\bar{n}_l + \bar{m}_l - 1$ alors, en le remplaçant par son expression donnée par (1.54), nous obtenons $c(P^{(1)}_{n_{k+1}} P^{(0)}_{\bar{n}_{l}}) = c(P^{(0)}_{\bar{n}_{l+1}-1} P^{(0)}_{\bar{n}_{l}}) \neq 0$,
et ainsi

$$\beta^{(l)}_{0} = \frac{c^{(1)}(P^{(1)}_{n_{k+1}} P^{(1)}_{n_{k+2}-1})}{c(P^{(0)}_{\bar{n}_{l+1}-1} P^{(0)}_{\bar{n}_{l}})}. \tag{1.63}$$

Pour les autres coefficients, nous prenons pour $i = (\bar{n}_{l+1} = n_{k+1} + 1), \ldots, (\bar{n}_{l+2} - 1 = n_{k+2} - 1)$, $U_i = P^{(0)}_{i}$ (polynômes réguliers ou complémentaires) et en imposant les orthogonalités (1.16) et (1.60), nous obtenons le système triangulaire

$$\begin{aligned} c^{(1)}(P^{(0)}_{\bar{n}_{l+1}} P^{(1)}_{n_{k+2}-1}) &= \beta^{(l+1)}_{\bar{m}_{l+1}-1} c(P^{(0)}_{\bar{n}_{l+1}} P^{(0)}_{\bar{n}_{l+2}-1}) \\ c^{(1)}(P^{(0)}_{\bar{n}_{l+1}+1} P^{(1)}_{n_{k+2}-1}) &= \beta^{(l+1)}_{\bar{m}_{l+1}-2} c(P^{(0)}_{\bar{n}_{l+1}+1} P^{(0)}_{\bar{n}_{l+2}-2}) + \beta^{(l+1)}_{\bar{m}_{l+1}-1} c(P^{(0)}_{\bar{n}_{l+1}+1} P^{(0)}_{\bar{n}_{l+2}-1}) \\ \cdots\cdots\cdots\cdots\cdots\cdots &\cdots \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ c^{(1)}(P^{(0)}_{\bar{n}_{l+2}-1} P^{(1)}_{n_{k+2}-1}) &= \beta^{(l+1)}_{0} c(P^{(0)}_{\bar{n}_{l+2}-1} P^{(0)}_{\bar{n}_{l+1}}) + \beta^{(l+1)}_{1} c(P^{(0)}_{\bar{n}_{l+2}-1} P^{(0)}_{\bar{n}_{l+1}+1}) \\ & \quad + \cdots + \beta^{(l+1)}_{\bar{m}_{l+1}-1} c(P^{(0)}_{\bar{n}_{l+2}-1} P^{(0)}_{\bar{n}_{l+2}-1}). \end{aligned} \tag{1.64}$$

D'après les relations (1.60), nous avons

$$c(P^{(0)}_{\bar{n}_{l+1}+i} P^{(0)}_{\bar{n}_{l+1}+j}) = c(P^{(0)}_{\bar{n}_{l+1}+f} P^{(0)}_{\bar{n}_{l+1}+h}) \neq 0 \tag{1.65}$$

$\forall\, i, j, f$ et h dans $\mathbb{N}$ vérifiant $i + j = f + h = \bar{m}_{l+1} - 1$, alors le système (1.64) est non-singulier.
Nous avons aussi $c^{(1)}(P^{(1)}_{n_{k+1}} P^{(1)}_{n_{k+2}-1}) \neq 0$, et donc le coefficient $\beta^{(l)}_{0}$ est différent de zéro. En remarquant que $P^{(0)}_{n_{k+2}}(0) = \beta^{(l)}_{0} \neq 0$, le Théorème 1.1.2, nous permet de confirmer

l'existence du polynôme $P^{(1)}_{n_{k+2}}$.

Exprimons maintenant le polynôme $P^{(1)}_{n_{k+2}} - P^{(0)}_{\bar{n}_{l+2}}$ de degré $n_{k+2}-1$ dans la base

$$\{P^{(1)}_{n_0}, \xi P^{(1)}_{n_0}, \ldots, \xi^{m_{k-1}-1}P^{(1)}_{n_{k-1}}, P^{(1)}_{n_k}, \xi P^{(1)}_{n_k}, \ldots, P^{(1)}_{n_{k+1}}, \xi P^{(1)}_{n_{k+1}}, \ldots, \xi^{m_{k+1}-1}P^{(1)}_{n_{k+1}}\}.$$

Nous obtenons

$$\begin{aligned} P^{(1)}_{n_{k+2}}(\xi) \;=\; & -\lambda^{(0)}_0 P^{(1)}_{n_0}(\xi) - \lambda^{(0)}_1 \xi P^{(1)}_{n_0}(\xi) - \cdots - \lambda^{(0)}_{m_0-1}\xi^{m_0-1}P^{(1)}_{n_0}(\xi) \\ & - \cdots\cdots\cdots\cdots\cdots\cdots\cdots - \lambda^{(k-1)}_{m_{k-1}-1}\xi^{m_{k-1}-1}P^{(1)}_{n_{k-1}}(\xi) \\ & - \lambda^{(k)}_0 P^{(1)}_{n_k}(\xi) - \lambda^{(k)}_1 \xi P^{(1)}_{n_k}(\xi) - \cdots - \lambda^{(k)}_{m_k-1}\xi^{m_k-1}P^{(1)}_{n_k}(\xi) \\ & - \lambda^{(k+1)}_0 P^{(1)}_{n_{k+1}}(\xi) - \lambda^{(k+1)}_1 \xi P^{(1)}_{n_{k+1}}(\xi) - \cdots - \lambda^{(k+1)}_{m_{k+1}-1}\xi^{m_{k+1}-1}P^{(1)}_{n_{k+1}}(\xi) \\ & + P^{(0)}_{\bar{n}_{l+2}}(\xi). \end{aligned} \tag{1.66}$$

Les orthogonalités (1.16) nous permettent d'avoir

$$\lambda^{(j)}_i = 0 \quad \text{pour} \quad i = 0, \ldots, \bar{m}_j - 1 \quad ; \quad \lambda^{(k+1)}_i = 0 \quad \text{pour} \quad i = 1, \ldots, m_{k+1}-1, \quad \text{et} \quad j = 0, \ldots, k$$

Alors, $P^{(1)}_{n_{k+2}}$ est calculé par la relation

$$P^{(1)}_{n_{k+2}}(\xi) = P^{(0)}_{\bar{n}_{l+2}}(\xi) - \lambda^{(k+1)}_0 P^{(1)}_{n_{k+1}}(\xi). \tag{1.67}$$

En multipliant cette relation par le polynôme *complémentaire* $P^{(1)}_{n_{k+2}-1}$ donné par (1.59) et en appliquant $c^{(1)}$, nous obtenons

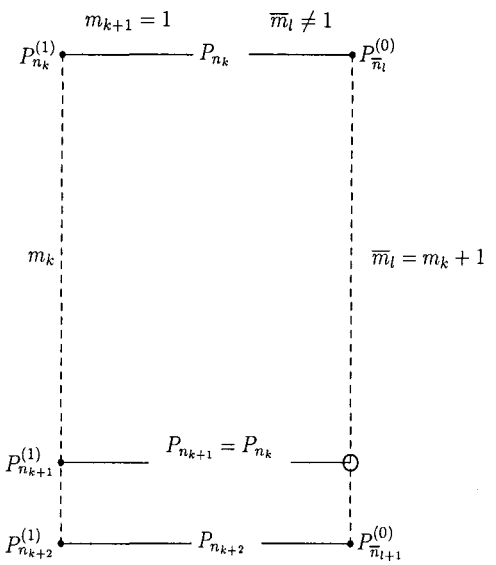
$$c(\xi P^{(1)}_{n_{k+2}-1}P^{(0)}_{\bar{n}_{l+2}}) = \lambda^{(k+1)}_0 c^{(1)}(P^{(1)}_{n_{k+2}-1}P^{(1)}_{n_{k+1}}).$$

Les orthogonalités (1.15) appliquées à $P^{(1)}_{n_{k+1}}$ nous donnent $c^{(1)}(P^{(1)}_{n_{k+2}-1}P^{(1)}_{n_{k+1}}) \neq 0$.
D'autre part, en remplaçant $P^{(0)}_{n_{k+2}}$ par son expression donnée par (1.61) et en utilisant les relations (1.16) appliquées à $P^{(0)}_{\bar{n}_{l+2}}$, nous trouvons

$$\lambda^{(k+1)}_0 = \frac{c(P^{(0)}_{\bar{n}_{l+2}}P^{(0)}_{\bar{n}_{l+2}})}{c^{(1)}(P^{(1)}_{n_{k+2}-1}P^{(1)}_{n_{k+1}})}. \tag{1.68}$$

A l'itération n_{k+2} le polynôme régulier $P_{n_{k+2}}$ existe et est de degré exactement égal à $n_{k+2} = \bar{n}_{l+2}$.
Dans ce premier cas, nous avons donc obtenu les relations de récurrence permettant de calculer les polynômes réguliers $P^{(1)}_{n_{k+2}}$ et $P^{(0)}_{\bar{n}_{l+2}}$ de même degré n_{k+2} en passant par les polynômes $P^{(1)}_{n_{k+1}}$ et $P^{(0)}_{\bar{n}_{l+1}}$.

FIG. 1.3 – Deuxième cas : $\bar{m}_l \neq 1$ et $m_{k+1} = 1$



- *Deuxième cas* : $\bar{m}_l \neq 1$ et $m_{k+1} = 1$. Ce cas est présenté dans la figure 1.3.

Nous avons $n_k = \bar{n}_l$, et puisque $\bar{m}_l \neq 1$ alors d'après le Théorème 1.2.3, nous avons $m_k = \bar{m}_l - 1$ ce qui implique $n_{k+1} = \bar{n}_{l+1} - 1$. Cela veut dire que nous devons d'abord calculer le polynôme régulier $P^{(1)}_{n_{k+1}}$. Pour cela nous procédons, comme dans le premier cas, en construisant les polynômes *complémentaires* donnés par la relation (1.51). Ainsi le polynôme $P^{(1)}_{n_{k+1}}$ est calculé grâce aux relations (1.54), (1.55) et (1.56).
D'autre part, nous avons $H^{(0)}_{\bar{n}_{l+1}} = H^{(0)}_{n_{k+1}+1} \neq 0$. Et puisque $m_{k+1} = 1$ alors $H^{(1)}_{n_{k+1}+1} \neq 0$ et $n_{k+2} = n_{k+1} + 1 = \bar{n}_{l+1}$. Dans ce cas, le polynôme $P^{(0)}_{\bar{n}_{l+1}}$ de degré $\bar{n}_{l+1} = n_{k+1} + 1$ ne peut pas être calculé comme dans le premier cas par la relation (1.58). Par contre, il peut être exprimé dans la base

$$\{P^{(0)}_{\bar{n}_0}, \xi P^{(0)}_{\bar{n}_0}, \ldots, \xi^{\bar{m}_0-1} P^{(0)}_{\bar{n}_0}, P^{(0)}_{\bar{n}_1}, \xi P^{(0)}_{\bar{n}_1}, \ldots, \xi^{\bar{m}_{l-1}-1} P^{(0)}_{\bar{n}_{l-1}}, P^{(0)}_{\bar{n}_l}, \xi P^{(0)}_{n_l}, \ldots, \xi^{\bar{m}_l-1} P^{(0)}_{\bar{n}_l}, \xi P^{(1)}_{n_{k+1}}\}$$

ce qui nous permet d'écrire

$$\begin{aligned} P^{(0)}_{\bar{n}_{l+1}}(\xi) \;=\; & -\gamma^{(0)}_0 P^{(0)}_{\bar{n}_0}(\xi) - \gamma^{(0)}_1 \xi P^{(0)}_{\bar{n}_0}(\xi) - \cdots - \gamma^{(0)}_{\bar{m}_0-1} \xi^{\bar{m}_0-1} P^{(0)}_{\bar{n}_0}(\xi) \\ & - \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ & - \gamma^{(l)}_0 P^{(0)}_{\bar{n}_l}(\xi) - \gamma^{(l)}_1 \xi P^{(0)}_{\bar{n}_l}(\xi) - \cdots - \gamma^{(l)}_{\bar{m}_l-1} \xi^{\bar{m}_l-1} P^{(0)}_{\bar{n}_l}(\xi) \\ & + \xi P^{(1)}_{n_{k+1}}(\xi). \end{aligned}$$

En utilisant les orthogonalités (1.15) et (1.16), nous obtenons

$$\gamma_i^{(j)} = 0 \quad \text{pour} \quad \begin{cases} i = 0, \ldots, \bar{m}_j - 1 \\ j = 0, \ldots, l-1 \end{cases} \quad ; \quad \gamma_i^{(l)} = 0 \quad \text{pour} \quad i = 1, \ldots, \bar{m}_l - 1.$$

Ainsi, le polynôme $P^{(0)}_{\bar{n}_{l+1}}$ satisfait la récurrence

$$P^{(0)}_{\bar{n}_{l+1}}(\xi) = \xi P^{(1)}_{n_{k+1}}(\xi) - \gamma_0^{(l)} P^{(0)}_{\bar{n}_l}(\xi). \tag{1.69}$$

Si nous multiplions cette relation par $P^{(1)}_{n_{k+1}}$ et lui appliquons la fonctionnelle c, alors nous obtenons

$$c(\xi P^{(1)}_{n_{k+1}} P^{(1)}_{n_{k+1}}) = \gamma_0^{(l)} c(P^{(1)}_{n_{k+1}} P^{(0)}_{\bar{n}_l}).$$

Ici le polynôme $P^{(1)}_{n_{k+1}}$ est donné par la relation (1.54) que nous pouvons écrire sous la forme

$$P^{(1)}_{n_{k+1}} = P^{(0)}_{\bar{n}_{l+1}-1} + \phi_{n_{k+1}-1},$$

où $\phi_{n_{k+1}-1}$ est un polynôme de degré égal à $n_{k+1} - 1$. Grâce aux orthogonalités (1.16), nous avons

$$c(P^{(1)}_{n_{k+1}} P^{(0)}_{\bar{n}_l}) = c(P^{(0)}_{\bar{n}_{l+1}-1} P^{(0)}_{\bar{n}_l}) \neq 0,$$

d'où

$$\gamma_0^{(l)} = \frac{c(\xi P^{(1)}_{n_{k+1}} P^{(1)}_{n_{k+1}})}{c(P^{(0)}_{\bar{n}_{l+1}-1} P^{(0)}_{\bar{n}_l})}. \tag{1.70}$$

Maintenant nous considérons la base des polynômes de degré au plus $\bar{n}_{l+1}$

$$\{P^{(1)}_{n_0}, \xi P^{(1)}_{n_0}, \ldots, \xi^{m_0-1} P^{(1)}_{n_0}, P^{(1)}_{n_1}, \xi P^{(1)}_{n_1}, \ldots, \xi^{m_k-1} P^{(1)}_{n_k}, P^{(1)}_{n_{k+1}}, P^{(0)}_{\bar{n}_{l+1}}\}.$$

En exprimant le polynôme $P^{(1)}_{n_{k+2}}$ de degré $n_{k+2} = n_{k+1} + 1 = \bar{n}_{l+1}$ dans cette base, nous avons la relation suivante

$$\begin{aligned} P^{(1)}_{n_{k+2}}(\xi) \quad = \quad & -\lambda_0^{(0)} P^{(1)}_{n_0}(\xi) - \lambda_1^{(0)} \xi P^{(1)}_{n_0}(\xi) - \cdots - \lambda^{(0)}_{m_0-1} \xi^{m_0-1} P^{(1)}_{n_0}(\xi) \\ & - \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ & - \lambda_0^{(k)} P^{(1)}_{n_k}(\xi) - \lambda_1^{(k)} \xi P^{(1)}_{n_k}(\xi) - \cdots - \lambda^{(k)}_{m_k-1} \xi^{m_k-1} P^{(1)}_{n_k}(\xi) \\ & - \lambda_0^{(k+1)} P^{(1)}_{n_{k+1}}(\xi) + P^{(0)}_{\bar{n}_{l+1}}(\xi). \end{aligned}$$

Les orthogonalités (1.15) et (1.16) nous donnent

$$\lambda_i^{(j)} = 0 \quad \text{pour } i = 0, \ldots, m_j - 1 \qquad \text{et } j = 0, \ldots, k.$$

Ainsi

$$P^{(1)}_{n_{k+2}}(\xi) = P^{(0)}_{\bar{n}_{l+1}}(\xi) - \lambda_0^{(k+1)} P^{(1)}_{n_{k+1}}(\xi). \tag{1.71}$$

Pour le calcul du coefficient $\lambda_0^{(k+1)}$, nous multiplions cette relation par $P_{n_{k+1}}^{(1)}$ et nous lui appliquons la fonctionnelle $c^{(1)}$ afin d'obtenir

$$c^{(1)}(P_{n_{k+1}}^{(1)} P_{\bar{n}_{l+1}}^{(0)}) = \lambda_0^{(k+1)} c^{(1)}(P_{n_{k+1}}^{(1)} P_{n_{k+1}}^{(1)}).$$

Or, puisque $m_{k+1} = 1$, alors les orthogonalités (1.16) appliquées au polynôme $P_{n_{k+1}}^{(1)}$ nous permettent d'avoir $c^{(1)}(P_{n_{k+1}}^{(1)} P_{n_{k+1}}^{(1)}) \neq 0$ et donc nous obtenons $\lambda_0^{(k+1)}$.
De plus, en remplaçant $P_{\bar{n}_{l+1}}^{(0)}$ par son expression donnée par (1.69), et grâce aux orthogonalités (1.16), nous pouvons écrire $c(P_{\bar{n}_{l+1}}^{(0)} P_{\bar{n}_{l+1}}^{(0)}) = c(\xi P_{n_{k+1}}^{(1)} P_{\bar{n}_{l+1}}^{(0)})$.
Ainsi, $\lambda_0^{(k+1)}$ peut être calculé à partir de la relation suivante

$$\lambda_0^{(k+1)} = \frac{c(P_{\bar{n}_{l+1}}^{(0)} P_{\bar{n}_{l+1}}^{(0)})}{c^{(1)}(P_{n_{k+1}}^{(1)} P_{n_{k+1}}^{(1)})}. \tag{1.72}$$

Nous avons ainsi trouvé les trois récurrences permettant le calcul des polynômes $P_{\bar{n}_{l+1}}^{(0)}$ et $P_{n_{k+2}}^{(1)}$ de degré $n_{k+2} = \bar{n}_{l+1}$ en passant par le polynôme $P_{n_{k+1}}^{(1)}$.
A l'itération n_{k+1} le polynôme $P_{n_{k+1}}$ existe mais n'est pas de degré égal à n_{k+1}. Il est donc égal à P_{n_k} et nous retrouvons à l'itération suivante n_{k+2} le polynôme régulier $P_{n_{k+2}}$ de degré exactement égal à $n_{k+2} = \bar{n}_{l+1}$.

- *Troisième cas* : $\bar{m}_l = 1$ et $m_k \neq 1$. Ce cas est présenté dans la figure 1.4.

Ici nous avons $\bar{n}_{l+1} = \bar{n}_l + 1 = n_k + 1$ et, puisque $m_k \neq 1$, alors $H_{n_k+1}^{(0)} \neq 0$ tandis que $H_{n_k+1}^{(1)} = 0$. Cela implique, grâce au lemme 1.2.1, que

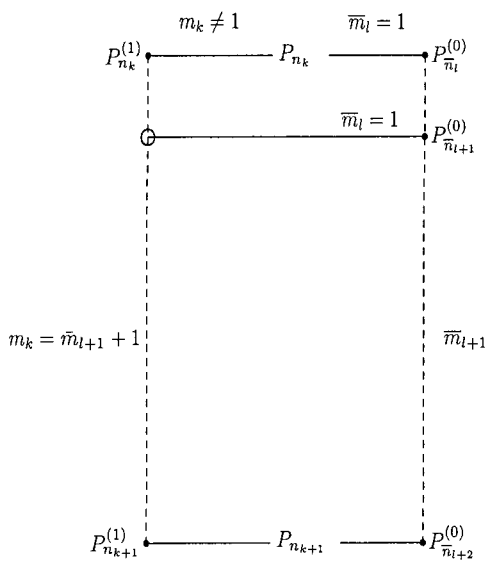
$$P_{\bar{n}_{l+1}}^{(0)} = P_{n_k+1}^{(0)} = \xi P_{n_k}^{(1)}. \tag{1.73}$$

Le Théorème 1.2.3 nous permet de déduire que $m_k = \bar{m}_{l+1} + 1$, donc $n_{k+1} = \bar{n}_{l+2}$.
Ce résultat peut aussi être obtenu grâce au lemme 1.2.2.
Le polynôme $P_{\bar{n}_{l+2}}^{(0)}$ peut être calculé d'une façon similaire à celle donnée dans le *Premier cas*. Ainsi nous commençons par construire les polynômes *complémentaires* suivants

$$\begin{aligned} P_{n_k+i}^{(0)} &= \xi P_{n_k+i-1}^{(1)} - \mu_{i-1}^{(k)} P_{n_k+i-1}^{(0)} \qquad \text{pour} \quad 2 \leq i \leq m_k - 1, \\ P_{n_k+i}^{(1)} &= P_{n_k+i}^{(0)} - \nu_i^{(k)} P_{n_k+i-1}^{(1)} \qquad \text{pour} \quad 1 \leq i \leq m_k - 1, \end{aligned} \tag{1.74}$$

avec $\mu_i^{(k)}$ et $\nu_i^{(k)}$ choisis de façon arbitraire.

FIG. 1.4 – Troisième cas : $\bar{m}_l = 1$ et $m_k \neq 1$



Les conditions d'orthogonalité (1.15) et (1.16) nous permettent d'avoir les relations

$$\begin{aligned}
&\text{pour tout} \quad j = 1, \ldots, m_k - 1 \\
&\qquad c^{(1)}(\xi^i P^{(1)}_{n_k+j}) = 0 \quad \text{pour} \quad i = 0, \ldots, n_k + m_k - 2 - j, \\
&\qquad\qquad\qquad\quad \neq 0 \quad \text{pour} \quad i = n_k + m_k - 1 - j. \\
&\text{pour tout} \quad j = 2, \ldots, m_k - 1 \\
&\qquad c(\xi^i P^{(0)}_{n_k+j}) = 0 \quad \text{pour} \quad i = 0, \ldots, n_k + m_k - 1 - j, \\
&\qquad\qquad\qquad\quad \neq 0 \quad \text{pour} \quad i = n_k + m_k - j.
\end{aligned} \tag{1.75}$$

Considérons la base suivante

$$\{P^{(0)}_{\bar{n}_0}, \xi P^{(0)}_{\bar{n}_0}, \ldots, \xi^{\bar{m}_{l-1}-1} P^{(0)}_{\bar{n}_{l-1}}, P^{(0)}_{\bar{n}_l}, P^{(0)}_{\bar{n}_{l+1}}, P^{(0)}_{\bar{n}_{l+1}+1}, \ldots, P^{(0)}_{\bar{n}_{l+2}-1}, \xi P^{(1)}_{n_{k+1}-1}\}.$$

En exprimant le polynôme $P^{(0)}_{\bar{n}_{l+2}}$ de degré $\bar{n}_{l+2} = n_{k+1}$ dans cette base, nous obtenons

$$\begin{aligned}
P^{(0)}_{\bar{n}_{l+2}}(\xi) \;=\; & -\beta^{(0)}_0 P^{(0)}_{\bar{n}_0}(\xi) - \beta^{(0)}_1 \xi P^{(0)}_{\bar{n}_0}(\xi) - \cdots - \beta^{(0)}_{\bar{m}_0-1} \xi^{\bar{m}_0-1} P^{(0)}_{\bar{n}_0}(\xi) \\
& -\beta^{(1)}_0 P^{(0)}_{\bar{n}_1}(\xi) - \beta^{(1)}_1 \xi P^{(0)}_{\bar{n}_1}(\xi) - \cdots - \beta^{(1)}_{\bar{m}_1-1} \xi^{\bar{m}_1-1} P^{(0)}_{\bar{n}_1}(\xi) \\
& \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\
& -\beta^{(l)}_0 P^{(0)}_{\bar{n}_l}(\xi) - \beta^{(l+1)}_0 P^{(0)}_{\bar{n}_{l+1}}(\xi) - \beta^{(l+1)}_1 P^{(0)}_{\bar{n}_{l+1}+1}(\xi) - \cdots - \beta^{(l+1)}_{\bar{m}_{l+1}-1} P^{(0)}_{\bar{n}_{l+2}-1}(\xi) \\
& + \xi P^{(1)}_{n_{k+1}-1}(\xi).
\end{aligned}$$

Les orthogonalités (1.16) et (1.75) nous permettent de montrer que

$$\beta_i^{(j)} = 0 \qquad \text{pour} \quad i = 0, \ldots, \bar{m}_j - 1,$$
$$\text{et} \quad j = 0, \ldots, l-1.$$

Le polynôme $P^{(0)}_{\bar{n}_{l+2}} = P^{(0)}_{n_{k+1}}$ vérifie donc la relation

$$P^{(0)}_{\bar{n}_{l+2}}(\xi) = \xi P^{(1)}_{n_{k+1}-1}(\xi) - \beta^{(l+1)}_{\bar{m}_{l+1}-1} P^{(0)}_{\bar{n}_{l+2}-1}(\xi) - \cdots - \beta^{(l+1)}_0 P^{(0)}_{\bar{n}_{l+1}}(\xi) - \beta^{(l)}_0 P^{(0)}_{\bar{n}_l}(\xi). \quad (1.76)$$

Cette relation est semblable à la relation (1.61) donnée dans le premier cas.
En multipliant (1.76) par $P^{(1)}_{n_k}$ et en lui appliquant la fonctionnelle c nous obtenons

$$c^{(1)}(P^{(1)}_{n_k} P^{(1)}_{n_{k+1}-1}) = \beta^{(l)}_0 c(P^{(1)}_{n_k} P^{(0)}_{\bar{n}_l}).$$

Puisque $n_k = \bar{n}_l$ et $\bar{m}_l = 1$ alors, d'après les orthogonalités (1.16), nous avons

$$c(P^{(1)}_{n_k} P^{(0)}_{\bar{n}_l}) = c(P^{(0)}_{\bar{n}_l} P^{(0)}_{\bar{n}_l}) \neq 0.$$

D'où

$$\beta^{(l)}_0 = \frac{c^{(1)}(P^{(1)}_{n_k} P^{(1)}_{n_{k+1}-1})}{c(P^{(0)}_{\bar{n}_l} P^{(0)}_{\bar{n}_l})}. \quad (1.77)$$

Pour les autres coefficients de la relation (1.76, nous multiplions pour $i \in \{(\bar{n}_{l+1} = n_k + 1), \ldots, (\bar{n}_{l+2} - 1 = n_{k+1} - 1)\}$ la relation (1.76) par $P^{(0)}_i$ (polynôme régulier ou complémentaire) et en imposant les orthogonalités (1.16) et (1.75), nous obtenons le système triangulaire

$$\begin{aligned}
c^{(1)}(P^{(0)}_{\bar{n}_{l+1}} P^{(1)}_{n_{k+1}-1}) &= \beta^{(l+1)}_{\bar{m}_{l+1}-1} c(P^{(0)}_{\bar{n}_{l+1}} P^{(0)}_{\bar{n}_{l+2}-1}) \\
c^{(1)}(P^{(0)}_{\bar{n}_{l+1}+1} P^{(1)}_{n_{k+1}-1}) &= \beta^{(l+1)}_{\bar{m}_{l+1}-2} c(P^{(0)}_{\bar{n}_{l+1}+1} P^{(0)}_{n_{k+1}-2}) + \beta^{(l+1)}_{\bar{m}_{l+1}-1} c(P^{(0)}_{\bar{n}_{l+1}+1} P^{(0)}_{n_{k+1}-1}) \\
\cdots\cdots\cdots\cdots\cdots\cdots &\ \cdots\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \quad (1.78) \\
c^{(1)}(P^{(0)}_{\bar{n}_{l+2}-1} P^{(1)}_{n_{k+1}-1}) &= \beta^{(l+1)}_0 c(P^{(0)}_{\bar{n}_{l+2}-1} P^{(0)}_{\bar{n}_{l+1}}) + \beta^{(l+1)}_1 c(P^{(0)}_{\bar{n}_{l+2}-1} P^{(0)}_{\bar{n}_{l+1}+1}) \\
&\quad + \cdots + \beta^{(l+1)}_{\bar{m}_{l+1}-1} c(P^{(0)}_{\bar{n}_{l+2}-1} P^{(0)}_{\bar{n}_{l+2}-1}).
\end{aligned}$$

D'après les relations (1.75), nous avons

$$c(P^{(0)}_{\bar{n}_{l+1}+i} P^{(0)}_{\bar{n}_{l+1}+j}) = c(P^{(0)}_{\bar{n}_{l+1}+f} P^{(0)}_{\bar{n}_{l+1}+h}) \neq 0 \quad (1.79)$$

$\forall\ i, j, f$ et h dans $I\!N$ vérifiant $i + j = f + h = \bar{m}_{l+1} - 1$, et donc le système (1.78) est non-singulier.
Exprimons, maintenant, le polynôme $P^{(1)}_{n_{k+1}} - P^{(0)}_{\bar{n}_{l+2}}$ de degré $n_{k+1} - 1$ dans la base

$$\{P^{(1)}_{n_0}, \xi P^{(1)}_{n_0}, \ldots, \xi^{m_{k-1}-1} P^{(1)}_{n_{k-1}}, P^{(1)}_{n_k}, \xi P^{(1)}_{n_k}, \ldots, \xi^{m_k - 1} P^{(1)}_{n_k}\}.$$

Nous obtenons

$$\begin{aligned} P^{(1)}_{n_{k+1}}(\xi) &= -\lambda^{(0)}_0 P^{(1)}_{n_0}(\xi) - \lambda^{(0)}_1 \xi P^{(1)}_{n_0}(\xi) - \cdots - \lambda^{(0)}_{m_0-1}\xi^{m_0-1} P^{(1)}_{n_0}(\xi) \\ &\quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ &\quad - \lambda^{(k)}_0 P^{(1)}_{n_k}(\xi) - \lambda^{(k)}_1 \xi P^{(1)}_{n_k}(\xi) - \cdots - \lambda^{(k)}_{m_k-1}\xi^{m_k-1} P^{(1)}_{n_k}(\xi) \\ &\quad + P^{(0)}_{\bar{n}_{l+2}}(\xi). \end{aligned}$$

Les orthogonalités (1.16) nous permettent d'avoir

$$\lambda^{(j)}_i = 0 \quad \text{pour} \quad i = 0, \ldots, \bar{m}_j - 1, \quad ; \quad \lambda^{(k)}_i = 0 \quad \text{pour} \quad i = 1, \ldots, m_k - 1$$
$$\text{et} \quad j = 0, \ldots, k-1.$$

Ainsi, $P^{(1)}_{n_{k+1}}$ pourra être calculé par la relation

$$P^{(1)}_{n_{k+1}}(\xi) = P^{(0)}_{\bar{n}_{l+2}}(\xi) - \lambda^{(k)}_0 P^{(1)}_{n_k}(\xi). \tag{1.80}$$

En multipliant cette relation par le polynôme *complémentaire* $P^{(1)}_{n_{k+1}-1}$ donné par (1.74), et en appliquant $c^{(1)}$, nous obtenons

$$c(\xi P^{(1)}_{n_{k+1}-1} P^{(0)}_{\bar{n}_{l+2}}) = \lambda^{(k)}_0 c^{(1)}(P^{(1)}_{n_{k+1}-1} P^{(1)}_{n_k}).$$

Les orthogonalités (1.15) appliquées à $P^{(1)}_{n_k}$ nous donnent $c^{(1)}(P^{(1)}_{n_{k+1}-1} P^{(1)}_{n_k}) \neq 0$.
D'autre part, en remplaçant le polynôme $P^{(0)}_{n_{k+1}}$ par son expression, les conditions (1.16), nous permettent d'avoir $c(P^{(0)}_{\bar{n}_{l+2}} P^{(0)}_{\bar{n}_{l+2}}) = c(\xi P^{(1)}_{n_{k+1}-1} P^{(0)}_{\bar{n}_{l+2}})$ et ainsi $\lambda^{(k)}_0$ est donné par

$$\lambda^{(k)}_0 = \frac{c(P^{(0)}_{\bar{n}_{l+2}} P^{(0)}_{\bar{n}_{l+2}})}{c^{(1)}(P^{(1)}_{n_{k+1}-1} P^{(1)}_{n_k})}. \tag{1.81}$$

A l'itération n_{k+1} le polynôme $P_{n_{k+1}}$ existe et est de degré exactement égal à n_{k+1}.
Nous avons ainsi les relations de récurrence pour le calcul des polynômes $P^{(1)}_{n_{k+1}}$ et $P^{(0)}_{\bar{n}_{l+2}}$ du même degré n_{k+1}.
Ces différentes relations de récurrence permettent de calculer les polynômes réguliers $P^{(1)}_k$ et $P^{(0)}_k$ dans les trois cas possibles de *breakdown*, et nous trouvons ainsi une généralisation des relations (1.41). Ces relations peuvent être résumées dans l'algorithme suivant

Algorithme 1.3.3 : *Relations de récurrence à deux termes de* $P_k^{(1)}$ *et* $P_k^{(0)}$.

1. Initialisation : $P_0^{(1)} = 1$, $P_0^{(0)} = 1$, $l = k = 0$, $\bar{n}_0 = 0$ et $n_0 = 0$.

2. Boucle :
   $m_k = 1$.
3. Si $c(P_{\bar{n}_l}^{(0)} P_{\bar{n}_l}^{(0)}) = 0$.
   Tant que $c^{(1)}(P_{n_k}^{(1)} P_{n_k+m_k-1}^{(1)}) = 0$.
   $m_k = m_k + 1$.
   Calculer $P_{n_k+m_k-1}^{(1)}$ et $P_{n_k+m_k-1}^{(0)}$ à partir de (1.51).
   Fin (tant que).
   $\bar{m}_l = m_k + 1$, $n_{k+1} = n_k + m_k$ et $\bar{n}_{l+1} = n_{k+1} + 1$.
   Calculer $P_{n_{k+1}}^{(0)}$ à partir de (1.51).
   Calculer $P_{n_{k+1}}^{(1)}$ à partir de (1.54), (1.55) et (1.56).
   Si $c^{(1)}(P_{n_{k+1}}^{(1)} P_{n_{k+1}}^{(1)}) = 0$.
   Calculer $P_{\bar{n}_{l+1}}^{(0)}$ à partir de (1.58).
   Calculer $P_{n_{k+1}+1}^{(1)}$ à partir de (1.59).
   $m_{k+1} = 2$
   Tant que $c^{(1)}(P_{n_{k+1}}^{(1)} P_{n_{k+1}+m_{k+1}-1}^{(1)}) = 0$.
   $m_{k+1} = m_{k+1} + 1$.
   Calculer $P_{n_{k+1}+m_{k+1}-1}^{(1)}$ et $P_{n_{k+1}+m_{k+1}-1}^{(0)}$ à partir de (1.59).
   Fin (tant que).
   $\bar{m}_{l+1} = m_{k+1} - 1$, $n_{k+2} = n_{k+1} + m_{k+1}$ et $\bar{n}_{l+2} = \bar{n}_{l+1} + \bar{m}_{l+1}$.
   Calculer $P_{n_{k+2}}^{(0)}$ à partir de (1.61), (1.63) et (1.64).
   Calculer $P_{n_{k+2}}^{(1)}$ à partir de (1.67) et (1.68).
   $l = l + 2$.
   Sinon
   $m_{k+1} = 1$ et $n_{k+2} = n_{k+1} + m_{k+1}$.
   Calculer $P_{\bar{n}_{l+1}}^{(0)}$ à partir de (1.69) et (1.70).
   Calculer $P_{n_{k+2}}^{(1)}$ à partir de (1.71) et (1.72).
   $l = l + 1$.
   Fin (si).
   $k = k + 2$.

4. Sinon
   $\bar{m}_l = 1$ et $\bar{n}_{l+1} = \bar{n}_l + 1$.
   Si $c^{(1)}(P^{(1)}_{n_k} P^{(1)}_{n_k}) = 0$.
      Calculer $P^{(0)}_{\bar{n}_{l+1}}$ à partir de (1.73).
      Calculer $P^{(1)}_{n_k+1}$ à partir de (1.74).
      $m_k = 2$.
      Tant que $c^{(1)}(P^{(1)}_{n_k} P^{(1)}_{n_k+m_k-1}) = 0$.
         $m_k = m_k + 1$.
         Calculer $P^{(1)}_{n_k+m_k-1}$ et $P^{(0)}_{n_k+m_k-1}$ à partir de (1.74).
      Fin (tant que).
      $\bar{m}_{l+1} = m_k - 1$, $n_{k+1} = n_k + m_k$ et $\bar{n}_{l+2} = \bar{n}_{l+1} + \bar{m}_{l+1}$.
      Calculer $P^{(0)}_{\bar{n}_{l+2}}$ à partir de (1.76), (1.77) et (1.78).
      Calculer $P^{(1)}_{n_{k+1}}$ à partir de (1.80) et (1.81).
      $l = l + 2$.
   Sinon
      $n_{k+1} = n_k + m_k$ et $\bar{n}_{l+1} = \bar{n}_l + m_l$.
      Calculer $P^{(0)}_{\bar{n}_{l+1}}$ et $P^{(1)}_{n_{k+1}}$ à partir de (1.41) et (1.42).
      $l = l + 1$.
   Fin (si).
   $k = k + 1$.
Fin (si).
Fin.

D'après (1.24) et (1.40), les vecteurs réguliers sont définis par

$$\begin{aligned} v_{\bar{n}_l} &= P^{(0)}_{\bar{n}_l}(A)v_0, & w_{\bar{n}_l} &= P^{(0)}_{\bar{n}_l}(A^T)w_0 \\ p_{n_k} &= P^{(1)}_{n_k}(A)v_0, & q_{n_k} &= P^{(1)}_{n_k}(A)w_0. \end{aligned} \tag{1.82}$$

Pour $i = 1, \ldots, \bar{m}_l - 1$ et $j = 1, \ldots, m_k - 1$, nous introduisons les vecteurs *complémentaires* définis par

$$\begin{aligned} v_{\bar{n}_l+i} &= P^{(0)}_{\bar{n}_l+i}(A)v_0, & w_{\bar{n}_l+i} &= P^{(0)}_{\bar{n}_l+i}(A^T)w_0 \\ p_{n_k+j} &= P^{(1)}_{n_k+j}(A)v_0, & q_{n_k+j} &= P^{(1)}_{n_k+j}(A)w_0, \end{aligned} \tag{1.83}$$

avec $P^{(0)}_{\bar{n}_l+i}$ et $P^{(1)}_{n_k+j}$ des polynômes *complémentaires*.
Nous notons

$$V_l = [v_{\bar{n}_l} \;\; v_{\bar{n}_l+1} \;\; \ldots \;\; v_{\bar{n}_{l+1}-1}], \qquad W_l = [w_{\bar{n}_l} \;\; w_{\bar{n}_l+1} \;\; \ldots \;\; w_{\bar{n}_{l+1}-1}]$$

et

$$P_k = [p_{n_k} \;\; p_{n_k+1} \;\; \ldots \;\; p_{n_{k+1}-1}], \qquad Q_k = [q_{n_k} \;\; q_{n_k+1} \;\; \ldots \;\; q_{n_{k+1}-1}].$$

En traduisant l'algorithme 1.3.3, nous obtenons le processus de Lanczos par des relations de récurrence à deux termes dans lequel nous évitons le *true* et le *ghost-breakdown*.

Algorithme 1.3.4 *:Le processus de Lanczos par des récurrences à deux termes avec look-ahead.*

1. Initialisation : Choisir v_0 et $w_0 \in \mathbb{R}^N$,
$p_0 = v_0$, $q_0 = w_0$, $l = k = 0$, $\bar{n}_0 = 0$, $n_0 = 0$.
Prendre $V_0 = v_0$, $W_0 = w_0$, $P_0 = p_0$ et $Q_0 = q_0$.

2. Tant que ($\|v_{\bar{n}_l}\| \neq 0$ et $\|w_{\bar{n}_l}\| \neq 0$) :
$m_k = 1$.
3. Si $(w_{\bar{n}_l}, v_{\bar{n}_l}) = 0$.
Tant que $(q_{n_k}, Ap_{n_k+m_k-1}) = 0$.
$m_k = m_k + 1$.
$v_{\bar{n}_l+m_k-1} = Ap_{n_k+m_k-2} - \mu^{(k)}_{m_k-1} v_{\bar{n}_l+m_k-2}$.
$w_{\bar{n}_l+m_k-1} = A^T q_{n_k+m_k-2} - \mu^{(k)}_{m_k-1} w_{\bar{n}_l+m_k-2}$.
$V_l = [V_l \quad v_{\bar{n}_l+m_k-1}]$, $W_l = [W_l \quad w_{\bar{n}_l+m_k-1}]$.
$p_{n_k+m_k-1} = v_{\bar{n}_l+m_k-1} - \nu^{(k)}_{m_k-1} p_{n_k+m_k-2}$.
$q_{n_k+m_k-1} = w_{\bar{n}_l+m_k-1} - \nu^{(k)}_{m_k-1} q_{n_k+m_k-2}$.
$P_k = [P_k \quad p_{n_k+m_k-1}]$, $Q_k = [Q_k \quad q_{n_k+m_k-1}]$.
Fin (tant que).
$\bar{m}_l = m_k + 1$, $n_{k+1} = n_k + m_k$ et $\bar{n}_{l+1} = n_{k+1} + 1$.
$v_{\bar{n}_{l+1}-1} = Ap_{n_{k+1}-1} - \mu^{(k)}_{m_k} v_{\bar{n}_{l+1}-2}$.
$w_{\bar{n}_{l+1}-1} = A^T q_{n_{k+1}-1} - \mu^{(k)}_{m_k} w_{\bar{n}_{l+1}-2}$.
$V_l = [V_l \quad v_{\bar{n}_{l+1}-1}]$, $W_l = [W_l \quad w_{\bar{n}_{l+1}-1}]$.

$\alpha_k = (Q_k{}^T A P_k)^{-1} Q_k{}^T A v_{\bar{n}_{l+1}-1}$.
Si $k = 0$.
$\alpha_0^{(-1)} = 0$.
Sinon
$$\alpha_0^{(k-1)} = \frac{(w_{\bar{n}_l}, v_{\bar{n}_{l+1}-1})}{(q_{n_{k-1}}, Ap_{n_k-1})}.$$
Fin(si).

$p_{n_{k+1}} = v_{\bar{n}_{l+1}-1} - P_k\alpha_k - \alpha_0^{(k-1)} p_{n_{k-1}}$,
$q_{n_{k+1}} = w_{\bar{n}_{l+1}-1} - Q_k\alpha_k - \alpha_0^{(k-1)} q_{n_{k-1}}$.
$P_{k+1} = p_{n_{k+1}}, \qquad Q_{k+1} = q_{n_{k+1}}$.
$k = k+1$

Sinon.

$\bar{m}_l = 1, \qquad \bar{n}_{l+1} = \bar{n}_l + \bar{m}_l$.

Fin (si).

4. Si $(q_{n_k}, Ap_{n_k}) = 0$.

$v_{\bar{n}_{l+1}} = Ap_{n_k}$,
$w_{\bar{n}_{l+1}} = A^T q_{n_k}$.
$V_{l+1} = v_{\bar{n}_{l+1}}, \qquad W_{l+1} = w_{\bar{n}_{l+1}}$.
$l = l+1$.
$p_{n_k+1} = v_{\bar{n}_l} - \nu_1^{(k+1)} p_{n_k}$.
$q_{n_k+1} = w_{\bar{n}_l} - \nu_1^{(k+1)} q_{n_k}$.
$P_k = [P_k \quad p_{n_k+1}], \qquad Q_k = [Q_k \quad q_{n_k+1}]$.
$m_k = 2$
Tant que $(q_{n_k}, Ap_{n_k+m_k-1}) = 0$.
$m_k = m_k + 1$.
$v_{\bar{n}_l+m_k-2} = Ap_{n_k+m_k-2} - \mu_{m_k-2}^{(k)} v_{\bar{n}_l+m_k-3}$
$w_{\bar{n}_l+m_k-2} = A^T q_{n_k+m_k-2} - \mu_{m_k-2}^{(k)} w_{\bar{n}_l+m_k-3}$
$V_l = [V_l \quad v_{\bar{n}_l+m_k-2}], \qquad W_l = [W_l \quad \bar{w}_{n_l+m_k-2}]$.
$p_{n_k+m_k-1} = v_{\bar{n}_l+m_k-2} - \nu_{m_k-1}^{(k)} p_{n_k+m_k-2}$
$q_{n_k+m_k-1} = w_{\bar{n}_l+m_k-2} - \nu_{m_k-1}^{(k)} q_{n_k+m_k-2}$
$P_k = [P_k \quad p_{n_k+m_k-1}], \qquad Q_k = [Q_k \quad q_{n_k+m_k-1}]$.
Fin (tant que).
$\bar{m}_l = m_k - 1$, $n_{k+1} = n_k + m_k$ et $\bar{n}_{l+1} = n_{k+1}$.
$\beta_l = (W_l^T V_l)^{-1} W_l^T A p_{n_{k+1}-1}$.
$\beta_0^{(l)} = \dfrac{(q_{n_k}, Ap_{n_{k+1}-1})}{(w_{\bar{n}_{l-1}}, v_{\bar{n}_l-1})}$.
$v_{\bar{n}_{l+1}} = Ap_{n_{k+1}-1} - V_l\beta_l - \beta_0^{(l)} v_{\bar{n}_{l-1}}$,
$w_{\bar{n}_{l+1}} = A^T q_{n_{k+1}-1} - W_l\beta_l - \beta_0^{(l)} w_{\bar{n}_{l-1}}$.

Sinon

$m_k = 1, \quad n_{k+1} = \bar{n}_{l+1}$.
$\gamma_0^{(l)} = \dfrac{(q_{n_k}, Ap_{n_k})}{(w_{\bar{n}_l}, v_{\bar{n}_{l+1}-1})}$.
$v_{\bar{n}_{l+1}} = Ap_{n_k} - \gamma_0^{(l)} v_{\bar{n}_l}$.
$w_{\bar{n}_{l+1}} = A^T q_{n_k} - \gamma_0^{(l)} w_{\bar{n}_l}$.

Fin(si)

5. $V_{l+1} = v_{\bar{n}_{l+1}}, \qquad W_{l+1} = w_{\bar{n}_{l+1}}.$
$\lambda_0^{(k+1)} = \dfrac{(w_{\bar{n}_{l+1}}, v_{\bar{n}_{l+1}})}{(q_{n_k}, Ap_{n_{k+1}-1})}.$
$p_{n_{k+1}} = v_{\bar{n}_{l+1}} - \lambda_0^{(k+1)} p_{n_k}.$
$q_{n_{k+1}} = w_{\bar{n}_{l+1}} - \lambda_0^{(k+1)} q_{n_k}.$
$P_{k+1} = p_{n_{k+1}}, \qquad Q_{k+1} = q_{n_{k+1}}.$
$l = l + 1.$
$k = k + 1\ .$

Fin(Tant que).

Dans cet algorithme, nous résolvons des systèmes d'équations avec les matrices $W_l^T V_l$ et $Q_k^T A P_k$. Nous avons, à partir des relations (1.65), le résultat suivant

$$(w_{\bar{n}_{l+1}-1}, v_{\bar{n}_l}) = (w_{\bar{n}_{l+1}-2}, v_{\bar{n}_l+1}) = \cdots = (w_{\bar{n}_l}, v_{\bar{n}_{l+1}-1}) = d_l \neq 0.$$

Cela prouve que la matrice $W_l^T V_l = D_l$ de dimension $\bar{m}_l \times \bar{m}_l$ est inversible et qu'elle est de la forme

$$D_l = \begin{bmatrix} 0 & \cdots & 0 & d_l \\ \vdots & \iddots & \iddots & * \\ 0 & \iddots & \iddots & \vdots \\ d_l & * & \cdots & * \end{bmatrix}.$$

D'autre part, les relations (1.57) nous donnent

$$(q_{n_{k+1}-1}, Ap_{n_k}) = (q_{n_{k+1}-2}, Ap_{n_k+1}) = \cdots = (q_{n_k}, Ap_{n_{k+1}-1}) = e_k \neq 0,$$

Ainsi, la matrice $Q_k^T A P_k = E_k$ de dimension $m_k \times m_k$ est donc inversible et a la forme suivante

$$E_k = \begin{bmatrix} 0 & \cdots & 0 & e_k \\ \vdots & \iddots & \iddots & * \\ 0 & \iddots & \iddots & \vdots \\ e_k & * & \cdots & * \end{bmatrix}.$$

Notons $V^{(n)}$ et $W^{(n)}$ les deux matrices de dimension $N \times (n+1)$ qui regroupent les vecteurs réguliers et complémentaires comme suit

$$V^{(n)} = [V_0\ V_1\ \cdots\ V_l] \qquad \text{et} \qquad W^{(n)} = [W_0\ W_1\ \cdots\ W_l],$$

tels que, pour tout $i = 0, \ldots, l$ nous ayons

$$V_i = \begin{cases} [v_{\bar{n}_i}\ v_{\bar{n}_i+1}\ \cdots\ v_{\bar{n}_{l+1}-1}], & \text{si} \quad 0 \leq i < l, \\ [v_{\bar{n}_l}\ v_{\bar{n}_l+1}\ \cdots\ v_n], & \text{si} \quad i = l, \end{cases}$$

et

$$W_i = \begin{cases} [w_{\tilde{n}_i}\ w_{\tilde{n}_i+1}\ \cdots\ w_{\tilde{n}_{i+1}-1}], & \text{si}\quad 0 \le i < l, \\ [w_{\tilde{n}_l}\ w_{\tilde{n}_l+1}\ \cdots\ w_n], & \text{si}\quad i = l, \end{cases}$$

avec $l = l(n)$ l'unique entier égal à l'indice du dernier vecteur régulier précédant v_n, (*i.e* $\tilde{n}_l \le n < \tilde{n}_{l+1}$). Les relations (1.52) et (1.60) nous permettent d'écrire

$$W^{(n)^T} V^{(n)} = \begin{bmatrix} D_0 & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & D_l \end{bmatrix}.$$

Cela prouve que les vecteurs $(v_i)_{i=0}^n$ et $(w_i)_{i=0}^n$ sont biorthogonaux par blocs.
De façon similaire, nous notons $P^{(n)}$ et $Q^{(n)}$ les deux matrices de dimension $N \times (n+1)$ qui regroupent les vecteurs réguliers et complémentaires p_n et q_n comme suit

$$P^{(n)} = [P_0\ P_1\ \cdots\ P_k] \qquad \text{et} \qquad Q^{(n)} = [Q_0\ Q_1\ \cdots\ Q_k],$$

tels que, pour tout $j = 0, \ldots, k$ nous ayons

$$P_j = \begin{cases} [p_{n_j}\ p_{n_j+1}\ \cdots\ p_{n_{j+1}-1}], & \text{si}\quad 0 \le j < k, \\ [p_{n_k}\ p_{n_k+1}\ \cdots\ p_n], & \text{si}\quad j = k, \end{cases}$$

et

$$Q_j = \begin{cases} [q_{n_j}\ q_{n_j+1}\ \cdots\ q_{n_{j+1}-1}], & \text{si}\quad 0 \le j < k, \\ [q_{n_k}\ w_{n_k+1}\ \cdots\ w_n], & \text{si}\quad j = k, \end{cases}$$

avec $k = k(n)$ l'unique entier égal à l'indice du dernier vecteur régulier précédant p_n, c'est-à-dire $n_k \le n < n_{k+1}$.
Les vecteurs $(p_i)_{i=0}^n$ et $(q_i)_{i=0}^n$ sont A-biorthogonaux par bloc. En effet, grâce aux relations (1.52) et (1.60), nous avons

$$Q^{(n)^T} A P^{(n)} = \begin{bmatrix} E_0 & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & E_k \end{bmatrix}.$$

Pour une itération n quelconque, les relations de récurrence données dans l'algorithme 1.3.4 nous permettent de retrouver les mêmes relations que (1.46) que l'on réécrit

$$\begin{aligned} V^{(n)} &= P^{(n)} U_n, & AP^{(n)} &= V^{(n+1)} L_n, \\ W^{(n)} &= Q^{(n)} U_n, & A^T Q^{(n)} &= W^{(n+1)} L_n, \end{aligned} \tag{1.84}$$

avec U_n une matrice carrée triangulaire supérieure de dimension $(n+1) \times (n+1)$ et L_n matrice de Hessenberg supérieure de dimension $(n+2) \times (n+1)$.
En éliminant P_n dans (1.84), nous obtenons

$$AV^{(n)} = V^{(n+1)} L_n U_n, \tag{1.85}$$

et en comparant ce résultat avec (1.38), nous obtenons

$$H_e^{(n)} = L_n U_n. \tag{1.86}$$

Les matrices L_n et U_n représentent une décomposition de la matrice tridiagonale par bloc $H_e^{(n)}$ générée par le processus de tridiagonalisation de Lanczos avec *look-ahead.*
Nous pouvons récupérer la matrice tridiagonale par bloc $H^{(n)}$, obtenue par le processus de Lanczos avec *look-ahead*, à partir du produit $\bar{L}_n U_n$ avec $\bar{L}_n$ est la matrice de dimension $(n+1) \times (n+1)$ qui vérifie

$$L_n = \begin{bmatrix} & \bar{L}_n & & \\ 0 & \cdots & 0 & 1 \end{bmatrix}.$$

1.4 Résultats numériques

Dans cette section, nous donnons quelques résultats numériques que nous avons obtenu en utilisant le processus de Lanczos dans le calcul des valeurs propres d'une matrice. Nous comparons les résultats obtenus par le processus classique de Lanczos avec ceux obtenus par le processus de Lanczos par des récurrences à deux termes.

Considérons une matrice A donnée de dimension $N \times N$ et $(\lambda_i)_{1 \leq i \leq N}$ ses valeurs propres. Nous appliquons chacun des deux processus de Lanczos pour obtenir la matrice $H^{(N)}$, et nous calculons les valeurs propres $(\mu_i)_{1 \leq i \leq N}$ de cette matrice par la commande `eig`.
Ici, nous cherchons à associer chaque valeur propre de A à une seule valeur propre de $H^{(N)}$. Pour cela, nous appliquons la procédure suivante :
Nous construisons d'abord la matrice D de dimension $N \times N$ avec les coefficients $d_{i,j} = |\lambda_i - \mu_j|$ pour $i, j = 1, \ldots, N$. Ensuite, nous localisons la distance minimale dans cette matrice que nous notons $d_{i(1),j(1)}$, cette valeur représente l'erreur des deux valeurs propres associées $\lambda_{i(1)}$ et $\mu_{j(1)}$. Ces deux valeurs propres sont, ensuite, supprimées de la liste en prenant dans la matrice D, $d_{i(1),k} = d_{k,j(1)} = \infty$ pour $k = 1, \ldots, N$.
Cette opération est répétée jusqu'au dernier couple $(\lambda_{i(N)}, \mu_{j(N)})$. Nous obtenons ainsi les valeurs $d_{min} = d_{i(1),j(1)}$ et $d_{max} = d_{i(N),j(N)}$ qui représentent respectivement la plus petite et la plus grande erreur des valeurs propres.
Un problème peut survenir dans cette procédure lorsque deux distances $d_{i,j}$ sont égales. Or, cette situation est peu probable à cause des erreurs d'arrondi.

Pour détecter le *breakdown* dans les deux processus avec *look-ahead*, nous allons considérer qu'une quantité est nulle si sa valeur absolue est inférieure à $\epsilon = 10^{-10}$.
Nos tests ont été réalisés à l'aide du logiciel MATLAB (version 7).

Exemple 1.4.1

Prenons A une matrice triangulaire supérieure de dimension $N \times N$ ayant les valeurs propres

$$\lambda_i = \{ \text{ approximation de } 10(1 - 1/i) \text{ avec quatre décimales } \}, \quad \text{pour} \quad i = 1, \ldots, N.$$

Les autres éléments de la matrice A sont choisis dans $[-10, 10]$ d'une manière aléatoire. Nous appliquons les deux processus de Lanczos avec $v_0 = w_0 = (1, \ldots, 1)^T$. Les tableaux 1.1 et 1.2 représentent seulement les valeurs propres réelles significatives de la matrice $H^{(N)}$ obtenue par chaque algorithme. Nous remarquons qu'en utilisant le processus de Lanczos

valeur propre exacte	Lanczos classique	Lanczos par des récurrences à 2 termes
0.0000	$0.12038744596415 \ 10^{-7}$	$0.58323124108028 \ 10^{-9}$
5.0000	3.91144739590432	4.81960575212682
7.5000	-	7.62145556671008
8.3333	8.30202135210024	-
8.7500	-	8.74763314560834
9.0000	-	9.15367632952191
d_{max}	4.6347	2.0908
d_{min}	$0.1204 \ 10^{-7}$	$0.5832 \ 10^{-9}$

TAB. 1.1 - N=100

valeur propre exacte	Lanczos classique	Lanczos par des récurrences à 2 termes
0.0000	-	$- 0.41652809015366 \ 10^{-6}$
6.6666	-	6.64310823100008
7.5000	7.67434889653407	7.43474807606191
9.3333	-	9.33400112048892
d_{max}	5.5466	4.36199
d_{min}	$0.6361 \ 10^{-4}$	$0.4156 \ 10^{-6}$

TAB. 1.2 - N=200

par des récurrences à deux termes nous obtenons les meilleures approximations des valeurs

propres λ_i . En plus, nous obtenons plus de valeurs propres réelles qu'avec le processus classique de Lanczos. Cependant, un problème de dépassement de capacité peut affecter les deux processus de Lanczos lorsque nous prenons $N = 200$. Une normalisation des vecteurs des deux algorithmes, en utilisant la norme euclidienne, permet d'éviter ce problème.

Exemple 1.4.2

Dans cet exemple, nous appliquons les deux processus de Lanczos avec *look-ahead* à des matrices qui donnent le *breakdown*. Les valeurs propres de ces matrices sont calculées par la commande `eig`. Bien sur, ici nous supposons que les valeurs propres des matrices sont calculées correctement par la commande `eig`. C'est pourquoi nous donnons aussi la norme infini du vecteur obtenu par la commande `eigsens` qui représente une mesure de la sensibilité des valeurs propres par rapport à la perturbation de la matrice.

Nous commençons par prendre la matrice

$$A = \begin{pmatrix} 0 & 1 & & & & & \\ -1 & 0 & 1 & & & & \\ & -1 & 0 & 1 & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & \ddots & \ddots & \ddots & \\ & & & & -1 & 0 & 1 \\ & & & & & -1 & 0 \end{pmatrix}$$

Nous choisissons $v_0 = w_0 \in \mathbb{R}^N$ d'une manière aléatoire avec la commande *rand*. Un *breakdown* survient à chaque itération impaire dans le processus de Lanczos par des récurrences à deux termes, il s'agit du problème de *true-breakdown*. Un saut $m_{2k} = 2$ permet au processus de Lanczos (par des récurrences à deux termes avec *look-ahead*) d'éviter ce problème.

Le processus classique de Lanczos ne subit aucun *breakdown*. Cependant, les matrices $H^{(2k+1)}$ calculées à chaque itération impaire, par cet algorithme, sont singulières. Cela explique le problème du *true-breakdown* qui se produit dans le premier algorithme et qui est dû à la décomposition LU de cette matrice. Nous obtenons les résultats donnés dans le tableau 1.3.

Considérons maintenant la matrice

$$A = \begin{pmatrix} 0 & 0 & 0 & \cdots & 0 & -1 \\ 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 0 \end{pmatrix}.$$

N	Lanczos classique	Lanczos par des récurrences à 2 termes	$\|eigsens(A)\|_\infty$
100	$d_{min} = 4.8107\ 10^{-18}$ $d_{max} = 8.2164\ 10^{-15}$	$d_{min} = 0$ $d_{max} = 5.1071\ 10^{-15}$	1.0000
200	$d_{min} = 4.3863\ 10^{-18}$ $d_{max} = 9.1038\ 10^{-15}$	$d_{min} = 1.0957\ 10^{-18}$ $d_{max} = 8.1601\ 10^{-15}$	1.0000
400	$d_{min} = 2.6469\ 10^{-23}$ $d_{max} = 2.6470\ 10^{-15}$	$d_{min} = 0$ $d_{max} = 1.0881\ 10^{-15}$	1.0000

TAB. 1.3 -

En prenant $v_0 = (-N, 1, 2, 3, \ldots, N-1)^T$ et $w_0 = (1, \ldots, 1)^T$, il se produit dans les deux algorithmes un *breakdown* à la quatrième itération. Le processus classique de Lanczos avec *look-ahead* détecte un *ghost-breakdown* et fait un saut de longueur $\bar{m}_3 = N-5$ pour calculer à l'itération $\bar{n}_4 = N-2$ les vecteurs réguliers $v_{\bar{n}_4}$ et $w_{\bar{n}_4}$. Ce résultat est obtenu aussi par le processus de Lanczos par des récurrences à deux termes avec *look-ahead* qui détecte à la fois le *true* et le *ghost-breakdown* et fait un saut de longueur égale à $m_3 = N-6$ pour d'abord calculer les vecteurs p_{n_4} et q_{n_4} à l'itération $n_4 = N-3$ et ensuite il calcul les vecteurs réguliers $v_{\bar{n}_4}$ et $w_{\bar{n}_4}$ à l'itération $n_4 + 1 = N-2$. Nous retrouvons ici la situation représentée dans la figure 1.3. Les résultats obtenus sont donnés dans le tableau 1.4.

N	Lanczos classique	Lanczos par des récurrences à 2 termes	$\|eigsens(A)\|_\infty$
100	$d_{min} = 2.0088\ \ 10^{-11}$ $d_{max} = 5.4771\ \ 10^{-6}$	$d_{min} = 9.1208\ 10^{-12}$ $d_{max} = 1.1476\ 10^{-6}$	1.0000
200	$d_{min} = 1.0371\ 10^{-8}$ $d_{max} = 0.0158$	$d_{min} = 4.0512\ 10^{-9}$ $d_{max} = 0.0162$	1.0000
400	$d_{min} = 6.3266\ 10^{-10}$ $d_{max} = 0.6022$	$d_{min} = 3.5274\ 10^{-11}$ $d_{max} = 0.0035$	1.0000

TAB. 1.4 -

N	Lanczos classique	Lanczos par des récurrences à 2 termes	$\|eigsens(A)\|_\infty$
100	$d_{min} = 8.5687\ 10^{-12}$ $d_{max} = 2.0766$	$d_{min} = 8.0242\ 10^{-12}$ $d_{max} = 0.4629$	$6.7520\ 10^{+7}$
200	$d_{min} = 3.6677\ 10^{-9}$ $d_{max} = 0.1745$	$d_{min} = 3.1317\ 10^{-9}$ $d_{max} = 0.1176$	$2.3956\ 10^{+14}$
400	$d_{min} = 2.6732\ 10^{-3}$ $d_{max} = 0.1866$	$d_{min} = 3.2169\ 10^{-4}$ $d_{max} = 0.1896$	$6.1139\ 10^{+14}$

TAB. 1.5 –

Enfin, nous prenons la matrice

$$\begin{pmatrix} 2 & 1 & & & & & & \\ 0 & 2 & 1 & & & & & \\ 1 & 0 & 2 & 1 & & & & \\ & \ddots & \ddots & \ddots & \ddots & & & \\ & & \ddots & \ddots & \ddots & \ddots & & \\ & & & 1 & 0 & 2 & 1 & \\ & & & & 1 & 0 & 2 & \end{pmatrix}.$$

Pour $v_0 = (1, 0, \ldots, 0)^T$ et $w_0 = (-1, -1, -1, 0 \ldots, 0)^T$, les deux processus de Lanczos avec *look-ahead* détectent un *breakdown* à la première itération, il s'agit d'un *ghost-breakdown* pour le processus classique qui fait un saut de longueur égal à $\bar{m}_0 = 3$ pour calculer les vecteurs réguliers $v_{\bar{n}_1}$ et $w_{\bar{n}_1}$ à l'itération $\bar{n}_1 = 3$ alors que le processus de Lanczos par des récurrence à deux termes détecte un *ghost* et *true-breakdown* et fait un saut $m_0 = 2$ pour calculer les vecteurs p_{n_1} et q_{n_1} à l'itération $n_1 = 2$ et ensuite il calcule les vecteurs réguliers $v_{\bar{n}_1}$ et $w_{\bar{n}_1}$ à l'itération $\bar{n}_1 = 3$. Comme pour la matrice précédente, nous somme dans le cas présenté dans la figure 1.3. Les résultats obtenus sont donnés dans le tableau 1.5.

1.5 Conclusion

Dans ce chapitre, nous avons étudié le problème du *breakdown* qui risque de se produire dans le calcul des polynômes orthogonaux formels. En particulier, ceux utilisés dans la mise en œuvre de la méthode de Lanczos. Un schéma représentant les différents cas de *breakdown* possibles a été donné. Compte-tenu de ces résultats, nous avons proposé un formalisme qui permet de généraliser toutes les relations de récurrence dans le cas d'un éventuel *breakdown*. Cela nous a permis, d'abord, de retrouver le processus classique de Lanczos avec *look-ahead*, ensuite, d'éviter les deux types de *breakdown* susceptibles de ce produire dans le processus de Lanczos construit par des récurrences à deux termes. Une

deuxième application de ce formalisme et des résultats obtenus sera donnée dans le prochain chapitre, elle consiste à éviter les problèmes de *breakdown* qui affectent l'algorithme du Gradient Biconjuguée.

Les résultats numériques qui ont été établis dans le présent chapitre confirment le résultat déjà connu, à savoir que les algorithmes basés sur des récurrences à deux termes sont plus stables numériquement que ceux qui sont basés sur des récurrences à trois termes. Ce résultat reste vraie même dans le cas des relations de récurrence avec *look-ahead.*

Chapitre 2

Problème du breakdown dans la méthode de Lanczos

La méthode de Lanczos [50, 51] est l'une des méthodes les plus utilisées pour la résolution des systèmes d'équations linéaires. Cette méthode est mise en œuvre en utilisant différentes relations de récurrence satisfaites par les familles adjacentes des polynômes orthogonaux formels [19]. Ce qui a donné les méthodes de type Lanczos.
Les divisions par zéro, susceptibles de se produire dans la construction de ces polynômes orthogonaux, génèrent des situations de *breakdown* dans la méthode de Lanczos.
Il existe plusieurs algorithmes, à base de *look-ahead*, qui apportent des solutions aux problèmes de *breakdown* dans la méthode de Lanczos. Nous allons citer quatre d'entre elles en donnant leur interprétation polynômiale et nous y introduisons un préconditionneur.
Le Gradient Biconjugué [29], connu aussi sous le nom de Lanczos/Orthomin [47], est une extension du Gradient conjugué aux matrices non-symétriques. Dans cet algorithme, nous construisons les vecteurs de direction à partir de récurrence à deux termes, ce qui le rend plus stable que les autres algorithmes de type Lanczos. Malheureusement, il risque de s'y produire deux types de *breakdown*; le premier n'est autre que le *ghost-breakdown* qui affecte le processus de Lanczos, le deuxième se produit lorsque les vecteurs d'itérations ne sont pas définis par les conditions de Galerkin ce qui se traduit par la singularité de la matrice tridiagonale de Lanczos. Il s'agit ici du *true-breakdown* qui intervient, comme nous l'avons cité dans le premier chapitre, dans la décomposition LU de la matrice de Lanczos, décomposition qui est à la base de l'algorithme du Gradient Biconjugué.
Parmi les algorithmes qui proposent des solutions au problème de *breakdown* dans le Gradient Biconjugué, nous avons le BMRZ [15, 17] et le CSBCG [21, 22] qui traitent uniquement le cas du *true-breakdown*, le QMR [35] est une version du Gradient Biconjugué qui utilise la quasi-minimisation du résidu pour éviter le problème du *true-breakdown*, et qui résout le *ghost-breakdown* en appliquant le *look-ahead* au processus de Lanczos.

Ici, nous allons utiliser les résultats du premier chapitre, en particulier le rapport entre le *true* et le *ghost-breakdown*, afin d'appliquer le *look-ahead* au Gradient Biconjugué et ainsi lui éviter ces deux types de *breakdown*.

Ce chapitre est organisé comme suite :

Au paragraphe 2.1 nous rappelons brièvement la méthode de Lanczos. En 2.1.1 nous donnons son approche matricielle. En 2.1.2 nous donnons son approche polynômiale. En 2.1.3 nous expliquons le problème du *breakdown* qui peut affecter les méthodes de type Lanczos.

Au paragraphe 2.2 nous appliquons le *look-ahead* au Gradient Biconjugué en utilisant le formalisme proposé dans le premier chapitre ainsi que les résultats obtenus sur le rapport entre le *true* et le *ghost-breakdown*.

Au paragraphe 2.3 nous utilisons les polynômes orthogonaux formels pour introduire le préconditionneur dans quelques méthodes de type Lanczos qui utilisent le *look-ahead*. En 2.3.1 nous étudions la méthode CSBCG. En 2.3.2 nous étudions la méthode QMR avec *look-ahead*. En 2.3.3 nous nous intéressons à la méthode BIORES non générique. En 2.3.4 nous traitons les algorithmes MRZ-stab et HMRZ-stab.

Dans le dernier paragraphe, nous comparons les méthodes étudiées dans ce chapitre, dans le cas d'un *breakdown*, a travers des tests numériques.

2.1 Méthode de Lanczos

Considérons le système linéaire

$$Ax = b, \tag{2.1}$$

où $A \in \mathbb{R}^{N\times N}$, $b \in \mathbb{R}^N$ et $x \in \mathbb{R}^N$.

La méthode de Lanczos consiste à construire une suite de vecteurs (x_k) de la façon suivante :

- choisir deux vecteurs arbitraires x_0 et $y \neq 0$ dans $\mathbb{R}^n$,
- calculer $r_0 = b - Ax_0$,
- déterminer x_k tel que

$$\begin{aligned} x_k - x_0 \in \mathcal{K}_k(A, r_0) &= span(r_0, Ar_0, \ldots, A^{k-1}r_0) \\ r_k = b - Ax_k \perp \mathcal{K}_k(A^T, y) &= span(y, A^T y, \ldots, A^{T^{k-1}} y). \end{aligned} \tag{2.2}$$

Nous obtenons ainsi une méthode itérative qui fournit (en théorie, c'est à dire s'il n'y avait pas d'erreurs dues à l'arithmétique de l'ordinateur) la solution exacte x en N itérations au

plus.
A partir de (2.2), $x_k - x_0$ peut s'écrire sous la forme suivante

$$x_k - x_0 = -a_1 r_0 - \cdots - a_k A^{k-1} r_0,$$

ce qui donne

$$r_k = r_0 + a_1 A r_0 + \cdots + a_k A^k r_0. \tag{2.3}$$

La condition d'orthogonalité dans (2.2) s'écrit

$$(y, A^i r_k) = 0, \quad \text{pour } i = 0, \ldots, k-1, \tag{2.4}$$

et nous obtenons le système d'équations linéaires

$$a_1(y, A^{i+1} r_0) + \cdots + a_k(y, A^{i+k} r_0) = -(y, A^i r_0), \text{ pour } i = 0, \ldots, k-1. \tag{2.5}$$

Si le déterminant du système (2.5) est différent de zéro, alors x_k existe et est déterminé de façon unique par les relations (2.2).
Bien sûr, calculer x_k en résolvant pour tout k le système qui donne $a_1, a_2, \ldots, a_k$ n'est pas faisable en pratique ; c'est pourquoi nous allons utiliser les polynômes orthogonaux formels pour obtenir des algorithmes récursifs qui permettent de calculer x_k.
Nous allons donner dans les deux sous-sections suivantes les deux approches, matricielle et polynômiale, de la méthode de Lanczos.

2.1.1 Approche matricielle

Soit V_k une matrice dont les colonnes sont les vecteurs $r_0, Ar_0, \ldots, A^{k-1} r_0$. D'après (2.2), $x_k - x_0$ peut s'écrire sous la forme d'une combinaison de cette base, ainsi

$$x_k - x_0 = -V_k a,$$

avec $a = (a_1, a_2, \ldots, a_k)^T$, le vecteur a est bien la solution du système (2.5).
A partir de la définition du résidu, nous obtenons

$$r_k = r_0 + A V_k a.$$

Afin de calculer l'expression de a, nous prenons la matrice $W_k = [y, A^T y, \ldots, (A^T)^{k-1} y]$.
Les conditions d'orthogonalité (2.2) peuvent s'écrire $W_k^T r_k = 0$, ce qui nous donne

$$W_k^T r_0 + W_k^T A V_k a = 0,$$

alors

$$a = -[W_k^T A V_k]^{-1} W_k^T r_0.$$

Donc

$$r_k = \mathbf{P_k} r_0 = (I - \mathbf{Q_k}) r_0$$

avec

$$\mathbf{Q_k} = AV_k [W_k^T AV_k]^{-1} W_k^T = \mathbf{Q_k}^2.$$

Nous remarquons que $\mathbf{Q_k}$ et $\mathbf{P_k}$ représentent deux projections obliques.
En particulier, si $W_k = AV_k$ alors $\mathbf{Q_k} = \mathbf{Q_k}^T$. Dans ce cas $\mathbf{Q_k}$ et $\mathbf{P_k}$ représentent deux projections orthogonales.
Notons $H_k = W_k^T AV_k$. Cette matrice est la matrice du système (2.5), elle est identique à la matrice de Hankel $M_k^{(1)}$ donnée dans le chapitre 1.
Prenons $d = W_k^T r_0$; alors nous avons

$$a = -H_k^{-1} d.$$

Bien sûr, ceci est possible seulement lorsque la matrice H_k est non-singulière. Alors, nous pouvons calculer x_k à partir de la relation

$$x_k = x_0 - V_k H_k^{-1} d.$$

2.1.2 Approche polynômiale

Considérons le polynôme

$$P_k(\xi) = 1 + a_1 \xi + \cdots + a_k \xi^k = 1 + \xi R_{k-1}(\xi).$$

avec $R_{k-1}(\xi) = a_1 + \cdots + a_k \xi^{k-1}$ et $(a_1, \ldots, a_k)$ solution du système (2.5).
D'après la relation (2.3), nous avons

$$r_k = P_k(A) r_0. \tag{2.6}$$

Si nous définissons la fonctionnelle linéaire c sur l'espace des polynômes par

$$c(\xi^i) = c_i = (y, A^i r_0) \quad \text{pour} \quad i = 1, 2, \ldots \tag{2.7}$$

alors, pour tout polynôme R

$$c(R(\xi)) = (y, R(A) r_0).$$

Les conditions d'orthogonalité (2.4) peuvent donc s'écrire sous la forme

$$c(\xi^i P_k(\xi)) = 0 \quad \text{pour } i = 0, \ldots, k-1. \tag{2.8}$$

Ces conditions montrent que P_k est le polynôme de degré au plus k de la famille des polynômes orthogonaux formels par rapport à la fonctionnelle linéaire c.

Ces polynômes sont normalisés par la condition $P_k(0) = 1$, alors ils peuvent s'écrire sous la forme

$$P_k(\xi) = \begin{vmatrix} 1 & \xi & \cdots & \xi^k \\ c_0 & c_1 & \cdots & c_k \\ \vdots & \vdots & & \vdots \\ c_{k-1} & c_k & \cdots & c_{2k-1} \end{vmatrix} \Bigg/ \begin{vmatrix} c_1 & \cdots & c_k \\ \vdots & & \vdots \\ c_k & \cdots & c_{2k-1} \end{vmatrix}.$$

Comme nous l'avons déjà vu dans le chapitre 1, ces polynômes existent si et seulement si $det(H_k) = H_k^{(1)} \neq 0$. Alors dans ce cas

$$r_k = P_k(A)r_0 = \begin{vmatrix} r_0 & Ar_0 & \cdots & A^k r_0 \\ c_0 & c_1 & \cdots & c_k \\ \vdots & \vdots & & \vdots \\ c_{k-1} & c_k & \cdots & c_{2k-1} \end{vmatrix} \Bigg/ \begin{vmatrix} c_1 & \cdots & c_k \\ \vdots & & \vdots \\ c_k & \cdots & c_{2k-1} \end{vmatrix}.$$

Les polynômes P_k peuvent être calculés par des relations de récurrence. Ces différentes relations conduisent aux différentes méthodes de types Lanczos : *gradient conjugué, Orthores, Orthodir, Orthomin, BIORES, BIODIR, BICG,...*
Toutes ces variantes possèdent la même propriété de convergence finie, à savoir que $\exists k \leq N$ (N dimension du système à résoudre) tel que $r_k = 0$, c'est-à-dire $x_k = x = A^{-1}b$.

2.1.3 Breakdown dans les méthodes de type Lanczos

Les méthodes de type Lanczos [50, 51] présentent un intérêt pratique uniquement lorsque nous pouvons calculer récursivement les polynômes P_k, ainsi que r_k et x_k. Le premier algorithme pour cela est le Gradient Conjugué (CG) dû à Hestenes et Stiefel [44]. Cet algorithme est appliqué seulement dans le cas où la matrice A est symétrique définie positive. Dans ce cas l'algorithme est toujours bien défini. Une extension du CG pour des matrices quelconques a été donnée par Fletcher [29], il s'agit de l'algorithme du Gradient Biconjugué (BICG). D'autres algorithmes pour mettre en œuvre la méthode de Lanczos peuvent être dérivés en utilisant différentes relations de récurrence satisfaites par le polynôme P_k [19]. Les trois algorithmes les plus utilisés sont donnés dans le tableau 2.1.
Dans toutes les méthodes de type Lanczos interviennent des coefficients avec un produit scalaire comme dénominateur. Si l'un de ces produits scalaires est nul nous avons un *breakdown* dans l'algorithme (division par zéro), dans ce cas l'algorithme doit être arrêté. Si un produit scalaire dans un dénominateur est voisin de zéro, il se produit ce que l'on appelle un *near-breakdown* qui provoque une propagation importante d'erreurs numériques (une quantité voisine de zéro provient de la différence de deux nombres voisins ce qui provoque une erreur de cancellation).

TAB. 2.1 –

nom de l'algorithme	récurrences utilisées
Lanczos/Orthodir	$P_{k+1} \longleftarrow P_k, P_k^{(1)}$ $P_{k+1}^{(1)} \longleftarrow P_k^{(1)}, P_{k-1}^{(1)}$
Lanczos/Orthores	$P_{k+1} \longleftarrow P_k, P_{k-1}$
Lanczos/Orthomin	$P_{k+1} \longleftarrow P_k, Q_k$ $Q_{k+1} \longleftarrow P_{k+1}, Q_k$

Q_k est le $k^{ème}$ polynôme orthogonal par rapport à la fonctionnelle $c^{(1)}$ ayant le même coefficient du terme de degré k que P_k.

Nous distinguons deux types de *breakdown* :

- Le *true-breakdown* qui correspond à l'inexistence des polynômes P_k et $P_k^{(1)}$,
- Le *ghost-breakdown* dû au fait que le polynôme P_k n'est pas de degré exactement k, ce qui est équivalent à l'inexistence de $P_k^{(0)}$.

Pour remédier à cette situation, plusieurs stratégies ont été élaborées; Dans la section 2.3, nous nous intéressons à quatre d'entre elles : HMRZ-stab qui est une variante de (Method of Recursive Zoom), (Non Generic BIORES), CSBG (Composite Steps Biconjugate Gradient) et QMR (Quasi Minimal Residual). Nous proposons ici l'interprétation polynômiale de ces méthodes, les polynômes orthogonaux formels vont nous permettre d'introduire un préconditionneur dans ces algorithmes d'une manière très simple.
Dans la section suivante, nous donnons un nouvel algorithme qui est une version du Gradient Biconjugué sans *breakdown*, nous utilisons la technique du *look-ahead* pour éviter les situations du *ghost* et *true-breakdown* en se basant sur les résultats établies dans le chapitre 1.

2.2 Traitement du breakdown dans le Gradient Biconjugué

Dans cette section, nous nous intéressons aux problèmes de *breakdown* qui peuvent affecter l'algorithme du Gradient Biconjugué. Cet algorithme a été cité pour la première fois par Lanczos [50, 51]. Ensuite il a été écrit sous forme d'algorithme par Fletcher[29]. Il est connu aussi sous le nom de Lanczos/Orthomin [81]. Ici, nous proposons une variante de l'algorithme Gradient Biconjugué dans laquelle nous évitons le *ghost* et le *true-breakdown* qui risquent de s'y produire. Nous utilisons pour cela le formalisme proposé dans le premier chapitre ainsi que les résultats obtenus sur le rapport entre ces deux types de *breakdown*.

2.2.1 L'algorithme du Gradient Biconjugué

Cet algorithme est mis en œuvre à partir des relations de récurrence

$$\begin{aligned} P_{k+1}(\xi) &= P_k(\xi) - \alpha_{k+1}\xi Q_k(\xi), \\ Q_{k+1}(\xi) &= P_{k+1}(\xi) + \beta_{k+1}Q_k(\xi), \end{aligned} \tag{2.9}$$

avec $P_0(\xi) = Q_0(\xi) = 1$.
Le polynôme Q_k vérifie

$$Q_k(\xi) = (-1)^k \frac{H_k^{(0)}}{H_k^{(1)}} P_k^{(1)}(\xi).$$

En supposant que $H_k^{(1)} \neq 0$ et $H_k^{(0)} \neq 0$, les deux polynômes P_k et Q_k existent $\forall k$, et sont de degré exactement égal à k. Le polynôme Q_k est ainsi le $k^{ème}$ polynôme orthogonal par rapport à la fonctionnelle $c^{(1)}$ ayant le même coefficient du terme de degré k que P_k.
D'autre part, d'après sa définition, le polynôme $P_k^{(0)}$ vérifie

$$P_k^{(0)}(0) = (-1)^k \frac{H_k^{(0)}}{H_k^{(1)}}.$$

Alors, les polynômes Q_k peuvent être calculés par la relation

$$Q_k(\xi) = \frac{P_k^{(1)}(\xi)}{P_k^{(0)}(0)}. \tag{2.10}$$

Ces polynômes vérifient les conditions d'orthogonalité

$$c(\xi^i Q_k(\xi)) = 0 \quad \text{pour} \quad i = 1, \ldots, k-1. \tag{2.11}$$

Grâce aux conditions (2.8) et (2.11), les coefficients α_{k+1} et β_{k+1} vérifient les relations

$$\begin{aligned} \alpha_{k+1} &= c(P_k Q_k)/c(\xi Q_k Q_k), \\ \beta_{k+1} &= -c(\xi P_{k+1} Q_k)/c(\xi Q_k^2). \end{aligned}$$

Comme Q_k et P_k ont le même coefficient de degré k, $Q_k = P_k + q_{k-1}$ avec q_{k-1} un polynôme de degré au plus égal à $k-1$, alors

$$c(P_k Q_k) = c(P_k^2) + c(P_k q_{k-1}) = c(P_k^2),$$

nous en déduisons

$$\alpha_{k+1} = \frac{c(P_k^2)}{c(\xi Q_k^2)}.$$

D'autre part, nous avons $c(P_{k+1}^2) = -\alpha_{k+1} c(\xi P_{k+1} Q_k)$, alors

$$\begin{aligned} \beta_{k+1} &= \frac{c(P_{k+1}^2)}{\alpha_{k+1} c(\xi Q_k^2)} \\ &= \frac{c(P_{k+1}^2)}{c(P_k^2)}. \end{aligned}$$

Nous remarquons qu'à l'itération $k+1$, nous risquons d'avoir deux divisions par zéro :
- Le *ghost-breakdown*, dans le cas où $c(P_k^2) = 0 \Leftrightarrow H_{k+1}^{(0)} = 0$.
- Le *true-breakdown*, dans le cas où $c(\xi Q_k Q_k) \Leftrightarrow H_{k+1}^{(1)} = 0$.

Si nous prenons

$$\begin{aligned} r_k &= P_k(A)r_0, & \tilde{r}_k &= P_k(A^T)y, \\ z_k &= Q_k(A)r_0, & \tilde{z}_k &= Q_k(A^T)y, \end{aligned}$$

les récurrences (2.9) deviennent

$$\begin{aligned} r_{k+1} &= r_k - \alpha_{k+1}Az_k & z_{k+1} &= r_{k+1} + \beta_{k+1}z_k \\ \tilde{r}_{k+1} &= \tilde{r}_k - \alpha_{k+1}A^T\tilde{z}_k & \tilde{z}_{k+1} &= \tilde{r}_{k+1} + \beta_{k+1}\tilde{z}_k, \end{aligned}$$

et en utilisant (2.7), les coefficient α_{k+1} et β_{k+1} s'écrivent

$$\alpha_{k+1} = \frac{(\tilde{r}_k, r_k)}{(\tilde{z}_k, Az_k)}, \qquad \beta_{k+1} = \frac{(\tilde{r}_{k+1}, r_{k+1})}{(\tilde{r}_k, r_k)}.$$

En rassemblant ces formules nous obtenons l'algorithme du BICG.

Algorithme 2.2.1 : *Gradient Biconjugé "BICG"*

1. Initialisation : Choisir x_0 et $y \in \mathbb{R}^N$, $r_0 = b - Ax_0$.
   $z_0 = r_0$, $\tilde{z}_0 = \tilde{r}_0 = y$
   $\rho_0 = (\tilde{r}_0, r_0)$
   $k = 0$
2. Tant que $\|r_k\| \neq 0$
   $w_k = A^T p_k$
   $\alpha_{k+1} = \rho_k/(\tilde{z}_k, Az_k)$
   $x_{k+1} = x_k + \alpha_{k+1}z_k$
   $r_{k+1} = r_k - \alpha_{k+1}Az_k$
   $\tilde{r}_{k+1} = \tilde{r}_k - \alpha_{k+1}A^T\tilde{z}_k$
   $\rho_{k+1} = (\tilde{r}_{k+1}, r_{k+1})$
   $\beta_{k+1} = \rho_{k+1}/\rho_k$
   $z_{k+1} = r_{k+1} + \beta_{k+1}z_k$
   $\tilde{z}_{k+1} = \tilde{r}_{k+1} + \beta_{k+1}\tilde{z}_k$
   $k = k+1$
   Fin.

Dans le cas symétrique $A = A^T$, si nous prenons $y = r_0$ alors nous retrouvons l'algorithme du Gradient Conjuguée [44].

Dans L'algorithme du BICG nous pouvons résoudre simultanément le système dual $A^T x^* = b^*$ en prenant $y = \tilde{r}_0 = b^* - A^T x_0^*$ avec x_0^* une approximation initiale de x^*, il suffit de rajouter dans l'algorithme (2.2.1) la récurrence $x_{k+1}^* = x_k^* + \alpha_{k+1}\tilde{z}_k$
Il est possible de mettre en œuvre l'algorithme du BICG en évitant les produits par la matrice A^T et en utilisant une méthode de Lanczos de type produit. Il suffit pour cela de prendre $r_k = U_k(A)P_k(A)r_0$. Le choix $U_k = P_k$ nous donne l'algorithme du CGS dû à Sonneveld [73]. Van der Vorst [77] utilise $U_k(\xi) = (1 - a_k\xi)U_k(\xi)$ avec $U_0(\xi) = 1$, le paramètre a_k étant choisi de façon à minimiser la norme du résidu. D'autres choix ont permis à Brezinski et Redivo-Zaglia [11] et à Gutknecht [43] d'introduire les méthodes LTPM (Lanczos-type product methods).
Comme nous l'avons déjà signalé dans la sous-section 1.3.4, l'algorithme du BICG peut être obtenu directement à partir de l'algorithme 1.3.2 (*processus de Lanczos par des relations de récurrence à deux termes*) ; il suffit pour cela de prendre $v_0 = r_0$ et $w_0 = y$. Nous obtenons donc les vecteurs bi-orthogonaux $(v_0, \ldots, v_k)$ et $(w_0, \ldots, w_k)$, les vecteurs A bi-conjugués $(p_0, \ldots, p_k)$ et $(q_0, \ldots, q_k)$ et une décomposition LU de la matrice tridiagonale de Lanczos. Les résidus r_k et $\tilde{r}_k$ ont les mêmes directions que les vecteurs v_k et w_k, respectivement.
La méthode BICG présente l'avantage de posséder une récurrence courte. Par contre, elle requiert à chaque itération deux produits matrice-vecteur, par A et A^T. De plus elle peut rencontrer un problème de *ghost-breakdown* dû à la singularité de la matrice de Lanczos. La factorisation LU (sans pivotage) de cette dernière n'est pas toujours possible, ce qui risque de provoquer le *true-breakdown*. Ce problème a pu être évité par Bank et Chan [22] en faisant une décomposition LU par bloc 2×2 suivant les éléments de la diagonale sous la condition qu'il ne se produit pas de *ghost-breakdown*. La méthode du BMRZ [13, 17], proposé par Brezinski, Redivo-Zaglia et Sadok, est une version du BiCG sans *true-brekdown*, mais qui reste sous la menace d'un *ghost-breakdown*. Les résultats trouvés dans le chapitre 1, nous donnent le rapport entre ces deux types de breakdown, ce qui va nous permettre, dans la suite de ce chapitre, d'appliquer le look-ahead au Gradient Biconjugé d'une façon à éviter à la fois le *true* et le *ghost-breakdown*.

2.2.2 Formules de récurrence

Dans cette partie, nous allons appliquer la stratégie du *look-ahead* afin de construire des relations de récurrence sans *breakdown* qui généralisent les récurrences (2.9), le schéma donné dans la figure (1.1) nous donne les différents cas de *breakdown* possibles pour ce type de relation.
Gardons les mêmes notations que celles données dans la sous-section (1.2.2).
Nous notons par $0 = n_0 < n_1 < n_2 < \cdots < n_k < \cdots$ les degrés des polynômes $P_{n_k}^{(1)}$ qui existent (i.e $H_{n_k}^{(1)} \neq 0$).
Notons m_k la longueur du saut qui existe entre $P_{n_k}^{(1)}$ et $P_{n_{k+1}}^{(1)}$, i.e $n_{k+1} = n_k + m_k$.

D'après le Théorème 1.1.1, m_k est défini par les conditions

$$\begin{aligned} c^{(1)}(\xi^i P_{n_k}^{(1)}) &= 0 \quad \text{pour} \quad i = 0, \ldots, n_k + m_k - 2, \\ &\neq 0 \quad \text{pour} \quad i = n_k + m_k - 1. \end{aligned} \tag{2.12}$$

Notons par $0 = \bar{n}_0 < \bar{n}_1 < \bar{n}_2 < \cdots < \bar{n}_l < \cdots$ les indices des polynômes unitaires réguliers $P_{\bar{n}_l}^{(0)}$ qui existent (i.e $H_{\bar{n}_l}^{(0)} \neq 0$). Ces polynômes vérifient les relations suivantes

$$\begin{aligned} c(\xi^i P_{\bar{n}_l}^{(0)}) &= 0 \quad \text{pour} \quad i = 0, \ldots, \bar{n}_l + \bar{m}_l - 2, \\ &\neq 0 \quad \text{pour} \quad i = \bar{n}_l + \bar{m}_l - 1 \end{aligned} \tag{2.13}$$

où $\bar{m}_l$ est le saut entre $P_{\bar{n}_l}^{(0)}$ et $P_{\bar{n}_{l+1}}^{(0)}$, $\bar{n}_{l+1} = \bar{n}_l + \bar{m}_l$.
Dans le cas où le polynôme P_{n_k} est de degré exactement égal à n_k, il existe $l \in \mathbb{N}$ tel que $n_k = \bar{n}_l$, dans ce cas $H_{n_k}^{(0)} = H_{\bar{n}_l}^{(0)} \neq 0$ et $H_{\bar{n}_l}^{(1)} = H_{n_k}^{(1)} \neq 0$.

Nous commençons par supposer qu'à une itération n_k donnée, le polynôme régulier P_{n_k} est de degré exactement n_k. Dans ce cas le polynôme Q_{n_k} existe et il est degré égal à n_k, ce qui implique qu'il existe $l \in \mathbb{N}$ tel que $n_k = \bar{n}_l$. De plus $P_{\bar{n}_l}^{(0)}(0) \neq 0$.
Les relations d'orthogonalité (2.12) sont équivalentes à

$$\begin{aligned} c^{(1)}(\xi^i Q_{n_k}) = 0 \quad &\text{pour} \quad i = 0, \ldots, n_k + m_k - 2 \\ \neq 0 \quad &\text{pour} \quad i = n_k + m_k - 1 \end{aligned} \tag{2.14}$$

► Si $c(P_{n_k} P_{n_k}) \neq 0$, alors d'après le Lemme 1.1.4, nous avons $H_{n_k+1}^{(0)} \neq 0$.
Compte tenu du schéma établi dans la figure (1.1), nous savons que $H_{n_{k+1}}^{(0)} \neq 0$. Ainsi, à l'itération n_{k+1}, les deux polynômes réguliers $P_{n_{k+1}}$ et $Q_{n_{k+1}}$ sont de degré exactement n_{k+1}.
Considérons la base suivante

$$\{P_{\bar{n}_0}^{(0)}, \xi P_{\bar{n}_0}^{(0)}, \ldots, \xi^{\bar{m}_{l-1}-1} P_{\bar{n}_{l-1}}^{(0)}, P_{n_k}, \xi Q_{n_k}, \ldots, \xi^{m_k} Q_{n_k}\}.$$

En exprimant le polynôme $P_{n_{k+1}}$ dans cette base, nous obtenons

$$\begin{aligned} P_{n_{k+1}} = \; & \beta_{m_k}^{(k)} \xi^{m_k} Q_{n_k} - \cdots\cdots\cdots - \beta_1^{(k)} \xi Q_{n_k} - \beta_0^{(k)} P_{n_k} \\ & - \gamma_{\bar{m}_{l-1}-1}^{(l-1)} \xi^{\bar{m}_{l-1}-1} P_{\bar{n}_{l-1}}^{(0)} - \cdots - \gamma_1^{(l-1)} \xi P_{\bar{n}_{l-1}}^{(0)} - \gamma_0^{(l-1)} P_{\bar{n}_{l-1}}^{(0)} \\ & \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ & - \gamma_{m_0-1}^{(0)} \xi^{m_0-1} P_{\bar{n}_0}^{(0)} - \cdots\cdots\cdots - \gamma_1^{(0)} \xi P_{\bar{n}_0}^{(0)} - \gamma_0^{(0)} P_{\bar{n}_0}^{(0)}, \end{aligned}$$

Les relations d'orthogonalité (1.16) et (2.14) nous donnent

$$\begin{aligned} \gamma_i^{(j)} = 0, \qquad \text{pour} \qquad & j = 0, \ldots, l-1 \\ & i = 0, \ldots, \bar{m}_j - 1. \end{aligned}$$

La condition de normalisation $P_{n_{k+1}}(0) = 1$ donne $\beta_0^{(k)} = -1$ ce qui nous permet d'écrire

$$P_{n_{k+1}} = P_{n_k} - \beta_1^{(k)} \xi Q_{n_k} - \cdots\cdots - \beta_{m_k}^{(k)} \xi^{m_k} Q_{n_k}. \tag{2.15}$$

Pour calculer les coefficients figurant dans cette relation, nous la multiplions par les polynômes $\xi^j Q_{n_k}$ pour $j = 0, \ldots, m_k - 1$ et nous appliquons la fonctionnelle c. Grâce aux relations d'orthogonalité (2.8) et (2.14), nous obtenons le système suivant

$$\begin{array}{rcl} 0 & = & \beta_{m_k-1}^{(k)} c^{(1)}(\xi^{m_k-1} Q_{n_k} Q_{n_k}) + \beta_{m_k}^{(k)} c^{(1)}(\xi^{m_k} Q_{n_k} Q_{n_k}) \\ \cdots & \cdots & \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ 0 & = & \beta_1^{(k)} c^{(1)}(\xi^{m_k-1} Q_{n_k} Q_{n_k}) + \cdots + \beta_{m_k}^{(k)} c^{(1)}(\xi^{2m_k-2} Q_{n_k} Q_{n_k}) \end{array} \tag{2.16}$$

et

$$c(Q_{n_k} P_{n_k}) = \beta_{m_k}^{(k)} c^{(1)}(\xi^{m_k-1} Q_{n_k} Q_{n_k}).$$

Les polynômes P_{n_k} et Q_{n_k} sont de même degré, et il existe un polynôme $\tilde{q}$ de degré au plus $n_k - 1$ tel que

$$P_{n_k} = Q_{n_k} + \tilde{q}$$

ce qui nous permet d'avoir

$$c(P_{n_k} P_{n_k}) = c(Q_{n_k} P_{n_k}),$$

et

$$\beta_{m_k}^{(k)} = \frac{c(P_{n_k} P_{n_k})}{c^{(1)}(\xi^{m_k-1} Q_{n_k} Q_{n_k})}. \tag{2.17}$$

Enfin, nous exprimons le polynôme $Q_{n_{k+1}} - P_{n_{k+1}}$ de degré $n_{k+1} - 1$ dans la base

$$\{P_{n_0}^{(1)}, \xi P_{n_1}^{(1)}, \ldots, \xi^{m_{k-1}-1} P_{n_{k-1}}^{(1)}, Q_{n_k}, \xi Q_{n_k}, \ldots, \xi^{m_k-1} Q_{n_k}\},$$

ce qui nous donne

$$\begin{array}{rcl} Q_{n_{k+1}} - P_{n_{k+1}} & = & \lambda_0^{(0)} P_{n_0}^{(1)} + \lambda_1^{(0)} \xi P_{n_0}^{(1)} + \cdots + \lambda_{m_0-1}^{(0)} \xi^{m_0-1} P_{n_0}^{(1)} \\ & & + \lambda_0^{(1)} P_{n_1}^{(1)} + \lambda_1^{(1)} \xi P_{n_1}^{(1)} + \cdots + \lambda_{m_1-1}^{(1)} \xi^{m_1-1} P_{n_1}^{(1)} \\ & & \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ & & + \lambda_0^{(k)} Q_{n_k} + \lambda_1^{(k)} \xi Q_{n_k} + \cdots + \lambda_{m_k-1}^{(k)} \xi^{m_k-1} Q_{n_k}. \end{array}$$

Compte tenu des relations d'orthogonalité (2.14), nous pouvons écrire

$$Q_{n_{k+1}} = P_{n_{k+1}} + \lambda_0^{(k)} Q_{n_k} \tag{2.18}$$

avec

$$-c(\xi^{m_k}Q_{n_k}P_{n_{k+1}}) = \lambda_0^{(k)}c^{(1)}(\xi^{m_k-1}Q_{n_k}Q_{n_k}).$$

À partir de (2.15), nous avons

$$c(P_{n_{k+1}}P_{n_{k+1}}) = -\beta_{m_k}^{(k)}c(\xi^{m_k}Q_{n_k}P_{n_{k+1}})$$

et ainsi

$$\lambda_0^{(k)} = \frac{c(P_{n_{k+1}}P_{n_{k+1}})}{\beta_{m_k}^{(k)}c^{(1)}(\xi^{m_k-1}Q_{n_k}Q_{n_k})}.$$

En tenant compte de l'expression de $\beta_{m_k}^{(k)}$ donné dans (2.17), nous obtenons

$$\lambda_0^{(k)} = \frac{c(P_{n_{k+1}}P_{n_{k+1}})}{c(P_{n_k}P_{n_k})}. \tag{2.19}$$

► Si $c(P_{n_k}P_{n_k}) = 0$ alors, d'après le Lemme 1.1.4, nous avons $H_{n_k+1} = 0$. En plus d'après le schéma donné dans la figure (1.1)

$$H_{n_k+i}^{(0)} = 0 \qquad \text{pour} \qquad i = 1, \ldots, m_k,$$

où m_k est le saut donné par les relations (2.14).
Il en résulte l'inexistence des polynômes $P_{n_k+i}^{(0)}$ pour $i = 1, \ldots, m_k$, et par conséquent :

- le polynôme $P_{n_{k+1}}$ n'est pas de degré n_{k+1} et il vérifie

$$P_{n_{k+1}} = P_{n_k}.$$

- Le polynôme $Q_{n_{k+1}}$ n'existe pas.

Le calcul du polynôme $Q_{n_{k+1}}$ étant impossible, nous le remplaçons par le polynôme intermédiaire

$$\bar{Q}_{n_{k+1}} = \frac{P_{n_{k+1}}^{(1)}}{P_{n_k}^{(0)}(0)}.$$

Ce polynôme a le même coefficient du terme de plus haut degré que le polynôme $P_{n_{k+1}}$.
Ces polynômes intermédiaires vont nous permettre de compléter la base des polynômes réguliers $\{Q_{n_k}\}$. Il est important de noter que ces polynômes vérifient aussi les conditions d'orthogonalité (2.21).
Pour donner la relation qui nous permet de calculer le polynôme $\bar{Q}_{n_{k+1}}$, nous commençons par exprimer $P_{n_{k+1}}^{(1)}$ dans la base

$$\{P_{n_0}^{(1)}, \xi P_{n_0}^{(1)}, \ldots, P_{n_{k-1}}^{(1)}, \ldots, P_{n_k}^{(1)}, \xi P_{n_k}^{(1)}, \ldots, \xi^{m_k}P_{n_k}^{(1)}\}.$$

Nous obtenons

$$P^{(1)}_{n_{k+1}} = \xi^{m_k} P^{(1)}_{n_k} - \alpha^{(k)}_{m_k-1}\xi^{m_k-1}P^{(1)}_{n_k} - \cdots - \alpha^{(k)}_1 \xi P^{(1)}_{n_k} - \alpha^{(k)}_0 P^{(1)}_{n_k} - \cdots - \alpha^{(k-1)}_0 P^{(1)}_{n_{k-1}} - \cdots - \alpha^{(0)}_0 P^{(1)}_{n_0}.$$

À partir des relations (2.14), nous avons

$$\alpha^{(j)}_i = 0, \quad \text{pour} \quad \begin{matrix} j &=& 0,\ldots,k-2, \\ i &=& 0,\ldots,m_j-1 \end{matrix} \qquad \alpha^{(k-1)}_i = 0, \quad \text{pour} \quad i = 1,\ldots,m_{k-1}-1,$$

et ainsi

$$P^{(1)}_{n_{k+1}} = \xi^{m_k} P^{(1)}_{n_k} - \alpha^{(k)}_{m_k-1}\xi^{m_k-1}P^{(1)}_{n_k} - \cdots - \alpha^{(k)}_1 \xi P^{(1)}_{n_k} - \alpha^{(k)}_0 P^{(1)}_{n_k} - \alpha^{(k-1)}_0 P^{(1)}_{n_{k-1}}.$$

En divisant tous les termes de cette relation par $P^{(0)}_{n_k}(0)$, nous obtenons

$$\bar{Q}_{n_{k+1}} = \xi^{m_k} Q_{n_k} - \alpha^{(k)}_{m_k-1}\xi^{m_k-1}Q_{n_k} - \cdots - \alpha^{(k)}_1 \xi Q_{n_k} - \alpha^{(k)}_0 Q_{n_k} - \beta_0 Q_p, \qquad (2.20)$$

Q_p correspond au polynôme régulier, sinon intermédiaire, de degré n_{k-1}. En effet, nous avons

$$\beta_0 = \alpha^{(k-1)}_0 \frac{P^{(0)}_{n_f}(0)}{P^{(0)}_{n_k}(0)} \qquad \text{et} \qquad Q_p = \frac{P^{(1)}_{n_{k-1}}}{P^{(0)}_{n_f}(0)}.$$

si $P^{(0)}_{n_{k-1}}(0) \neq 0$ alors $n_f = n_{k-1}$ et $Q_p = Q_{n_{k-1}}$,

sinon $n_f = n_{k-2}$ et $Q_p = \bar{Q}_{n_{k-1}} = \dfrac{P^{(1)}_{n_{k-1}}}{P^{(0)}_{n_{k-2}}(0)}$.

Pour le calcul des coefficients $(\alpha^{(k)}_i)_{0\leq i\leq m_k-1}$, nous multiplions la relation (2.20) par les polynômes $\xi^i Q_{n_k}$ pour $i = 0,\ldots,m_k-1$, et en appliquant la fonctionnelle $c^{(1)}$, nous obtenons grâce aux relations (2.14) le système suivant

$$\begin{aligned}
c^{(1)}(\xi^{m_k}Q_{n_k}Q_{n_k}) &= \alpha^{(k)}_{m_k-1}c^{(1)}(\xi^{m_k-1}Q_{n_k}Q_{n_k}) \\
c^{(1)}(\xi^{m_k+1}Q_{n_k}Q_{n_k}) &= \alpha^{(k)}_{m_k-2}c^{(1)}(\xi^{m_k-1}Q_{n_k}Q_{n_k}) + \alpha^{(k)}_{m_k-1}c^{(1)}(\xi^{m_k}Q_{n_k}Q_{n_k}) \\
\cdots\cdots\cdots\cdots &\quad \cdots \quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \qquad (2.21) \\
c^{(1)}(\xi^{2m_k-1}Q_{n_k}Q_{n_k}) &= \alpha^{(k)}_0 c^{(1)}(\xi^{m_k-1}Q_{n_k}Q_{n_k}) + \alpha^{(k)}_1 c^{(1)}(\xi^{m_k}Q_{n_k}Q_{n_k}) + \cdots \\
&\quad + \alpha^{(k)}_{m_k-2}c^{(1)}(\xi^{2m_k-3}Q_{n_k}Q_{n_k}) + \alpha^{(k)}_{m_k-1}c^{(1)}(\xi^{2m_k-2}Q_{n_k}Q_{n_k}).
\end{aligned}$$

De la même manière, nous multiplions la relation (2.20) par $\xi^{m_{k-1}-1}\tilde{Q}_p$ et nous appliquons les orthogonalités suivantes

$$\begin{matrix} c^{(1)}(\xi^i Q_p) = 0 & \text{pour} & i = 0,\ldots,n_k-2 \\ \neq 0 & \text{pour} & i = n_k-1 \end{matrix}$$

pour obtenir

$$c^{(1)}(\xi^{m_k+m_{k-1}-1}Q_{n_k}Q_p) = \beta_0 c^{(1)}(\xi^{m_{k-1}-1}Q_pQ_p). \qquad (2.22)$$

Ici, nous cherchons à établir une généralisation des relations (2.9). À l'itération n_{k+1}, nous avons l'existence du polynôme $P_{n_{k+1}}$ qui n'est autre que le polynôme P_{n_k}, ainsi que le polynôme intermédiaire $\bar{Q}_{n_{k+1}}$ qui remplace le polynôme inexistant $Q_{n_{k+1}}$. Ceci n'est pas suffisant puisqu'il nous faut trouver des polynômes réguliers ayant le même degré. Compte-tenu des résultats donnés dans le Théorème 1.2.3, nous savons que $H^{(0)}_{n_{k+2}} \neq 0$, donc il nous suffit de faire un deuxième saut m_{k+1} afin de trouver les polynômes réguliers $Q_{n_{k+2}}$ et $P_{n_{k+2}}$ de degré n_{k+2}. Le saut m_{k+1} peut être déterminé par les conditions

$$\begin{aligned} c^{(1)}(\xi^i \bar{Q}_{n_{k+1}}) &= 0 \quad \text{pour} \quad i = 0, \ldots, n_{k+1} + m_{k+1} - 2 \\ &\neq 0 \quad \text{pour} \quad i = n_{k+1} + m_{k+1} - 1. \end{aligned}$$

Dans ce qui va suivre, nous allons donner les relations qui permettent de calculer les polynômes $Q_{n_{k+2}}$ et $P_{n_{k+2}}$.

Pour cela, nous commençons par exprimer le polynôme $P_{n_{k+2}}$ dans la base

$$\{P^{(0)}_{\bar{n}_0}, \xi P^{(0)}_{\bar{n}_0}, \ldots, \xi^{\bar{m}_{l-1}-1} P^{(0)}_{\bar{n}_{l-1}}, P_{n_k}, \xi Q_{n_k}, \ldots, \xi^{m_k} Q_{n_k}, \xi \bar{Q}_{n_{k+1}}, \ldots, \xi^{m_{k+1}} \bar{Q}_{n_{k+1}}\}$$

où les $P^{(0)}_{\bar{n}_i}$ sont les polynômes unitaires définis dans la sous-section 1.2.2.

Nous avons la relation

$$\begin{aligned} P_{n_{k+2}} = \; & \beta^{(k+1)}_{m_{k+1}} \xi^{m_{k+1}} \bar{Q}_{n_{k+1}} - \cdots - \beta^{(k+1)}_1 \xi \bar{Q}_{n_{k+1}} - \beta^{(k)}_{m_k} \xi^{m_k} Q_{n_k} - \cdots - \beta^{(k)}_1 \xi Q_{n_k} - \beta^{(k)}_0 P_{n_k} \\ & - \gamma^{(l-1)}_{\bar{m}_{l-1}-1} \xi^{\bar{m}_{l-1}-1} P^{(0)}_{\bar{n}_{l-1}} - \cdots - \gamma^{(l-1)}_0 P^{(0)}_{\bar{n}_{l-1}} \cdots\cdots\cdots \\ & - \gamma^{(0)}_{m_0-1} \xi^{m_0-1} P^{(0)}_{\bar{n}_0} - \cdots - \gamma^{(0)}_1 \xi P^{(0)}_{\bar{n}_0} - \gamma^{(0)}_0 P^{(0)}_{\bar{n}_0}. \end{aligned}$$

En utilisant les orthogonalités (1.16) et (2.14), nous obtenons

$$\gamma^{(j)}_i = 0, \quad \text{pour} \quad \begin{matrix} j = 0, \ldots, l-1, \\ i = 0, \ldots, \bar{m}_j - 1. \end{matrix} \quad \text{et} \quad \beta^{(k)}_i = 0, \quad \text{pour} \quad i = 1, \ldots, m_k$$

La condition de normalisation $P_{n_{k+2}}(0) = 1$ nous donne $\beta^{(k)}_0 = -1$. En plus nous avons $P_{n_{k+1}} = P_{n_k}$, ce qui nous permet d'écrire

$$P_{n_{k+2}} = P_{n_{k+1}} - \beta^{(k+1)}_1 \xi \bar{Q}_{n_{k+1}} - \cdots - \beta^{(k+1)}_{m_{k+1}} \xi^{m_{k+1}} \bar{Q}_{n_{k+1}}. \tag{2.23}$$

Nous multiplions cette relation par les polynômes $\xi^j \bar{Q}_{n_{k+1}}$ pour $j = 0, \ldots, m_{k+1} - 1$ et nous appliquons la fonctionnelle c. Grâce aux relations d'orthogonalité, nous obtenons le système suivant

$$\begin{aligned} c(\bar{Q}_{n_{k+1}} P_{n_{k+1}}) &= \beta^{(k+1)}_{m_{k+1}} c^{(1)}(\xi^{m_{k+1}-1} \bar{Q}_{n_{k+1}} \bar{Q}_{n_{k+1}}) \\ 0 &= \beta^{(k+1)}_{m_{k+1}-1} c^{(1)}(\xi^{m_{k+1}-1} \bar{Q}_{n_{k+1}} \bar{Q}_{n_{k+1}}) + \beta^{(k+1)}_{m_{k+1}} c^{(1)}(\xi^{m_{k+1}} \bar{Q}_{n_{k+1}} \bar{Q}_{n_{k+1}}) \\ \cdots \quad &\cdots \quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ 0 &= \beta^{(k+1)}_1 c^{(1)}(\xi^{m_{k+1}-1} \bar{Q}_{n_{k+1}} \bar{Q}_{n_{k+1}}) + \cdots + \beta^{(k+1)}_{m_{k+1}} c^{(1)}(\xi^{2m_{k+1}-2} \bar{Q}_{n_{k+1}} \bar{Q}_{n_{k+1}}). \end{aligned} \tag{2.24}$$

Maintenant, nous considérons la base des polynômes de degré au plus $n_{k+2}-1$

$$\{P_{n_0}^{(1)}, \xi P_{n_1}^{(1)}, \ldots, \xi^{m_{k-1}-1} P_{n_{k-1}}^{(1)}, Q_{n_k}, \xi Q_{n_k}, \ldots, \xi^{m_k-1} Q_{n_k}, \bar{Q}_{n_{k+1}}, \ldots, \xi^{m_{k+1}-1} \bar{Q}_{n_{k+1}}\}.$$

Ainsi, en exprimant le polynôme $Q_{n_{k+2}} - P_{n_{k+2}}$ (de degré $n_{k+2}-1$) dans cette base, nous obtenons

$$\begin{aligned} Q_{n_{k+2}} - P_{n_{k+2}} &= \lambda_0^{(0)} P_{n_0}^{(1)} + \lambda_1^{(0)} \xi P_{n_0}^{(1)} + \cdots + \lambda_{m_0-1}^{(0)} \xi^{m_0-1} P_{n_0}^{(1)} \\ &\quad + \lambda_0^{(1)} P_{n_1}^{(1)} + \lambda_1^{(1)} \xi P_{n_1}^{(1)} + \cdots + \lambda_{m_1-1}^{(1)} \xi^{m_1-1} P_{n_1}^{(1)} \\ &\quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ &\quad + \lambda_0^{(k+1)} \bar{Q}_{n_{k+1}} + \lambda_1^{(k+1)} \xi \bar{Q}_{n_{k+1}} + \cdots + \lambda_{m_{k+1}-1}^{(k+1)} \xi^{m_{k+1}-1} \bar{Q}_{n_{k+1}}, \end{aligned}$$

et, compte-tenu des relations d'orthogonalité (2.14), nous trouvons

$$Q_{n_{k+2}} = P_{n_{k+2}} + \lambda_0^{(k+1)} \bar{Q}_{n_{k+1}} \tag{2.25}$$

avec

$$-c(\xi^{m_{k+1}} \bar{Q}_{n_{k+1}} P_{n_{k+2}}) = \lambda_0^{(k+1)} c^{(1)}(\xi^{m_{k+1}-1} \bar{Q}_{n_{k+1}} \bar{Q}_{n_{k+1}}).$$

À partir de la relation (2.23), nous avons

$$c(P_{n_{k+2}} P_{n_{k+2}}) = -\beta_{m_{k+1}}^{(k+1)} c(\xi^{m_{k+1}} \bar{Q}_{n_{k+1}} P_{n_{k+2}}),$$

alors

$$\lambda_0^{(k+1)} = \frac{c(P_{n_{k+2}} P_{n_{k+2}})}{\beta_{m_{k+1}}^{(k+1)} c^{(1)}(\xi^{m_{k+1}-1} \bar{Q}_{n_{k+1}} \bar{Q}_{n_{k+1}})}$$

et en tenant compte de l'expression de $\beta_{m_{k+1}}^{(k+1)}$ donné dans la première relation de (2.24), nous obtenons

$$\lambda_0^{(k+1)} = \frac{c(P_{n_{k+2}} P_{n_{k+2}})}{c(\bar{Q}_{n_{k+1}} P_{n_{k+1}})}. \tag{2.26}$$

Reprenons l'expression de β_0 donnée par (2.22). Dans le cas où $Q_p = Q_{n_{k-1}}$, nous avons

$$\beta_0 = \frac{c^{(1)}(\xi^{m_k+m_{k-1}-1} Q_{n_k} Q_{n_{k-1}})}{c^{(1)}(\xi^{m_{k-1}-1} Q_{n_{k-1}} Q_{n_{k-1}})}$$

et les polynômes P_{n_k} et Q_{n_k} sont calculés par les relations

$$\begin{aligned} P_{n_k} &= P_{n_{k-1}} - \beta_1^{(k-1)} \xi Q_{n_{k-1}} - \cdots\cdots\cdots - \beta_{m_{k-1}}^{(k-1)} \xi^{m_{k-1}} Q_{n_{k-1}} \\ Q_{n_k} &= P_{n_k} + \lambda_0^{(k-1)} Q_{n_{k-1}}. \end{aligned}$$

À partir des conditions d'orthogonalité (2.8) et (2.14), nous pouvons écrire

$$c^{(1)}(\xi^{m_k-1}Q_{n_k}Q_{n_k}) = c^{(1)}(\xi^{m_k-1}Q_{n_k}P_{n_k}) = \beta_{m_{k-1}}^{(k-1)}c^{(1)}(\xi^{m_k+m_{k-1}-1}Q_{n_k}Q_{n_{k-1}}).$$

Comme pour (2.17), le coefficient $\beta_{m_{k-1}}^{(k-1)}$ est donné par

$$\beta_{m_{k-1}}^{(k-1)} = \frac{c(P_{n_{k-1}}P_{n_{k-1}})}{c^{(1)}(\xi^{m_{k-1}-1}Q_{n_{k-1}}Q_{n_{k-1}})},$$

et l'expression du coefficient β_0 devient

$$\beta_0 = \frac{c^{(1)}(\xi^{m_k-1}Q_{n_k}Q_{n_k})}{c(P_{n_{k-1}}P_{n_{k-1}})}. \tag{2.27}$$

De façon similaire, nous pouvons montrer que, dans le cas où $Q_p = \bar{Q}_{n_{k-1}}$, nous avons

$$\beta_0 = \frac{c^{(1)}(\xi^{m_k-1}Q_{n_k}Q_{n_k})}{c(\bar{Q}_{n_{k-1}}P_{n_{k-1}})}. \tag{2.28}$$

Remarque 2.2.1

- *Pour obtenir les relations de récurrence précédentes, nous avons complété l'ensemble des polynômes réguliers par des polynômes complémentaires de la forme* $\xi^i Q_{n_j}^{(1)}$ *ou* $\xi^i \bar{Q}_{n_j}^{(1)}$.
- *Le ghost et le true-breakdown affectent les relations de récurrence selon le schéma donné dans figure (1.1). Il en résulte trois situations. Ici, nous avons traité deux cas seulement puisque le troisième n'est qu'un cas particulier du premier, il suffit que le saut* m_{k+1} *soit égal à* 1.
- *Le calcul du polynôme intermédiaire* $\bar{Q}_{n_{k+1}}$ *nécessite le stockage du polynôme "intermédiaire ou régulier" existant à l'itération* n_{k-1}.

Afin de mettre en œuvre l'algorithme du BICG avec *look-ahead*, nous posons

$$r_k = P_k(A)r_0, \qquad \tilde{r}_k = P_k(A^T)y,$$
$$z_k = Q_k(A)r_0, \qquad \tilde{z}_k = Q_k(A^T)y.$$

Lorsque les polynômes réguliers inexistants Q_k sont remplacés par les polynômes intermédiaires $\bar{Q}_k$, les vecteurs de direction z_k et $\tilde{z}_k$ sont déterminés par

$$z_k = \bar{Q}_k(A)r_0, \qquad \tilde{z}_k = \bar{Q}_k(A^T)y.$$

Nous posons

$$b_i^{(k)} = (\tilde{z}_{n_k}, A^{m_k+i}z_{n_k}).$$

► Dans le cas où $(\tilde{r}_{n_k}, r_{n_k}) \neq 0$, les relations (2.15) et (2.18) deviennent

$$\begin{aligned} r_{n_{k+1}} &= r_{n_k} - \beta_1^{(k)} A z_{n_k} - \cdots - \beta_{m_k}^{(k)} A^{m_k} z_{n_k}, \\ \tilde{r}_{n_{k+1}} &= \tilde{r}_{n_k} - \beta_1^{(k)} (A^T) \tilde{z}_{n_k} - \cdots - \beta_{m_k}^{(k)} (A^T)^{m_k} \tilde{z}_{n_k}, \\ z_{n_{k+1}} &= r_{n_{k+1}} + \lambda_0^{(k)} z_{n_k}, \\ \tilde{z}_{n_{k+1}} &= \tilde{r}_{n_{k+1}} + \lambda_0^{(k)} \tilde{z}_{n_k}. \end{aligned}$$

Les coefficients figurant dans ces relations sont déterminés grâce à (2.16), (2.17) et (2.19) par

$$\left\{ \begin{array}{l} \beta_{m_k}^{(k)} b_0^{(k)} = (\tilde{r}_{n_k}, r_{n_k}) \\ \beta_{m_k-1}^{(k)} b_0^{(k)} + \beta_{m_k}^{(k)} b_1^{(k)} = 0 \\ \cdots\cdots\cdots\cdots\cdots\cdots \\ \beta_1^{(k)} b_0^{(k)} + \cdots\cdots + \beta_{m_k}^{(k)} b_{m_k-1}^{(k)} = 0 \end{array} \right. \tag{2.29}$$

et

$$\lambda_0^{(k)} = \frac{(\tilde{r}_{n_{k+1}}, r_{n_{k+1}})}{(\tilde{r}_{n_k}, r_{n_k})}.$$

► Dans le cas où $(\tilde{r}_{n_k}, r_{n_k}) = 0$, nous avons

$$r_{n_{k+1}} = r_{n_k} \qquad \text{et} \qquad \tilde{r}_{n_{k+1}} = \tilde{r}_{n_k}$$

et les relations (2.20) deviennent

$$\begin{aligned} z_{n_{k+1}} &= A^{m_k} z_{n_k} - \alpha_{m_k-1}^{(k)} A^{m_k-1} z_{n_k} - \cdots - \alpha_0^{(k)} z_{n_k} - \beta_0 z_{n_{k-1}}, \\ \tilde{z}_{n_{k+1}} &= (A^T)^{m_k} \tilde{z}_{n_k} - \alpha_{m_k-1}^{(k)} (A^T)^{m_k-1} \tilde{z}_{n_k} - \cdots - \alpha_0^{(k)} \tilde{z}_{n_k} - \beta_0 \tilde{z}_{n_{k-1}}. \end{aligned}$$

Les coefficients $\alpha_i^{(k)}$, pour $i = 0, \ldots, m_k - 1$, sont solution du système linéaire

$$\left\{ \begin{array}{l} \alpha_{m_k-1}^{(k)} b_0^{(k)} = b_1^{(k)} \\ \alpha_{m_k-2}^{(k)} b_0^{(k)} + \alpha_{m_k-1}^{(k)} b_1^{(k)} = b_2^{(k)} \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ \alpha_0^{(k)} b_0^{(k)} + \cdots\cdots\cdots + \alpha_{m_k-1}^{(k)} b_{m_k-1}^{(k)} = b_{m_k}^{(k)}. \end{array} \right. \tag{2.30}$$

Les relations (2.27) et (2.28) nous donnent l'expression du coefficient β_0 ;

$$\beta_0 = \frac{b_0^{(k)}}{(\tilde{r}_{n_{k-1}}, r_{n_{k-1}})} \qquad \text{si} \qquad (\tilde{r}_{n_{k-1}}, r_{n_{k-1}}) \neq 0$$

$$\beta_0 = \frac{b_0^{(k)}}{(\tilde{z}_{n_{k-1}}, r_{n_{k-1}})} \qquad \text{si} \qquad (\tilde{r}_{n_{k-1}}, r_{n_{k-1}}) = 0.$$

Les relations (2.23) et (2.25) conduisent aux récurrences

$$\begin{aligned} r_{n_{k+2}} &= r_{n_k} - \beta_1^{(k+1)} A z_{n_{k+1}} - \cdots - \beta_{m_{k+1}}^{(k+1)} A^{m_{k+1}} z_{n_{k+1}}. \\ \tilde{r}_{n_{k+2}} &= \tilde{r}_{n_k} - \beta_1^{(k+1)} (A^T) \tilde{z}_{n_{k+1}} - \cdots - \beta_{m_{k+1}}^{(k+1)} (A^T)^{m_{k+1}} \tilde{z}_{n_{k+1}}. \end{aligned}$$

$$\begin{aligned} z_{n_{k+2}} &= r_{n_{k+2}} + \lambda_0^{(k+1)} z_{n_{k+1}} \\ \tilde{z}_{n_{k+2}} &= \tilde{r}_{n_{k+2}} + \lambda_0^{(k+1)} \tilde{z}_{n_{k+1}} \end{aligned}$$

Les coefficients figurant dans ces relations sont déterminés grâce à (2.24) et (2.26) par

$$\begin{cases} \beta_{m_{k+1}}^{(k+1)} b_0^{(k+1)} = (\tilde{z}_{n_{k+1}}, r_{n_{k+1}}) \\ \beta_{m_{k+1}-1}^{(k+1)} b_0^{(k+1)} + \beta_{m_{k+1}}^{(k+1)} b_1^{(k+1)} = 0 \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ \beta_1^{(k+1)} b_0^{(k+1)} + \cdots\cdots + \beta_{m_{k+1}}^{(k+1)} b_{m_{k+1}-1}^{(k+1)} = 0 \end{cases}$$

et

$$\lambda_0^{(k+1)} = \frac{(\tilde{r}_{n_{k+2}}, r_{n_{k+2}})}{(\tilde{z}_{n_{k+1}}, r_{n_{k+1}})}.$$

Les relations ainsi établies définissent l'algorithme du BICG avec look-ahead.

Algorithme 2.2.2 : *Gradient Biconjugué avec look-ahead.*

1. Initialisation : Choisir x_0 et $y \in \mathbb{R}^N$;
   $r_0 = b - Ax_0$, $z_0 = r_0$, $\tilde{z}_0 = y$, $k = 0$ et $n_0 = 0$;
   $\rho_0^{(k)} = (\tilde{r}_{n_k}, r_{n_k})$;

2. Tant que $r_{n_k} \neq 0$:

3. Si $\rho_0^{(k)} = 0$;
$b_0^{(k)} = (\tilde{z}_{n_k}, Az_{n_k})$;
$m_k = 1$;
Tant que $b_0^{(k)} = 0$;
$m_k = m_k + 1$;
$b_0^{(k)} = (\tilde{z}_{n_k}, A^{m_k} z_{n_k})$;
Fin (tant que) ;
$n_{k+1} = n_k + m_k$;
$x_{n_{k+1}} = x_{n_k}$ et $r_{n_{k+1}} = r_{n_k}$;
Si $k = 0$;
$\beta_0 = 0$;
Sinon
$$\beta_0 = \frac{b_0^{(k)}}{\rho_0^{(k-1)}} ;$$
Fin(si) ;
Pour $i = 1, \ldots, m_k$;
$b_i^{(k)} = (\tilde{z}_{n_k}, A^{m_k+i} z_{n_k})$;
calculer $\alpha_{m_k-i}^{(k)}$ à partir de 2.30 ;
Fin (pour) ;
$z_{n_{k+1}} = A^{m_k} z_{n_k} - \alpha_{m_k-1}^{(k)} A^{m_k-1} z_{n_k} - \cdots - \alpha_0^{(k)} z_{n_k} - \beta_0 z_{n_{k-1}}$;
$\tilde{z}_{n_{k+1}} = (A^T)^{m_k} \tilde{z}_{n_k} - \alpha_{m_k-1}^{(k)} (A^T)^{m_k-1} \tilde{z}_{n_k} - \cdots - \alpha_0^{(k)} \tilde{z}_{n_k} - \beta_0 \tilde{z}_{n_{k-1}}$;
$\rho_0^{(k+1)} = (\tilde{z}_{n_{k+1}}, r_{n_{k+1}})$;
$k = k + 1$
Fin (si).

4. $b_0^{(k)} = (\tilde{z}_{n_k}, Az_{n_k})$;
$m_k = 1$;
Tant que $b_0^{(k)} = 0$;
$m_k = m_k + 1$;
$b_0^{(k)} = (\tilde{z}_{n_k}, A^{m_k} z_{n_k})$;
Fin (tant que) ;
$n_{k+1} = n_k + m_k$;
Pour $i = 1, \ldots, m_k$;
$b_i^{(k)} = (\tilde{z}_{n_k}, A^{m_k+i} z_{n_k})$;
calculer $\beta_{m_k-i+1}^{(k)}$ à partir de 2.29 ;
Fin (pour) ;

$$
\begin{aligned}
&x_{n_{k+1}} = x_{n_k} + \beta_1^{(k)} z_{n_k} + \cdots + \beta_{m_k}^{(k)} A^{m_k-1} z_{n_k} ;\\
&r_{n_{k+1}} = r_{n_k} - \beta_1^{(k)} A z_{n_k} - \cdots - \beta_{m_k}^{(k)} A^{m_k} z_{n_k} ;\\
&\tilde{r}_{n_{k+1}} = \tilde{r}_{n_k} - \beta_1^{(k)} (A^T) \tilde{z}_{n_k} - \cdots - \beta_{m_k}^{(k)} (A^T)^{m_k} \tilde{z}_{n_k} ;\\
&\rho_0^{(k+1)} = (\tilde{r}_{n_{k+1}}, r_{n_{k+1}}) ;\\
&\lambda_0^{(k)} = \frac{\rho_0^{(k+1)}}{\rho_0^{(k)}} ;\\
&z_{n_{k+1}} = r_{n_{k+1}} + \lambda_0^{(k)} z_{n_k} ;\\
&\tilde{z}_{n_{k+1}} = \tilde{r}_{n_{k+1}} + \lambda_0^{(k)} \tilde{z}_{n_k} ;\\
&k = k + 1 ;\\
&\text{Fin(tant que).}
\end{aligned}
$$

Dans cet algorithme, le coefficient $\rho_0^{(k)}$ nous permet de détecter le *ghost-breakdown*. Dans le cas où $\rho_0^{(k)} \neq 0$, nous utilisons des récurrences semblables à celles utilisées dans l'algorithme BMRZ [13, 17]. Ici, un seul saut de taille m_k suffit pour calculer les itérés réguliers à l'itération n_{k+1}. Dans le cas contraire, nous avons une stagnation du résidu $r_{n_{k+1}} = r_{n_k}$. Nous calculons alors les vecteurs de direction par des récurrences semblables à celles de MRZ-stab, et en faisant un deuxième saut m_{k+1}, nous sommes sûrs d'avoir les itérés réguliers de l'itération n_{k+2}. Les tailles des sauts sont obtenus grâce au test $b_0^{(k)} = (\tilde{z}_{n_k}, A^{m_k} z_{n_k}) \neq 0$.

2.3 Quelques méthodes de Lanczos préconditionnées

Dans cette section, nous allons utiliser les polynômes orthogonaux formels pour introduire le préconditionneur dans quelques méthodes de type Lanczos qui utilisent le *look-ahead*. Pour cela, nous commençons par établir la connexion entre les polynômes orthogonaux formels et la méthode du Lanczos préconditionnée. Ensuite, en donnant l'interprétation polynômiale de chacune des méthodes étudiées, nous construisons leurs algorithmes avec préconditionneur. Cette procédure a déjà été utilisée par Brezinski et al. [18] pour le Gradient Biconjuguée. Ici, nous traitons les méthodes : CSBCG [21], QMR avec *look-ahead* [34], BIORES non générique[40], MRZ-stab et HMRZ-stab [17].

La matrice A risque d'être mal conditionnée. Pour cela nous allons multiplier l'équation (2.1) par M^{-1}, avec M une approximation de A dont l'inverse est facile à calculer (un système avec cette matrice est facile a résoudre) et à stocker. Ainsi nous obtenons un nouveau système à résoudre

$$M^{-1} A \, x = M^{-1} b, \tag{2.31}$$

pour lequel nous allons appliquer la méthode de Lanczos.

Nous commençons par noter $\tilde{r}_k = M^{-1}b - M^{-1}Ax_k$. Les conditions (2.2) nous donnent

$$x_k - x_0 \in \mathcal{K}_k(M^{-1}A, \tilde{r}_0), \tag{2.32}$$

$$\tilde{r}_k \perp \mathcal{K}_k((M^{-1}A)^T, y). \tag{2.33}$$

Puisque $\tilde{r}_k = M^{-1}r_k$, alors les conditions (2.2) s'écrivent aussi

$$x_k - x_0 \in M^{-1}\,\mathcal{K}_k(AM^{-1}, r_0), \tag{2.34}$$

$$r_k = b - Ax_k \perp \mathcal{K}_k((AM^{-1})^T, z) \quad \text{avec } z = M^{-T}y. \tag{2.35}$$

D'après (2.32) et (2.34), $x_k - x_0$ peut s'écrire sous la forme

$$x_k - x_0 = -a_1M^{-1}r_0 - \cdots - a_k(M^{-1}A)^{k-1}M^{-1}r_0.$$

En multipliant cette équation par A et en soustrayant b, nous obtenons

$$(Ax_k - b) - (Ax_0 - b) = -a_1AM^{-1}r_0 - \cdots - a_kA(M^{-1}A)^{k-1}M^{-1}r_0,$$

d'où

$$\begin{aligned} r_k &= r_0 + a_1AM^{-1}r_0 + \cdots + a_kA(M^{-1}A)^{k-1}M^{-1}r_0 \\ &= r_0 + a_1AM^{-1}r_0 + \cdots + a_k(AM^{-1})^k r_0 \end{aligned}$$

et

$$\tilde{r}_k = \tilde{r_0} + a_1M^{-1}A\tilde{r_0} + \cdots + a_k(M^{-1}A)^k\tilde{r_0}.$$

Pour simplifier les notations, nous posons

$$A_l = M^{-1}A \quad \text{et} \quad A_r = AM^{-1}.$$

Ainsi

$$\begin{aligned} r_k &= r_0 + a_1A_rr_0 + \cdots + a_kA_r^kr_0, \\ \tilde{r_k} &= \tilde{r_0} + a_1A_l\tilde{r_0} + \cdots + a_kA_l^k\tilde{r_0}. \end{aligned} \tag{2.36}$$

La condition d'orthogonalité (2.33) peut s'écrire sous la forme

$$(y, A_l^i\tilde{r}_k) = 0 \quad \text{pour } i = 0, \ldots, k-1, \tag{2.37}$$

et à partir de la relation (2.36), nous obtenons le système d'équations linéaires

$$(y, A_l^i\tilde{r}_0) + a_1(y, A_l^{i+1}\tilde{r}_0) + \cdots + a_k(y, A_l^{i+k}\tilde{r}_0) = 0, \quad i = 0, \ldots, k-1. \tag{2.38}$$

De façon similaire, la condition d'orthogonalité (2.35) donne

$$(z, A_r^ir_k) = 0 \quad \text{pour } i = 0, \ldots, k-1, \tag{2.39}$$

et à partir de la relation (2.36), nous avons le système d'équations linéaires

$$(z, A_r^i r_0) + a_1(z, A_r^{i+1} r_0) + \cdots + a_k(z, A_r^{i+k} r_0) = 0, \quad i = 0, \ldots, k-1. \tag{2.40}$$

Si les déterminants des systèmes (2.38) et (2.40) (qui sont identiques puisque $(z, A_r^i r_0) = (y, A_l^i \tilde{r}_0)$) sont différents de zéro, alors x_k existe et il est déterminé de façon unique soit par (2.32) et (2.33) soit par (2.34) et (2.35).
Considérons le polynôme

$$P_k(\xi) = 1 + a_1\xi + \cdots + a_k\xi^k.$$

D'après les relations (2.36), nous avons

$$r_k = P_k(A_r)r_0 \qquad \text{et} \qquad \tilde{r_k} = P_k(A_l)\tilde{r}_0.$$

Si l'on pose

$$c_i = (z, A_r^i r_0) = (y, A_l^i \tilde{r}_0) \quad \text{pour } i = 1, 2, \ldots$$

et si l'on définit la fonctionnelle linéaire c sur l'espace des polynômes par

$$c(\xi^i) = c_i \quad \text{pour } i = 1, 2, \ldots$$

alors, pour tout polynôme R,

$$c(R(\xi)) = (z, R(A_r)r_0) = (y, R(A_l)\tilde{r}_0).$$

Les conditions d'orthogonalité (2.37) (ainsi que (2.39)) s'écrivent sous la forme

$$c(\xi^i P_k(\xi)) = 0 \quad \text{pour } i = 0, \ldots, k-1. \tag{2.41}$$

Ces conditions montrent que P_k est le polynôme de degré au plus k appartenant à la famille des polynômes orthogonaux formels par rapport à la fonctionnelle c, normalisé par la condition $P_k(0) = 1$.

2.3.1 Algorithme du Gradient Biconjugué à pas composés

L'algorithme du Gradient Biconjugué à pas composés (CSBCG), proposé par Chan et Bank [21, 22], est une simple modification du Gradient Binconjugé (BICG) dans lequel on utilise les relations de récurrence

$$\begin{aligned} P_{k+1}(\xi) &= P_k(\xi) - \alpha_k \xi Q_k(\xi), \\ Q_{k+1}(\xi) &= P_{k+1}(\xi) + \beta_{k+1} Q_k(\xi), \end{aligned} \tag{2.42}$$

où

$$Q_k(\xi) = (-1)^k (H_k^{(0)} / H_k^{(1)}) P_k^{(1)}(\xi),$$

avec $(-1)^k(H_k^{(0)}/H_k^{(1)})$ = coefficient de plus haut degré de P_k.
Le BICG subit les deux types de breakdown (*ghost* et *true*). L'algorithme du (CSBCG) consiste à simplifier ce problème en évitant le *true-breakdown* sous la condition que le *ghost-breakdown* ne puisse pas apparaître dans le BICG. Donc, dans tout ce qui va suivre, nous allons supposer que $H_k^{(0)} \neq 0 \ \forall k$.
Nous commençons par donner le lemme suivant, qui est un cas particulier du lemme 1.2.2.

Lemme 2.3.1
Si $\forall k \ H_k^{(0)} \neq 0$, deux déterminant consécutif de Hankel $H_k^{(1)}$ et $H_{k+1}^{(1)}$ ne peuvent être nuls.

Remarque 2.3.1 *Sous la condition $H_k^{(0)} \neq 0$, nous avons*
- *La longueur du saut entre les degrés des polynômes réguliers $P_k^{(1)}$ ne peut pas dépasser deux.*
- *Les polynômes P_k, lorsqu'ils existent, sont de degré égal à k.*

Nous supposons maintenant que nous avons un *true-breakdown* à la $k^{\text{ème}}$ itération (i.e $H_{k+1}^{(1)} = 0$). Les polynômes P_{k+1} et Q_{k+1} n'existent pas alors que P_k, Q_k, P_{k+2} et Q_{k+2} existent. L'idée est alors de calculer directement P_{k+2} et Q_{k+2} sans passer par P_{k+1} et Q_{k+1}.
Puisque $H_k^{(0)} \neq 0 \ \forall k$, les polynômes unitaires $P_k^{(0)}$ existent pour tout k, et si nous notons a_{k+2} le coefficient de plus haut degré de P_{k+2}, alors nous pouvons exprimer le polynôme

$$P_{k+2} - a_{k+2}\xi P_{k+1}^{(0)}$$

de degré $k+1$ dans la base

$$\{P_0^{(0)}, P_1^{(0)}, \ldots, P_{k-1}^{(0)}, P_k, \xi Q_k, \}.$$

En imposant les conditions d'orthogonalité (2.41), nous obtenons

$$P_{k+2} = a_{k+2}\xi P_{k+1}^{(0)} + b_k \xi Q_k + c_k P_k.$$

Grâce à la condition de normalisation $P_{k+2}(0) = 1$, nous pouvons écrire

$$P_{k+2} = a_{k+2}\xi P_{k+1}^{(0)} + b_k \xi Q_k + P_k.$$

Par définition, le polynôme Q_{k+2} est de degré $k+2$ et il a le même coefficient de plus haut degré que P_{k+2} et donc le polynôme $Q_{k+2} - P_{k+2}$ est de degré $k+1$. Ainsi, nous pouvons l'exprimer dans la base

$$\{P_0^{(0)}, P_1^{(0)}, \ldots, P_{k-1}^{(0)}, Q_k, P_{k+1}^{(0)}\},$$

et en imposant les conditions d'orthogonalité (2.41), nous obtenons

$$Q_{k+2} = P_{k+2} + d_{k+1}P_{k+1}^{(0)} + d_k Q_k.$$

De façon similaire, en exprimant le polynôme $P_{k+1}^{(0)}$ dans la base

$$\{P_0^{(0)}, P_1^{(0)}, \ldots, P_{k-1}^{(0)}, P_k, tQ_k\}$$

nous obtenons

$$P_{k+1}^{(0)} = \lambda_k P_k + \gamma_k \xi Q_k.$$

$P_{k+1}^{(0)}$ est un polynôme unitaire et d'après la définition de Q_k nous avons

$$\gamma_k = (-1)^k H_k^{(1)} / H_k^{(0)}.$$

En utilisant les conditions d'orthogonalité (2.41), nous obtenons

$$\lambda_k = -\gamma_k \frac{c(\xi P_k^{(0)} Q_k)}{c(P_k^{(0)} P_k)}.$$

D'après la définition de P_k et $P_k^{(0)}$, nous avons la relation

$$P_k(\xi) = (-1)^{(k)} \frac{H_k^{(0)}}{H_k^{(1)}} P_k^{(0)}(\xi)$$

et si nous posons

$$\rho_k = c(P_k P_k) \quad \text{et} \quad \sigma_k = c(\xi P_k Q_k),$$

alors nous pouvons écrire

$$\lambda_k = -\frac{\gamma_k \sigma_k}{\rho_k}.$$

Nous considérons maintenant le polynôme

$$\bar{P}_{k+1} = -\frac{\rho_k}{\gamma_k} P_{k+1}^{(0)}.$$

D'après ce qui précède, nous avons la récurrence suivante

$$\bar{P}_{k+1} = \sigma_k P_k - \rho_k \xi Q_k. \tag{2.43}$$

D'autre part, P_k et Q_k ont le même coefficient de plus haut degré, ce qui nous permet d'écrire

$$P_k = Q_k + R_{k-1},$$

où R_{k-1} est un polynôme de degré au plus égal à $k-1$.
Alors nous obtenons

$$\begin{aligned} c(\xi P_k Q_k) &= c(\xi Q_k Q_k) + c(\xi R_{k-1} Q_k) \\ &= c(\xi Q_k Q_k), \end{aligned}$$

donc

$$\sigma_k = c(\xi Q_k Q_k).$$

Remarque 2.3.2

1. $\sigma_k = 0$ *dans le cas du true-breakdown.*
2. $\rho_k = 0$ *dans le cas du ghost-breakdown.*

Comme nous l'avons déjà signalé dans la Remarque (2.3.1), sous la condition $H_k^{(0)} \neq 0 \; \forall k$, les polynômes P_k lorsqu'ils existent, sont de degré exactement k, et nous avons

$$\rho_k = c(P_k P_k) = ((-1)^k H_k^{(0)}/H_k^{(1)})^2 c({P_k^{(0)}}^2) \neq 0.$$

Les récurrences concernant P_{k+2} et Q_{k+2}, établies précédemment, peuvent s'écrire sous la forme

$$P_{k+2} = P_k - \xi[Q_k, \bar{P}_{k+1}]f_k \tag{2.44}$$
$$Q_{k+2} = P_{k+2} + [Q_k, \bar{P}_{k+1}]g_k, \tag{2.45}$$

avec $f_k = [f_k^{(1)}, f_k^{(2)}]^T$ et $g_k = [g_k^{(1)}, g_k^{(2)}]^T$ deux vecteurs de $\mathbb{R}^2$ qu'on peut déterminer grâce aux conditions d'orthogonalité.
En multipliant (2.44) par Q_k et en appliquant la fonctionnelle c, nous obtenons

$$\begin{aligned} c(Q_k P_{k+2}) &= c(P_k Q_k) - c(\xi Q_k Q_k) f_k^{(1)} - c(\xi Q_k \bar{P}_{k+1}) f_k^{(2)}, \\ 0 &= c(P_k Q_k) - c(\xi Q_k Q_k) f_k^{(1)} - c(\xi Q_k \bar{P}_{k+1}) f_k^{(2)}. \end{aligned} \tag{2.46}$$

En multipliant (2.44) par $\bar{P}_{k+1}$ et en appliquant la fonctionnelle c, nous avons

$$\begin{aligned} c(\bar{P}_{k+1} P_{k+2}) &= c(\bar{P}_{k+1} P_k) - c(\xi \bar{P}_{k+1} Q_k) f_k^{(1)} - c(\xi \bar{P}_{k+1} \bar{P}_{k+1}) f_k^{(2)}, \\ 0 &= c(\xi \bar{P}_{k+1} Q_k) f_k^{(1)} + c(\xi \bar{P}_{k+1} \bar{P}_{k+1}) f_k^{(2)}. \end{aligned} \tag{2.47}$$

De même pour la relation (2.45), en appliquant la fonctionnelle $c^{(1)}$, nous obtenons

$$\begin{aligned} c^{(1)}(Q_k Q_{k+2}) &= c(\xi P_k P_{k+2}) + c(\xi Q_k Q_k) g_k^{(1)} + c(\xi Q_k \bar{P}_{k+1}) g_k^{(2)}, \\ 0 &= c(\xi Q_k Q_k) g_k^{(1)} + c(\xi Q_k \bar{P}_{k+1}) g_k^{(2)}, \end{aligned} \tag{2.48}$$

et

$$\begin{aligned} c^{(1)}(\bar{P}_{k+1} Q_{k+2}) &= c(\xi \bar{P}_{k+1} P_{k+2}) + c(\xi \bar{P}_{k+1} Q_k) g_k^{(1)} + c(\xi \bar{P}_{k+1} \bar{P}_{k+1}) g_k^{(2)}, \\ 0 &= c(\xi \bar{P}_{k+1} P_{k+2}) + c(\xi \bar{P}_{k+1} Q_k) g_k^{(1)} + c(\xi \bar{P}_{k+1} \bar{P}_{k+1}) g_k^{(2)}. \end{aligned} \tag{2.49}$$

Pour le calcul des vecteurs f_k et g_k, nous utilisons les deux relations suivantes, obtenues à partir de (2.43) et (2.44)

$$\begin{aligned} c(\xi Q_k \bar{P}_{k+1}) &= -c(\bar{P}_{k+1}^2)/\rho_k, \\ c(\, P_{k+2}^2) &= -c(\xi P_{k+2} \bar{P}_{k+1}) f_k^{(2)}. \end{aligned}$$

Les relations (2.46), (2.47), (2.48) et (2.49) s'écrivent alors sous la forme

$$\begin{bmatrix} \sigma_k & -\theta_{k+1}/\rho_k \\ -\theta_{k+1}/\rho_k & \zeta_{k+1} \end{bmatrix} \begin{bmatrix} f_k^{(1)} & g_k^{(1)} \\ f_k^{(2)} & g_k^{(2)} \end{bmatrix} = \begin{bmatrix} \rho_k & 0 \\ 0 & \rho_{k+2}/f_k^{(2)} \end{bmatrix} \tag{2.50}$$

avec

$$\begin{aligned} \zeta_{k+1} &= c(\xi \bar{P}_{k+1}^2), \\ \theta_{k+1} &= c(\bar{P}_{k+1}^2). \end{aligned}$$

Le système (2.50) a une solution explicite donnée par

$$\begin{aligned} f_k &= \frac{\rho_k^2}{\delta_k} [\zeta_{k+1}\rho_k, \theta_{k+1}], \\ g_k &= [\frac{\rho_{k+2}}{\rho_k}, \sigma_k \frac{\rho_{k+2}}{\theta_{k+1}}], \end{aligned} \tag{2.51}$$

où

$$\delta_k = \sigma_k \zeta_{k+1} \rho_k^2 - \theta_{k+1}^2.$$

À partir des relations de récurrence établies ci-dessus, nous allons mettre en œuvre la méthode du CSBCG avec préconditionneur. Nous commençons par le cas où à une itération k nous avons un *true-breakdown*. Nous allons construire deux algorithmes dont l'un nous donne le résidu non préconditionné et l'autre le résidu préconditionné. Ensuite nous allons combiner les deux pour en extraire un nouvel algorithme dans lequel nous effectuons un saut de deux ce qui nous permet de calculer le résidu r_{k+2} sans avoir à calculer r_{k+1}, avec le moins d'opérations possible.
Dans le premier algorithme nous allons calculer récursivement le résidu $r_{k+2} = P_{k+2}(A_r)r_0$. Posons

$$p_k = Q(A_r)r_0 \quad \text{et} \quad z_{k+1} = \bar{P}_{k+1}(A_r)r_0.$$

Les relations (2.43), (2.44) et (2.45) deviennent

$$\begin{aligned} z_{k+1} &= \sigma_k r_k - \rho_k A_r p_k, \\ r_{k+2} &= r_k - A_r[p_k, z_{k+1}]f_k, \\ p_{k+2} &= r_{k+2} + [p_k, z_{k+1}]g_k. \end{aligned} \tag{2.52}$$

En remplaçant le résidu par son expression initiale $r_k = b - Ax_k$, nous obtenons

$$x_{k+2} = x_k + M^{-1}[p_k, z_{k+1}]f_k.$$

Le second algorithme nous permet de calculer le résidu préconditionné $\tilde{r}_{k+2} = P_{k+2}(A_l)\tilde{r}_0$. Posons

$$\tilde{p}_k = Q(A_l)\tilde{r}_0 \quad \text{et} \quad \tilde{z}_{k+1} = \bar{P}_{k+1}(A_l)\tilde{r}_0.$$

Les relations (2.43), (2.44) et (2.45) deviennent

$$\begin{aligned} \tilde{z}_{k+1} &= \sigma_k \tilde{r}_k - \rho_k A_l \tilde{p}_k, \\ \tilde{r}_{k+2} &= \tilde{r}_k - A_l[\tilde{p}_k, \tilde{z}_{k+1}]f_k, \\ \tilde{p}_{k+2} &= \tilde{r}_{k+2} + [\tilde{p}_k, \tilde{z}_{k+1}]g_k, \end{aligned} \tag{2.53}$$

et le vecteur x_k vérifie

$$x_{k+2} = x_k + [\tilde{p}_k, \tilde{z}_{k+1}]f_k.$$

Remarque 2.3.3

Nous constatons que $\tilde{r}_k = M^{-1}r_k$, $\tilde{p}_k = M^{-1}p_k$ et $\tilde{z}_k = M^{-1}z_k$, alors

- *nous pouvons obtenir les relations (2.53) directement à partir des relations (2.52),*
- *les deux algorithmes obtenus ne sont pas indépendants.*

Pour calculer les coefficients des relations (2.52) et (2.53), nous définissons d'abord les vecteurs suivants

$$\begin{aligned} r'_k &= P_k(A_l^T)y, & \tilde{r}'_k &= P_k(A_r^T)z, \\ z'_k &= \bar{P}_k(A_l^T)y, & \tilde{z}'_k &= \bar{P}_k(A_r^T)z, \\ p'_k &= Q_k(A_l^T)y, & \tilde{p}'_k &= Q_k(A_r^T)z. \end{aligned} \tag{2.54}$$

Ces vecteurs sont calculés en utilisant les relations de récurrence (2.43), (2.44) et (2.45). Rappelant la définition de la fonctionnelle c

$$c(\xi^i) = (z, A_r^i r_0) = (y, A_l^i \tilde{r}_0),$$

alors

$$\begin{aligned} \sigma_k &= c(\xi Q_k Q_k) \\ &= (z, A_r Q_k(A_r) Q_k(A_r) r_0) \\ &= (Q_k(A_r)^T z, A_r Q_k(A_r) r_0) = (\tilde{p}'_k, A_r p_k) \\ &= (\tilde{p}'_k, A\tilde{p}_k). \end{aligned} \tag{2.55}$$

De façon similaire, nous trouvons

$$\begin{aligned} \sigma_k &= c(\xi Q_k Q_k) \\ &= (y, A_l Q_k(A_l) Q_k(A_l) \tilde{r}_0) \\ &= (Q_k(A_l)^T y, A_l Q_k(A_l) \tilde{r}_0) = (p'_k, A_l \tilde{p}_k), \end{aligned}$$

et, de même,

$$\begin{aligned} \rho_k &= c(P_k P_k) \\ &= (P_k(A_r^T)z, P_k(A_r)r_0) = (\tilde{r}'_k, r_k) \\ &= (P_k(A_l)^T y, P_k(A_l)\tilde{r}_0) = (r'_k, \tilde{r}_k). \end{aligned} \tag{2.56}$$

Nous avons également

$$\begin{aligned} \zeta_{k+1} &= c(\xi \bar{P}_{k+1}^2), \\ &= (z, A_r \bar{P}_{k+1}^2(A_r)r_0) = (\bar{P}_{k+1}(A_r^T)z, A_r \bar{P}_{k+1}(A_r)r_0) = (\tilde{z}'_{k+1}, A\tilde{z}_{k+1}), \\ &= (y, A_l \bar{P}_{k+1}^2(A_l)\tilde{r}_0) = (\bar{P}_{k+1}(A_l^T)y, A_l \bar{P}_{k+1}(A_l)r_0) = (z'_{k+1}, A_l\tilde{z}_{k+1}), \end{aligned} \tag{2.57}$$

et

$$\begin{aligned} \theta_{k+1} &= c(\bar{P}_{k+1}^2), \\ &= (z, \bar{P}_{k+1}^2(A_r)r_0) = (\bar{P}_{k+1}(A_r^T)z, \bar{P}_{k+1}(A_r)r_0) = (\tilde{z}'_{k+1}, z_{k+1}), \\ &= (y, \bar{P}_{k+1}^2(A_l)\tilde{r}_0) = (\bar{P}_{k+1}(A_l^T)y, \bar{P}_{k+1}(A_l)\tilde{r}_0) = (z'_{k+1}, \tilde{z}_{k+1}). \end{aligned} \tag{2.58}$$

Les coefficients f_k et g_k sont calculés à partir de (2.51).

Remarque 2.3.4
Nous avons $\tilde{p}'_k = M^{-T}p'_k$, $\tilde{r}'_k = M^{-T}r'_k$ *et* $\tilde{z}'_k = M^{-T}z'_k$.
Cela nous permet d'affirmer que les deux expressions de σ_k *(ainsi que* ρ_k, ζ_{k+1}, θ_{k+1}*) sont exactement les mêmes.*

En combinant ces deux algorithmes, nous pouvons construire un algorithme complet qui nous permet de calculer le résidu r_{k+2} à une itération k où se produit un *true-breakdown*. Cet algorithme est nommé (2×2 *step*).
Nous avons donc les relations suivantes

$$\begin{aligned} z_{k+1} &= \sigma_k r_k - \rho_k A\tilde{p}_k, \\ r_{k+2} &= r_k - A[\tilde{p}_k, \tilde{z}_{k+1}]f_k, \\ \tilde{p}_{k+2} &= \tilde{r}_{k+2} + [\tilde{p}_k, \tilde{z}_{k+1}]g_k. \end{aligned} \tag{2.59}$$

A partir de (2.54) et des relations (2.43), (2.44) et (2.45) nous obtenons

$$\begin{aligned} z'_{k+1} &= \sigma_k r'_k - \rho_k A\tilde{p}'_k, \\ r'_{k+2} &= r'_k - A[\tilde{p}'_k, \tilde{z}'_{k+1}]f_k, \\ \tilde{p}'_{k+2} &= \tilde{r}'_{k+2} + [\tilde{p}'_k, \tilde{z}'_{k+1}]g_k, \\ x_{k+2} &= x_k + [\tilde{p}_k, \tilde{z}_{k+1}]f_k, \end{aligned} \tag{2.60}$$

avec

$$\tilde{z}_{k+1} = M^{-1} z_{k+1}, \quad \tilde{r}_k = M^{-1} r_k,$$

et

$$\tilde{z}'_{k+1} = M^{-T} z'_{k+1}, \quad \tilde{r}'_k = M^{-T} r'_k.$$

Les coefficients ρ_k et σ_k sont donnés dans (2.55) et (2.56) et sont calculés par

$$\begin{aligned} \sigma_k &= (\tilde{p}'_k, A\tilde{p}_k), \\ \rho_k &= (r'_k, \tilde{r}_k), \end{aligned}$$

et, d'autre part, f_k et g_k sont calculés à partir de (2.51) avec

$$\begin{aligned} \zeta_{k+1} &= (\tilde{z}'_{k+1}, A\tilde{z}_{k+1}) \\ \theta_{k+1} &= (\tilde{z}'_{k+1}, z_{k+1}). \end{aligned}$$

Remarque 2.3.5

D'après (2.56) et (2.58), nous pouvons aussi prendre $\rho_k = (\tilde{r}'_k, r_k)$ *et* $\theta_{k+1} = (z'_{k+1}, \tilde{z}_{k+1})$, *ce qui ne change rien à l'algorithme.*

S'il n'y a pas de *true-breakdown* à l'itération k, nous pouvons calculer le résidu r_{k+1} en utilisant les relations de récurrence (2.42). Dans ce cas nous somme amenés à construire un algorithme, nommé (1×1 *step*), qui est une variante du BIGG avec préconditionneur. En appliquant les conditions d'orthogonalité aux relations (2.42), nous obtenons

$$\begin{aligned} \alpha_k &= \frac{c(P_k P_k)}{c(\xi P_k Q_k)} = \rho_k / \sigma_k, \\ \beta_{k+1} &= -\frac{c(\xi Q_k P_{k+1})}{c(\xi Q_k Q_k)} \\ &= \frac{c(P_{k+1} P_{k+1})}{\alpha_k c(\xi Q_k Q_k)} = \frac{\rho_{k+1}}{\alpha_k \sigma_k} = \rho_{k+1} / \rho_k. \end{aligned}$$

Alors la relation de récurrence de P_{k+1} s'écrit

$$P_{k+1} = P_k - \frac{\rho_k}{\sigma_k} \xi Q_k.$$

En comparant ce résultat avec (2.43), nous voyons que

$$P_{k+1} = \frac{\bar{P}_{k+1}}{\sigma_k}$$

et les relations (2.42) s'écrivent aussi

$$\begin{aligned} P_{k+1}(\xi) &= P_k(\xi) - \alpha_k \xi Q_k(\xi), \\ Q_{k+1}(\xi) &= \frac{\bar{P}_{k+1}}{\sigma_k}(\xi) + \beta_{k+1} Q_k(\xi). \end{aligned}$$

Ainsi, nous obtenons

$$\begin{aligned} x_{k+1} &= x_k + \alpha_k \tilde{p}_k, \\ r_{k+1} &= r_k - \alpha_k A\tilde{p}_k, \qquad & r'_{k+1} &= r'_k - \alpha_k A^T \tilde{p}'_k, \\ \tilde{p}_{k+1} &= \tilde{z}_{k+1}/\sigma_k + \beta_{k+1}\tilde{p}_k, \qquad & \tilde{p}'_{k+1} &= \tilde{z}'_{k+1}/\sigma_k + \beta_{k+1}\tilde{p}'_k. \end{aligned}$$

Pour le calcul de ρ_{k+1} dans le (1×1 *step*), et afin de réduire le nombre d'opérations à effectuer, nous prenons

$$\begin{aligned} \rho_{k+1} &= c(P_{k+1}P_{k+1}) = c(\tilde{P}_{k+1}\tilde{P}_{k+1})/\sigma_k^2 \\ &= (\tilde{z}'_{k+1}, z_{k+1})/\sigma_k^2 = \theta_{k+1}/\sigma_k^2. \end{aligned}$$

Les relations ainsi établies définissent l'algorithme CSBCG avec un préconditionneur M.

Algorithme 2.3.1 : *CSBCG avec préconditionneur*

1. Initialisation
   Choisir x_0 et y
   $r_0 = b - Ax_0$; $\quad r'_0 = y$
   Résoudre $M\tilde{p}_0 = r_0$
   $M^T\tilde{p}'_0 = y$
   $\rho_0 = (r'_0, \tilde{p}_0)$
   $k \longleftarrow 0$

2. Tant que $r_k \neq 0$
   $\sigma_k = (\tilde{p}'_k, A\tilde{p}_k)$
   $z_{k+1} = \sigma_k r_k - \rho_k A\tilde{p}_k$; $\qquad z'_{k+1} = \sigma_k r'_k - \rho_k A^T\tilde{p}'_k$
   Résoudre $M\tilde{z}_{k+1} = z_{k+1}$
   $M^T\tilde{z}'_{k+1} = z'_{k+1}$
   $\theta_{k+1} = (\tilde{z}'_{k+1}, z_{k+1})$
   $\zeta_{k+1} = (\tilde{z}'_{k+1}, A\tilde{z}_{k+1})$

3. Si 1×1 *step*
   $\alpha_k = \rho_k/\sigma_k$
   $\rho_{k+1} = \theta_{k+1}/\sigma_k^2$
   $\beta_{k+1} = \rho_{k+1}/\rho_k$

$x_{k+1} = x_k + \alpha_k \tilde{p}_k$
$r_{k+1} = r_k - \alpha_k A\tilde{p}_k$; $\quad r'_{k+1} = r'_k - \alpha_k A^T \tilde{p}'_k$
$\tilde{p}_{k+1} = \tilde{z}_{k+1}/\sigma_k + \beta_{k+1}\tilde{p}_k$; $\quad \tilde{p}'_{k+1} = \tilde{z}'_{k+1}/\sigma_k + \beta_{k+1}\tilde{p}'_k$
$k \longleftarrow k+1$

4. Si 2×2 *step*
$\delta_k = \sigma_k \zeta_{k+1} \rho_k^2 - \theta_{k+1}^2$
$\alpha_k = \zeta_{k+1}\rho_k^3/\delta_k$
$\alpha_{k+1} = \theta_{k+1}\rho_k^2/\delta_k$
$x_{k+2} = x_k + \alpha_k \tilde{p}_k + \alpha_{k+1}\tilde{z}_{k+1}$
$r_{k+2} = r_k - \alpha_k A\tilde{p}_k - \alpha_{k+1} A\tilde{z}_{k+1}$; $\quad r'_{k+2} = r'_k - \alpha_k A^T \tilde{p}'_k - \alpha_{k+1} A^T \tilde{z}'_{k+1}$
Résoudre $M\tilde{r}_{k+2} = r_{k+2}$
$M^T \tilde{r}'_{k+2} = r'_{k+2}$
$\rho_{k+2} = (r'_{k+2}, \tilde{r}_{k+2})$
$\beta_{k+1} = \rho_{k+2}/\rho_k$
$\beta_{k+2} = \rho_{k+2}\sigma_k/\theta_{k+1}$
$\tilde{p}_{k+2} = \tilde{r}_{k+2} + \beta_{k+1}\tilde{p}_k + \beta_{k+2}\tilde{z}_{k+1}$; $\quad \tilde{p}'_{k+2} = \tilde{r}'_{k+2} + \beta_{k+1}\tilde{p}'_k + \beta_{k+2}\tilde{z}'_{k+1}$
$k \longleftarrow k+2$

Dans le cas du (2×2 *step*) Les composantes de f_k et g_k sont notées respectivement par α_k, α_{k+1}, β_k et β_{k+1}.

Comme il a été signalé dans la Remarque 2.3.2, afin de détecter le *true-breakdown*, nous pouvons utiliser le test $\sigma_k \neq 0$, mais pour rendre la méthode du BiCG plus stable, un autre test a été proposé par Chan. Il consiste à utiliser le (1×1 *step*) si $\|r_{k+1}\| \leq \|r_k\|$, et le (2×2 *step*) si $\|r_{k+2}\| < \|r_{k+1}\|$. Ceci est équivalent mathématiquement à utiliser le (2×2 *step*) seulement lorsque

$$\|r_{k+1}\| > max\{\|r_k\|, \|r_{k+2}\|\}.$$

Évidemment ce test doit être effectué sans calculer les résidus r_{k+1} et r_{k+2}. Alors nous allons remplacer le test $\|r_{k+1}\| \leq \|r_k\|$ par $\|z_{k+1}\| \leq |\sigma_k|\|r_k\|$, et en posant

$$\nu_{k+2} = \delta_k r_{k+2} = \delta_k r_k - \zeta_{k+1}\rho_k^3 A\tilde{p}_k - \theta_{k+1}\rho_k^2 A\tilde{z}_{k+1},$$

nous avons

$$\begin{aligned} \|r_{k+1}\| < \|r_{k+2}\| &\Leftrightarrow \|z_{k+1}\| < |\sigma_k|\|r_{k+2}\|, \\ &\Leftrightarrow |\delta_k|\|z_{k+1}\| < |\sigma_k|\|\nu_{k+2}\|. \end{aligned}$$

Le test $\|r_{k+1}\| < \|r_{k+2}\|$ est donc remplacé par $|\delta_k|\|z_{k+1}\| < |\sigma_k|\|\nu_{k+2}\|$.

Nous notons que ce nouveau test nous permet d'éviter l'*overflow* dans le cas où les valeurs de $\|r_{k+1}\|$ et $\|r_{k+2}\|$ sont très grandes. D'autre part, dans le cas où $\|r_{k+1}\| > max\{\|r_k\|, \|r_{k+2}\|\}$, si nous utilisons le procédé (1×1 *step*) cela provoque des pics dans la courbe représentant la norme du résidu. Par conséquent, en prenant le procédé (2×2 *step*), toujours dans le même cas, nous arrivons à réduire ces pics, et ainsi obtenir une courbe qui est beaucoup plus lisse.
En résumé, nous effectuons le test suivant

si $\|z_{k+1}\| \leq |\sigma_k|\|r_k\|$, alors

(1×1 *step*)

sinon

$\|\nu_{k+2}\| = \|\delta_k r_{k+2} = \delta_k r_k - \zeta_{k+1}\rho_k^3 A\tilde{p}_k - \theta_{k+1}\rho_k^2 A\tilde{z}_{k+1}\|$

si $|\sigma_k|\|\nu_{k+2}\| \leq |\delta_k|\|z_{k+1}\|$

(2×2 *step*)

sinon

(1×1 *step*)

Remarque 2.3.6
Comme il a été souligné dans la Remarque 2.3.2, si nous avons un true-breakdown à la k^{eme} *itération alors* $\sigma_k = 0$ *et en utilisant le test précédent, nous effectuons une itération* (2×2 *step*). *Ainsi, ce test permet à la fois de détecter le true-breakdown sans utiliser de tolérance donnée et de lisser la courbe représentant la norme du résidu.*

2.3.2 la méthode du QMR avec look-ahead

Dans ce paragraphe, nous allons présenter la méthode du *Quasi-minimal residual* avec *look-ahead*. Le principe est le même que celui utilisé dans la méthode du GMRES [63], il consiste à réduire le système étudié et à le résoudre au sens des moindres carrés. Dans le QMR, on construit une base bi-orthogonale du sous-espace de Krylov à partir du processus de Lanczos, alors que dans le GMRES on utilise le processus d'Arnoldi pour avoir une base orthogonale. En utilisant le processus de Lanczos, le QMR peut rencontrer un problème de *breakdown* qu'on peut éviter en utilisant la technique du *look-ahead*.

Nous notons que la procédure du QMR a été proposée pour la première fois par R. Freund pour la résolution d'un système linéaire dans le cas symétrique complexe [30] et qu'elle a, ensuite, été adaptée au cas non symétrique par Freund et Nachtigal [34]. Freund et Hochbruk ont prouvé que cet algorithme peut être appliqué même pour les systèmes carrés singuliers [32].
Le QMR est donc une méthode de sous-espace de Krylov, dont la base est construite à partir du processus de Lanczos et les itérés sont caractérisés par les propriétés de "quasi-minimisation" de la norme du résidu (d'où son nom). L'algorithme du QMR est mathématiquement équivalent à celui du BICG qui utilise le processus de Lanczos et dont les itérés sont déterminés par la condition de bi-orthogonalité de Petrov-Galerkin. Comme nous l'avons vu dans l'algorithme du BICG, nous avons deux types de breakdown possibles : le *ghost-breakdown*, causé par une division par zéro dans le processus de Lanczos et le *true-breakdown*, dû à la singularité de la matrice tridiagonale générée par ce même processus. Le principe de "quasi-minimisation" utilisé dans le QMR permet d'éviter le *true-breakdown*. Par ailleurs, le *ghost-breakdown* est plus compliqué à éliminer, et Freund et Nachtigal [34] ont utilisé le processus de Lanczos avec *look-ahead* afin d'éviter ce type de breakdown, ce qui a donné l'algorithme du QMR avec *look-ahead*.
Cet algorithme est considéré comme une éventuelle alternative pour stabiliser la méthode du BICG. En effet, la "quasi-minimisation" de la norme du résidu employée dans le QMR permet d'éliminer les pics qui caractérisent la méthode du BICG, ainsi on obtient une convergence lisse et presque monotone.

Nous notons aussi que dans le cas où la matrice A du système à résoudre est symétrique (*i.e* $A = A^T$), le QMR est équivalent à l'algorithme du MINRES dû à Paige et Saunders [56].

Les algorithmes du *QMR* avec *look-ahead* donnés par Freund et Nachtigal [34, 35] exploitent des versions du processus de Lanczos dans lesquelles on traite, en plus du *breakdown*, le *near-breakdown*. La mise en œuvre de ces algorithmes est assez compliquées, c'est pourquoi le QMR sans *look-ahead* est le plus utilisé. Ici, nous nous intéressons seulement aux situations de *breakdown*. Nous proposons une version du QMR adaptée uniquement à ce problème et nous utilisons pour cela l'algorithme du processus de Lanczos (1.3.1). L'algorithme obtenu servira à des comparaisons avec les autres algorithmes de ce chapitre dans le cas d'un exact *breakdown*.

L'algorithme du QMR

Comme nous l'avons déjà évoqué auparavant, l'algorithme du QMR est une méthode de sous-espace de Krylov, donc les approximations x_{n+1} de la solution $x = A^{-1}b$ du système

linéaire (1) vérifient

$$x_{n+1} \in x_0 + \mathcal{K}_{n+1}(r_0, A), \qquad n = 0, 1, \ldots \tag{2.61}$$

avec $x_0 \in \mathbb{R}^N$ un approximation initiale de x choisie arbitrairement et $r_0 = b - Ax_0$ son résidu.
Comme nous l'avons vu dans la sous-section 1.3.3, nous pouvons construire une base de $\mathcal{K}_{n+1}(r_0, A)$ en utilisant le processus de Lanczos avec *look-ahead*.
En effet, si nous prenons $v_0 = r_0$ et un vecteur arbitraire $w_0 \in \mathbb{R}^N$, $w_0 \neq 0$, alors l'algorithme 1.3.1 nous permet de construire les vecteurs $\{v_0, v_1, \ldots, v_n\}$ qui forment une base génératrice de $\mathcal{K}_{n+1}(r_0, A)$ ainsi que la matrice de Hessenberg $H_e^{(n)}$ de dimension $(n+2) \times (n+1)$. Et nous avons la relation

$$AV^{(n)} = V^{(n+1)} H_e^{(n)}, \tag{2.62}$$

avec $V^{(n)} = [v_0, \ldots, v_n]$ définie dans (1.39).
La relation (2.61) peut s'écrire

$$x_{n+1} = x_0 + V^{(n)} z_n \qquad \text{avec} \quad z_n \in \mathbb{R}^{n+1}. \tag{2.63}$$

En utilisant les résultats du processus de Lanczos, le résidu correspondant à (2.63) vérifie

$$r_{n+1} = b - Ax_{n+1} = r_0 - AV^{(n)} z_n = r_0 - V^{(n+1)} {H_e}^{(n)} z_n = V^{(n+1)}(e_1 - {H_e}^{(n)} z_n) \tag{2.64}$$

avec $e_1 = [1\ 0\ \cdots\ 0]^T \in \mathbb{R}^{n+2}$. Ensuite, nous introduisons dans (2.64) la matrice diagonale de poids de dimension $(n+2) \times (n+2)$

$$\Omega_n = diag(\omega_0, \omega_1, \ldots, \omega_{n+1}), \qquad \omega_j > 0 \quad \forall j = 0, \ldots, n+1.$$

C'est une matrice arbitraire dont nous discuterons le choix plus tard.
L'expression du résidu r_{n+1} devient

$$\begin{aligned} r_{n+1} &= V^{(n+1)} \Omega_n^{-1} \Omega_n (e_1 - {H_e}^{(n)} z_n) \\ &= V^{(n+1)} \Omega_n^{-1} (f_n - \Omega_n {H_e}^{(n)} z_n), \qquad \text{avec } f_n = \omega_0 e_1. \end{aligned} \tag{2.65}$$

Le vecteur z_n est choisi tel que $\|r_{n+1}\|$ soit minimal. Cependant, puisque la base $\{v_j\}_{j=0}^{n+1}$ n'est pas nécessairement une base orthogonale, alors, indépendamment du choix de Ω_n, la matrice $V^{(n+1)}\Omega_n$ n'est pas toujours orthogonale. Donc, pour résoudre le problème de minimisation au sens des moindres carrés

$$\min_{x_{n+1}} \|b - Ax_{n+1}\| = \min_{z_n} \|V^{(n+1)} \Omega_n^{-1} (f_n - \Omega_n {H_e}^{(n)} z_n)\| \tag{2.66}$$

nous effectuons la factorisation QR de $V^{(n+1)}\Omega_n^{-1}$. Cela nécessite un stockage de l'ordre de $\mathcal{O}(Nn)$ et un coût algorithmique de l'ordre de $\mathcal{O}(Nn^2)$, ce qui est très coûteux.

Alors, au lieu de résoudre (2.66), nous prenons z_n solution du problème de moindres carrés suivant

$$\|f_n - \Omega_n {H_e}^{(n)} z_n\| = \min_{z \in \mathbb{R}^{n+1}} \|f_n - \Omega_n {H_e}^{(n)} z\| \tag{2.67}$$

qui est une minimisation du second facteur de la représentation (2.65) du résidu. C'est ce que l'on appelle *quasi-minimisation* du résidu, d'où le nom de cette méthode (QMR).

D'après sa définition, la matrice $\Omega_n {H_e}^{(n)}$, de dimension $(n+2) \times (n+1)$, est de rang maximal puisque les éléments de la sous-diagonale de ${H_e}^{(n)}$ sont tous différents de zéro et $\omega_j > 0$, ce qui garanti l'existence et l'unicité de z_n et définit un unique x_{n+1} grâce à (2.63).

Pour résoudre le problème de moindres carrés (2.67), nous effectuons la factorisation QR de la matrice de Hessenberg $\Omega_n {H_e}^{(n)}$ en utilisant les rotations de Givens. Ainsi nous obtenons

$$\Omega_n {H_e}^{(n)} = {Q^{(n)}}^T \begin{bmatrix} & {R_e}^{(n)} & \\ 0 & \cdots & 0 \end{bmatrix} \tag{2.68}$$

avec $Q^{(n)}$ matrice orthogonale de dimension $(n+2)\times(n+2)$ et ${R_e}^{(n)}$ triangulaire supérieure inversible de dimension $(n+1) \times (n+1)$. Donc la relation (2.67) devient

$$\begin{aligned} \min_z \left\| f_n - \Omega_n {H_e}^{(n)} z \right\| &= \min_z \left\| (Q^{(n)})^T \left(Q^{(n)} f_n - \begin{bmatrix} {R_e}^{(n)} \\ 0 \end{bmatrix} z \right) \right\|, \\ &= \min_z \left\| Q^{(n)} f_n - \begin{bmatrix} {R_e}^{(n)} \\ 0 \end{bmatrix} z \right\| = \min_z \left\| \begin{bmatrix} t^{(n+1)} \\ \tilde{\tau}_{n+2} \end{bmatrix} - \begin{bmatrix} {R_e}^{(n)} \\ 0 \end{bmatrix} z \right\|, \\ &= \min_z \left\| \begin{bmatrix} t^{(n+1)} - {R_e}^{(n)} z \\ \tilde{\tau}_{n+2} \end{bmatrix} \right\|, \end{aligned} \tag{2.69}$$

où

$$Q^{(n)} f_n = \begin{bmatrix} t^{(n+1)} \\ \tilde{\tau}_{n+2} \end{bmatrix} = \begin{bmatrix} \tau_1 \\ \vdots \\ \tau_{n+1} \\ \tilde{\tau}_{n+2} \end{bmatrix} \in \mathbb{R}^{n+2}. \tag{2.70}$$

Puisque ${R_e}^{(n)}$ est inversible, alors nous avons les deux résultats suivants

$$z_n = \left({R_e}^{(n)}\right)^{-1} t^{(n+1)} \qquad \text{et} \qquad \left\| f_n - \Omega_n {H_e}^{(n)} z \right\| = |\tilde{\tau}_{n+2}|. \tag{2.71}$$

Préconditionnement du QMR

Soit M une matrice de dimension $N \times N$, qui représente un bon préconditionneur de A. Au lieu de résoudre le système (2.1), nous appliquons l'algorithme du QMR au système équivalent

$$M^{-1} A x = M^{-1} b.$$

Cela permet de calculer les itérés x_n et les résidus $\tilde{r}_{n+1} = b_l - A_l x_{n+1}$ du système préconditionné, où $A_l = M^{-1}A$ et $b_l = M^{-1}b$.
Nous récupérons le résidu du système (1) grâce à $r_{n+1} = M\tilde{r}_{n+1}$. Ainsi le QMR avec préconditionnement est donné comme suit

Algorithme 2.3.2 : *QMR avec look-ahead préconditionné*

0) Choisir $x_0 \in \mathbb{R}^N$ et $w_0 \in \mathbb{R}^N$, $w_0 \neq 0$. Choisir $\omega_0 > 0$
$\tilde{r}_0 = M^{(-1)}(b - Ax_0)$, $v_0 = \tilde{r}_0$

Pour $n = 0, 1, \ldots$:

1) Appliquer la n^{eme} itération de l'algorithme(1.3.1)(pour la matrice A_l) pour obtenir V^{n+1}, V^n et ${H_e}^{(n)}$ vérifiant $A_l V^n = V^{n+1} {H_e}^{(n)}$.
2) Choisir $\omega_{n+1} > 0$ et faire la factorisation QR (2.68) de $\Omega_n {H_e}^{(n)}$. Déterminer le vecteur $t^{(n+1)}$ et $\tilde{r}_{n+2}$ dans (2.70)
3) Calculer

$$x_{n+1} = x_0 + V^{(n)}({R_e}^{(n)})^{-1} t^{(n+1)}; \tag{2.72}$$

4) Si x_{n+1} atteint la convergence, on arrête.

Fin.

Quelques détails de mise en œuvre du QMR

Dans cette partie, nous allons donner quelques détails sur la mise en œuvre des étapes 2), 3) et 4) de l'algorithme (2.3.2), et en particulier le calcul des itérés x_n que l'on peut développer en utilisant une relation de récurrence courte. Cette approche est basée sur une technique qui a été utilisée pour la première fois par Paige et Saunders dans les méthodes de MINRES et SYMMLQ pour le cas des matrices réelles symétriques [56].

La factorisation QR

La matrice Ω_n permet de faire un "*scaling*" de ${H_e}^{(n)}$, le choix le plus simple étant de prendre $w_j = 1$. Ici, afin d'avoir un algorithme plus stable et puisque que nous avons utilisé la version classique de Lanczos (sans normalisation), nous allons normaliser les colonnes de $V^{(n+1)}$ en prenant $\omega_j = \|v_j\|$, pour $j = 0, \ldots, n+1$.
La factorisation QR de $\Omega_n {H_e}^{(n)}$ est obtenue en utilisant les rotations de Givens, ce qui est très avantageux puisque $\Omega_n {H_e}^{(n)}$ est une matrice de Hessenberg. Nous commençons par

calculer la matrice orthogonale $Q^{(n)}$ à partir de la relation suivante

$$\begin{aligned} Q^{(n)} &= G_{n+1}\begin{bmatrix} G_n & 0 \\ 0 & 1 \end{bmatrix}\cdots\begin{bmatrix} G_1 & 0 \\ 0 & I_n \end{bmatrix} \\ &= G_{n+1}\begin{bmatrix} Q^{(n-1)} & 0 \\ 0 & 1 \end{bmatrix}, \end{aligned} \tag{2.73}$$

telle que pour tout $j = 1, \ldots, n+1$

$$G_j = \begin{bmatrix} I_{j-1} & 0 & 0 \\ 0 & c_j & s_j \\ 0 & -s_j & c_j \end{bmatrix} \quad \text{avec} \quad c_j \in I\!R, \quad s_j \in I\!R, \quad c_j^2 + s_j^2 = 1.$$

Nous notons que nous arrivons à obtenir la factorisation QR de $\Omega_n {H_e}^{(n)}$ à partir de celle de $\Omega_{n-1}{H_e}^{(n-1)}$ à l'itération précédente n. Donc à l'itération $n+1$, il suffit de connaître G_{n+1} pour pouvoir calculer $Q^{(n)}$ à partir de $Q^{(n-1)}$. Puisque G_{n+1} n'agit pas sur les n premières colonnes de $R_e^{(n)}$, alors nous pouvons obtenir $R_e^{(n)}$ à partir de $R_e^{(n-1)}$ en calculant seulement sa dernière colonne.

En effet, à partir de (2.68) et (2.73), nous avons

$$\begin{aligned} Q^{(n)}\Omega_n {H_e}^{(n)} &= G_{n+1}\begin{bmatrix} Q^{(n-1)} & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} \Omega_{n-1}H^{(n)} \\ 0 \ \cdots \ 0 \ \omega_{n+1} \end{bmatrix} \\ &= G_{n+1}\begin{bmatrix} Q^{(n-1)}(\Omega_{n-1}H^{(n)}) \\ 0 \ \cdots \ 0 \ \omega_{n+1} \end{bmatrix} = \begin{bmatrix} R_e^{(n)} \\ 0 \ \cdots \ 0 \end{bmatrix}. \end{aligned} \tag{2.74}$$

D'où

$$Q^{(n-1)}(\Omega_{n-1}H^{(n)}) = R^{(n)}$$

ce qui représente la factorisation de $\Omega_{n-1}H^{(n)}$.

Alors

$$G_{n+1}\begin{bmatrix} R^{(n)} \\ 0 \ \cdots \ 0 \ \omega_{n+1} \end{bmatrix} = \begin{bmatrix} R_e^{(n)} \\ 0 \ \cdots \ 0 \end{bmatrix}.$$

Donc, la différence entre $R^{(n)}$ et $R_e^{(n)}$ se limite aux éléments de leurs dernières colonnes. Donc pour obtenir $R_e^{(n)}$ il suffit de calculer sa dernière colonne

$$[\mu_1, \ldots, \mu_{n+1}]^T = R_e^{(n)} e_{n+1}^{(n+1)}. \tag{2.75}$$

En utilisant les premières rotations de Givens, nous obtenons

$$\begin{bmatrix} \mu_1 \\ \vdots \\ \mu_n \\ \mu \\ \nu \end{bmatrix} = \begin{bmatrix} G_n & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} G_{n-1} & 0 \\ 0 & I_2 \end{bmatrix}\cdots\begin{bmatrix} G_1 & 0 \\ 0 & I_n \end{bmatrix}\Omega_n {H_e}^{(n)}\begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}. \tag{2.76}$$

Ainsi, nous avons toutes les composantes du vecteur (2.75), sauf la dernière μ_{n+1}. Maintenant, en multipliant (2.76) par la matrice de rotation G_{n+1}, la dernière composante ($\nu = \omega_{n+1}$) de (2.76) devrait s'annuler. Alors nous devons choisir c_{n+1} et s_{n+1} tels que

$$\begin{bmatrix} c_{n+1} & s_{n+1} \\ -s_{n+1} & c_{n+1} \end{bmatrix} \begin{bmatrix} \mu \\ \nu \end{bmatrix} = \begin{bmatrix} c_{n+1}\mu + s_{n+1}\nu \\ 0 \end{bmatrix} = \begin{bmatrix} \mu_{n+1} \\ 0 \end{bmatrix}.$$

Pour réaliser cela, nous prenons

$$\begin{aligned} &c_{n+1} = \frac{|\mu|}{\sqrt{\mu^2+\nu^2}}, \qquad s_{n+1} = c_{n+1}\frac{\nu}{\mu}, \qquad \text{si} \quad \mu \neq 0, \\ &c_{n+1} = 0, \qquad\qquad s_{n+1} = 1, \qquad\qquad \text{si} \quad \mu = 0. \end{aligned} \tag{2.77}$$

De plus, nous remarquons que $|s_{n+1}\mu_{n+1}| = \omega_{n+1}$.
Le vecteur $t^{(n+1)}$ donné dans (2.70) vérifie la relation suivante

$$\begin{bmatrix} t^{(n+1)} \\ \tilde{\tau}_{n+2} \end{bmatrix} = G_{n+1} \begin{bmatrix} t^{(n)} \\ \tilde{\tau}_{n+1} \\ 0 \end{bmatrix}$$

et à partir de la définition de G_{n+1}, nous obtenons

$$\tau_{n+1} = c_{n+1}\tilde{\tau}_{n+1} \qquad \text{et} \qquad \tilde{\tau}_{n+2} = -s_{n+1}\tilde{\tau}_{n+1}. \tag{2.78}$$

Le calcul des itérés

Nous pouvons réduire (2.72) à une relation plus courte permettant de calculer x_{n+1} en fonction de x_n. Nous commençons par définir les vecteurs p_j comme suit

$$P^{(n)} = [p_0\ p_1\ \ldots\ p_n] = V^{(n)}(R_e^{(n)})^{-1}. \tag{2.79}$$

À partir de (2.70), (2.72) et (2.78), nous obtenons la relation de récurrence suivante

$$x_{n+1} = x_n + \tau_{n+1}p_n. \tag{2.80}$$

Il reste à indiquer comment calculer le vecteur p_n.
Nous regroupons $P^{(n)}$ par blocs selon la répartition (1.38) de $V^{(n)}$ comme suit

$$P^{(n)} = [P_0\ P_2\ \cdots\ P_l], \tag{2.81}$$

où

$$P_k = \begin{cases} [p_{n_k}\ p_{n_k+1}\ \cdots\ p_{n_{k+1}-1}], & \text{si} \quad 0 \leq k < l, \\ [p_{n_l}\ p_{n_l+1}\ \cdots\ p_n], & \text{si} \quad k = l. \end{cases}$$

D'après la définition (2.79) de $P^{(n)}$, nous avons

$$V^{(n)} = P^{(n)}R_e^{(n)}. \tag{2.82}$$

Comme nous l'avons déjà vu, la matrice $\Omega_n H_e^{(n)}$ est tridiagonale par bloc, donc la matrice $R_e^{(n)}$ est de la forme

$$R_e^{(n)} = \begin{bmatrix} \sigma_0 & \epsilon_1 & \vartheta_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \vartheta_l \\ & & & \ddots & \epsilon_l \\ & & & & \sigma_l \end{bmatrix},$$

où σ_j et ϵ_j ont les mêmes dimensions que λ_j et θ_j (sous-matrices de $H^{(n)}$). $R_e^{(n)}$ est non singulière, alors σ_j est aussi non singulière puisqu'elle est triangulaire supérieure. Donc (2.82) peut s'écrire sous la forme suivante

$$V_j = P_j\sigma_j + P_{j-1}\epsilon_j + P_{j-2}\vartheta_j.$$

Ainsi, nous obtenons une relation de récurrence courte pour calculer P_j

$$P_j = (V_j - P_{j-1}\epsilon_j - P_{j-2}\vartheta_j)(\sigma_j)^{-1}.$$

p_n est obtenu en l'extrayant de P_j, ce qui va nous permettre de calculer x_{n+1} selon (2.80). Enfin, pour l'étape 4) de l'algorithme (2.3.2), on arrête les itérations lorsque

$$\|r_{n+1}\| \leq tol.\|r_0\|,$$

où *tol* est la tolérance souhaitée. Il est possible de calculer le résidu par une relation de récurrence sans utiliser la définition $r_{n+1} = b - Ax_{n+1}$. En effet, en utilisant (2.65), (2.68) et (2.70) nous obtenons

$$r_{n+1} = \tilde{\tau}_{n+2}\, y_{n+1} \tag{2.83}$$

avec

$$y_{n+1} = V^{(n+1)}(\Omega_n)^{-1}(Q^{(n)})^T \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}$$

et d'après (2.73), les deux vecteurs successifs y_n et y_{n+1} vérifient la relation suivante

$$y_{n+1} = -s_{n+1}\, y_n + \frac{c_{n+1}}{\omega_{n+1}}\, v_{n+1}.$$

En utilisant le résultat (2.78), on a

$$\begin{aligned} r_{n+1} &= \tilde{\tau}_{n+2}\, y_{n+1} = (-s_{n+1})(-s_{n+1}\tilde{\tau}_{n+1})y_n + \frac{c_{n+1}\tilde{\tau}_{n+2}}{\omega_{n+1}}\, v_{n+1} \\ &= (s_{n+1})^2\, r_n + \frac{c_{n+1}\tilde{\tau}_{n+2}}{\omega_{n+1}}\, v_{n+1}. \end{aligned}$$

Le QMR est donc une méthode de sous-espace de Krylov qui utilise le *look-ahead* pour éviter le problème du *ghost-breakdown*. En plus, dans cette méthode, la décomposition LU qui génére le problème du *true-breakdown* dans la méthode du BICG est remplacée par une factorisation QR, ce qui se fait en utilisant la quasi minimisation du résidu à la place des conditions de Galerkin. Dans cette partie, nous avons traité la mise en œuvre de la méthode du QMR dans laquelle nous utilisons le processus de Lanczos avec *look-ahead* 1.3.1. Nous pouvons faire la même chose en travaillant avec le processus de Lanczos par les récurrences à deux termes (1.3.4). Dans ce cas la matrice de Hessenberg $H_e^{(n)}$ sera remplacée par sa décomposition donnée dans la sous-section 1.3.5.

2.3.3 Algorithme du BIORES non générique

Dans cette section nous allons étudier le Nongeneric BIORES [40], qui est la version sans breakdown de l'algorithme BIORES. Ce dernier utilise la relation à trois termes

$$P_{k+1} = \eta_k[(\xi - \gamma_k)P_k - \delta_k P_{k-1}]. \tag{2.84}$$

Cette relation n'est satisfaite que lorsque les polynômes P_k existent et sont de degré exactement k. Ainsi, le BIORES peut être sujet au *true* et au *ghost-breakdown*. Pour remédier à cette situation, nous allons utiliser les polynômes orthogonaux unitaires $P_k^{(0)}$ pour donner la relation de récurrence équivalente à (2.84) dans le cas général.
Nous gardons les mêmes notations que précédemment, avec $0 = \bar{n}_0 < \bar{n}_1 < \bar{n}_2 < \cdots$ les indices des polynômes réguliers unitaires $P_{\bar{n}_l}^{(0)}$ par rapport à la fonctionnelle c.
$\bar{m}_l$ est la longueur du saut qui existe entre $P_{\bar{n}_l}^{(0)}$ et $P_{\bar{n}_{l+1}}^{(0)}$, i.e $\bar{n}_{l+1} = \bar{n}_l + \bar{m}_l$.
Pour $\bar{n}_l < j < \bar{n}_{l+1}$, nous introduisons les polynômes complémentaires

$$P_j^{(0)}(\xi) = \omega_{j-\bar{n}_l}(\xi)P_{\bar{n}_l}^{(0)}(\xi) \tag{2.85}$$

appelés *insuffisants* par Gutknecht [40], avec ω_i polynôme arbitraire de degré i.
Ces polynômes servent à compléter l'ensemble des polynômes réguliers $P_{\bar{n}_l}^{(0)}$ et ils peuvent être calculés de façon récursive. Le choix le plus simple est de prendre $\omega_l(\xi) = \xi^l$, dans ce cas nous avons

$$P_{j+1}^{(0)}(\xi) = \xi P_j^{(0)}(\xi), \qquad \bar{n}_l \leq j \leq \bar{n}_{l+1} - 2.$$

Dans tout ce qui va suivre, nous prenons $\omega_l(\xi)$ le polynôme qui vérifie la relation de récurrence à trois termes suivante

$$\omega_{l+1}(\xi) = (\xi - \alpha_l^\omega)\omega_l(\xi) - \beta_l^\omega \omega_{l-1}(\xi), \tag{2.86}$$

avec $\omega_0(\xi) = 1$, $\omega_{-1}(\xi) = 0$ et $\beta_0^\omega = 0$.
Alors, les polynômes complémentaires définis par (2.85) peuvent être calculés par la relation

de récurrence suivante

$$P^{(0)}_{j+1}(\xi) = (\xi - \alpha^{\omega}_{j-\bar{n}_l})P^{(0)}_j(\xi) - \beta^{\omega}_{j-\bar{n}_l}P^{(0)}_{j-1}(\xi) \qquad \bar{n}_l \le j \le \bar{n}_{l+1} - 2. \tag{2.87}$$

Nous considérons maintenant la base des polynômes de degré $\bar{n}_{l+1} - 1$ suivante

$$\{P^{(0)}_{\bar{n}_0}, t^1_0 P^{(0)}_{\bar{n}_0}, \ldots, t^{\bar{m}_0-1}_0 P^{(0)}_{\bar{n}_0}, P^{(0)}_{\bar{n}_1}, t^1_1 P^{(0)}_{\bar{n}_1}, \ldots, t^{\bar{m}_1-1}_1 P^{(0)}_{\bar{n}_1}, \ldots, P^{(0)}_{\bar{n}_l}, t^1_l P^{(0)}_{\bar{n}_l}, \ldots, t^{\bar{m}_l-1}_l P^{(0)}_{\bar{n}_l}\},$$

où les t^j_i sont des polynômes arbitraires de degré exactement j.
En exprimant, dans cette base, le polynôme $P^{(0)}_{\bar{n}_{l+1}} - \omega_{\bar{m}_l}P^{(0)}_{\bar{n}_l}$ de degré $\bar{n}_{l+1} - 1$, nous pouvons écrire

$$\begin{aligned} P^{(0)}_{\bar{n}_{l+1}} &= \alpha^{(0)}_{\bar{n}_0}P^{(0)}_{n_0} + \alpha^{(1)}_{\bar{n}_0}t^1_0 P^{(0)}_{\bar{n}_0} + \ldots + \alpha^{(\bar{m}_0-1)}_{\bar{n}_0}t^{\bar{m}_0-1}_0 P^{(0)}_{\bar{n}_0} \\ &+ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ &+ \alpha^{(0)}_{\bar{n}_l}P^{(0)}_{\bar{n}_l} + \alpha^{(1)}_{\bar{n}_l}t^1_l P^{(0)}_{\bar{n}_l} + \ldots + \alpha^{(\bar{m}_l-1)}_{\bar{n}_l}t^{\bar{m}_l-1}_l P^{(0)}_{\bar{n}_l} \\ &+ \omega_{\bar{m}_l}P^{(0)}_{\bar{n}_l}. \end{aligned}$$

Compte-tenu des relations d'orthogonalité (2.13), nous obtenons

$$P^{(0)}_{\bar{n}_{l+1}}(\xi) = (\omega_{\bar{m}_l}(\xi) - a_l(\xi))P^{(0)}_{\bar{n}_l}(\xi) - \beta_l P^{(0)}_{\bar{n}_{l-1}}(\xi), \tag{2.88}$$

avec

$$a_l(\xi) = -\sum_{j=0}^{\bar{m}_l-1} \alpha^{(j)}_{\bar{n}_l}t^j_l(\xi) \quad \text{et} \quad \beta_l = -\alpha^{(0)}_{\bar{n}_{l-1}}.$$

a_l est un polynôme de degré $\bar{m}_l - 1$, que nous pouvons d'ailleurs exprimer dans la base $(\omega_0, \ldots, \omega_{\bar{m}_l-1})$

$$a_l(\xi) = \sum_{i=0}^{\bar{m}_l-1} \alpha_i\omega_i(\xi).$$

En multipliant (2.86) par $P^{(0)}_{\bar{n}_l}$, nous avons

$$\omega_{\bar{m}_l}P^{(0)}_{\bar{n}_l} = \xi P^{(0)}_{\bar{n}_{l+1}-1} - \alpha^{\omega}_{\bar{m}_l-1}P^{(0)}_{\bar{n}_{l+1}-1} - \beta^{\omega}_{\bar{m}_l-1}P^{(0)}_{\bar{n}_{l+1}-2}.$$

Ainsi, la récurrence (2.88) devient

$$\begin{aligned} P^{(0)}_{\bar{n}_{l+1}}(\xi) &= (\xi - \alpha_{\bar{m}_l-1} - \alpha^{\omega}_{\bar{m}_l-1})P^{(0)}_{\bar{n}_{l+1}-1}(\xi) - (\alpha_{\bar{m}_l-2} + \beta^{\omega}_{\bar{m}_l-1})P^{(0)}_{\bar{n}_{l+1}-2}(\xi) \\ &\quad -\alpha_{\bar{m}_l-3}P^{(0)}_{\bar{n}_{l+1}-3}(\xi) - \cdots - \alpha_0 P^{(0)}_{\bar{n}_l}(\xi) - \beta_l P^{(0)}_{\bar{n}_{l-1}}(\xi). \end{aligned} \tag{2.89}$$

Pour le calcul des coefficients figurant dans ces relations, nous utilisons les relations d'orthogonalité (2.13) que nous pouvons aussi écrire sous la forme suivante

$$\begin{aligned} c(P^{(0)}_k P^{(0)}_{k'}) &= 0 \quad \text{si} \quad \begin{cases} i \neq j & \text{où bien} \\ i = j \quad \text{et} \quad k + k' < \bar{n}_j + \bar{n}_{j+1} - 1 \end{cases} \\ c(P^{(0)}_k P^{(0)}_{k'}) &\neq 0 \quad \text{si} \quad i = j \quad \text{et} \quad k + k' = \bar{n}_j + \bar{n}_{j+1} - 1 \end{aligned} \tag{2.90}$$

avec $\bar{n}_i \leq k < \bar{n}_{i+1}$ et $\bar{n}_j \leq k' < \bar{n}_{j+1}$.

Pour calculer β_l, nous multiplions (2.89) par $P^{(0)}_{\bar{n}_l-1}$ et en utilisant les relations (2.90), nous obtenons

$$\beta_l c(P^{(0)}_{\bar{n}_l-1}P^{(0)}_{\bar{n}_l-1}) = c(\xi P^{(0)}_{\bar{n}_l-1}P^{(0)}_{\bar{n}_{l+1}-1}). \tag{2.91}$$

Nous procédons de la même façon pour obtenir les coefficients $(\alpha_i)_{0\leq i\leq \bar{m}_l-1}$: nous multiplions (2.89) par $P^{(0)}_{\bar{n}_l+i}$ pour $i = 0, \ldots, \bar{m}_l - 1$ et grâce aux relations (2.90), nous avons

$$\begin{aligned} \alpha_{\bar{m}_l-1}c(P^{(0)}_{\bar{n}_l}P^{(0)}_{\bar{n}_{l+1}-1}) &= c(\xi P^{(0)}_{\bar{n}_l}P^{(0)}_{\bar{n}_{l+1}-1}) - \alpha^{\omega}_{\bar{m}_l-1}c(P^{(0)}_{\bar{n}_l}P^{(0)}_{\bar{n}_{l+1}-1}) \qquad (2.92)\\ \sum_{s=1}^{i+1}\alpha_{\bar{m}_l-s}c(P^{(0)}_{\bar{n}_l+i}P^{(0)}_{\bar{n}_{l+1}-s}) &= c(\xi P^{(0)}_{\bar{n}_l+i}P^{(0)}_{\bar{n}_{l+1}-1}) - \alpha^{\omega}_{\bar{m}_l-1}c(P^{(0)}_{\bar{n}_l+i}P^{(0)}_{\bar{n}_{l+1}-1})\\ &\quad -\beta^{\omega}_{\bar{m}_l-1}c(P^{(0)}_{\bar{n}_l+i}P^{(0)}_{\bar{n}_{l+1}-2}) \end{aligned}$$

pour $i = 1, \ldots, \bar{m}_l - 1$.
Ainsi, nous obtenons un système triangulaire supérieur à résoudre. Remarquons que dans le cas où $m_l = 1$, nous avons une seule équation à résoudre.

La mise en œuvre de l'algorithme du BIORES non générique avec préconditionneur consiste à utiliser les relations de récurrence du polynôme $P^{(0)}_k$ en remplaçant directement la matrice A par $M^{-1}A$. Nous commençons par considérer les vecteurs suivants

$$u_k = P^{(0)}_k(A_r)u_0\Gamma_k, \qquad \tilde{u}'_k = P^{(0)}_k(A^T_r)z\Gamma'_k,$$

et

$$\tilde{u}_k = P^{(0)}_k(A_l)\tilde{u}_0\Gamma_l, \qquad u'_k = P^{(0)}_k(A^T_l)y\Gamma'_k,$$

où Γ_k et $\tilde{\Gamma}_k$ sont des facteurs qui seront déterminés ultérieurement, u_0 et y sont choisis arbitrairement, avec $z = M^{-T}y$ et $\tilde{u}_0 = M^{-1}u_0$.
Nous remarquons que

$$\tilde{u}_k = M^{-1}u_k, \qquad \tilde{u}'_k = M^{-T}u'_k. \tag{2.93}$$

En remplaçant dans la récurrence (2.87) ξ par A_r et A^T_r, nous obtenons

$$\begin{aligned} u_{j+1} &= (A_r u_j - \alpha^{\omega}_{j-\bar{n}_l}u_j)\gamma_{j+1,1} - \beta^{\omega}_{j-\bar{n}_l}u_{j-1}\gamma_{j+1,2},\\ \tilde{u}'_{j+1} &= (A^T_r\tilde{u}'_j - \alpha^{\omega}_{j-\bar{n}_l}\tilde{u}'_j)\gamma'_{j+1,1} - \beta^{\omega}_{j-\bar{n}_l}\tilde{u}'_{j-1}\gamma'_{j+1,2}. \end{aligned}$$

Nous faisons la même chose pour A_l et A^T_l, d'où

$$\begin{aligned} \tilde{u}_{j+1} &= (A_l\tilde{u}_j - \alpha^{\omega}_{j-\bar{n}_l}\tilde{u}_j)\gamma_{j+1,1} - \beta^{\omega}_{j-\bar{n}_l}\tilde{u}_{j-1}\gamma_{j+1,2}\\ u'_{j+1} &= (A^T_l u'_j - \alpha^{\omega}_{j-\bar{n}_l}u'_j)\gamma'_{j+1,1} - \beta^{\omega}_{j-\bar{n}_l}u'_{j-1}\gamma'_{j+1,2}. \end{aligned}$$

D'après (2.93), ces relations s'écrivent aussi

$$\begin{aligned} u_{j+1} &= (A\tilde{u}_j - \alpha^{\omega}_{j-\bar{n}_l} u_j)\gamma_{j+1,1} - \beta^{\omega}_{j-\bar{n}_l} u_{j-1}\gamma_{j+1,2} \\ \tilde{u}'_{j+1} &= (M^{-T}A^T\tilde{u}'_j - \alpha^{\omega}_{j-\bar{n}_l}\tilde{u}'_j)\gamma'_{j+1,1} - \beta^{\omega}_{j-\bar{n}_l}\tilde{u}'_{j-1}\gamma'_{j+1,2} \end{aligned} \tag{2.94}$$

et

$$\begin{aligned} \tilde{u}_{j+1} &= (M^{-1}A\tilde{u}_j - \alpha^{\omega}_{j-\bar{n}_l}\tilde{u}_j)\gamma_{j+1,1} - \beta^{\omega}_{j-\bar{n}_l}\tilde{u}_{j-1}\gamma_{j+1,2} \\ u'_{j+1} &= (A^T\tilde{u}'_j - \alpha^{\omega}_{j-\bar{n}_l}u'_j)\gamma'_{j+1,1} - \beta^{\omega}_{j-\bar{n}_l}u'_{j-1}\gamma'_{j+1,2} \end{aligned} \tag{2.95}$$

$$\text{pour} \qquad \bar{n}_l \le j \le \bar{n}_{l+1} - 2,$$

où

$$\gamma_{n,i} = \Gamma_n/\Gamma_{n-i}, \qquad \gamma'_{n,i} = \Gamma'_n/\Gamma'_{n-i} \quad (n \in I\!N, i \in I\!N),$$

avec

$$\Gamma_0 = \Gamma'_0 = 1$$

et

$$\begin{aligned} \Gamma_n &= \gamma_{1,1}\gamma_{2,1}\cdots\gamma_{n,1} \qquad \gamma_{n,i} = \gamma_{n,1}\gamma_{n-1,1}\cdots\gamma_{n-i+1,1} \\ \Gamma'_n &= \gamma'_{1,1}\gamma'_{2,1}\cdots\gamma'_{n,1} \qquad \gamma'_{n,i} = \gamma'_{n,1}\gamma'_{n-1,1}\cdots\gamma'_{n-i+1,1}. \end{aligned} \tag{2.96}$$

De même, à partir de la relation (2.89), nous obtenons en utilisant (2.93)

$$\begin{aligned} u_{\bar{n}_{l+1}} &= [A\tilde{u}_{\bar{n}_{l+1}-1} - (\alpha_{\bar{m}_l-1} + \alpha^{\omega}_{\bar{m}_l-1})u_{\bar{n}_{l+1}-1}]\gamma_{\bar{n}_{l+1},1} \\ &\quad -(\alpha_{\bar{m}_l-2} + \beta^{\omega}_{\bar{m}_l-1})u_{\bar{n}_{l+1}-2}\gamma_{\bar{n}_{l+1},2} - \alpha_{\bar{m}_l-3}u_{\bar{n}_{l+1}-3}\gamma_{\bar{n}_{l+1},3} - \cdots \\ &\quad -\alpha_0 u_{\bar{n}_l}\gamma_{\bar{n}_{l+1},\bar{m}_l} - \beta_l u_{\bar{n}_{l-1}}\gamma_{\bar{n}_{l+1},\bar{m}_l+\bar{m}_{l-1}} \end{aligned}$$

$$\begin{aligned} \tilde{u}'_{\bar{n}_{l+1}} &= [M^{-T}A^T\tilde{u}'_{\bar{n}_{l+1}-1} - (\alpha_{\bar{m}_l-1} + \alpha^{\omega}_{\bar{m}_l-1})\tilde{u}'_{\bar{n}_{l+1}-1}]\gamma'_{\bar{n}_{l+1},1} \\ &\quad -(\alpha_{\bar{m}_l-2} + \beta^{\omega}_{\bar{m}_l-1})\tilde{u}'_{\bar{n}_{l+1}-2}\gamma'_{\bar{n}_{l+1},2} - \alpha_{\bar{m}_l-3}u'_{\bar{n}_{l+1}-3}\gamma'_{\bar{n}_{l+1},3} - \cdots \\ &\quad -\alpha_0\tilde{u}'_{\bar{n}_l}\gamma'_{\bar{n}_{l+1},\bar{m}_l} - \beta_l\tilde{u}'_{\bar{n}_{l-1}}\gamma'_{\bar{n}_{l+1},\bar{m}_l+\bar{m}_{l-1}}, \end{aligned}$$

ainsi que

$$\begin{aligned} \tilde{u}_{\bar{n}_{l+1}} &= [M^{-1}A\tilde{u}_{\bar{n}_{l+1}-1} - (\alpha_{\bar{m}_l-1} + \alpha^{\omega}_{\bar{m}_l-1})\tilde{u}_{\bar{n}_{l+1}-1}]\gamma_{\bar{n}_{l+1},1} \\ &\quad -(\alpha_{\bar{m}_l-2} + \beta^{\omega}_{\bar{m}_l-1})\tilde{u}_{\bar{n}_{l+1}-2}\gamma_{\bar{n}_{l+1},2} - \alpha_{\bar{m}_l-3}\tilde{u}_{\bar{n}_{l+1}-3}\gamma_{\bar{n}_{l+1},3} - \cdots \\ &\quad -\alpha_0\tilde{u}_{\bar{n}_l}\gamma_{\bar{n}_{l+1},\bar{m}_l} - \beta_l\tilde{u}_{\bar{n}_{l-1}}\gamma_{\bar{n}_{l+1},\bar{m}_l+\bar{m}_{l-1}} \end{aligned}$$

$$\begin{aligned} u'_{\bar{n}_{l+1}} &= [A^T\tilde{u}'_{\bar{n}_{l+1}-1} - (\alpha_{\bar{m}_l-1} + \alpha^{\omega}_{\bar{m}_l-1})u'_{\bar{n}_{l+1}-1}]\gamma'_{\bar{n}_{l+1},1} \\ &\quad -(\alpha_{\bar{m}_l-2} + \beta^{\omega}_{\bar{m}_l-1})u'_{\bar{n}_{l+1}-2}\gamma'_{\bar{n}_{l+1},2} - \alpha_{\bar{m}_l-3}u'_{\bar{n}_{l+1}-3}\gamma'_{\bar{n}_{l+1},3} - \cdots \\ &\quad -\alpha_0 u'_{\bar{n}_l}\gamma'_{\bar{n}_{l+1},\bar{m}_l} - \beta_l u'_{\bar{n}_{l-1}}\gamma'_{\bar{n}_{l+1},\bar{m}_l+\bar{m}_{l-1}}. \end{aligned}$$

Les coefficients β_l et $(\alpha_i)_{0\leq i\leq \bar{m}_l-1}$ sont déterminés grâce à (2.91) et (2.92), que nous pouvons aussi écrire, d'après la définition de la fonctionnelle c, comme suit

$$\begin{aligned}
\beta_l \gamma_{\bar{n}_{l+1}-1,\bar{m}_l+\bar{m}_{l-1}-1}(\tilde{u}'_{\bar{n}_l-1}, u_{n_{l-1}}) &= (\tilde{u}'_{\bar{n}_l-1}, A\tilde{u}_{\bar{n}_{l+1}-1}) \\
\alpha_{\bar{m}_l-1}(\tilde{u}'_{\bar{n}_l}, u_{\bar{n}_{l+1}-1}) &= (\tilde{u}'_{\bar{n}_l}, A\tilde{u}_{\bar{n}_{l+1}-1}) - \alpha^{\omega}_{\bar{m}_l-1}(\tilde{u}'_{\bar{n}_l}, u_{\bar{n}_{l+1}-1}) \\
\sum_{s=1}^{i+1} \alpha_{\bar{m}_l-s}\gamma_{\bar{n}_{l+1}-1,s-1}(\tilde{u}'_{\bar{n}_l+i}, u_{\bar{n}_{l+1}-s}) &= (\tilde{u}'_{\bar{n}_l+i}, A\tilde{u}_{\bar{n}_{l+1}-1}) - \alpha^{\omega}_{\bar{m}_l-1}(\tilde{u}'_{\bar{n}_l+i}, u_{\bar{n}_{l+1}-1}) \\
&\quad -\beta^{\omega}_{\bar{m}_l-1}\gamma_{n_{l+1}-1,1}(\tilde{u}'_{\bar{n}_l+i}, u_{\bar{n}_{l+1}-2})
\end{aligned}$$

pour $i = 1, \ldots, \bar{m}_l - 1$.

Pour la mise en œuvre de cet algorithme, nous allons nous contenter d'utiliser les récurrences de u_k et u'_k alors que $\tilde{u}_k$ et $\tilde{u}'_k$ seront calculés à partir de (2.93). En rassemblant toutes ces formules nous obtenons l'algorithme suivant

Algorithme 2.3.3 *:Nongeneric BIORES avec préconditionneur.*

1. Initialisation
   Choisir u_0 et y
   $u'_0 = y, \quad \bar{n}_0 = 0$
2. Itérations :
   Pour $l = 0, 1 \ldots$
   Résoudre $M\tilde{u}_{\bar{n}_l} = u_{\bar{n}_l}$ et $M^T\tilde{u}'_{\bar{n}_l} = u'_{\bar{n}_l}$
   $\bar{m}_l = 1 + \min\{i \in I\!N$ tel que $(\tilde{u}'_{\bar{n}_l+i}, u_{\bar{n}_l}) \neq 0\}$
   $\bar{n}_{l+1} = \bar{n}_l + \bar{m}_l$

   – Tant que $\bar{n}_l \leq j \leq \bar{n}_{l+1} - 2$
   $u_{j+1} = (A\tilde{u}_j - \alpha^{\omega}_{j-\bar{n}_l}u_j)\gamma_{j+1,1} - \beta^{\omega}_{j-\bar{n}_l}u_{j-1}\gamma_{j+1,2}$,
   $u'_{j+1} = (A^T\tilde{u}'_j - \alpha^{\omega}_{j-\bar{n}_l}u'_j)\gamma'_{j+1,1} - \beta^{\omega}_{j-\bar{n}_l}u'_{j-1}\gamma'_{j+1,2}$,
   Résoudre $M\tilde{u}_{j+1} = u_{j+1}$ et $M^T\tilde{u}'_{j+1} = u'_{j+1}$
   – Déterminer les paramètres β_l et $(\alpha_i)_{1\leq i\leq \bar{m}_l-1}$ par (2.97)

   $$\begin{aligned}
   u_{\bar{n}_{l+1}} = [&A\tilde{u}_{\bar{n}_{l+1}-1} - (\alpha_{\bar{m}_l-1} + \alpha^{\omega}_{\bar{m}_l-1})u_{\bar{n}_{l+1}-1}]\gamma_{\bar{n}_{l+1},1} \\
   &-(\alpha_{\bar{m}_l-2} + \beta^{\omega}_{\bar{m}_l-1})u_{\bar{n}_{l+1}-2}\gamma_{\bar{n}_{l+1},2} - \alpha_{\bar{m}_l-3}u_{\bar{n}_{l+1}-3}\gamma_{\bar{n}_{l+1},3} - \cdots \\
   &-\alpha_0 u_{\bar{n}_l}\gamma_{\bar{n}_{l+1},\bar{m}_l} - \beta_l u_{\bar{n}_{l-1}}\gamma_{\bar{n}_{l+1},\bar{m}_l+\bar{m}_{l-1}},
   \end{aligned}$$

$$
\begin{aligned}
u'_{\bar{n}_{l+1}} = [A^T \tilde{u}'_{\bar{n}_{l+1}-1} &- (\alpha_{\bar{m}_l-1} + \alpha^{\omega}_{\bar{m}_l-1}) u'_{\bar{n}_{l+1}-1}]\gamma'_{\bar{n}_{l+1},1} \\
&-(\alpha_{\bar{m}_l-2} + \beta^{\omega}_{\bar{m}_l-1}) u'_{\bar{n}_{l+1}-2}\gamma'_{\bar{n}_{l+1},2} - \alpha_{\bar{m}_l-3} u'_{\bar{n}_{l+1}-3}\gamma'_{\bar{n}_{l+1},3} - \cdots \\
&-\alpha_0 u'_{\bar{n}_l}\gamma'_{\bar{n}_{l+1},\bar{m}_l} - \beta_l u'_{\bar{n}_{l-1}}\gamma'_{\bar{n}_{l+1},\bar{m}_l+\bar{m}_{l-1}},
\end{aligned}
$$

Fin (pour).

Cet algorithme peut être utilisé pour la résolution du système $M^{-1}Ax = M^{-1}b$ en construisant une suite de vecteurs z_k et une suite de scalaires ρ_k liées par la relation

$$u_k = \rho_k b - Az_k, \tag{2.97}$$

définis comme suit

$$
\begin{aligned}
z_{j+1} &= -(\tilde{u}_j + \alpha^{\omega}_{j-\bar{n}_l} z_j)\gamma_{j+1,1} - \beta^{\omega}_{j-\bar{n}_l} z_{j-1}\gamma_{j+1,2}, \\
\rho_{j+1} &= -\alpha^{\omega}_{j-\bar{n}_l}\rho_j\gamma_{j+1,1} - \beta^{\omega}_{j-\bar{n}_l}\rho_{j-1}\gamma_{j+1,2} \qquad \text{pour} \quad \bar{n}_l \le j \le \bar{n}_{l+1} - 2,
\end{aligned}
$$

$$
\begin{aligned}
z_{\bar{n}_{l+1}} &= -[\tilde{u}_{\bar{n}_{l+1}-1} + (\alpha_{\bar{m}_l-1} + \alpha^{\omega}_{\bar{m}_l-1}) z_{\bar{n}_{l+1}-1}]\gamma_{\bar{n}_{l+1},1} \\
&\quad -(\alpha_{\bar{m}_l-2} + \beta^{\omega}_{\bar{m}_l-1}) z_{\bar{n}_{l+1}-2}\gamma_{\bar{n}_{l+1},2} - \alpha_{\bar{m}_l-3} z_{\bar{n}_{l+1}-3}\gamma_{\bar{n}_{l+1},3} - \cdots \\
&\quad -\alpha_0 z_{\bar{n}_l}\gamma_{\bar{n}_{l+1},\bar{m}_l} - \beta_l z_{\bar{n}_{l-1}}\gamma_{\bar{n}_{l+1},\bar{m}_l+\bar{m}_{l-1}}, \\
\rho_{\bar{n}_{l+1}} &= -(\alpha_{\bar{m}_l-1} + \alpha^{\omega}_{\bar{m}_l-1})\rho_{\bar{n}_{l+1}-1}\gamma_{\bar{n}_{l+1},1} - (\alpha_{\bar{m}_l-2} + \beta^{\omega}_{\bar{m}_l-1})\rho_{\bar{n}_{l+1}-2}\gamma_{\bar{n}_{l+1},2} \\
&\quad -\alpha_{\bar{m}_l-3}\rho_{\bar{n}_{l+1}-3}\gamma_{\bar{n}_{l+1},3} - \cdots - \alpha_0\rho_{\bar{n}_l}\gamma_{\bar{n}_{l+1},\bar{m}_l} - \beta_l\rho_{\bar{n}_{l-1}}\gamma_{\bar{n}_{l+1},\bar{m}_l+\bar{m}_{l-1}}.
\end{aligned}
\tag{2.98}
$$

Les approximations x_k de la solution exacte $x = A^{-1}b$ sont données par $x_k = z_k/\rho_k$.

Explication

D'après le Lemme 1.1.3, nous avons

$$P_k(\xi) = \frac{P_k^{(0)}(\xi)}{P_k^{(0)}(0)}.$$

Le résidu est défini par $r_k = P_k(A_r)r_0$. Si nous notons $\rho_k = P_k^{(0)}(0)\Gamma_k$, alors nous pouvons écrire

$$r_k = \frac{u_k}{\rho_k}.$$

Le résidu r_k s'écrit aussi $r_k = b - Ax_k$, d'où la relation (2.97), avec $z_k = \rho_k x_k$.
Les relations de récurrence (2.87) et (2.89) permettent de calculer ρ_k. Par contre z_k est calculé en remplaçant u_k par son expression dans sa relation de récurrence.
Ainsi, d'après (2.87), pour $\bar{n}_l \le j \le \bar{n}_{l+1} - 2$, nous avons

$$
\begin{aligned}
\rho_{j+1} &= P_{j+1}^{(0)}(0)\Gamma_{j+1} = -\alpha^{\omega}_{j-\bar{n}_l}\rho_j\frac{\Gamma_{j+1}}{\Gamma_j} - \beta^{\omega}_{j-\bar{n}_l}\rho_{j-1}\frac{\Gamma_{j+1}}{\Gamma_{j-1}} \\
&= -\alpha^{\omega}_{j-\bar{n}_l}\rho_j\gamma_{j+1,1} - \beta^{\omega}_{j-\bar{n}_l}\rho_{j-1}\gamma_{j+1,2}
\end{aligned}
$$

et

$$\begin{aligned} u_{j+1} &= (A\tilde{u}_j - \alpha^{\omega}_{j-\bar{n}_l} u_j)\gamma_{j+1,1} - \beta^{\omega}_{j-\bar{n}_l} u_{j-1}\gamma_{j+1,2} \\ &= [A\tilde{u}_j - \alpha^{\omega}_{j-\bar{n}_l}(b\rho_j - Az_j)]\gamma_{j+1,1} - \beta^{\omega}_{j-\bar{n}_l}(b\rho_{j-1} - Az_{j-1})\gamma_{j+1,2} \\ &= A[(\tilde{u}_j + \alpha^{\omega}_{j-\bar{n}_l} z_j)\gamma_{j+1,1} + \beta^{\omega}_{j-\bar{n}_l} z_{j-1}\gamma_{j+1,2}] + b\rho_{j+1}. \end{aligned}$$

En utilisant (2.97), nous obtenons

$$z_{j+1} = -(\tilde{u}_j + \alpha^{\omega}_{j-\bar{n}_l} z_j)\gamma_{j+1,1} - \beta^{\omega}_{j-\bar{n}_l} z_{j-1}\gamma_{j+1,2},$$

et de façon similaire, nous obtenons les expressions de $z_{\bar{n}_{l+1}}$ et $\rho_{\bar{n}_{l+1}}$, voir (2.98).
Les paramètres α^{ω}_l et β^{ω}_l figurant dans cet algorithme sont choisis de façon arbitraire. Les facteurs $\gamma_{l+1,i}$ ("scale factors") doivent vérifier (2.96). Il est clair que si nous prenons $\gamma_{n,1} = \gamma'_{n,1} = 1, \quad \forall n$, c'est-à-dire

$$\Gamma_n = \gamma_{n,i} = \Gamma'_n = \gamma'_{n,i} = 1 \quad (\forall\, n, i),$$

l'algorithme se simplifie. Mais dans la pratique, ce choix peut causer un problème de dépassement de capacité "overflow" ou "underflow" il est même conseillé de normaliser les deux suites de vecteurs biorthogonaux u_k et u'_k par deux suites de facteurs Γ_k et Γ'_k différents.

Remarque 2.3.7
Cet algorithme subit un true-breakdown à l'itération k lorsque $\rho_k = 0$ et un incurable-breakdown lorsque $(u'_{N-1}, u_{\bar{n}_l}) = 0$, où N est la dimension du système étudié.

Par définition $\rho_k = P^{(0)}_k(0)\Gamma_l$ et, d'après le Théorème 1.1.2, nous avons

$$P^{(0)}_k(0) = 0 \quad \text{si et seulement si} \quad H^{(1)}_k = 0.$$

Supposons qu'à l'itération $\bar{n}_l$ nous ayons $\rho_{\bar{n}_l} \neq 0$, alors $H^{(1)}_{\bar{n}_l} \neq 0$. Or, nous savons, d'après le Lemme 1.2.2, que si $H^{(1)}_{\bar{n}_{l+1}} = 0$ alors $H^{(1)}_{\bar{n}_{l+2}} \neq 0$.
Cela veut dire que si nous avons un *true-breakdow* à l'itération $\bar{n}_{l+1}$ ($\rho_{\bar{n}_{l+1}} = 0$), ce qui n'empêche pas le calcul des itérés de l'algorithme 2.3.3, alors, en calculant ceux de l'itération $\bar{n}_{l+2}$ nous sommes sûrs d'avoir $\rho_{\bar{n}_{l+2}} \neq 0$. Il suffit donc, dans le cas d'un *true-breakdown*, d'effectuer un deuxième saut.

2.3.4 Algorithme HMRZ-stab

L'algorithme Lanczos/Orthodir [81] utilise les relations de récurrence suivantes

$$\begin{aligned} P_{k+1}(\xi) &= P_k(\xi) - \lambda_{k+1} P^{(1)}_k(\xi) \\ P^{(1)}_{k+1}(\xi) &= (\xi - \beta_{k+1}) P^{(1)}_k(\xi) - \gamma_{k+1} P^{(1)}_{k-1}(\xi) \end{aligned} \tag{2.99}$$

où les coefficients λ_{k+1}, β_{k+1} et γ_{k+1} sont calculés sous la condition $H_k^{(1)} \neq 0, \forall\ k$.
Dans le cas contraire, il se produit un *true-breakodwn* qui arrête l'algorithme. Le *ghost-breakdown*, lui, provoque une stagnation dans l'algorithme mais n'empêche pas le calcul des itérés. Le MRZ, HMRZ, MRZ-stab et HMRZ-stab [17] sont des versions sans *breakdown* de cet algorithme. Ici, nous nous intéressons aux algorithmes MRZ-stab et HMRZ-stab qui sont les plus stables parmi toutes les variantes de MRZ.
D'après Draux [25], Struble [74] et Brezinski, Redivo-Zagla et Sadok [15], nous avons les relations

$$\begin{aligned} P_{n_{k+1}}^{(1)}(\xi) &= q_k(\xi)P_{n_k}^{(1)}(\xi) - C_{k+1}P_{n_{k-1}}^{(1)}(\xi), \\ P_{n_{k+1}}(\xi) &= P_{n_k}(\xi) - \xi\omega_k(\xi)P_{n_k}^{(1)}(\xi), \end{aligned} \tag{2.100}$$

où $P_{-1}^{(1)}(\xi) = 0$, $P_0^{(1)}(\xi) = 1$, $C_1 = 0$.
q_k est un polynôme unitaire de degré m_k et ω_k un polynôme de degré au plus $m_k - 1$.
Nous posons

$$\begin{aligned} \omega_k(\xi) &= \beta_0^{(k)} + \cdots + \beta_{m_k-1}^{(k)}\xi^{m_k-1} \\ q_k(\xi) &= \alpha_0^{(k)} + \cdots + \alpha_{m_k-1}^{(k)}\xi^{m_k-1} + \xi^{m_k}. \end{aligned}$$

Les coefficients $(\beta_i^{(k)})_{0\leq i\leq m_k-1}$, $(\alpha_i^{(k)})_{0\leq i\leq m_k-1}$ et C_{k+1}, sont obtenus en multipliant respectivement les deux relations de récurrence (2.100) par un polynôme U_i de degré i et en appliquant respectivement les fonctionnelles $c^{(1)}$ et c

$$\begin{aligned} c^{(1)}(U_iP_{n_{k+1}}^{(1)}) &= \alpha_0^{(k)}c^{(1)}(U_iP_{n_k}^{(1)}) + \cdots + \alpha_{m_k-1}^{(k)}c^{(1)}(\xi^{m_k-1}U_iP_{n_k}^{(1)}) + c^{(1)}(\xi^{m_k}U_iP_{n_k}^{(1)}) \\ &\quad -C_{k+1}c^{(1)}(U_iP_{n_{k-1}}^{(1)}) \\ c(U_iP_{n_{k+1}}) &= c(U_iP_{n_k}) - \beta_0^{(k)}c(\xi U_iP_{n_k}^{(1)}) - \cdots - \beta_{m_k-1}^{(k)}c(\xi^{m_k}U_iP_{n_k}^{(1)}). \end{aligned} \tag{2.101}$$

Le choix des polynômes arbitraires U_i permet de définir différentes variantes de la méthode MRZ [13, 17]. Le choix $U_i = \xi^i$ nous donne les algorithmes MRZ et HMRZ [13].
Ici, nous prenons $U_i = \xi^{i-n_k}P_{n_k}^{(1)}$. Les relations d'orthogonalité (2.12) peuvent être remplacées par

$$\begin{aligned} c(\xi^i{P_{n_k}^{(1)}}^2) &= 0 \qquad \text{pour} \qquad i = 0, \ldots, m_k - 2 \\ &\neq 0 \qquad \text{pour} \qquad i = m_k - 1. \end{aligned} \tag{2.102}$$

Les coefficients $(\beta_i^{(k)})_{0\leq i\leq m_k-1}$ et $(\alpha_i^{(k)})_{0\leq i\leq m_k-1}$ se calculent en imposant les orthogonalités (2.102) dans (2.101) pour $i = n_k, \ldots, n_k + m_k - 1$, ainsi nous obtenons les deux systèmes

$$\begin{aligned} &\beta_{m_k-1}^{(k)}c^{(1)}(\xi^{m_k-1}{P_{n_k}^{(1)}}^2) = c(P_{n_k}^{(1)}P_{n_k}) \\ &\beta_{m_k-2}^{(k)}c^{(1)}(\xi^{m_k-1}{P_{n_k}^{(1)}}^2) + \beta_{m_k-1}^{(k)}c^{(1)}(\xi^{m_k}{P_{n_k}^{(1)}}^2) = c(\xi P_{n_k}^{(1)}P_{n_k}) \\ &\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ &\beta_0^{(k)}c^{(1)}(\xi^{m_k-1}{P_{n_k}^{(1)}}^2) + \cdots + \beta_{m_k-1}^{(k)}c^{(1)}(\xi^{2m_k-2}{P_{n_k}^{(1)}}^2) = c(\xi^{m_k-1}P_{n_k}^{(1)}P_{n_k}). \end{aligned} \tag{2.103}$$

et

$$\begin{aligned}
&\alpha^{(k)}_{m_k-1}c^{(1)}(\xi^{m_k-1}{P^{(1)}_{n_k}}^2) = -c^{(1)}(\xi^{m_k}{P^{(1)}_{n_k}}^2)\\
&\alpha^{(k)}_{m_k-2}c^{(1)}(\xi^{m_k-1}{P^{(1)}_{n_k}}^2) + \alpha^{(k)}_{m_k-1}c^{(1)}(\xi^{m_k}{P^{(1)}_{n_k}}^2) = -c^{(1)}(\xi^{m_k+1}{P^{(1)}_{n_k}}^2) \qquad (2.104)\\
&\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\\
&\alpha^{(k)}_{0}c^{(1)}(\xi^{m_k-1}{P^{(1)}_{n_k}}^2) + \cdots + \alpha^{(k)}_{m_k-1}c^{(1)}(\xi^{2m_k-2}{P^{(1)}_{n_k}}^2) = -c^{(1)}(\xi^{2m_k-1}{P^{(1)}_{n_k}}^2).
\end{aligned}$$

Pour le calcul de C_{k+1}, nous prenons $U_i = \xi^{m_{k-1}-1}P^{(1)}_{n_{k-1}}$, ce qui nous donne

$$c^{(1)}(\xi^{m_k+m_{k-1}-1}P^{(1)}_{n_k}P^{(1)}_{n_{k-1}}) = C_{k+1}c^{(1)}(\xi^{m_{k-1}-1}{P^{(1)}_{n_{k-1}}}^2).$$

Le polynôme $P^{(1)}_{n_k}$ est de degré n_k. Alors il existe un polynôme $\tilde{q}_{n_k-1}$ de degré au plus $n_k - 1$ tel que

$$P^{(1)}_{n_k} = \xi^{m_{k-1}}P^{(1)}_{n_{k-1}} + \tilde{q}_{n_k-1}$$

et d'après les orthogonalités (2.12), nous avons

$$\begin{aligned}
c^{(1)}(\xi^{m_k-1}{P^{(1)}_{n_k}}^2) &= c^{(1)}(\xi^{m_k+m_{k-1}-1}P^{(1)}_{n_k}P^{(1)}_{n_{k-1}}) + c^{(1)}(\xi^{m_k-1}P^{(1)}_{n_k}\tilde{q}_{n_k-1})\\
&= c^{(1)}(\xi^{m_k+m_{k-1}-1}P^{(1)}_{n_k}P^{(1)}_{n_{k-1}}),
\end{aligned}$$

d'où

$$C_{k+1} = \frac{c^{(1)}(\xi^{m_k-1}{P^{(1)}_{n_k}}^2)}{c^{(1)}(\xi^{m_{k-1}-1}{P^{(1)}_{n_{k-1}}}^2)}.$$

En posant

$$\begin{aligned}
r_k &= P_{n_k}(A_r)r_0 \qquad z_k = P^{(1)}_{n_k}(A_r)r_0,\\
\tilde{r}_k &= P_{n_k}(A_l)\tilde{r}_0 \qquad \tilde{z}_k = P^{(1)}_{n_k}(A_l)\tilde{r}_0,\\
z'_k &= P^{(1)}_{n_k}(A_r^T)z \qquad \tilde{z}'_k = P^{(1)}_{k}(A_l^T)y,
\end{aligned}$$

nous remarquons que

$$z'_k = M^{-T}\tilde{z}'_k \quad \text{et} \quad \tilde{z}_k = M^{-1}z_k.$$

Si nous posons

$$\begin{aligned}
&\text{pour} \quad i = 0,\ldots,m_k \qquad \tilde{z}_{k,i} = M^{-1}z_{k,i} \quad \text{et} \quad z_{k,i+1} = A\tilde{z}_{k,i}\\
&\text{pour} \quad j = 0,\ldots,m_k - 1 \qquad z'_{k,j} = M^{-T}\tilde{z}'_{k,j} \quad \text{et} \quad \tilde{z}'_{k,j+1} = A^T z'_{k,j}
\end{aligned}$$

avec $z_{k,0} = z_k$, $\tilde{z}_{k,0} = \tilde{z}_k$, $\tilde{z}'_{k,0} = \tilde{z}'_k$ et $z'_{k,0} = z'_k$, alors

$$\begin{aligned}
z_{k,i} &= {A_r}^i z_k, \qquad \tilde{z}_{k,i} = {A_l}^i\tilde{z}_k\\
\text{et} \quad z'_{k,i} &= {A_r^T}^i z'_k \qquad \tilde{z}'_{k,i} = {A_l^T}^i\tilde{z}'_k.
\end{aligned}$$

Les récurrences (2.100) donnent

$$\begin{aligned} x_{k+1} &= x_k + [\beta_0^{(k)}\tilde{z}_{k,0} + \beta_1^{(k)}\tilde{z}_{k,1} + \cdots + \beta_{m_k-1}^{(k)}\tilde{z}_{k,m_k-1}], \\ r_{k+1} &= r_k - [\beta_0^{(k)}z_{k,1} + \beta_1^{(k)}z_{k,2} + \cdots + \beta_{m_k-1}^{(k)}z_{k,m_k}], \\ z_{k+1} &= \alpha_0^{(k)}z_{k,0} + \alpha_1^{(k)}z_{k,1} + \cdots + \alpha_{m_k-1}^{(k)}z_{k,m_k-1} + z_{k,m_k} - C_{k+1}z_{k-1}, \\ z'_{k+1} &= \alpha_0^{(k)}z'_{k,0} + \alpha_1^{(k)}z'_{k,1} + \cdots + \alpha_{m_k-1}^{(k)}z'_{k,m_k-1} + z'_{k,m_k} - C_{k+1}z'_{k-1}, \end{aligned} \tag{2.105}$$

ainsi que

$$\begin{aligned} x_{k+1} &= x_k + [\beta_0^{(k)}\tilde{z}_{k,0} + \beta_1^{(k)}\tilde{z}_{k,1} + \cdots + \beta_{m_k-1}^{(k)}\tilde{z}_{k,m_k-1}], \\ \tilde{r}_{k+1} &= \tilde{r}_k - [\beta_0^{(k)}\tilde{z}_{k,1} + \beta_1^{(k)}\tilde{z}_{k,2} + \cdots + \beta_{m-1}^{(k)}\tilde{z}_{k,m_k}], \\ \tilde{z}_{k+1} &= \alpha_0^{(k)}\tilde{z}_{k,0} + \alpha_1^{(k)}\tilde{z}_{k,1} + \cdots + \alpha_{m-1}^{(k)}\tilde{z}_{k,m_k-1} + \tilde{z}_{k,m_k} - C_{k+1}\tilde{z}_{k-1}, \\ \tilde{z}'_{k+1} &= \alpha_0^{(k)}\tilde{z}'_{k,0} + \alpha_1^{(k)}\tilde{z}'_{k,1} + \cdots + \alpha_{m_k-1}^{(k)}\tilde{z}'_{k,m_k-1} + \tilde{z}'_{k,m_k} - C_{k+1}\tilde{z}'_{k-1}. \end{aligned} \tag{2.106}$$

De même, les deux systèmes (2.103) et (2.104) deviennent

$$\begin{cases} \beta_{m_k-1}^{(k)}b_0^{(k)} = d_0^{(k)} \\ \beta_{m_k-2}^{(k)}b_0^{(k)} + \beta_{m_k-1}^{(k)}b_1^{(k)} = d_1^{(k)} \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ \beta_0^{(k)}b_0^{(k)} + \cdots + \beta_{m_k-1}^{(k)}b_{m_k-1}^{(k)} = d_{m_k-1}^{(k)} \end{cases} \tag{2.107}$$

et

$$\begin{cases} b_0^{(k)} = C_{k+1}b_0^{(k-1)} \\ \alpha_{m_k-1}^{(k)}b_0^{(k)} + b_1^{(k)} = 0 \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ \alpha_0^{(k)}\ b_0^{(k)} + \alpha_1^{(k)}b_1^{(k)} + \cdots + \alpha_{m_k-1}^{(k)}b_{m_k-1}^{(k)} + b_{m_k}^{(k)} = 0, \end{cases} \tag{2.108}$$

avec

$$\begin{aligned} d_i^{(k)} = c(\xi^i P_{n_k}^{(1)} P_{n_k}) &= (P_{n_k}^{(1)}(A_r^T)z, A_r^i P_{n_k}(A_r)r_0) = (z'_k, A_r^i r_k) = (z'_{k,i}, r_k) \\ &\quad (P_{n_k}^{(1)}(A_l^T)y, A_l^i P_{n_k}(A_l)\tilde{r}_0) = (\tilde{z}'_k, A_l^i\tilde{r}_k) = (\tilde{z}'_{k,i}, \tilde{r}_k), \end{aligned} \tag{2.109}$$

$$\begin{aligned} b_i^{(k)} &= c^{(1)}(\xi^{m_k+i-1}{P_{n_k}^{(1)}}^2) \\ &= (P_{n_k}^{(1)}(A_r^T)z, A_r^{m_k+i}P_{n_k}^{(1)}(A_r)r_0) = (z'_k, A_r^{m_k+i}z_k) = (z'_{k,i}, z_{k,m_k}) \\ &\quad (P_{n_k}^{(1)}(A_l^T)y, A_l^{m_k+i}P_{n_k}^{(1)}(A_l)\tilde{r}_0) = (\tilde{z}'_k, A_l^{m_k+i}\tilde{z}_k) = (\tilde{z}'_{k,i}, \tilde{z}_{k,m_k}). \end{aligned} \tag{2.110}$$

Nous obtenons deux systèmes à résoudre que nous pouvons réunir en un seul système avec deux seconds membres différents selon le schéma suivant

$$
\begin{array}{cccccc}
\beta_0^{(k)} & \beta_1^{(k)} & \cdots & \cdots & \beta_{m_k-1}^{(k)} & \text{pour } \beta^{(k)} \quad \text{pour } \alpha^{(k)} \\
\alpha_0^{(k)} & \alpha_1^{(k)} & \cdots & \cdots & \alpha_{m_k-1}^{(k)} & \\
\downarrow & \downarrow & & & \downarrow & \downarrow \quad \downarrow \\
b_0^{(k)} & b_1^{(k)} & \cdots & \cdots & b_{m_k-1}^{(k)} & d_{m_k-1}^{(k)} \quad -b_{m_k}^{(k)} \\
 & b_0^{(k)} & \cdots & \cdots & b_{m_k-2}^{(k)} & d_{m_k-2}^{(k)} \quad -b_{m_k-1}^{(k)} \\
 & & \ddots & & \vdots & \vdots \quad \vdots \\
 & & & \vdots & \vdots & \vdots \quad \vdots \\
 & 0 & & b_0^{(k)} & b_1^{(k)} & d_1^{(k)} \quad -b_2^{(k)} \\
 & & & & b_0^{(k)} & d_0^{(k)} \quad -b_1^{(k)}
\end{array}
\tag{2.111}
$$

Il y a deux voies pour construire l'algorithme complet de MRZ-stab avec préconditionneur : soit nous prenons les récurrences des vecteurs z_k et z'_k soit celles de $\tilde{z}_k$ et $\tilde{z}'_k$. Les deux versions sont équivalentes et donnent théoriquement le même algorithme. Ici, nous utilisons les récurrences de $\tilde{z}_k$ et $\tilde{z}'_k$.

Algorithme 2.3.4 *:MRZ-stab avec préconditionneur.*

- Initialisation
  Choisir x_0 et y
  $\tilde{z}_{-1} = 0$; $\tilde{z}'_{-1} = 0$; $r_0 = b - Ax_0$
  résoudre $M\tilde{z}_0 = r_0$;
  $\tilde{z}'_0 = y$; $n_0 = 0$; $C_1 = 0$; $m_{-1} = 0$; $k = 0$;
- Tant que $r_k \neq 0$;
  $\tilde{z}_{k,0} = \tilde{z}_k$;
  $\tilde{z}'_{k,0} = \tilde{z}'_k$;
  $z_{k,1} = A\tilde{z}_{k,0}$;
  Résoudre $M\tilde{z}_{k,1} = z_{k,1}$
  $b_0^{(k)} = (\tilde{z}'_{k,0}, \tilde{z}_{k,1})$;
  $m_k = 1$;

Tant que $b_0^{(k)} = 0$ do
$m_k = m_k + 1$
$z_{k,m_k} = A\tilde{z}_{k,m_{k-1}}$
Résoudre $M\tilde{z}_{k,m_k} = z_{k,m_k}$
$b_0^{(k)} = (\tilde{z}'_{k,0}, \tilde{z}_{k,m_k})$
Fin (tant que)
Si $k \neq 0$
$C_{k+1} = b_0^{(k)}/b_0^{(k-1)}$
Fin (si)
Pour $i = 1, \cdots, m_k$
Résoudre $M^T z'_{k,i-1} = \tilde{z}'_{k,i-1}$
$d_{i-1}^{(k)} = (z'_{k,i-1}, r_k)$
$\tilde{z}'_{k,i} = A^T z'_{k,i-1}$
$b_i^{(k)} = (\tilde{z}'_{k,i}, \tilde{z}_{k,m_k})$
Calculer $\beta_{m_k-i}^{(k)}$
Calculer $\alpha_{m_k-i}^{(k)}$
Fin(pour)
$x_{k+1} = x_k + \beta_0^{(k)}\tilde{z}_{k,0} + \beta_1^{(k)}\tilde{z}_{k,1} + \cdots + \beta_{m_k-1}^{(k)}\tilde{z}_{k,m_k-1}$
$r_{k+1} = r_k - [\beta_0^{(k)} z_{k,1} + \beta_1 z_{k,2} + \cdots + \beta_{m_k-1}^{(k)} z_{k,m_k}]$
$\tilde{z}_{k+1} = \alpha_0^{(k)}\tilde{z}_{k,0} + \alpha_1^{(k)}\tilde{z}_{k,1} + \cdots + \alpha_{m_k-1}^{(k)}\tilde{z}_{k,m_k-1} + \tilde{z}_{k,m_k} - C_{k+1}\tilde{z}_{k-1}$
$\tilde{z}'_{k+1} = \alpha_0^{(k)}\tilde{z}'_{k,0} + \alpha_1^{(k)}\tilde{z}'_{k,1} + \cdots + \alpha_{m_k-1}^{(k)}\tilde{z}'_{k,m_k-1} + \tilde{z}'_{k,m_k} - C_{k+1}\tilde{z}'_{k-1}$
$n_{k+1} = n_k + m_k$
$k = k + 1$
Fin

Nous remarquons que dans le cas où $c(\xi^{n_k} P_{n_k}) = 0$, il se produit un *ghost-breakodwn* dans les récurrences (2.100) qui se traduit non par une division par zéro, mais plutôt par une annulation des numérateurs $(d_i^{(k)})_{0 \leq i \leq m_k - 1}$. Cela implique l'annulation du polynôme w_k et nous retrouvons ainsi le résultat donné par le Lemme 1.2.3, $P_{n_{k+1}} = P_{n_k}$ ce qui nous donne $r_{k+1} = r_k$. Donc, le *ghost-breakdown* provoque une stagnation du résidu dans les algorithmes MRZ, HMRZ, MRZ-stab et HMRZ-stab, mais n'empêche pas le calcul des itérés. Ensuite nous avons, d'après le schéma donné dans la figure 1.1, la certitude que le prochain polynôme $P_{n_{k+2}}$ sera de degré n_{k+2}. Cela explique le fait que les variantes de MRZ arrivent à corriger le *true-breakdown* sans attacher aucune importance aux situations de *ghost-breakdown.*

D'autre part, afin de réduire le nombre de vecteurs à stocker dans l'algorithme MRZ et ses variantes, Brezinski et al. [17] ont utilisé l'algorithme de Horner pour le calcul des vecteurs $w_k(A)r_k$ et $q_k(A)z_k$ figurant dans ces algorithmes. Nous allons récupérer les

résultats donnés dans [17] et utiliser la même procédure pour calculer les coefficients des polynômes q_k et ω_k et réduire le nombre de vecteurs utilisés dans MRZ-stab préconditionné. Commençons par considérer le polynôme

$$\nu(\xi) = \gamma_0^{(k)} + \cdots + \gamma_{m_k-1}^{(k)} \xi^{m_k-1} + \xi^{m_k}.$$

Ce polynôme peut être calculé par la méthode de Horner

$$\begin{aligned} \nu^{(0)}(\xi) &= 1 \\ \nu_k^{(i)}(\xi) &= \xi \nu_k^{(i-1)}(\xi) + \gamma_{m_k-i}^{(k)} \qquad \text{pour} \quad i = 1, \cdots, m_k, \end{aligned}$$

et nous obtenons $\nu_k(\xi) = \nu_k^{(m_k)}(\xi)$.
Prenons les coefficients $\gamma_i^{(k)}$ solutions du système triangulaire inférieur

$$\begin{pmatrix} b_0^{(k)} & & & & & \\ b_1^{(k)} & b_0^{(k)} & & & 0 & \\ \vdots & b_1^{(k)} & \ddots & & & \\ \vdots & \vdots & \ddots & \ddots & & \\ b_{m_k-2}^{(k)} & b_{m_k-1}^{(k)} & \cdots & \ddots & b_0^{(k)} & \\ b_{m_k-1}^{(k)} & b_{m_k-2}^{(k)} & \cdots & \cdots & b_1^{(k)} & b_0^{(k)} \end{pmatrix} \begin{pmatrix} \gamma_{m_k-1}^{(k)} \\ \gamma_{m_k-2}^{(k)} \\ \vdots \\ \vdots \\ \gamma_1 \\ \gamma_0^{(k)} \end{pmatrix} = - \begin{pmatrix} b_1^{(k)} \\ b_2^{(k)} \\ \vdots \\ \vdots \\ b_{m_k-1}^{(k)} \\ b_{m_k}^{(k)} \end{pmatrix}$$

La matrice de ce système correspond à celle du système (2.111) et les coefficients $\alpha_i^{(k)}$ du polynôme $q_k(\xi)$ coïncident donc exactement avec les $\gamma_i^{(k)}$, ce qui nous donne

$$\nu_k(\xi) = \nu_k^{m_k}(\xi) = q_k(\xi).$$

Les coefficients $\beta_i^{(k)}$ du polynôme ω_k sont calculés par la relation

$$\omega_k(\xi) = \frac{1}{b_0^{(k)}} \sum_{j=0}^{m_k-1} d_{m_k-j-1}^{(k)} \nu_k^{(j)}(\xi)$$

En appliquant ces propriétés dans MRZ-stab préconditionné, nous obtenons l'algorithme de HMRZ-stab avec préconditionneur.

Algorithme 2.3.5 *:HMRZ-stab avec préconditionneur.*

- Initialisation
  Choisir x_0 et y ;
  $\tilde{z}_{-1} = 0, \quad \tilde{z}'_{-1} = 0, \quad r_0 = b - Ax_0, \quad C_1 = 0$;
  Résoudre $M\tilde{z}_0 = r_0$;
  $\tilde{z}'_0 = y, \quad n_0 = 0, \quad b_0^{(-1)} = 0, \quad k = 0$;
- Tant que $r_k \neq 0$;
  Résoudre $M^T\tilde{y}_k = \tilde{z}'_k$;
  $d_0^{(k)} = (\tilde{y}_k, r_k)$;
  $m_k = 1$;
  $y_k = A^T\tilde{y}_k$;
  $\tilde{v}_k = y_k$;
  $b_0^{(k)} = (y_k, \tilde{z}_k)$;
  Tant que $b_0^{(k)} = 0$;
  $m_k = m_k + 1$;
  Résoudre $M^T\tilde{y}_k = y_k$;
  $d_{m_k-1}^{(k)} = (\tilde{y}_k, r_k)$;
  $y_k = A^T\tilde{y}_k$;
  $b_0^{(k)} = (y_k, \tilde{z}_k)$;
  Fin(tant que) ;
  Si $k \neq 0$;
  $C_{k+1} = b_0^{(k)}/b_0^{(k-1)}$;
  Fin(si) ;
  $t_k = \tilde{z}_k$;
  $\tilde{z}_{k+1} = -C_{k+1}\tilde{z}_{k-1}$;
  $t'_k = \tilde{z}'_k$;
  $\tilde{z}'_{k+1} = -C_{k+1}\tilde{z}'_{k-1}$;
  $x_{k+1} = x_k$;
  $r_{k+1} = r_k$;
  Pour $i = 1, \ldots, m_k$;
  $u_k = At_k$;
  $\beta = d_{m_k-i}^{(k)}/b_0^{(k)}$;
  $x_{k+1} = x_{k+1} + \beta t_k$;
  $r_{k+1} = r_{k+1} - \beta u_k$;

Résoudre $M\tilde{u}_k = u_k$;
$\gamma = -(y_k, \tilde{u}_k)/b_0^{(k)}$;
$t_k = \tilde{u}_k + \gamma \tilde{z}_k$;
Si $i \neq 1$;
Résoudre $M^T \tilde{v}_k = t'_k$;
$v_k = A^T \tilde{v}_k$;
Fin (si) ;
$t'_k = v_k + \gamma \tilde{z}'_k$;
Fin (pour) ;
$\tilde{z}_{k+1} = \tilde{z}_{k+1} + t_k$;
$\tilde{z}'_{k+1} = \tilde{z}'_{k+1} + t'_k$;
$n_{k+1} = n_k + m_k$;
$k = k + 1$;
Fin

Dans l'algorithme HMRZ-stab, nous n'avons plus à stocker les $b_i^{(k)}$ mais seulement $b_0^{(k)}$, les coefficients des polynômes ω_k et q_k ne sont pas stockés non plus. Enfin l'avantage le plus important de cette variante est que le nombre de vecteurs à stocker ne dépend plus de m_k (voir tableau 2.2).
L'algorithme HMRZ-stab est le plus stable parmi toutes les variantes de MRZ [17].

Méthode	Produits scalaires	Produits Mat-vect(*)	Place mémoire(+)
MRZ	$4m_k - m_{k-1}$	m_k/m_k	$matrice + (M+7)N + 5M$
HMRZ	$4m_k$	m_k/m_k	$matrice + 8N + 2M$
MRZ-stab	$3m_k$	$m_k/2m_k - 1$	$matrice + (M+8)N + 4M$
HMRZ-stab	$3m_k$	$m_k/2m_k - 1$	$matrice + 12N + 1M$

(*) i/j, indique qu'il y a i produits matrice vecteur avec la matrix A et j produits matrice vecteur avec sa transposée A^T

(+) $M = \sup_k m_k$

TAB. 2.2 - Stockage et nombre d'opérations requis par quelques variantes MRZ entre les itérations n_k et n_{k+1}

2.4 Résultats numériques

Dans ce paragraphe, nous donnerons quelques résultats numériques qui nous permettrons de comparer les différents algorithmes étudiés précédemment. Compte tenu du fait que ces algorithmes sont conçus pour éviter le *breakdown* et non le *near-breakdown*, nous prendrons des exemples dans lesquelles nous avons un exact *breakdown*.
Tous les tests ont été réalisés à l'aide du logiciel MATLAB (version 7). Pour détecter le *breakdown* dans les méthodes étudiées, nous allons considérer qu'une quantité est nulle si sa valeur absolue est inférieure à une tolérance donnée ϵ, ici nous prenons $\epsilon = 10^{-10}$.

Exemple 2.4.1

Nous commençons cette partie par l'exemple de dimension 4 proposé par Joubert [49] où l'on considère la matrice

$$A = \begin{pmatrix} 1 & -1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 3 & -1 \\ 0 & 0 & 1 & 3 \end{pmatrix}.$$

En prenant $b = (0,2,2,4)^T$, $x_0 = (0,0,0,0)^T$ et $y = (1,1,1,1)^T$, cette matrice cause un *breakdown* à la deuxième itération dans la méthode du BICG. Il s'agit d'un *ghost-breakdown*, c'est d'ailleurs pourquoi la méthode du CSBCG s'arrête à la premiere itération car elle est incapable d'éviter ce type de *breakdown*. Par contre, nous obtenons la solution exacte pour les méthodes BIORES non générique, HMRZ-stab et BICG avec *look-ahead*. Nous remarquons que le HMRZ-stab ne détecte aucun breakdown. En revanche il obtient dans la première et la deuxième itération le même résidu ($\|r_1\| = \|r_2\| = 3.4641$). Il en est de même avec le BICG avec *look-ahead* qui détecte un *ghost-breakdown* à la deuxième itération ce qui y provoque une stagnation du résidu. L'algorithme du BIORES lui, fait un saut de $n_1 = 1$ à $n_2 = 3$ et confirme le résultat donné par le BICG avec *look-ahead*. Les résultats obtenus sont donnés dans la figure 2.1.

Exemple 2.4.2

Considérons l'exemple donné dans [15, 17]

$$\begin{pmatrix} 0 & 0 & 0 & \cdots & 0 & -1 \\ 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \\ 3 \\ \vdots \\ N \end{pmatrix} = \begin{pmatrix} -N \\ 1 \\ 2 \\ \vdots \\ N-1 \end{pmatrix}.$$

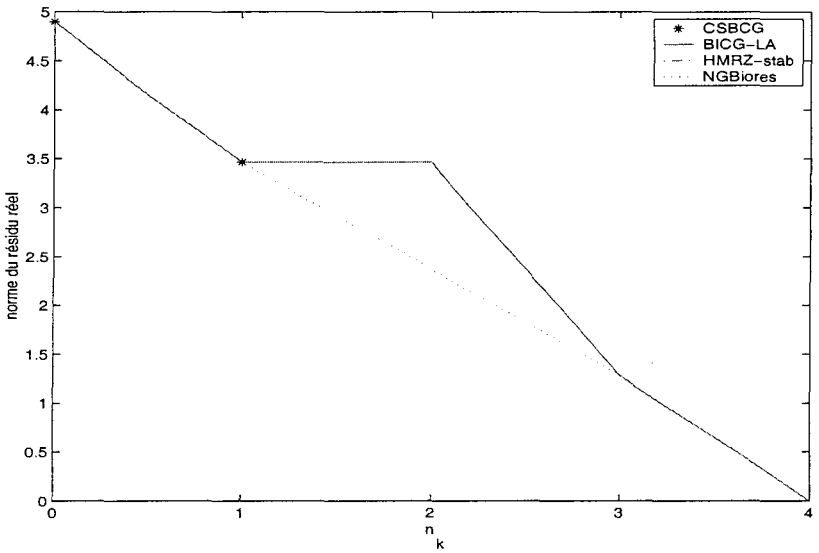


FIG. 2.1 – $N = 4$, *Matrice de Joubert*

Si nous prenons $y = (1, \ldots, 1)^T$ et $x_0 = (0, \ldots, 0)^T$ avec $N = 100$, la méthode de HMRZ-stab détecte un *true-breakdown* à l'itération $k = 4$ et fait un saut de longueur $m_3 = 94$. Le résidu calculé à l'itération $n_4 = 97$ est égal à celui déjà calculé à l'itération $n_3 = 3$ ($\|r_{97}\| = \|r_3\| = 247.40$). Cela s'explique par le fait que l'HMRZ-stab subit un *ghost-breakdown* à l'itération $n_4 = 97$ ce qui provoque une stagnation du résidu. Le BICG avec *look-ahead* détecte le *ghost-breakdown* et le *true-breakdown* à l'itération $k = 4$ et fait un saut de longueur $m_3 = 94$. À l'itération $n_4 = 97$, sans calculer le résidu, l'algorithme attribue au résidu r_{97} la valeur du résidu de l'itération précédant le saut, ($\|r_{97}\| = \|r_3\| = 247.40$). Ce résultat a été obtenu par le HMRZ-stab en calculant récursivement le résidu. Le BIORES non générique et le QMR avec *look-ahead* détectent un *ghost-breakdown* à la même itération et font un saut de longueur $m_3 = 95$. Les courbes obtenues sont données dans la figure 2.2.

Maintenant, prenons la matrice précédente avec $b = (1, \ldots, 1)^T$, $x_0 = (0, \ldots, 0)^T$ et $y = r_0$. Les quatre méthodes détectent un *breakdown* à l'itération $k = 3$. Pour le BICG avec *look-ahead* il se produit un *ghost* et un *true-breakdown*. Alors il effectue un saut de longueur $m_2 = 97$ et il attribue à l'itération $n_3 = 99$ le résidu obtenu à l'itération $n_2 = 2$ (i.e $\|r_{199}\| = \|r_2\| = 2.82$). Le même résultat est obtenu par le HMRZ-stab qui détecte un *true-breakdown* et fait un saut $m_2 = 97$ et, en calculant le résidu correspondant, il trouve $\|r_{99}\| = \|r_2\| = 2.82$. Par la suite, à l'itération $k = 100$, le BICG avec *look-ahead* calcule les vecteurs réguliers et obtient $\|r_{100}\| = 1.08\ 10^{-10}$, la méthode de HMRZ-stab donne $\|r_{100}\| = 2.8\ 10^{-13}$. Le QMR avec *look-ahead* et le BIORES non générique détectent

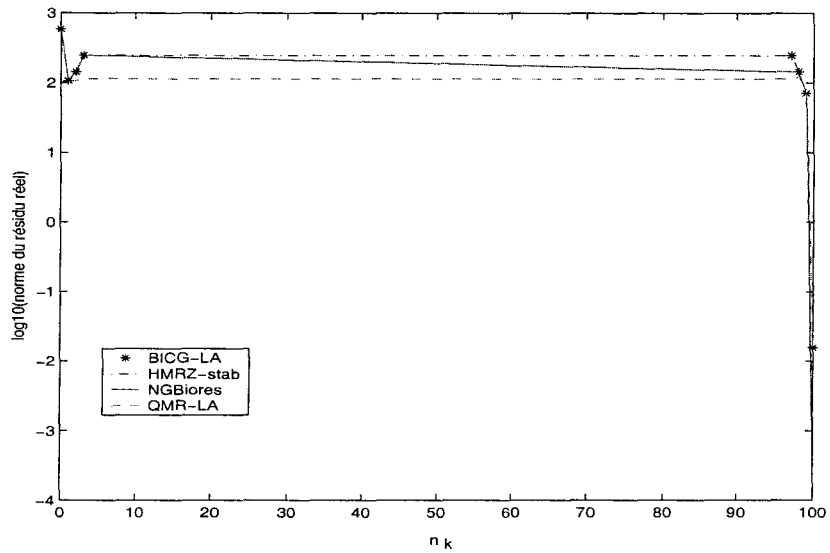


FIG. 2.2 – $N = 100$, $y = (1, \ldots, 1)^T$ et $x_0 = (0, \ldots, 0)^T$

un *ghost-breakdown* et font un saut de longueur 98. Et à l'itération 100, nous obtenons $\|r_{100}^{QMR-LA}\| = 1.89\ 10^{-12}$ et $\|r_{100}^{NGBiores}\| = 1.84\ 10^{-12}$. Les courbes représentatives sont données dans la figure 2.3.
Dans ces deux exemples, le CSBCG est incapable d'éviter ce genre de *breakdown*.

Exemple 2.4.3

Considérons l'exemple étudié par Brown dans [20]

$$\begin{pmatrix} a & 1 & & & & & \\ -1 & a & 1 & & & & \\ & -1 & a & 1 & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & \ddots & \ddots & \ddots & \\ & & & & -1 & a & 1 \\ & & & & & -1 & a \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 1 \\ \vdots \\ \vdots \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} a+1 \\ a \\ a \\ \vdots \\ \vdots \\ a \\ a-1 \end{pmatrix}$$

En prenant $a = 0$ et $y = r_0$, un *breakdown* survient à chaque itération impaire dans les méthodes HMRZ-stab, CSBCG et BICG avec *look-ahead*. Il s'agit d'un *true-breakdown* qui nécessite un saut de longueur $m_k = 2$.

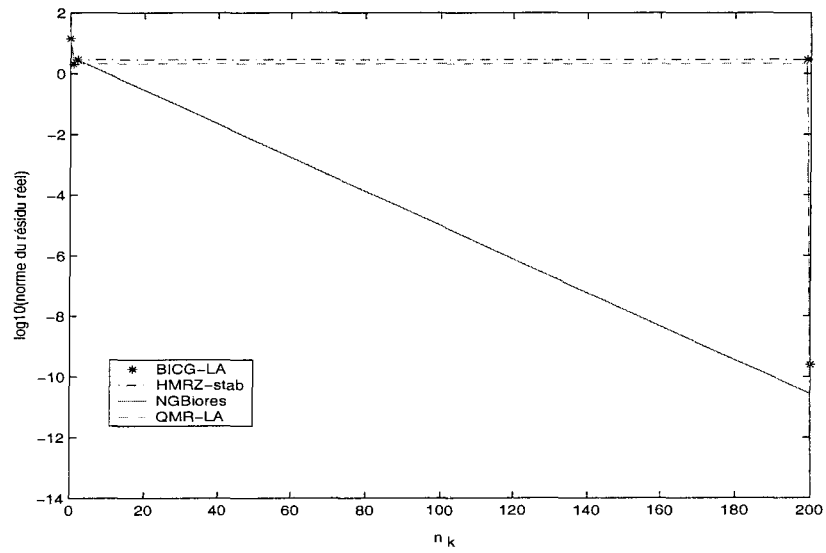


FIG. 2.3 - $N = 100$, $x_0 = (0, \ldots, 0)^T$, $b = (1, \ldots, 1)^T$ et $y = r_0$

Le QMR avec *look-ahead* et le BIORES non générique ne détectent aucun *breakdown*. En revanche dans ce dernier, le *true breakdown* ne permet pas le calcul des approximations $x_k^{NGBiores}$ et des résidus $r_k^{NGBiores}$ que lorsque k est pair. Dans la figure 2.4, nous avons pris $N = 500$ et x_0 un vecteur aléatoire. Le meilleur résidu est obtenu par la méthode CSBCG ($\|r_{100}\| = 1.61\ 10^{-11}$).
En prenant $N = 500$ et $x_0 = (0, \ldots, 0)^T$, toutes les méthodes donnent la solution exacte, voir la figure 2.5.

Exemple 2.4.4

Pour terminer, nous considérons le système linéaire proposé par Gutknecht [40]

$$\begin{pmatrix} 2 & 1 & & & & & & \\ 0 & 2 & 1 & & & & & \\ 1 & 0 & 2 & 1 & & & & \\ & \ddots & \ddots & \ddots & \ddots & & & \\ & & \ddots & \ddots & \ddots & \ddots & & \\ & & & 1 & 0 & 2 & 1 & \\ & & & & 1 & 0 & 2 & \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 1 \\ \vdots \\ \vdots \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 3 \\ 3 \\ 4 \\ \vdots \\ \vdots \\ 4 \\ 3 \end{pmatrix}.$$

Pour $N = 400$, $x_0 = (0, \ldots, 0)^T$ et $y = (0, 0, 0, 1, -1, \ldots, (-1)^{N-1}, 0)^T$, les méthodes HMRZ-stab, BICG avec *look-ahead*, QMR avec *look-ahead* et BIORES non générique

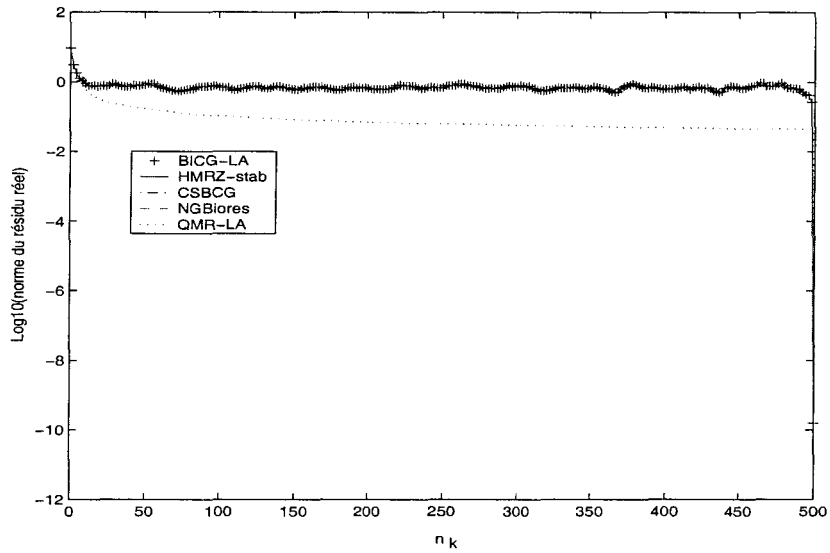


FIG. 2.4 – $N = 500$, $rand(N, 1)$ et $y = r_0$

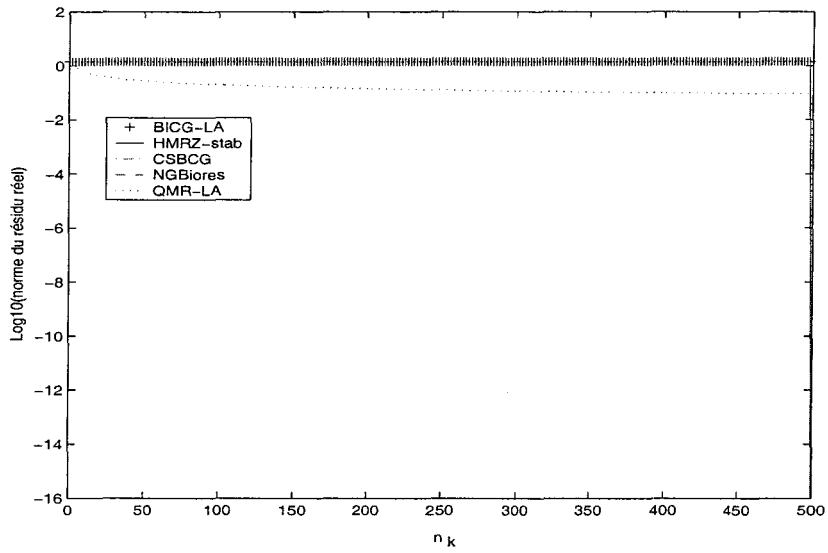


FIG. 2.5 – $N = 500$, $x_0 = (0, \ldots, 0)^T$ et $y = r_0$

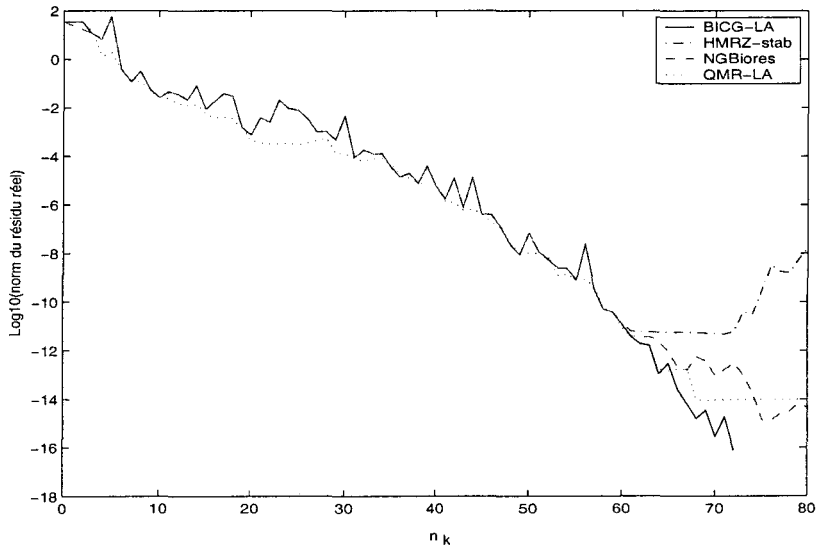


FIG. 2.6 – $N = 400$, $x_0 = (0, \ldots, 0)^T$ et $y = (0, 0, 0, 1, -1, \ldots, (-1)^{N-1}, 0)^T$

détectent un *breakdown* de longueur 2. Les meilleurs résidus obtenus sont $\|r_{72}^{BICG-LA}\| = 7.67\ 10^{-17}$, $\|r_{76}^{NGBiores}\| = 1.27\ 10^{-15}$, $\|r_{72}^{HMRZ-stab}\| = 4.64\ 10^{-12}$ et $\|r_{69}^{QMR-LA}\| = 8.49\ 10^{-15}$. Les courbes sont représentées dans la figure 2.6.

2.5 Conclusion

Dans ce chapitre, nous avons discuté et préconditionné quatre stratégies pour traiter le problème de *breakdown* dans la méthode de Lanczos pour la résolution des systèmes d'équations linéaires. Les résultats, sur les polynômes orthogonaux formels, obtenus dans la chapitre 1, nous ont permis de corriger les deux problèmes de *breakdown* qui se produisent dans l'algorithme du Gradient Biconjugué. Les résultats numériques montrent que tous les algorithmes qui ont été étudiés dans le présent chapitre, à l'exception du CSBCG, réussissent à éviter les différentes situations de *breakdown*. Il reste le problème du *incurable-breakdown* qui n'a d'autre solution que de relancer l'algorithme avec de nouveaux vecteurs initiaux. La nature du *look-ahead* peut causer un problème de dépassement de capacité (*overflow* ou *underflow*) dans ces algorithmes ; cette situation est due à une multiplication successive de la matrice A et A^T par les vecteurs de direction. Une normalisation par la norme euclidienne de ces vecteurs permet de rendre ces algorithmes plus stables. Dans le chapitre suivant, nous proposons une normalisation de l'algorithme HMRZ-stab par le produit scalaire.

Chapitre 3

Normalisation de MRZ-stab et application aux systèmes symétriques non définis positifs

3.1 Introduction

Les méthodes du Gradient Conjugué généralisées, telles que les méthodes Orthodir et Orthores dues à Young et Jea [81], la méthode Orthomin due à Vinsome [78], la méthode du GMRES due à Saad et Schultz [63], la méthode du Gradient Biconjugué due à Fletcher [29], et plusieurs autres méthodes [36, 53], sont très utilisées dans la résolution des grands systèmes linéaires

$$Ax = b, \tag{3.1}$$

où A est une matrice non-symétrique de dimension $N \times N$.
Jea et Young [47] ont proposé une simplification des méthodes du Gradient Conjugué généralisé qu'ils ont appelé méthodes de type Lanczos (Lanczos/Orthodir, Lanczos/Orthomin et Lanczos/Orthores). Théoriquement, en l'absence d'erreurs d'arrondi et des problèmes de *breakdown*, ces méthodes convergent en au plus N itérations. En comparant ces trois méthodes, Jea et Young [47, 81] ont montré que Lanczos/Orthomin converge si et seulement si Lanczos/Orthores converge, et que si ces deux méthodes convergent, alors Lanczos/Orthodir converge et les trois méthodes sont équivalentes. Donc, l'algorithme Lanczos/Orthodir devrait être le meilleur des trois. Cependant, en pratique, cette méthode souffre d'instabilité numérique due à l'effet des erreurs d'arrondi. Ici nous nous intéressons, particulièrement, aux problèmes de dépassement de capacité appelés "overflow" et "underflow" qui sont très fréquents dans cet algorithme, ces deux problèmes sont dus respectivement à une croissante ou décroissance rapide des produits scalaires provoquée par les multiplications successives des matrices A et A^T par les vecteurs de direction. Face

à ce problème, nous proposons, dans la première partie de ce chapitre, une normalisation de ces vecteurs ce qui permet de maintenir la convergence de l'algorithme et d'éviter tout dépassement provoqué par les produits scalaires. Cette technique sera appliquée à l'algorithme MRZ-stab qui est une version de Lanczos/Orthodir dans laquelle on évite le *true-breakdown*, ce dernier représente la seule division par zéro qui risque de se produire dans l'algorithme Lanczos/Orthodir.

Dans le cas où la matrice A est symétrique non définie positive, il existe plusieurs méthodes adaptées à ce type de problème. Nous avons, par exemple, la méthode du résidu minimal et la méthode de directions orthogonales dues à Fletcher [29]. Paige et Saunders [56] ont proposé des méthodes stables et efficaces basées sur la minimisation du résidu (MINRES) et les conditions de Galerkin (SYMMLQ).
Ici, nous nous intéressons aux méthodes de type Lanczos. Puisque la matrice A est symétrique, ces algorithmes peuvent être simplifiés à condition de bien choisir le vecteur de direction initial. Cependant, des divisions par zéro peuvent se produire dans ces algorithmes, cela est dû au fait que la matrice A est non définie positive. Ces situations de *breakdown* dépendent directement du vecteur de direction choisi à la première itération.
Dans la deuxième partie de ce chapitre, nous proposons une étude des situations de *breakdown* qui risquent de se produire dans les méthodes de type Lanczos lorsque la matrice du système étudié est symétrique non-définie positive. Dans un premier temps, nous prenons le résidu comme vecteur de direction initial. Dans ce cas, nous montrons que le *true-breakdown* est la seule division par zéro que nous pouvons avoir dans les algorithmes de type Lanczos, ces derniers sont basés sur une orthogonalisation du résidu. Ensuite, nous prenons le vecteur de direction initial égal au produit du résidu par la matrice A. Nous montrons que les méthodes de type Lanczos définissent une minimisation du résidu et que le *ghost-breakdown* est la seule division par zéro que subissent ces algorithmes. Dans ces deux cas, nous montrons que la longueur du saut à effectuer, pour éviter ces divisions par zéro, est égale à deux. Suite à ces résultats, ces deux choix vont nous permettre de simplifier l'algorithme MRZ-stab et ainsi donner des variantes de l'algorithme Lanczos/Orthodir pour le cas symétrique non défini positif.
Finalement des exemples numériques seront donnés dans le dernier paragraphe pour illustrer l'efficacité des algorithmes ainsi définis.

3.2 Le cas non-symétrique

L'algorithme Lanczos/Orthodir est une variante de la méthode de Lanczos qui construit à partir de deux vecteurs initiaux x_0 et y des approximations x_k de la solution de (3.1). La connexion entre les polynômes orthogonaux formels et la méthode de Lanczos permet de

retrouver l'algorithme Lanczos/Orthodir en utilisant les relations de récurrence suivantes

$$\begin{aligned} P_{k+1}(\xi) &= P_k(\xi) - \lambda_{k+1} P_k^{(1)}(\xi) \\ P_{k+1}^{(1)}(\xi) &= (\xi - \beta_{k+1}) P_k^{(1)}(\xi) - \gamma_{k+1} P_{k-1}^{(1)}(\xi). \end{aligned}$$

P_k (resp $P_k^{(1)}$) appartient à la famille des polynômes orthogonaux formels par rapport à la fonctionnelle c (resp $c^{(1)}$), où $c^{(1)}(\xi^i) = c(\xi^{i+1}) = (y, A^{i+1} r_0)$ avec $r_0 = b - Ax_0$.
Comme nous l'avons déjà vu dans la sous-section 2.3.4, il se produit dans ces relations un *true-breakdown* lorsque $H_k^{(1)} = 0$. Dans ce cas les polynômes P_k et $P_k^{(1)}$ n'existent pas et cela se traduit par une division par zéro dans l'algorithme Lanczos/Orthodir. D'autre part, lorsque $H_k^{(0)} = 0$ il se produit un *ghost-breakdown* ce qui provoque une stagnation du résidu mais n'empêche pas sa convergence. En appliquant la procédure du *look-ahead* à cet algorithme, nous obtenons la méthode MRZ-stab, ainsi que toutes les variantes MRZ [17], qui permettent d'éviter le *true-breakdown*.

Un autre type de problème peut survenir dans les algorithmes Lanczos/Orthodir et MRZ-stab : il s'agit des situations de dépassement de capacité qui provoquent des ruptures dans ces algorithmes. Cela se produit lorsque les produits scalaires calculés prennent des valeurs très grandes "Overflow" ou très petites "Underflow".
Dans la suite de ce paragraphe, nous allons effectuer une normalisation de l'algorithme MRZ-stab qui consiste à remplacer ses vecteurs de direction par des vecteurs normalisés, modification qui permet de rééquilibrer le rapport entre les produits scalaires calculés dans cet algorithme.

3.2.1 Algorithme MRZ-stab normalisé

Nous avons vu, dans le chapitre 2, que l'algorithme MRZ-stab est obtenu à partir des relations de récurrence suivantes

$$\begin{aligned} P_{k+1}(\xi) &= P_k(\xi) - \xi \omega_k(\xi) P_k^{(1)}(\xi), \\ P_{k+1}^{(1)}(\xi) &= q_k(\xi) P_k^{(1)}(\xi) - C_{k+1} P_{k-1}^{(1)}(\xi), \end{aligned} \tag{3.2}$$

où $P_{-1}^{(1)}(\xi) = 0$, $P_0^{(1)}(\xi) = 1$ et $C_1 = 0$.
Nous notons n_k le degré du $k^{\text{ème}}$ polynôme orthogonal formel régulier P_k et $P_k^{(1)}$, et m_k la longueur du saut entre $P_k^{(1)}$ et $P_{k+1}^{(1)}$ i.e $n_{k+1} = n_k + m_k$.
Le saut m_k est défini par les conditions

$$\begin{aligned} c^{(1)}(\xi^i P_k^{(1)}) &= 0 \qquad \text{pour } \; i = 0, \ldots, n_k + m_k - 2 \\ c^{(1)}(\xi^{n_k+m_k-1} P_k^{(1)}) &\neq 0. \end{aligned} \tag{3.3}$$

q_k est un polynôme unitaire de degré m_k et ω_k un polynôme de degré au plus $m_k - 1$.
Les coefficients des polynômes ω_k et q_k ainsi que C_{k+1} sont obtenus en imposant les condi-

tions (3.3) aux relations (3.2). Ainsi, si nous posons

$$\begin{aligned} \omega_k(\xi) &= \beta_0^{(k)} + \cdots + \beta_{m_k-1}^{(k)}\xi^{m_k-1}, \\ q_k(\xi) &= \alpha_0^{(k)} + \cdots + \alpha_{m_k-1}^{(k)}\xi^{m_k-1} + \xi^{m_k}, \end{aligned}$$

alors, les coefficients $\beta_i^{(k)}$ et $\alpha_i^{(k)}$ sont solutions des deux systèmes triangulaires

$$\begin{pmatrix} b_0^{(k)} & & & & & \\ b_1^{(k)} & b_0^{(k)} & & & 0 & \\ \vdots & b_1^{(k)} & \ddots & & & \\ \vdots & \vdots & \ddots & \ddots & & \\ b_{m_k-2}^{(k)} & b_{m_k-1}^{(k)} & \cdots & \ddots & b_0^{(k)} & \\ b_{m_k-1}^{(k)} & b_{m_k-2}^{(k)} & \cdots & \cdots & b_1^{(k)} & b_0^{(k)} \end{pmatrix} \begin{pmatrix} \beta_{m_k-1}^{(k)} & \alpha_{m_k-1}^{(k)} \\ \beta_{m_k-2}^{(k)} & \alpha_{m_k-2}^{(k)} \\ \vdots & \vdots \\ \vdots & \vdots \\ \beta_1^{(k)} & \alpha_1^{(k)} \\ \beta_0^{(k)} & \alpha_0^{(k)} \end{pmatrix} = \begin{pmatrix} d_0^{(k)} & -b_1^{(k)} \\ d_1^{(k)} & -b_2^{(k)} \\ \vdots & \vdots \\ \vdots & \vdots \\ d_{m_k-2}^{(k)} & -b_{m_k-1}^{(k)} \\ d_{m_k-1}^{(k)} & -b_{m_k}^{(k)} \end{pmatrix} \tag{3.4}$$

avec

$$\begin{aligned} b_i^{(k)} &= c^{(1)}(\xi^{m_k-1+i}{P_k^{(1)}}^2) \\ &= (P_k^{(1)}(A^T)y, A^{m_k+i}P_k^{(1)}(A)r_0) \qquad \text{pour } i = 0, \ldots, m_k \end{aligned}$$

et (3.5)

$$\begin{aligned} d_i^{(k)} &= c(\xi^i P_k^{(1)} P_k) \\ &= (P_k^{(1)}(A^T)y, A^i P_k(A)r_0) \qquad \text{pour } i = 0, \ldots, m_k - 1. \end{aligned}$$

Et enfin, $C_{k+1} = b_0^{(k)}/b_0^{(k-1)}$. D'après la condition (3.3), nous avons

$$b_0^{(k)} = c^{(1)}(\xi^{m_k-1}{P_k^{(1)}}^2) \neq 0.$$

Dans l'algorithme MRZ-stab, nous construisons deux suites de vecteur de direction z_k et $\tilde{z}_k$ à partir des relations de récurrence vérifiées par les polynômes $P_k^{(1)}$. Dans le cas où il ne se produit pas de *breakdown* ces vecteurs sont A-biorthogonaux. Sinon, ils vérifient les relations

$$(\tilde{z}_k, A^{i+1}z_j) = 0 \quad \text{si} \quad \begin{cases} j < k & \text{pour} \quad i = 0, \ldots, n_k - n_j + m_k - 2 \\ k < j & \text{pour} \quad i = 0, \ldots, n_j - n_k + m_j - 2 \\ k = j & \text{pour} \quad i = 0, \ldots, m_k - 2 \end{cases} \tag{3.6}$$

et

$$(\tilde{z}_k, A^{m_k}z_k) \neq 0.$$

Nous proposons ici une normalisation de ces deux suites de vecteurs de telle façon que

$$(\tilde{z}_k, A^{n_k+m_k}z_k) = 1. \tag{3.7}$$

Nous allons donc normaliser les vecteurs z_k et $\tilde{z}_k$ par deux facteurs différents. Pour cela, nous remplaçons les polynômes $P_k^{(1)}$ par le couple des polynômes

$$\widehat{P_k^{(1)}} = \frac{P_k^{(1)}}{\tau_k} \qquad \text{et} \qquad \overline{P_k^{(1)}} = \frac{P_k^{(1)}}{\upsilon_k},$$

avec τ_k et υ_k choisis tels que

$$c^{(1)}(\xi^{m_k-1}\overline{P_k^{(1)}}\widehat{P_k^{(1)}}) = \frac{c^{(1)}(\xi^{m_k-1}P_k^{(1)^2})}{\tau_k\upsilon_k} = 1.$$

Nous pouvons prendre

$$\tau_k = \sqrt{|b_0^{(k)}|} \qquad \text{et} \qquad \upsilon_k = signe[b_0^{(k)}]\ \tau_k.$$

Les vecteurs de direction sont donc définis par

$$z_k = \frac{P_k^{(1)}(A)r_0}{\tau_k} \qquad \text{et} \qquad \tilde{z}_k = \frac{P_k^{(1)}(A^T)y}{\upsilon_k},$$

et ils vérifient les relations (3.6) et (3.7). Ces vecteurs ne peuvent pas être calculés directement, car pour cela il faut connaître la valeur de m_k. C'est pourquoi nous définissons deux nouveaux vecteurs

$$s_k = \frac{P_k^{(1)}(A)r_0}{\tau_{k-1}} \qquad \text{et} \qquad \tilde{s}_k = \frac{P_k^{(1)}(A^T)y}{\upsilon_{k-1}}$$

avec $s_{-1} = r_0$ et $\tilde{s}_{-1} = y$.
La longueur du saut m_k est déterminée par la condition (3.3), et puisque $\tau_k \neq 0$ et $\upsilon_k \neq 0\ \ \forall k$, alors nous avons

$$\begin{aligned} (\tilde{s}_k, A^i s_k) &= 0 \qquad \text{pour } i = 1, \ldots, m_k - 1 \\ (\tilde{s}_k, A^{m_k} s_k) &\neq 0. \end{aligned}$$

Notons

$$\delta_k = \sqrt{|(\tilde{s}_k, A^{m_k} s_k)|} \qquad \text{et} \qquad \sigma_k = signe[(\tilde{s}_k, A^{m_k} s_k)]\ \delta_k$$

nous avons les propriétés suivantes

Proposition 3.2.1

1. $\delta_k = \tau_k/\tau_{k-1}$ *et* $\sigma_k = \upsilon_k/\upsilon_{k-1}$.
2. $z_k = s_k/\delta_k$ *et* $\tilde{z}_k = \tilde{s}_k/\sigma_k$.

Preuve :

D'après la définition de s_k et $\tilde{s}_k$, nous avons

$$\begin{aligned}(\tilde{s}_k, A^{m_k} s_k) &= (\frac{P_k^{(1)}(A^T)y}{\upsilon_{k-1}}, A^{m_k}\frac{P_k^{(1)}(A)r_0}{\tau_{k-1}}) = \frac{c(\xi^{m_k} {P_k^{(1)}}^2)}{\upsilon_{k-1}\tau_{k-1}} \\ &= \frac{b_0^{(k)}}{\upsilon_{k-1}\tau_{k-1}} = \frac{b_0^{(k)}}{b_0^{(k-1)}},\end{aligned}$$

alors

$$\delta_k = \sqrt{|(\tilde{s}_k, A^{m_k} s_k)|} = \sqrt{|\frac{b_0^{(k)}}{b_0^{(k-1)}}|} = \frac{\tau_k}{\tau_{k-1}}.$$

Ces deux résultats nous permettent d'écrire

$$\sigma_k = signe[(\tilde{s}_k, A^{m_k} s_k)]\ \delta_k = \frac{signe[b_0^{(k)}]}{signe[b_0^{(k-1)}]}\frac{\tau_k}{\tau_{k-1}} = \frac{\upsilon_k}{\upsilon_{k-1}}.$$

D'autre part, d'après les définitions de z_k et $\tilde{z}_k$, nous avons

$$z_k = \frac{P_k^{(1)}(A)r_0}{\tau_k} = \frac{P_k^{(1)}(A)r_0}{\tau_{k-1}}\frac{\tau_{k-1}}{\tau_k} = \frac{s_k}{\delta_k}$$

et

$$\tilde{z}_k = \frac{P_k^{(1)}(A^T)y}{\upsilon_k} = \frac{P_k^{(1)}(A^T)y}{\upsilon_{k-1}}\frac{\upsilon_{k-1}}{\upsilon_k} = \frac{\tilde{s}_k}{\sigma_k}.$$

□

Donc, pour obtenir les vecteurs z_{k+1} et $\tilde{z}_{k+1}$, nous calculons d'abord les vecteurs s_{k+1} et $\tilde{s}_{k+1}$ à partir des relations de récurrence (3.2). Ainsi, nous obtenons

$$\begin{aligned}s_{k+1} &= A^{m_k} z_k + \alpha_{m_k-1}^{(k)} A^{m_k-1} z_k + \cdots + \alpha_0^{(k)} z_k - C_{k+1}\frac{\tau_{k-1}}{\tau_k} z_{k-1} \\ \tilde{s}_{k+1} &= A^{m_k} \tilde{z}_k + \alpha_{m_k-1}^{(k)} A^{m_k-1} \tilde{z}_k + \cdots + \alpha_0^{(k)} \tilde{z}_k - C_{k+1}\frac{\upsilon_{k-1}}{\upsilon_k} \tilde{z}_{k-1}.\end{aligned} \tag{3.8}$$

Nous avons $C_{k+1} = b_0^{(k)}/b_0^{(k-1)}$ et $b_0^{(k)} = \tau_k \upsilon_k$, alors, en utilisant les résultats de la proposition 3.2.1, nous obtenons

$$C_{k+1}\frac{\tau_{k-1}}{\tau_k} = \frac{b_0^{(k)}}{\tau_k}\frac{\tau_{k-1}}{b_0^{(k-1)}} = \frac{\upsilon_k}{\upsilon_{k-1}} = \sigma_k$$

et

$$C_{k+1}\frac{\upsilon_{k-1}}{\upsilon_k} = \frac{b_0^{(k)}}{\upsilon_k}\frac{\upsilon_{k-1}}{b_0^{(k-1)}} = \frac{\tau_k}{\tau_{k-1}} = \delta_k.$$

Donc, les relations (3.8) s'écrivent

$$\begin{aligned} s_{k+1} &= A^{m_k} z_k + \alpha_{m_k-1}^{(k)} A^{m_k-1} z_k + \cdots + \alpha_0^{(k)} z_k - \sigma_k z_{k-1} \\ \tilde{s}_{k+1} &= A^{m_k} \tilde{z}_k + \alpha_{m_k-1}^{(k)} A^{m_k-1} \tilde{z}_k + \cdots + \alpha_0^{(k)} \tilde{z}_k - \delta_k \tilde{z}_{k-1}. \end{aligned} \tag{3.9}$$

Nous savons que le résidu est défini par $r_k = P_k(A) r_0$. Alors, d'après la première relation de (3.2), nous avons

$$r_{k+1} = r_k - \beta_0^{(k)} \tau_k A z_k - \beta_1^{(k)} \tau_k A^2 z_k - \cdots - \beta_{m_k-1}^{(k)} \tau_k A^{m_k} z_k.$$

Si nous posons

$$\tilde{\beta}_i^{(k)} = \beta_i^{(k)} \tau_k \qquad \text{pour } i = 0, \ldots, m_k - 1,$$

alors, nous pouvons écrire

$$r_{k+1} = r_k - \tilde{\beta}_0^{(k)} A z_k - \tilde{\beta}_1^{(k)} A^2 z_k - \cdots - \tilde{\beta}_{m_k-1}^{(k)} A^{m_k} z_k. \tag{3.10}$$

Prenons maintenant

$$\begin{aligned} \tilde{b}_i^{(k)} &= (\tilde{z}_k, A^{m_k+i} z_k) && \text{pour } i = 0, \ldots, m_k \\ \tilde{d}_i^{(k)} &= (\tilde{z}_k, A^i r_k) && \text{pour } i = 0, \ldots, m_k - 1. \end{aligned}$$

D'après les définitions de z_k et $\tilde{z}_k$, les relations (3.5) nous donnent

$$\begin{aligned} b_i^{(k)} &= (\upsilon_k \tilde{z}_k, \tau_k A^{m_k+i} z_k) \\ &= \tilde{b}_i^{(k)} \tau_k \upsilon_k = \tilde{b}_i^{(k)} b_0^{(k)} && \text{pour } i = 0, \ldots, m_k \\ d_i^{(k)} &= (\upsilon_k \tilde{z}_k, A^i r_k) \\ &= \tilde{d}_i^{(k)} \upsilon_k && \text{pour } i = 0, \ldots, m_k - 1. \end{aligned} \tag{3.11}$$

Remarquons que $\tilde{b}_0^{(k)} = (\tilde{z}_k, A^{m_k} z_k) = 1$.
Les coefficients $\beta_i^{(k)}$ et $\alpha_i^{(k)}$ sont solutions du système (3.4) et en utilisant les relations (3.11), nous pouvons calculer les coefficients $\alpha_i^{(k)}$ et $\tilde{\beta}_i^{(k)}$ à partir du système suivant

$$\begin{pmatrix} 1 & & & & \\ \tilde{b}_1^{(k)} & 1 & & & \\ \tilde{b}_2^{(k)} & \tilde{b}_1^{(k)} & \ddots & & \\ \vdots & \vdots & \ddots & \ddots & \\ \tilde{b}_{m_k-1}^{(k)} & \tilde{b}_{m_k-2}^{(k)} & \cdots & \tilde{b}_1^{(k)} & 1 \end{pmatrix} \begin{pmatrix} \tilde{\beta}_{m_k-1}^{(k)} & \alpha_{m_k-1}^{(k)} \\ \tilde{\beta}_{m_k-2}^{(k)} & \alpha_{m_k-2}^{(k)} \\ \vdots & \vdots \\ \vdots & \vdots \\ \tilde{\beta}_0^{(k)} & \alpha_0^{(k)} \end{pmatrix} = \begin{pmatrix} \tilde{d}_0^{(k)} & -\tilde{b}_1^{(k)} \\ \tilde{d}_1^{(k)} & -\tilde{b}_2^{(k)} \\ \vdots & \vdots \\ \vdots & \vdots \\ \tilde{d}_{m_k-1}^{(k)} & -\tilde{b}_{m_k}^{(k)} \end{pmatrix}. \tag{3.12}$$

Puisque $r_k = b - Ax_k$, l'approximation x_k vérifie la relation

$$x_{k+1} = x_k + \tilde{\beta}_0^{(k)} z_k + \tilde{\beta}_1^{(k)} A z_k + \cdots + \tilde{\beta}_{m_k-1}^{(k)} A^{m_k-1} z_k. \tag{3.13}$$

En rassemblant tous ces résultats, nous obtenons l'algorithme suivant, que nous appelons MRZ-stab normalisé.

Algorithme 3.2.1 *: MRZ-stab normalisé.*

1. Initialisation

   Choisir x_0 et $y \in \mathbb{R}^N$

   $z_{-1,0} = 0, \quad \tilde{z}_{-1,0} = 0$;

   $r_0 = b - Ax_0 \quad s_0 = r_0$;

   $\tilde{s}_0 = y \quad n_0 = 0 \quad k = 0$;

2. Tant que $r_k \neq 0$

   $s_{k,0} = s_k, \quad s_{k,1} = As_{k,0}$;

   $\rho_k = (\tilde{s}_k, s_{k,1}), \quad m_k = 1$;

3. Tant que $\rho_k = 0$

   $m_k = m_k + 1$;

   $s_{k,m_k} = As_{k,m_{k-1}}$;

   $\rho_k = (\tilde{s}_k, s_{k,m_k})$;

   Fin(tant que) ;

4. $\delta_k = \sqrt{|\rho_k|}, \quad \sigma_k = signe[\rho_k]\ \delta_k$;

   $z_{k,0} = s_{k,0}/\delta_k, \quad \tilde{z}_{k,0} = \tilde{s}_k/\sigma_k$;

   Pour $i = 1, \ldots, m_k$;

   $z_{k,i} = s_{k,i}/\delta_k$;

   $\tilde{z}_{k,i} = A^T \tilde{z}_{k,i-1}$;

   $\tilde{b}_i^{(k)} = (\tilde{z}_{k,i}, z_{k,m_k})$;

   $\tilde{d}_{i-1}^{(k)} = (\tilde{z}_{k,i-1}, r_k)$;

   Calculer $\tilde{\beta}_{m_k-i}^{(k)}$ et $\alpha_{m_k-i}^{(k)}$ à partir du système (3.12) ;

   Fin(pour)

5. $x_{k+1} = x_k + \tilde{\beta}_0^{(k)} z_{k,0} + \tilde{\beta}_1^{(k)} z_{k,1} + \cdots + \tilde{\beta}_{m_k-1}^{(k)} z_{k,m_k-1}$;

   $r_{k+1} = r_k - \tilde{\beta}_0^{(k)} z_{k,1} - \tilde{\beta}_1^{(k)} z_{k,2} - \cdots - \tilde{\beta}_{m_k-1}^{(k)} z_{k,m_k}$;

   $s_{k+1} = \alpha_0^{(k)} z_{k,0} + \alpha_1^{(k)} z_{k,1} + \cdots + \alpha_{m_k-1}^{(k)} z_{k,m_k-1} + z_{k,m_k} - \sigma_k z_{k-1,0}$;

   $\tilde{s}_{k+1} = \alpha_0^{(k)} \tilde{z}_{k,0} + \alpha_1^{(k)} \tilde{z}_{k,1} + \cdots + \alpha_{m_k-1}^{(k)} \tilde{z}_{k,m_k-1} + \tilde{z}_{k,m_k} - \delta_k \tilde{z}_{k-1,0}$;

6. $n_{k+1} = n_k + m_k$;

   $k = k + 1$.

Fin

Afin de réduire le nombre de vecteurs et de coefficients stockés dans l'algorithme 3.2.1, nous pouvons utiliser le schéma de Horner pour calculer $w_k(A)z_k$ et $q_k(A)z_k$, comme cela a été fait dans l'algorithme HMRZ-stab [17].
Dans la deuxième partie de cette section, nous proposons une modification de l'algorithme 3.2.1 qui porte sur le calcul des coefficients $(\alpha_i^{(k)})$ et $(\beta_i^{(k)})$. Cela va nous permettre de réduire le nombre de vecteurs et des coefficients à stocker sans passer par le schéma de Horner.

3.2.2 Algorithme MRZ-stab modifié

Dans l'algorithme 3.2.1, la résolution directe des deux systèmes (3.12) nécessite le stockage des coefficients $\tilde{b}_i^{(k)}$ et $d_{i-1}^{(k)}$ pour $i = 1, \ldots, m_k - 1$. Le calcul des itérés demande, en plus des coefficients $\alpha_{j-1}^{(k)}$ et $\beta_{j-1}^{(k)}$, le stockage des vecteurs $\tilde{z}_{k,j}$ et $z_{k,j}$ pour $j = 1, \ldots, m_k$. Nous pouvons réduire le nombre de vecteurs et de coefficients stockés en effectuant une simple modification dans la programmation de l'algorithme 3.2.1. Pour cela, nous commençons par donner les solutions du système (3.12).
Nous avons

$$\begin{aligned}
\tilde{\beta}_{m_k-1}^{(k)} &= \tilde{d}_0^{(k)}, \\
\tilde{\beta}_{m_k-2}^{(k)} &= \tilde{d}_1^{(k)} - \tilde{\beta}_{m_k-1}^{(k)}\tilde{b}_1^{(k)}, \\
\tilde{\beta}_{m_k-3}^{(k)} &= \tilde{d}_2^{(k)} - \tilde{\beta}_{m_k-1}^{(k)}\tilde{b}_2^{(k)} - \tilde{\beta}_{m_k-2}^{(k)}\tilde{b}_1^{(k)}, \\
\cdots \quad & \quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\
\tilde{\beta}_0^{(k)} &= \tilde{d}_{m_k-1}^{(k)} - \tilde{\beta}_{m_k-1}^{(k)}\tilde{b}_{m_k-1}^{(k)} - \cdots - \tilde{\beta}_1^{(k)}\tilde{b}_1^{(k)},
\end{aligned}$$

et

$$\begin{aligned}
\alpha_{m_k-1}^{(k)} &= -\tilde{b}_1^{(k)}, \\
\alpha_{m_k-2}^{(k)} &= -\tilde{b}_2^{(k)} - \alpha_{m_k-1}^{(k)}\tilde{b}_1^{(k)}, \\
\alpha_{m_k-3}^{(k)} &= -\tilde{b}_3^{(k)} - \alpha_{m_k-1}^{(k)}\tilde{b}_2^{(k)} - \alpha_{m_k-2}^{(k)}\tilde{b}_1^{(k)}, \\
\cdots \quad & \quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\
\alpha_0^{(k)} &= -\tilde{b}_{m_k}^{(k)} - \alpha_{m_k-1}^{(k)}\tilde{b}_{m_k-1}^{(k)} - \cdots - \alpha_1^{(k)}\tilde{b}_1^{(k)}.
\end{aligned}$$

En tenant compte des expressions des coefficients $\tilde{b}_i^{(k)}$ et $\tilde{d}_i^{(k)}$, données par les relations

$$\begin{aligned}
\tilde{b}_i^{(k)} &= (\tilde{z}_k, A^{m_k+i}z_k) \qquad \text{pour } i = 0, \ldots, m_k \\
\tilde{d}_i^{(k)} &= (\tilde{z}_k, A^i r_k) \qquad \text{pour } i = 0, \ldots, m_k - 1,
\end{aligned}$$

nous obtenons

$$\begin{aligned}
\tilde{\beta}^{(k)}_{m_k-1} &= (\tilde{z}_k, r_k),\\
\tilde{\beta}^{(k)}_{m_k-2} &= (A^T\tilde{z}_k, r_k) - (A^{Tm_k}\tilde{z}_k, \tilde{\beta}^{(k)}_{m_k-1}Az_k),\\
\tilde{\beta}^{(k)}_{m_k-3} &= (A^{T^2}\tilde{z}_k, r_k) - (A^{Tm_k}\tilde{z}_k, \tilde{\beta}^{(k)}_{m_k-1}A^2z_k + \tilde{\beta}^{(k)}_{m_k-2}Az_k)\\
\cdots &\quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\\
\tilde{\beta}^{(k)}_{0} &= (A^{Tm_k-1}\tilde{z}_k, r_k) - (A^{Tm_k}\tilde{z}_k, \tilde{\beta}^{(k)}_{m_k-1}A^{m_k-1}z_k + \cdots + \tilde{\beta}^{(k)}_{1}Az_k)
\end{aligned}$$

et

$$\begin{aligned}
\alpha^{(k)}_{m_k-1} &= -(A^{Tm_k}\tilde{z}_k, Az_k),\\
\alpha^{(k)}_{m_k-2} &= -(A^{Tm_k}\tilde{z}_k, A^2z_k + \alpha^{(k)}_{m_k-1}Az_k),\\
\alpha^{(k)}_{m_k-3} &= -(A^{Tm_k}\tilde{z}_k, A^3z_k + \alpha^{(k)}_{m_k-1}A^2z_k + \alpha^{(k)}_{m_k-2}Az_k),\\
\cdots &\quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\\
\alpha^{(k)}_{0} &= -(A^{Tm_k}\tilde{z}_k, A^{m_k}z_k + \alpha^{(k)}_{m_k-1}A^{m_k-1}z_k + \cdots + \alpha^{(k)}_{1}Az_k).
\end{aligned}$$

Nous remarquons que les expressions des vecteurs s_{k+1}, $\tilde{s}_{k+1}$, r_{k+1} et x_{k+1} données par les relations (3.9), (3.10) et (3.13) peuvent être récupérées à partir des expressions des coefficients $\alpha^{(k)}_0$ et $\tilde{\beta}^{(k)}_0$. Alors, d'après cette écriture, nous pouvons calculer les coefficients $(\alpha^{(k)}_i)$ et $(\tilde{\beta}^{(k)}_i)$ en mettant une boucle dans l'algorithme (3.2.1) qui n'exige pas le stockage des coefficients calculés et que nous exploiterons pour donner à la fin les vecteurs x_{k+1}, r_{k+1}, s_{k+1} et $\tilde{s}_{k+1}$.

Nous trouvons ainsi un nouvel algorithme qui est équivalent à l'algorithme 3.2.1, que nous appelons MRZ-stab modifié

Algorithme 3.2.2 : *MRZ-stab modifié.*

1. Initialisation

   Choisir x_0 et $y \in \mathbb{R}^N$

   $z_{-1} = 0, \quad \tilde{z}_{-1} = 0$;

   $r_0 = b - Ax_0 \quad s_0 = r_0$;

   $\tilde{s}_0 = y \quad n_0 = 0 \quad k = 0$

2. Tant que $r_k \neq 0$

   $\tilde{y} = A^T\tilde{s}_k$;

   $\rho_k = (\tilde{y}_k, s_k)$,

   $m_k = 1$;

3. Tant que $\rho_k = 0$
   $m_k = m_k + 1$
   $\tilde{y} = A^T \tilde{y}$
   $\rho_k = (\tilde{y}, s_k)$
   Fin(tant que)

4. $\delta_k = \sqrt{|\rho_k|}, \quad \sigma_k = signe[\rho_k]\ \delta_k$;
   $z_k = s_k/\delta_k,$
   $\tilde{z}_k = \tilde{s}_k/\sigma_k$;
   $\tilde{y} = \tilde{y}/\sigma_k,$
   $t = z_k,$
   $\tilde{u} = \tilde{z}_k,$
   $u = 0,$
   $v = z_k,$
   $\tilde{v} = \tilde{z}_k,$

5. Pour $i = 1, \ldots, m_k$
   $v = Av,$
   $\tilde{v} = A^T \tilde{v},$
   $\alpha = -(\tilde{y}, v),$
   $v = v + \alpha z_k,$
   $\tilde{v} = \tilde{v} + \alpha \tilde{z}_k,$
   $\tilde{\beta} = (\tilde{u}, r_k) - (\tilde{y}, u),$
   $\tilde{u} = A^T \tilde{u},$
   $t = u + \tilde{\beta} z_k,$
   $u = At,$
   Fin(pour)
   $x_{k+1} = x_k + t,$
   $r_{k+1} = r_k - u,$
   $s_{k+1} = v - \sigma_k z_{k-1},$
   $\tilde{s}_{k+1} = \tilde{v} - \delta_k \tilde{z}_{k-1},$

6. $n_{k+1} = n_k + m_k$
   $k = k + 1$

Fin

Dans cet algorithme, le nombre de vecteurs stockés est réduit à 12, il est indépendant de la longueur du saut m_k. Le nombre de coefficients stockés est limité à 6 coefficients alors

qu'en utilisant le schéma de Horner, nous sommes obligés de stocker les coefficients $\tilde{d}_i$ pour $i = 0, \ldots, m_k - 1$.
Des résultats numériques qui illustrent l'efficacité de cet algorithme sont donnés à la fin de ce chapitre.

Dans la section suivante, nous étudions les situations de *breakdown* qui se produisent dans les algorithmes de type Lanczos dans le cas où la matrice A est symétrique non définie positive. Nous allons voir que le choix de y permet la simplification de l'algorithme MRZ-stab modifié et peut éventuellement offrir des solutions aux problèmes de *breakdown*.

3.3 Le cas symétrique non défini positif

Dans le cas où la matrice du système (3.1) est symétrique non définie positive, la méthode de Lanczos consiste à construire une suite de vecteurs (x_k) de la façon suivante :

- Choisir deux vecteurs arbitraires x_0 et $y \neq 0$ dans $\mathbb{R}^N$,
- Calculer $r_0 = b - Ax_0$,
- Déterminer x_k tel que

$$\begin{aligned} x_k - x_0 \in \mathcal{K}_k(A, r_0) &= vect(r_0, Ar_0, \ldots, A^{k-1}r_0) \\ r_k = b - Ax_k \perp \mathcal{K}_k(A, y) &= vect(y, Ay, \ldots, A^{k-1}y). \end{aligned} \tag{3.14}$$

Sous ces conditions, nous remplaçons dans les algorithmes de type Lanczos la matrice transposée A^T par la matrice A, ce qui nous donne la possibilité de simplifier ces algorithmes à condition de bien choisir le vecteur initial y. En plus, les situations de *breakdown* qui se produisent dans la méthode de Lanczos dépendent directement du choix de ce vecteur.
Dans cette section, nous proposons une étude des situations de *breakdown* susceptibles de se produire dans les méthodes de type Lanczos lorsque la matrice du système étudié est symétrique non-définie positive. Nous commençons par prendre $y = r_0$, dans ce cas nous allons montrer qu'il se produit un *true-breakdown* dans les méthodes de type Lanczos, et qu'il suffit de faire un saut de longueur égale à deux pour éviter ce problème. Dans la deuxième partie, nous prendrons $y = Ar_0$, dans ce cas il se produit un *ghost-breakdown* que nous évitons aussi de la même manière. Ces deux choix vont nous permettre de simplifier l'algorithme MRZ-stab modifié et ainsi nous donnerons des variantes de l'algorithme Lanczos/Orthodir pour le cas symétrique non défini positif. Mais avant cela, nous commençons d'abord par donner quelques propriétés des sous-espaces de Krylov.

3.3.1 Quelques propriétés des sous-espaces de Krylov

Les espaces de Krylov $\mathcal{K}_k(A, r_0)$ forment évidement une famille croissante de sous-espaces, nécessairement bornée. Notons L la dimension maximale de ces sous-espaces de

Krylov. Nous avons donc

$$L = max\{k : dim\ \mathcal{K}_k(A, r_0) = k\}$$

Lemme 3.3.1
La suite des sous-espaces de Krylov $\mathcal{K}_k(A, r_0)$ est strictement croissante pour $k = 1, \ldots, L$, et stagne à partir de $k = L$.

Preuve :

Notons p le plus petit entier pour lequel $A^p r_0$ est dépendant des vecteurs précédents. Alors, les vecteurs $r_0, Ar_0, \ldots, A^{p-1}r_0$ sont linéairement indépendants et donc $\mathcal{K}_k(A, r_0)$ est de dimension k pour tout $1 \leq k \leq p$ et $A^p r_0 \in \mathcal{K}_p(A, r_0)$.
Montrons maintenant, par récurrence, que nous avons $A^{p+i}r_0 \in \mathcal{K}_p(A, r_0)$ pour tout $i \geq 0$; Supposons que $A^{p+i}r_0 \in \mathcal{K}_p(A, r_0)$ ce qui est équivalent à $A^{p+i}r_0 = \sum_{j=0}^{p-1} \alpha_j A^j r_0$; alors

$$\begin{aligned} A^{p+i+1}r_0 &= \sum_{j=0}^{p-2} \alpha_j A^{j+1} r_0 + \alpha_{p-1} A^p r_0 \\ &= \sum_{j=0}^{p-2} \alpha_j A^{j+1} r_0 + \alpha_{p-1} \sum_{j=0}^{p-1} \beta_j A^j r_0 \\ &= \sum_{j=0}^{p-1} \gamma_j A^j r_0. \end{aligned}$$

Nous avons donc $A^{p+i}r_0 \in \mathcal{K}_p(A, r_0)$, de sorte que $\mathcal{K}_{p+i}(A, r_0) = \mathcal{K}_p(A, r_0)$ pour tout $i \geq 0$.
Donc $L = p$ et $\mathcal{K}_1(A, r_0) \subsetneq \ldots \subsetneq \mathcal{K}_p(A, r_0) = \mathcal{K}_{p+i}(A, r_0)$ pour tout $i \geq 0$.

□

Proposition 3.3.1
Pour tout $k \leq L$, les vecteurs $r_0, Ar_0, \ldots, A^{k-1}r_0$ sont linéairement indépendants.

Théorème 3.3.1
La solution du problème $Ax = b$ appartient à l'espace affine $x_0 + \mathcal{K}_L$.

Preuve :
D'après le lemme 3.3.1, le sous-espace $\mathcal{K}_L$ est invariant. Donc, l'opérateur A définit une bijection de $\mathcal{K}_L$ dans $\mathcal{K}_L$. Le vecteur r_0 appartient à $\mathcal{K}_L$, donc, il a un antécédent dans $\mathcal{K}_L$. Alors, il existe un $z \in \mathcal{K}_L$ tel que $Az = r_0 = b - Ax_0 = A(x - x_0)$, d'où le résultat.

□

Donc, s'il ne se produit pas de *breakdown* dans l'algorithme de Lanczos, la solution est obtenue à l'itération L.

3.3.2 Orthogonalisation du résidu

Si nous prenons dans la méthode de Lanczos $y = r_0$, les conditions (3.14) s'écrivent

$$\begin{aligned} x_k - x_0 &\in \mathcal{K}_k(A, r_0) \\ r_k = A(x - x_k) &\perp \mathcal{K}_k(A, r_0). \end{aligned} \tag{3.15}$$

Dans le cas où la matrice A est définie positive, les conditions (3.15) sont équivalentes à minimiser l'erreur

$$\|e_k\|_A = \min_{z \in \mathcal{K}_k(A,r_0)} \|e_0 - z\|_A$$

où $e_k = x - x_k = x - (x_0 + z) = e_0 - z$.
La méthode du Gradient Conjugué est la méthode de type Lanczos, la plus puissante, utilisée dans le cas où A est symétrique définie positive. Les itérés x_k du Gradient Conjugué sont bien définis à partir des conditions (3.15) et l'algorithme ne subit aucun *breakdown*.

Ici, nous traitons le cas où la matrice A est symétrique non-définie positive. L'erreur et le résidu, définis par les conditions (3.15), ne vérifient pas de propriété de minimisation, nous avons tout simplement une orthogonalisation du résidu par rapport au sous-espace de Krylov $\mathcal{K}_k(A, r_0)$. Cependant, les itérés x_k ne sont pas tout le temps définis par cette propriété. En effet, prenons $V_k = [r_0, Ar_0, \ldots, A^{k-1}r_0]$. D'après les conditions (3.15), nous avons l'existence d'un vecteur $d_k \in \mathbb{R}^k$ tel que

$$\begin{aligned} x_k - x_0 &= V_k d_k \\ V_k^T r_k &= 0. \end{aligned} \tag{3.16}$$

La première relation de (3.16) devient

$$r_k = r_0 - AV_k d_k,$$

et les conditions d'orthogonalité nous donnent

$$V_k^T r_k = V_k^T r_0 - V_k^T A V_k d_k = 0.$$

D'où

$$d_k = (V_k^T A V_k)^{-1} V_k^T r_0.$$

Les itérés x_k sont bien définis lorsque la matrice $V_k^T A V_k$ est inversible, or comme A est non définie positive alors ce n'est pas toujours le cas. Lorsque la matrice $V_k^T A V_k$ est singulière il se produit un *breakdown* dans la méthode de Lanczos. Ce problème peut-être évité en utilisant les algorithmes de type Lanczos avec *look-ahead*.
Dans ce qui va suivre, nous allons donner une version simplifiée de l'algorithme MRZ-stab modifié pour le cas symétrique. Mais avant cela, nous commençons par étudier les situations de *breakdown* qui risquent de se produire dans les algorithmes de type Lanczos lorsque nous prenons $y = r_0$.

Comme nous l'avions déjà mentionné auparavant, lorsque $H_k^{(0)} = 0$ (resp $H_k^{(1)} = 0$) il se produit un *ghost-breakdown* (resp *true-breakdown*) dans les méthodes de type Lanczos. La fonctionnelle c est définie par rapport aux conditions (3.15) comme suit

$$c(\xi^i) = c_i = (r_0, A^i r_0). \tag{3.17}$$

Dans ce cas, les déterminants de Hankel $H_k^{(0)}$ et $H_k^{(1)}$ vérifient

$$H_k^{(0)} = \begin{vmatrix} c_0 & \cdots & c_{k-1} \\ \vdots & & \vdots \\ c_{k-1} & \cdots & c_{2k-2} \end{vmatrix} = \begin{vmatrix} (r_0, r_0) & \cdots & (r_0, A^{k-1} r_0) \\ \vdots & & \vdots \\ (r_0, A^{k-1} r_0) & \cdots & (r_0, A^{2k-2} r_0) \end{vmatrix} = det(V_k^T V_k).$$

$$H_k^{(1)} = \begin{vmatrix} c_1 & \cdots & c_k \\ \vdots & & \vdots \\ c_k & \cdots & c_{2k-1} \end{vmatrix} = \begin{vmatrix} (r_0, Ar_0) & \cdots & (r_0, A^k r_0) \\ \vdots & & \vdots \\ (r_0, A^k r_0) & \cdots & (r_0, A^{2k-1} r_0) \end{vmatrix} = det(V_k^T A V_k).$$

D'après la proposition 3.3.1, la matrice $V_k^T V_k$ est inversible pour tout $k \leq L$, avec L la dimension du sous-espace invariant $\mathcal{K}_L(A, r_0)$ par rapport à A. D'où $H_k^{(0)} \neq 0 \ \forall \ k \leq L$. Nous avons aussi le résultat suivant

Théorème 3.3.2
Pour tout $k < L$, si $H_k^{(1)} = 0$ alors $H_{k+1}^{(1)} \neq 0$.

Preuve :
Nous avons

$$V_{k+1}^T A V_{k+1} = \begin{pmatrix} (r_0, Ar_0) & (r_0, A^2 r_0) & \cdots & (r_0, A^k r_0) & (r_0, A^{k+1} r_0) \\ (Ar_0, Ar_0) & (Ar_0, A^2 r_0) & \cdots & (Ar_0, A^k r_0) & (Ar_0, A^{k+1} r_0) \\ \vdots & \vdots & & \vdots & \vdots \\ (A^{k-1} r_0, Ar_0) & (A^{k-1} r_0, A^2 r_0) & \cdots & (A^{k-1} r_0, A^k r_0) & (A^{k-1} r_0, A^{k+1} r_0) \\ (A^k r_0, Ar_0) & (A^k r_0, A^2 r_0) & \cdots & (A^k r_0, A^k r_0) & (A^k r_0, A^{k+1} r_0) \end{pmatrix}.$$

En utilisant l'identité de Sylvester, nous obtenons

$$H_{k+1}^{(1)} \ det[(AV_{k-1})^T A(AV_{k-1})] = H_k^{(1)} \ det[(AV_k)^T A(AV_k)] - (det[(AV_k)^T (AV_k)])^2.$$

D'après la proposition 3.3.1, les vecteurs $Ar_0, \ldots, A^k r_0$ sont linéairement indépendants $\forall \ k < L$. Alors la matrice $[(AV_k)^T (AV_k)]$ est inversible, d'où

$$det[(AV_k)^T (AV_k)] \neq 0; \quad \forall \ k < L.$$

Donc, si $H_k^{(1)} = 0$ alors $H_{k+1}^{(1)} \neq 0$.

□

Ce Théorème peut aussi être obtenu à partir du Lemme 1.2.2, en remarquant que le déterminant de Hankel $H_k^{(0)} \neq 0$.

Remarque 3.3.1

Si nous appliquons l'un des algorithmes de type Lanczos au système (3.1) dans le cas symétrique en prenant $y = r_0$, alors nous risquons d'avoir seulement un true-breakdown. En revanche, la taille du saut à effectuer pour éviter ce problème ne dépasse pas deux.

Comme nous l'avons déjà vu auparavant, l'algorithme MRZ-stab modifié nous permet d'éviter le *true-breakdown* en faisant des sauts de longueur m_k. Compte-tenu du résultat donné ci-dessus, si nous prenons dans cet algorithme $y = r_0$, alors le saut m_k est au plus égal à deux. D'autre part, nous calculons dans cet algorithme les vecteurs $\tilde{z}_k$ et $\tilde{s}_k$ par récurrence à partir du vecteur initial y. En prenant $y = r_0$, nous obtenons les propriétés suivantes

Proposition 3.3.2

Soit $\varrho_k = (s_k, A^{m_k} s_k)$, alors

(i) $\tilde{z}_k = signe[\varrho_k]\ z_k \quad et \quad \tilde{s}_{k+1} = signe[\varrho_k]\ s_{k+1}$.

(ii) $\delta_k = \sqrt{|\varrho_k|} \quad et \quad \sigma_k = signe[\varrho_{k-1}]\ signe[\varrho_k]\ \delta_k$.

Preuve :

Puisque $y = r_0$, la fonctionnelle c est donnée par (3.17) et les vecteurs z_k, $\tilde{z}_k$, s_k et $\tilde{s}_k$ sont définis par les relations suivantes

$$\begin{aligned} z_k &= \frac{P_k^{(1)}(A)r_0}{\tau_k}, \qquad \tilde{z}_k = \frac{P_k^{(1)}(A)r_0}{\upsilon_k}, \\ s_k &= \frac{P_k^{(1)}(A)r_0}{\tau_{k-1}}, \qquad \tilde{s}_k = \frac{P_k^{(1)}(A)r_0}{\upsilon_{k-1}}, \end{aligned} \tag{3.18}$$

où $\tau_k = \sqrt{|b_k^{(0)}|}$ et $\upsilon_k = signe[b_k^{(0)}]\ \tau_k$, avec $b_k^{(0)} = c(\xi^{m_k} {P_k^{(1)}}^2)$.

(i) Nous avons

$$\varrho_k = (s_k, A^{m_k} s_k) = \frac{c(\xi^{m_k} {P_k^{(1)}}^2)}{\tau_k^2},$$

donc

$$signe[\varrho_k] = signe[b_k^{(0)}].$$

D'autre part, d'après les relations (3.18), nous avons

$$\tilde{z}_k = signe[b_k^{(0)}]\ z_k \quad \text{et} \quad \tilde{s}_{k+1} = signe[b_k^{(0)}]\ s_{k+1},$$

ainsi, nous obtenons

$$\tilde{z}_k = signe[\varrho_k]\ z_k \quad et \quad \tilde{s}_{k+1} = signe[\varrho_k]\ s_{k+1}.$$

(ii) Nous avons

$$\delta_k = \sqrt{|(\tilde{s}_k, As_k)|} \quad \text{et} \quad \sigma_k = signe[(\tilde{s}_k, A^{m_k} s_k)]\ \delta_k.$$

En tenant compte du fait que $\tilde{s}_k = signe[\varrho_{k-1}]\ s_k$, nous obtenons

$$\delta_k = \sqrt{|(s_k, As_k)|} = \sqrt{|\varrho_k|}.$$

Nous avons aussi

$$signe[(\tilde{s}_k, A^{m_k} s_k)] = signe[\varrho_{k-1}] signe[(s_k, A^{m_k} s_k)] = signe[\varrho_{k-1}] signe[\varrho_k],$$

d'où

$$\sigma_k = signe[\varrho_{k-1}] signe[\varrho_k]\ \delta_k.$$

□

Ces propriétés nous permettent de simplifier l'algorithme MRZ-stab modifié puisque nous n'avons plus à calculer les vecteurs $\tilde{z}_k$ et $\tilde{s}_k$. Nous obtenons ainsi un algorithme dans lequel nous faisons un saut égal à deux dans le cas d'un *breakdown*, cet algorithme est appelé Lanczos/Orthodir OR.

Algorithme 3.3.1 *: Lanczos/Orthodir OR.*

1. Initialisation
   Choisir $x_0 \in I\!R^N$;
   $r_0 = b - Ax_0, \quad s_0 = r_0$;
   $z_p = 0, \quad \epsilon_p = 0 \quad k = 0$;

2. Tant que $r_k \neq 0$
   $\varrho_k = (s_k, As_k)$;

3. Si $\varrho_k \neq 0$;

$\epsilon_k = signe[\varrho_k], \quad \delta_k = \sqrt{\epsilon_k \varrho_k}$;

$z_k = \dfrac{s_k}{\delta_k}$;

$\theta_k = (z_k, r_k)\epsilon_k$;

$x_{k+1} = x_k + \theta_k z_k$;

$r_{k+1} = r_k - \theta_k A z_k$;

Si $k = 0$ alors $\nu_0 = 0$;

Sinon $\nu_k = \epsilon_p \, \epsilon_k \, \delta_k$;

Fin(Si)

$\mu_k = (Az_k, Az_k)\epsilon_k$;

$s_{k+1} = Az_k - \mu_k z_k - \nu_k z_p$;

$z_p = z_k$;

$\epsilon_p = \epsilon_k$;

$k = k + 1$;

4. Sinon

$\delta_k = \|As_k\|, \quad \epsilon_k = +1$;

$z_k = \dfrac{s_k}{\delta_k}$;

$\alpha_k = (\, z_k, r_k)$;

$\beta_k = (Az_k, r_k) - \alpha_k (Az_k, A^2 z_k)$;

$x_{k+2} = \alpha_k A z_k + \beta_k z_k + x_k$;

$r_{k+2} = r_k - \alpha_k A^2 z_k - \beta_k A z_k$;

Si $k = 0$, alors $\vartheta_0 = 0$;

Sinon $\vartheta_k = \epsilon_p \, \delta_k$;

Fin(si)

$u = A^2 z_k$;

$\lambda_k = (u, Az_k)$;

$u = u - \lambda_k A z_k$;

$\gamma_k = (A^2 z_k, u)$;

$s_{k+2} = u - \gamma_k z_k - \vartheta_k z_p$;

$z_p = z_k$;

$\epsilon_p = \epsilon_k$;

$k = k + 2$;

Fin(si) ;

Fin.

Cet algorithme est théoriquement équivalent à l'algorithme SYMMLQ [56]. Ce dernier est obtenu à partir des conditions (3.15), il utilise le processus de Lanczos pour transformer la matrice A en une matrice tridiagonale T_k, et résout le système obtenu en passant par une factorisation LQ de T_k ce qui lui permet d'éviter la situation de *breakdown*.

3.3.3 Minimisation du résidu

Si nous prenons dans la méthode de Lanczos $y = Ar_0$, les conditions (3.14) s'écrivent

$$\begin{aligned} x_k - x_0 &\in \mathcal{K}_k(A, r_0) \\ r_k &\perp A\mathcal{K}_k(A, r_0). \end{aligned} \tag{3.19}$$

Dans ce cas, nous avons le résultat suivant

Proposition 3.3.3
Les conditions (3.19) sont équivalentes à minimiser le résidu

$$\|r_k\| = \min_{z \in \mathcal{K}_k(A,r_0)} \|r_0 - Az\| \tag{3.20}$$

où $r_k = b - Ax_k = b - A(x_0 + z) = r_0 - Az$.

Preuve :
La condition d'orthogonalité donnée par (3.19) s'écrit

$$r_k^T u = 0, \ \forall u \in A\mathcal{K}_k(A, r_0).$$

Donc

$$r_k = (r_0 - Az) \in (A\mathcal{K}_k(A, r_0))^{\perp}, \tag{3.21}$$

$(A\mathcal{K}_k(A, r_0))^{\perp}$ représente le sous-espace orthogonal à $A\mathcal{K}_k(A, r_0)$.
La première équation de (3.19), nous donne

$$r_k \in r_0 + A\mathcal{K}_k(A, r_0). \tag{3.22}$$

D'après les relations (3.21) et (3.22), le résidu r_k est la projection orthogonale de r_0 sur $(A\mathcal{K}_k(A, r_0)^{\perp})$.
Le vecteur Az solution du problème aux moindres carrés (3.20) est bien la projection orthogonale de r_0 sur $A\mathcal{K}_k(A, r_0)$. D'où le résultat.

□

Dans ce cas, la méthode de Lanczos correspond à une projection orthogonale et les itérés x_k sont bien définis par les conditions (3.19). Or, certains algorithmes de type Lanczos, définis par ces conditions, subissent des divisions par zéro. C'est le cas, par exemple, de la

méthode CR (Conjugate Residual) [36, 53]. Cette méthode est mise en œuvre à partir de relations de récurrence à deux termes comme celles utilisées dans le Gradient Conjugué, sauf qu'ici nous prenons $y = Ar_0$. La méthode CR est définie lorsque la matrice A est définie positive.
D'après les conditions (3.19), la fonctionnelle c est définie comme suit

$$c(\xi^i) = c_i = (Ar_0, A^i r_0) = (r_0, A^{i+1} r_0).$$

D'après leurs définitions, les déterminants de Hankel s'écrivent

$$H_k^{(0)} = \begin{vmatrix} c_0 & \cdots & c_{k-1} \\ \vdots & & \vdots \\ c_{k-1} & \cdots & c_{2k-2} \end{vmatrix} = \begin{vmatrix} (r_0, Ar_0) & \cdots & (r_0, A^k r_0) \\ \vdots & & \vdots \\ (r_0, A^k r_0) & \cdots & (r_0, A^{2k-1} r_0) \end{vmatrix} = det[V_k^T A V_k].$$

$$H_k^{(1)} = \begin{vmatrix} c_1 & \cdots & c_k \\ \vdots & & \vdots \\ c_k & \cdots & c_{2k-1} \end{vmatrix} = \begin{vmatrix} (Ar_0, Ar_0) & \cdots & (Ar_0, A^k r_0) \\ \vdots & & \vdots \\ (Ar_0, A^k r_0) & \cdots & (Ar_0, A^{2k-1} r_0) \end{vmatrix} = det[(AV_k)^T A V_k].$$

avec $V_k = [r_0, Ar_0, \ldots, A^{k-1} r_0]$.
D'après la proposition 3.3.1, les vecteurs $Ar_0, A^2 r_0, \ldots, A^k r_0$ sont linéairement indépendants pour tout $k < L$. Donc la matrice $(AV_k)^T A V_k$ est inversible, d'où $H_k^{(1)} \neq 0 \; \forall \; k < L$.
D'autre part, nous avons montré, dans la preuve du Théorème 3.3.2, que si $det[V_k^T A V_k] = 0$ alors $det[V_{k+1}^T A V_{k+1}] \neq 0$, $\forall \; k < L$. Cela nous donne le résultat suivant

Théorème 3.3.3
Pour tout $k < L$, si $H_k^{(0)} = 0$ alors $H_{k+1}^{(0)} \neq 0$.

Les résultats donnés ci-dessus, montrent que seul le *ghost-breakdown* peut affecter les méthodes de type Lanczos définies par les conditions (3.19). En revanche, un saut de longueur égal à deux suffit pour éviter ce problème.
Donc, si nous appliquons la méthode MRZ-stab modifié dans ces conditions avec $y = Ar_0$, nous n'aurons pas de division par zéro à éviter et nous aurons $m_k = 1$ à toutes les itérations. Cependant, le *ghost-breakdown* peut éventuellement provoquer une stagnation du résidu pendant deux itérations au plus.
Le choix $y = Ar_0$ permet de simplifier l'algorithme MRZ-stab modifié, puisque nous obtenons $m_k = 1$, $\tilde{z}_k = Az_k$, $\tilde{s}_k = As_k$ et $\rho_k = \|As_k\|^2$. Nous remarquons que $\rho_k \neq 0$ sauf si $s_k = 0$, et dans ce cas nous avons déjà obtenu la solution du problème (3.1).
Nous obtenons ainsi une version sans *breakdown* de l'algorithme Lanczos/Orthodir basée sur une minimisation du résidu. L'algorithme obtenu est équivalent à l'algorithme MINRES [56] et il est appelé Lanczos/Orthodir MR.

Algorithme 3.3.2 *Lanczos/Orthodir MR*

1. Initialisation
   Choisir $x_0 \in \mathbb{R}^N$
   $r_0 = b - Ax_0, \quad s_0 = r_0,$
   $k = 0$
2. Tant que $r_k \neq 0$
   $u_k = As_k$;
   $\delta_k = \|u_k\|$;
   $z_k = \frac{s_k}{\delta_k}, \quad v_k = \frac{u_k}{\delta_k}$;
   $\theta_k = (v_k, r_k)$;
   $x_{k+1} = x_k + \theta_k z_k$;
   $r_{k+1} = r_k - \theta_k v_k$;
   Si $k = 0$ alors $\nu_0 = 0$;
   Sinon $\nu_k = \delta_k$;
   Fin(si)
   $\mu_k = (Av_k, v_k)$;
   $s_{k+1} = v_k - \mu_k z_k - \nu_k z_p$;
   $z_p = z_k$;
   $k = k + 1$;
Fin

3.4 Résultats numériques

Dans cette section, nous allons donner quelques résultats numériques. Nos tests ont été réalisés à l'aide du logiciel MATLAB (version 7).
Pour détecter le *breakdown* dans les méthodes étudiées, nous allons considérer qu'une quantité est nulle si sa valeur absolue est inférieure à une tolérance donnée ϵ, ici nous prenons $\epsilon = 10^{-10}$.

3.4.1 Le cas non-symétrique

Nous allons montrer que l'exécution de l'algorithme HMRZ-stab, qui est la version la plus stable du Lanczos/Orthodir permettant d'éviter *lebreakdown*, n'est pas à l'abri d'un problème d'" overflow" ou "underflow". Nous comparons cet algorithme avec MRZ-stab modifié.

Exemple 3.4.1

Nous considérons la matrice non-symétrique obtenue à partir de la discrétisation de l'équation elliptique aux dérivées partielles $Lu = f$ sur $[0,1] \times [0,1]$, où

$$Lu = -\Delta u + \beta \frac{\delta u}{\delta x},$$

avec les conditions de Dirichlet $u = 0$ aux bords, en utilisant un schéma à cinq points sur une grille uniforme 20×20 avec un pas $h = 1/21$. Ceci nous donne une matrice creuse non symétrique de dimension $N = 400$ avec 1920 éléments non nuls. Nous choisissons $\beta = 10^4$. L'application des algorithmes HMRZ-stab et MRZ-stab modifié avec $y = (0, 0, \ldots, 0, 0, 1)^T$, $x_0 = (0, \ldots, 0)^T$ et $b = (1, \ldots, 1)^T$ nous donne la figure 3.1.

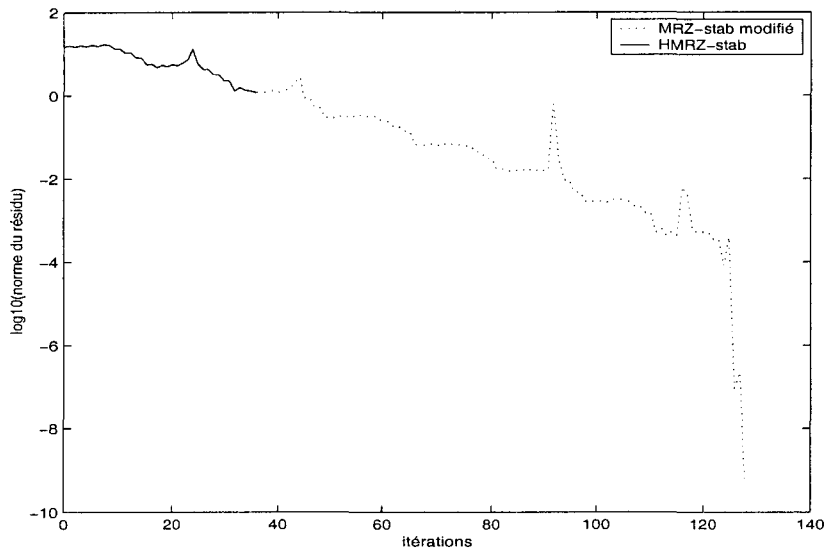


FIG. 3.1 – $N = 400$, $\beta = 10^4$, $b = (1, \ldots, 1)^T$

A l'itération $k = 38$, un "overflow" affecte l'algorithme HMRZ-stab, alors que dans le MRZ-stab modifié il n'apparaît aucun problème de dépassement de capacité.
Prenons maintenant $b = (1, 0, \ldots, 0)^T$, dans ce cas il se produit un *true-breakdown* dans les méthodes de type Lanczos. Ici les deux algorithmes évitent cette division par zéro en faisant un saut de longueur $m_k = 26$ à la première itération. Par la suite, le HMRZ-stab atteint le seuil "overflow" à l'itération $n_k = 49$ alors que le MRZ-stab modifié continue de converger pour donner la courbe représentée dans la figure 3.2.

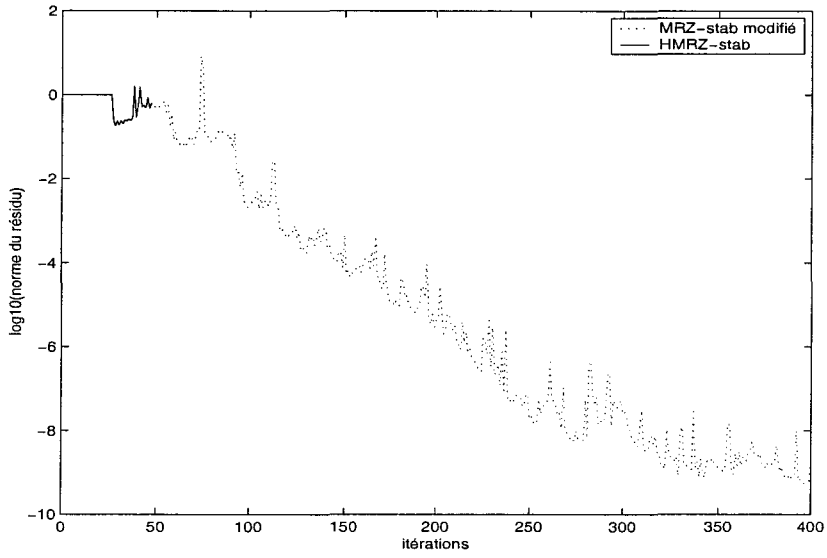


FIG. 3.2 – $N = 400$, $\beta = 10^4$, $b = (1, 0, \ldots, 0)^T$

Exemple 3.4.2

Dans cet exemple nous considérons la matrice A=sherman1, une matrice réelle non symétrique de dimension 1000×1000 du groupe **SHERMAN** et de la collection **Harwell-Boeing**. Elle contient 3750 éléments non nuls,
En prenant $b = (1, \ldots, 1)^T$, $x_0 = (0, \ldots, 0)^T$ et $y = b - Ax_0$, nous obtenons le résultat donné dans la figure 3.3.
L'algorithme HMRZ-stab subit un *breakdown* à l'itération $k = 12$ et une rupture due à un "overflow" à l'itération $k = 379$, ces deux situations sont provoquées par la propagation d'erreurs d'arrondi, alors que l'algorithme MRZ-stab modifié ne détecte aucun *breakdown* et continue de converger pour atteindre $r_{800} = 23.56\ 10^{-13}$.

Exemple 3.4.3

Prenons la matrice A= rdb 2048. C'est une matrice réelle non symétrique de dimension 2048×2048 de la collection **NEP** et du groupe **BRUSSEL**. Elle contient 12032 éléments non nuls.
Pour $b = (1, \ldots, 1)^T$, $x_0 = (0, \ldots, 0)^T$ et $y = b - Ax_0$, l'algorithme HMRZ-stab subit une rupture due à un "overflow" à l'itération $k = 160$ alors que l'algorithme MRZ-stab modifié converge normalement. Le résultat obtenu est donné dans la figure 3.4.

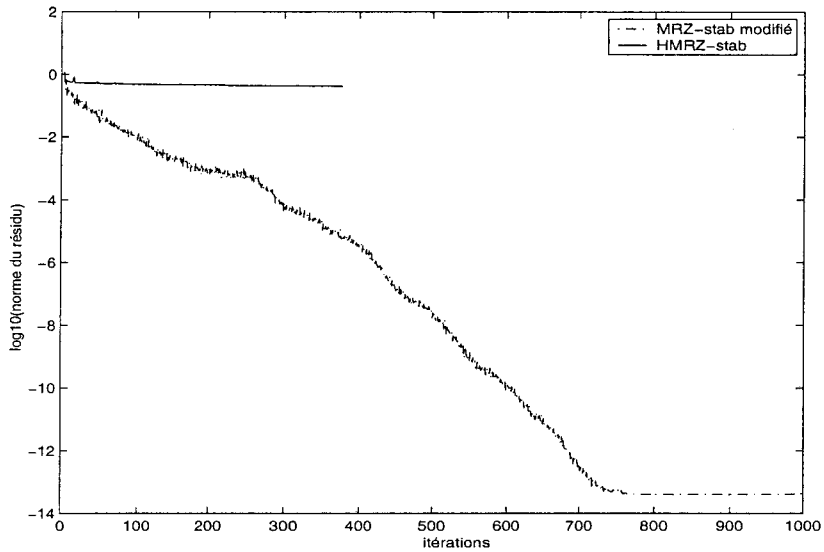


FIG. 3.3 – $N = 1000$, A=sherman1

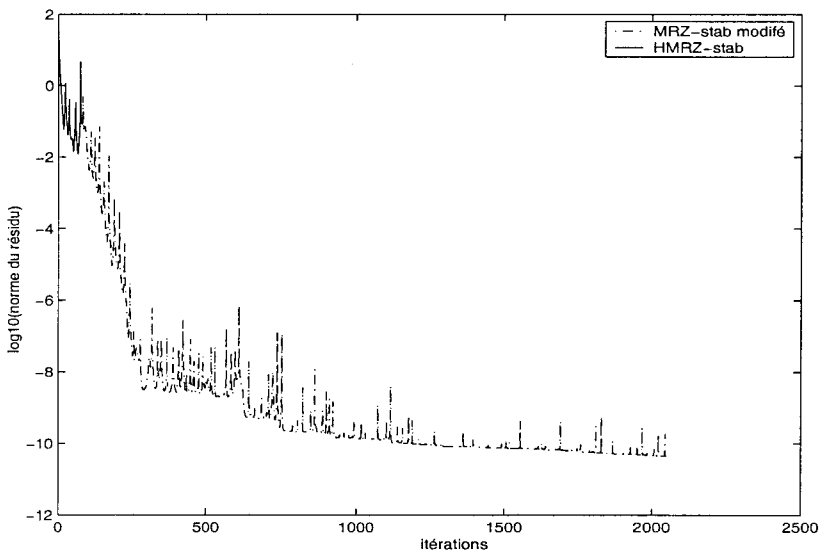


FIG. 3.4 – N=2048, A= rdb 2048

3.4.2 Le cas symétrique

Dans cette partie, nous comparons les algorithmes Lanczos/Orthodir MR et Lanczos/Orthodir OR avec les méthodes MINRES et SYMMLQ dans le cas des matrices symétriques non définies positives.

Exemple 3.4.4

Nous commençons par vérifier la stabilité numérique des algorithmes Lanczos/Orthodir MR et Lanczos/Orthodir MR. Les résultats obtenus par ces algorithmes sont comparés avec ceux du MINRES et SYMMLQ.
Prenons la matrice $A = QDQ^T$, avec Q une matrice orthogonale quelconque et

$$D = \begin{pmatrix} \lambda_1 & & & \\ & \ddots & & \\ & & \ddots & \\ & & & \lambda_N \end{pmatrix}$$

où $\lambda_i = cos(2(i-1)\pi/N)$ pour $i = 1, \ldots, N$.
En prenant $x_0 = (0, \ldots, 0)^T$ et $b = Q(1, \ldots, 1)^T$, nous pouvons calculer exactement les normes des résidus. En effet, nous avons d'après [66]

$$\|r_{2k}^{MR}\| = \|r_{2k+1}^{MR}\| = \sqrt{\frac{N}{2k+1}} \qquad \text{et} \qquad \|r_{2k}^{OR}\| = \sqrt{\frac{N}{2}}$$

avec r_k^{MR} (resp r_k^{OR}) désigne les résidus obtenus par la méthode de sous-espace de Krylov basée sur la minimisant du résidu (resp l'orthogonalisation du résidu). Dans cet exemple, il se produit un *breakdown* dans les méthodes de type Lanczos à chaque itération impaire. En appliquant les algorithmes Lanczos/Orthodir MR, Lanczos/Orthodir OR, MINRES et SYMMLQ, nous obtenons les résultats donnés dans les tableaux 3.1 et 3.2.
Les quatre algorithmes donnent les valeurs exactes de leurs résidus respectifs avec une légère instabilité due aux erreurs d'arrondi. Nous remarquons que l'algorithme Lanczos/Orthodir OR détecte un *true-breakdown* à chaque itération impaire et fait des sauts de longueur égale à deux, alors que dans le SYMMLQ nous avons une stagnation du résidu. Les deux algorithmes donnent les mêmes résultats. D'autre part, l'algorithme Lanczos/Orthodir MR subit un *ghost-breakdown* à chaque itération impaire, ce qui explique les stagnations de son résidu. Le même résultat est donné par le MINRES.

N=100

itération	Lanczos/Othodir OR	SYMMLQ	valeur exacte de la norme du résidu
1	saut	7.07106781186547	
2	7.07106781186547	7.07106781186547	7.07106781186547
3	saut	7.07106781186547	
10	7.07106781186547	7.07106781186547	7.07106781186547
11	saut	7.07106781186547	
12	7.07106781186547	7.07106781186547	7.07106781186547
13	saut	7.07106781186547	
97	saut	7.07106781186567	
98	7.07106781186563	7.07106781186566	7.07106781186547
99	saut	$2.014635\ 10^{-12}$	
100	$3.719722\ 10^{-12}$	$1.924893\ 10^{-13}$	

N=500

itération	Lanczos/Othodir OR	SYMMLQ	valeur exacte de la norme du résidu
1	saut	15.81138830084190	
2	15.81138830084190	15.81138830084190	15.81138830084190
3	saut	15.81138830084190	
250	15.81138830084182	15.81138830084188	15.81138830084190
251	saut	15.81138830084183	
252	15.81138830084174	15.81138830084183	15.81138830084190
498	15.81138830084136	15.81138830084106	15.81138830084190
499	saut	$7.563813\ 10^{-12}$	
500	$8.936\ 10^{-11}$	$7.556007\ 10^{-12}$	

N=1000

itération	Lanczos/Othodir OR	SYMMLQ	valeur exacte de la norme du résidu
1	saut	22.36067977499790	
2	22.36067977499790	22.36067977499790	22.36067977499790
3	saut	22.36067977499790	
498	22.36067977499721	22.36067977499918	22.36067977499790
499	saut	22.36067977499935	
500	22.36067977499758	22.36067977499936	22.36067977499790
997	saut	22.36067977500446	
998	22.36067977500651	22.36067977500446	22.36067977499790
999	saut	$1.631434\ 10^{-11}$	
1000	$8.94267\ 10^{-10}$	$1.627618\ 10^{-11}$	

TAB. 3.1 – Norme du résidu obtenu par Lanczos/Orthodir OR et SYMMLQ

N=100

itération	Lanczos/Othodir MR	MINRES	valeur exacte de la norme du résidu
1	10.000000	10.000000	10.000000
2	5.77350269189626	5.77350269189626	5.77350269189626
3	5.77350269189626	5.77350269189626	5.77350269189626
10	3.01511344577763	3.01511344577763	3.01511344577763
11	3.01511344577763	3.01511344577763	3.01511344577763
12	2.77350098112614	2.77350098112614	2.77350098112614
13	2.77350098112614	2.77350098112614	2.77350098112614
96	1.01534616513362	1.01534616513362	1.01534616513362
97	1.01534616513362	1.01534616513362	1.01534616513362
98	1.00503781525922	1.00503781525922	1.00503781525922
99	1.00503781525922	1.00503781525922	1.00503781525922
100	$1.067661\ 10^{-12}$	$2.906615\ 10^{-13}$	0.0

N=500

itération	Lanczos/Othodir MR	MINRES	valeur exacte de la norme du résidu
1	22.36067977499790	22.36067977499790	22.36067977499790
2	12.90994448735806	12.90994448735806	12.90994448735806
3	12.90994448735806	12.90994448735806	12.90994448735806
250	1.41139359234409	1.41139359234409	1.41139359234409
251	1.41139359234409	1.41139359234409	1.41139359234409
252	1.40580389278883	1.40580389278883	1.40580389278883
498	1.00100150250438	1.00100150250438	1.00100150250438
499	1.00100150250438	1.00100150250438	1.00100150250438
500	$5.884465\ 10^{-11}$	$9.047356\ 10^{-12}$	0.0

N=1000

itération	Lanczos/Othodir MR	MINRES	valeur exacte de la norme du résidu
1	31.62277660168379	31.62277660168379	31.62277660168379
2	18.25741858350554	18.25741858350554	18.25741858350554
3	1.825741858350554	18.25741858350554	18.25741858350554
498	1.41562990079756	1.41562990079757	1.41562990079754
499	1.41562990079756	1.41562990079757	1.41562990079754
500	1.41280146660172	1.41280146660173	1.41280146660170
501	1.41280146660172	1.41280146660174	1.41280146660170
997	1.00150338345975	1.00150338345975	1.00150338345970
998	1.00050037531282	1.00050037531282	1.00050037531277
999	1.00050037531282	1.00050037531282	1.00050037531277
1000	$2.840777\ 10^{-10}$	$1.095730\ 10^{-11}$	0.0

TAB. 3.2 – Norme du résidu obtenu par Lanczos/Orthodir MR et MINRES

Exemple 3.4.5

Considérons la matrice symétrique

$$A = \begin{bmatrix} 0 & F \\ F^T & 0 \end{bmatrix}$$

de dimension $N \times N$, avec F une matrice carrée de dimension $s \times s$ et $N = 2s$. En prenant $x_0 = (0, \ldots, 0)^T \in I\!R^N$ et $b = [b_1, b_2]^T \in I\!R^N$ avec $b_2 = (0, \ldots, 0)^T \in I\!R^s$, il se produit dans les méthodes de type Lanczos un *breakdown* à toutes les itérations impaires.

Dans un premier temps, nous prenons $F =$ **pde225** la matrice réelle non symétrique de dimension 225×225 de la collection **NEP** générée par **MATPDE**, cette matrice contient 1065 éléments non nuls.
En prenant $b_1 = (1, 1, \ldots, 1)^T$, les algorithmes Lanczos/Orthodir OR et SYMMLQ ont un comportement presque identique jusqu'à l'itération $k = 321$, à savoir que les deux donnent le même résidu à chaque itération paire, avec un saut de longueur deux pour le premier et une stagnation du résidu pour le deuxième aux itérations impaires (*true-breakdown*).
Même constat pour les algorithmes Lanczos/Orthodir MR et MINRES qui convergent de la même façon jusqu'à l'itération $k = 328$, avec un avantage, par la suite, en faveur du Lanczos/Orthodir MR qui a la meilleure convergence. Nous remarquons une stagnation des résidus de ces deux méthodes à chaque itération impaire (*ghost-breakdown*). Les résultats obtenus sont donnés dans la figure 3.5.

Considérons maintenant $F =$ **pde900** matrice non symétrique de dimension 900×900 de la collection **NEP** générée par **MATPDE**, cette matrice contient 4380 éléments non nuls. Les résultats obtenus sont donnés dans la figure 3.6. Les algorithmes étudiés se comportent de la même façon que dans le premier exemple.

Enfin, nous prenons F égal à la matrice donnée dans l'exemple 3.4.1. Nous obtenons une matrice A symétrique non définie positive de dimension 800×800. L'application des algorithmes Lanczos/Orthodir OR , Lanczos/Orthodir MR, SYMMLQ et MINRES nous donne les résultats présentés dans la figure 3.7. Nous remarquons le même comportement chez ces algorithmes que celui obtenu dans les deux exemples précédents.

Pour ces trois matrices, nous remarquons que les courbes du MINRES et du SYMMLQ présentent une stagnation numérique après un certain nombre d'itérations et que l'algorithme Lanczos/Orthodir donne les meilleurs résultats.

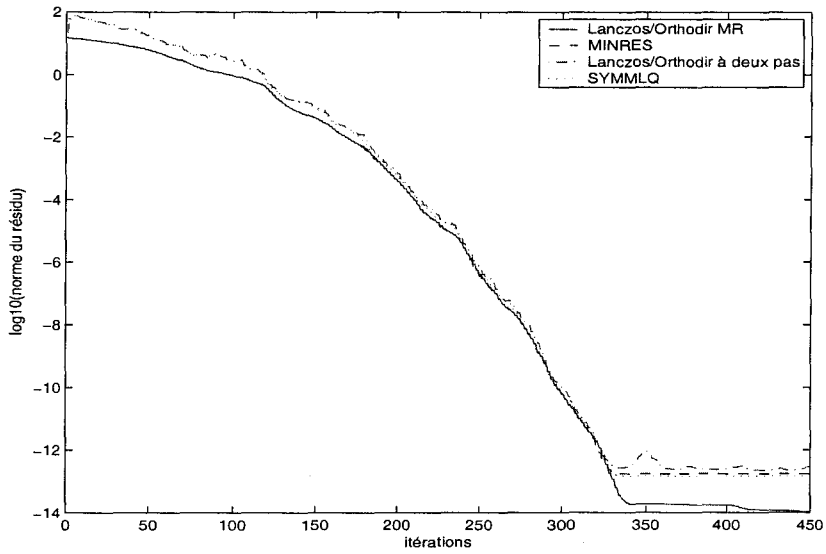


FIG. 3.5 – $N = 450$, $F = \mathbf{pde225}$

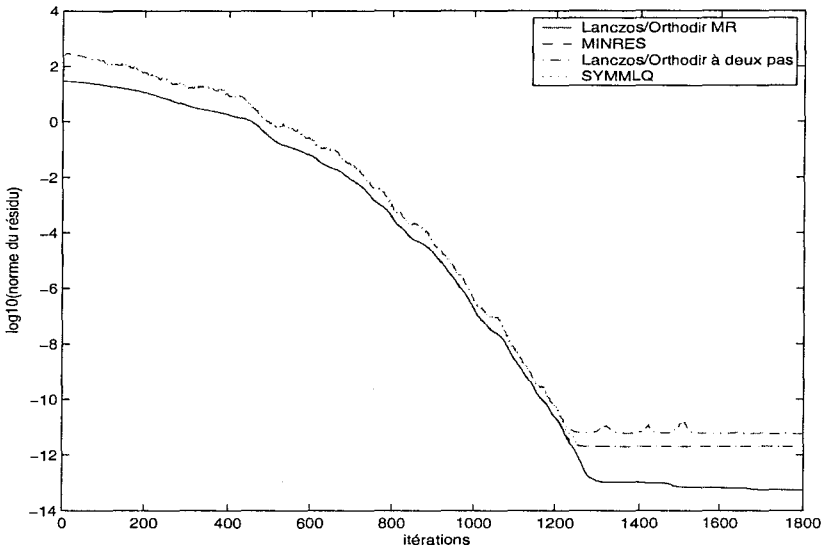


FIG. 3.6 – $N = 1800$, $F = \mathbf{pde900}$

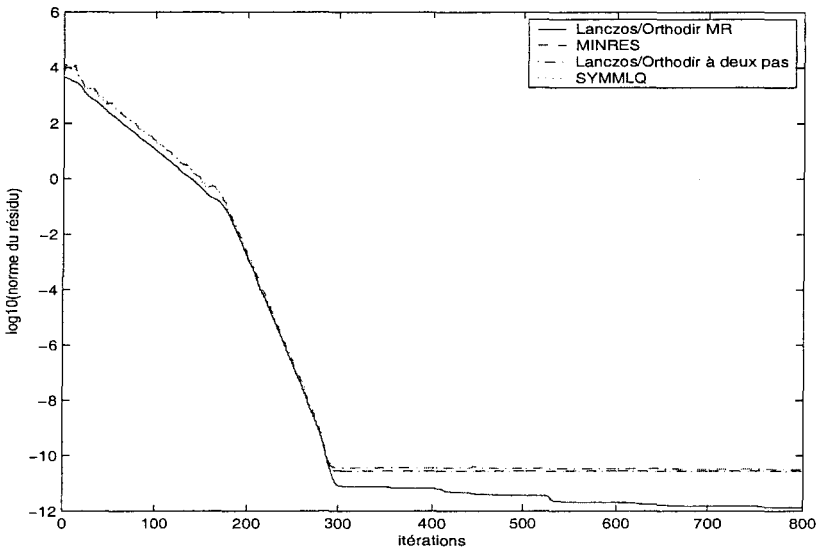


FIG. 3.7 – $N = 800$

3.5 Conclusion

Dans la première partie de ce chapitre, nous avons proposé une normalisation de l'algorithme MRZ-stab par le produit scalaire dénominateur, elle consiste à remplacer les vecteurs de direction de MRZ-stab par des vecteurs normalisés par un produit scalaire, ce qui nous a permis de réduire le nombre de vecteurs et coefficients stockés. L'algorithme obtenu se comporte d'une manière plus stable numériquement et permet d'éviter les dépassements de capacité appelés *overflow* et *underflow* provoqués par les multiplications successives des matrices A et A^T par les vecteurs de direction. Des variantes de cet algorithme ont été proposées dans le cadre d'une étude des problèmes de *breakdown* qui risquent de se produire dans les méthodes de type Lanczos dans le cas symétrique non défini positif. Nous avons montré que dans le cas d'un *breakdown* certains choix des vecteurs initiaux nous permettent de développer des algorithmes avec *look-ahead* en faisant un saut qui ne dépasse pas deux.
Des tests numériques qui approuvent ces résultats sont notamment proposés.

Une application des algorithmes Lanczos/Orthodir MR et Lanczos/Orthodir OR , proposés ici, peut être envisagée dans le cas où la matrice A du système est non-symétrique. Cela consiste à considérer le système symétrique augmenté

$$\begin{bmatrix} 0 & A \\ A^T & 0 \end{bmatrix} \begin{bmatrix} 0 \\ x \end{bmatrix} = \begin{bmatrix} b \\ 0 \end{bmatrix}$$

En appliquant les algorithmes Lanczos/Orthodir MR et Lanczos/Orthodir OR à ce système, nous évitons le problème de *breakdown* qui se produit dans les méthodes de type Lanczos à chaque itération impaire. Nous pouvons ainsi extraire deux algorithmes sans *breakdown* pour résoudre le système $Ax = b$. Les algorithmes obtenus peuvent être comparés aux algorithmes CGNE et CGNR.

Chapitre 4

Méthodes de projection oblique pour la résolution des systèmes linéaires avec plusieurs seconds membres

4.1 Introduction

Dans de nombreuses applications, nous sommes amenés à résoudre plusieurs systèmes linéaires avec la même matrice et différents seconds membres, comme, par exemple, les simulations numériques de propagation des ondes. Au lieu de résoudre chaque système indépendamment par une méthode itérative, il est plus efficace de traiter le problème d'une manière globale. Lorsque tous les seconds membres sont connus, notre problème s'écrit

$$AX = B \tag{4.1}$$

où A est une matrice réelle de dimension $N \times N$ quelconque, $B = [b_1, b_2, \ldots, b_s]$ et $X = [x_1, x_2, \ldots, x_s]$ sont deux matrices rectangulaires de dimension $N \times s$ avec $s \ll N$.

Plusieurs méthodes de sous-espace de Krylov par bloc ont été développées récemment. Parmi ces méthodes, nous trouvons le Bl-BICG [55, 70], le BGMRES introduit dans [79] et étudié dans [64], le Bl-QMR [33, 52] et le Bl-BiCGSTAB [28]. Notons que ces méthodes nécessitent l'application d'une procédure de déflation pour détecter et supprimer les dépendances linéaires qui peuvent exister entre les colonnes des itérés. Cette technique a été introduite dans [1] et appliquée au Bl-QMR [33].

Il existe une autre approche pour résoudre le système (4.1) : elle consiste à résoudre un système particulier appelé "single-seed" et à projeter les résidus des autres systèmes sur le sous-espace de Krylov obtenu. Le processus est répété jusqu'à ce que tous les systèmes soient résolus, voir [23, 48, 71, 72]. Cette technique est spécialement intéressante dans le cas où les seconds membres ne sont pas tous connus en même temps ; voir par exemple [62, 76].

Dans ce chapitre, nous allons utiliser une troisième approche pour résoudre le problème (4.1). Cette approche, qui a permis de donner l'algorithme du GMRES global [46], est basée sur une projection oblique par rapport aux sous-espaces de Krylov matriciels. Nous commencerons d'abord par développer le processus de Lanczos global, ainsi que les méthodes de type Lanczos globales telles que le Gradient Biconjugué global (Gl-BICG) et le BiCGSTAB global. Des situations de *breakdown* peuvent survenir dans ces algorithmes. Pour résoudre ce problème, nous donnerons des versions avec *look-ahead* de certaines méthodes de type Lanczos.
Dans ce travail, nous allons utiliser les notations suivantes : si X et Y sont deux matrices de dimension $N \times s$, nous considérons le produit scalaire $< X, Y >_F = tr(X^T Y)$ où $tr(Z)$ est la trace de la matrice carrée Z, X^T la matrice transposée de X. La norme associée à ce produit est la norme de Frobenius, elle sera notée $\| \, . \, \|_F$.
Deux matrices de $\mathbb{R}^{N\times s}$ sont dites F-orthogonales si elles sont orthogonales par rapport au produit scalaire $< ., . >_F$.
Si $V \in \mathbb{R}^{N\times s}$, le sous-espace de Krylov matriciel $\mathcal{K}_k(A, V)$ est le sous-espace de $\mathbb{R}^{N\times s}$ engendré par les matrice $V, AV, \ldots, A^{k-1}V$.

4.2 Le processus de Lanczos global

Le processus de Lanczos global est une méthode qui permet de transformer une matrice quelconque en une matrice tridiagonale en utilisant une projection oblique par rapport aux sous-espaces de Krylov matriciels. Cette projection définit la procédure de Lanczos globale qui va nous permettre d'introduire des algorithmes globaux pour résoudre les systèmes linéaires avec plusieurs seconds membres. Nous commençons par donner la définition des sous-espaces de krylov matriciels.

Soit V une matrice réelle de dimension $N \times s$. D'après la définition du sous-espace $\mathcal{K}_k(A, V)$, nous avons

$$Z \in \mathcal{K}_k(A, V) \Longleftrightarrow Z = \sum_{i=1}^{k} \alpha_i A^{i-1} V; \;\; \alpha_i \in \mathbb{R}; \;\; i = 1, \ldots, k.$$

En d'autres termes, $\mathcal{K}_k(A, V)$ est le sous-espace de $\mathbb{R}^{N\times s}$ qui contient toutes les matrices de dimension $N \times s$ s'écrivant sous la forme $Z = P(A)V$, où P est un polynôme de degré au plus égal à $k-1$. Cela veut dire que chaque vecteur colonne de Z est associé à un sous-espace de Krylov.
Remarquons que $\mathcal{K}_k(A, V)$ est différent du sous-espace de Krylov par bloc $\tilde{\mathcal{K}}_k(A, V)$ utilisé

dans les méthodes par bloc. En effet

$$Z \in \tilde{\mathcal{K}}_k(A,V) \Longleftrightarrow Z = \sum_{i=1}^{k} A^{i-1} V \Omega_i;\ \ \Omega_i \in I\!R^{s\times s},\ i = 1,\ldots,k.$$

Dans ce cas, chaque vecteur colonne de Z est associé à la somme de s sous-espaces de Krylov.

Le polynôme minimal de A par rapport à V est le polynôme unitaire non nul de plus petit degré tel que $P(A)V = 0$, le degré m de ce polynôme ne peut dépasser N. Compte-tenu de cette définition, nous avons la propriété suivante

Propriété 4.2.1

Si m est le degré du polynôme minimal de A par rapport à V, alors nous avons

1. *$\mathcal{K}_m(A,V)$ est invariant par rapport à A.*
2. *$dim\ \mathcal{K}_k(A,V) = min(k,m)$.*

Soient V_1 et W_1 deux matrices de dimension $N \times s$, notons $\mathcal{K}_k(A,V_1)$ et $\mathcal{K}_k(A^T,W_1)$ les sous-espaces de Krylov matriciels engendrés respectivement par $\{V_1, AV_1, \ldots, A^{(k-1)}V_1\}$ et $\{W_1, A^TW_1, \ldots, A^{T(k-1)}W_1\}$. Le processus de Lanzos global construit deux bases bi-orthogonales $\{V_1, V_2, \ldots, V_k\}$ et $\{W_1, W_2, \ldots, W_k\}$ des deux sous-espaces de Krylov matriciels $\mathcal{K}_k(A,V_1)$ et $\mathcal{K}_k(A^T,W_1)$ respectivement, telles que

$$< V_i, W_j >_F = tr(V_i^T\, W_j) = \delta_{ij};\ i,j = 1,\ldots,k.$$

L'algorithme est défini comme suit

Algorithme 4.2.1 *:Le processus de Lanczos global.*

1. Initialisation
   Choisir V_1 et W_1 deux matrices de dimension $N \times s$ telles que $< V_1, W_1 >_F = 1$
   $\beta_1 = \delta_1 = 0$ et $W_0 = V_0 = 0$.
2. Pour $j = 1,2,\ldots,k$
   $\alpha_j = tr(W_j^T AV_j)$
   $\tilde{V}_{j+1} = AV_j - \alpha_j V_j - \beta_j V_{j-1}$
   $\tilde{W}_{j+1} = A^TW_j - \alpha_j W_j - \delta_j W_{j-1}$
   $\delta_{j+1} = \mid tr(\tilde{V}_{j+1}^T \tilde{W}_{j+1}) \mid^{1/2}$
   $\beta_{j+1} = tr(\tilde{V}_{j+1}^T \tilde{W}_{j+1})/\delta_{j+1}$
   $V_{j+1} = \tilde{V}_{j+1}/\delta_{j+1}$
   $W_{j+1} = \tilde{W}_{j+1}/\beta_{j+1}$
   Fin

Dans cet algorithme, il se produit un *breakdown* à la $j^{\text{ème}}$ itération lorsque

$$tr(\tilde{V}_{j+1}^T \tilde{W}_{j+1}) = 0.$$

Nous allons voir par la suite que la technique du *look-ahead* permet d'éviter ce problème, cette technique sera appliquée aux méthodes de type Lanczos.
Si $\tilde{V}_{j+1} = 0$, alors le sous-espace de Krylov matriciel $\mathcal{K}_j(A, V_1)$ est invariant par rapport à A. Donc $j \geq m$, avec m le degré du polynôme minimal de A par rapport à V_1. Dans ce cas nous obtenons la solution exacte du problème (4.1).
Nous remarquons que l'algorithme 4.2.1 n'exige aucune inversion de matrice, c'est pourquoi le problème de la dépendance linéaire des vecteurs de $V_1, AV_1, \ldots$ et $W_1, A^T W_1, \ldots$ ne se pose pas ici. Donc nous n'avons pas besoin de procédure de déflation pour supprimer les dépendances et les presque-dépendances linéaires de ces vecteurs.

Prenons maintenant $\mathcal{V}_k = [V_1, \ldots, V_k]$ et $\mathcal{W}_k = [W_1, \ldots, W_k]$ deux matrices de dimension $N \times ks$. Soit T_k la matrice tridiagonale de dimension $k \times k$ définie par

$$T_k = \begin{pmatrix} \alpha_1 & \beta_2 & & \\ \delta_2 & \alpha_2 & \ddots & \\ & \ddots & \ddots & \beta_k \\ & & \delta_k & \alpha_k \end{pmatrix}$$

où α_i, β_i et δ_i sont les scalaires définis dans l'algorithme 4.2.1. Notons que dans l'algorithme de Lanczos par bloc, la matrice correspondante à T_k est une matrice tridiagonale par bloc de dimension $ks \times ks$.
Considérons la matrice

$$\tilde{T}_k = \begin{pmatrix} T_k \\ \delta_{k+1} e_k^T \end{pmatrix},$$

avec $e_k = (0, \ldots, 0, 1)^T \in \mathbb{R}^k$.
Dorénavant, nous utiliserons la notation $*$ donnée dans [46] pour définir le produit suivant

$$\mathcal{V}_k * y = \sum_{i=1}^{k} y^i V_i = \mathcal{V}_k (y \otimes I_s) \tag{4.2}$$

où $y = (y^1, y^2, \ldots, y^k)^T$ est un vecteur quelconque de $\mathbb{R}^k$. Le produit $\otimes$ est défini par

$$y \otimes I_s = \begin{pmatrix} y_1 I_s & & 0 \\ & \ddots & \\ 0 & & y_k I_s \end{pmatrix}.$$

Et de façon similaire, nous avons

$$\mathcal{V}_k * T_k = [\mathcal{V}_k * T_{.,1}, \mathcal{V}_k * T_{.,2}, \ldots, \mathcal{V}_k * T_{.,k}], \tag{4.3}$$

où $T_{.,i}$ est la $i^{\text{ème}}$ colonne de la matrice T_k.
En utilisant ces notations, nous avons le résultat suivant

Propriété 4.2.2
S'il ne se produit pas de breakdown dans l'algorithme de Lanczos global jusqu'à la $k^{ème}$ itération, alors $\{V_1, \ldots, V_k\}$ et $\{W_1, \ldots, W_k\}$ forment des bases des sous-espaces de Krylov matriciels $\mathcal{K}_k(A, V_1)$ et $\mathcal{K}_k(A^T, W_1)$ respectivement. Et nous avons les relations suivantes

$$A\,\mathcal{V}_k = \mathcal{V}_k * T_k + \delta_{k+1}[0, \ldots, 0, V_{k+1}], \tag{4.4}$$
$$A\,\mathcal{V}_k = \mathcal{V}_{k+1} * \tilde{T}_k. \tag{4.5}$$

Preuve :
À partir de la définition du produit $*$ et de la structure de la matrice T_k, nous avons pour $j = 1, \ldots, k-1$

$$\begin{aligned} \mathcal{V}_k * T_{.,j} &= \alpha_j V_j + \delta_{j+1} V_{j+1} + \beta_j V_{j-1} \\ &= AV_j, \end{aligned}$$

et

$$\begin{aligned} \mathcal{V}_k * T_{.,k} &= \beta_k V_{k-1} + \alpha_k V_k \\ &= AV_k + \delta_{k+1} V_{k+1}. \end{aligned}$$

Alors, nous obtenons

$$A\,\mathcal{V}_k = \mathcal{V}_k * T_k + \delta_{k+1}[0, \ldots, 0, V_{k+1}].$$

D'autre part, nous avons

$$\begin{aligned} \mathcal{V}_{k+1} * \tilde{T}_k &= [\mathcal{V}_k, V_{k+1}] * \tilde{T}_k \\ &= \mathcal{V}_k * T_k + \delta_{k+1} e_k^T V_{k+1} \\ &= \mathcal{V}_k * T_k + \delta_{k+1}[0, \ldots, 0, V_{k+1}]. \end{aligned}$$

En utilisant la relation (4.4), nous obtenons le résultat de (4.5).

□

Considérons maintenant le système linéaire (4.1). Soient X_0 une approximation initiale de la solution et $R_0 = B - AX_0$ le résidu correspondant. La méthode de Lanczos globale pour la résolution du système (4.1) consiste à déterminer une suite d'approximations X_k telles que

$$X_k - X_0 = Z_k \in \mathcal{K}_k(A, R_0) \tag{4.6}$$

et

$$R_k = B - AX_k \perp_F \mathcal{K}_k(A^T, \tilde{R}_0), \tag{4.7}$$

où $\tilde{R}_0$ est une matrice de dimension $N \times s$ choisie de façon à ce que $< R_0, \tilde{R}_0 >_F \neq 0$.
Si nous notons $\mathcal{S}_k$ la projection oblique sur $A\mathcal{K}_k(A, R_0)$ orthogonalement à $\mathcal{K}_k(A^T, \tilde{R}_0)$, alors les relations (4.6) et (4.7) s'écrivent

$$R_k = R_0 - \mathcal{S}_k R_0. \tag{4.8}$$

Prenons $\{V_1, \ldots, V_k\}$ et $\{W_1, \ldots, W_k\}$ l'ensemble des matrices construites par l'algorithme 4.2.1 et qui engendrent les sous-espaces de Krylov matriciels $\mathcal{K}_k(A, R_0)$ et $\mathcal{K}_k(A^T, \tilde{R}_0)$, respectivement, avec les initialisations $V_1 = R_0 / \parallel R_0 \parallel_F$ et W_1 est tel que $< V_1, W_1 >_F = 1$.
La relation (4.6) nous permet d'écrire

$$X_k = X_0 + \mathcal{V}_k * y_k \tag{4.9}$$

où y_k est le vecteur de $\mathbb{R}^k$ obtenu par la relation

$$< R_0 - A\mathcal{V}_k * y_k, W_i >_F = 0,$$

ce qui est équivalent à

$$< R_0, W_i >_F = < A\mathcal{V}_k * y_k, W_i >_F \quad \text{pour} \quad i = 1, \ldots, k. \tag{4.10}$$

Cette relation peut aussi s'écrire sous la forme

$$\sum_{j=1}^{k} y_k^j tr(W_1^T A V_j) = \parallel R_0 \parallel_F$$

et

$$\sum_{j=1}^{k} y_k^j tr(W_i^T A V_j) = 0 \quad \text{pour} \quad i = 2, \ldots, k.$$

Nous obtenons ainsi le système linéaire suivant

$$T_k y_k = \parallel R_0 \parallel_F e_1^{(k)}, \tag{4.11}$$

où $e_1^{(k)}$ est le premier vecteur de la base canonique de $\mathbb{R}^k$.
Si la matrice tridiagonale T_k est non-singulière, alors les approximations X_k, obtenues par la méthode de Lanczos globale, sont données par la relation suivante

$$X_k = X_0 + \parallel R_0 \parallel_F \mathcal{V}_k * T_k^{-1} e_1^{(k)}. \tag{4.12}$$

Maintenant, nous allons donner l'expression de la norme du résidu R_k, ce qui nous permet de vérifier si la convergence est atteinte sans passer par les approximations X_k.
Le résidu R_k est donné par

$$R_k = R_0 - A\mathcal{V}_k * y_k.$$

Grâce à la relation (4.4) et au fait que $R_0 = \| R_0 \|_F V_1$, nous avons

$$R_k = \| R_0 \|_F V_1 - \mathcal{V}_k * T_k y_k - \delta_{k+1}[0, 0, \ldots, V_{k+1}] * y_k.$$

D'autre part, puisque $V_1 = \mathcal{V}_k * e_1^{(k)}$, alors

$$R_k = \mathcal{V}_k * (\| R_0 \|_F e_1^{(k)} - T_k y_k) - \delta_{k+1}[0, 0, \ldots, V_{k+1}] * y_k.$$

Enfin, en utilisant la relation (4.11), nous obtenons

$$\| R_k \|_F = | \delta_{k+1} y_k^k | \| V_{k+1} \|_F, \tag{4.13}$$

où y_k^k est la dernière composante du vecteur y_k.

Remarque 4.2.1

1. *Si m est le degré du polynôme minimal de A par rapport à R_0, alors $\mathcal{K}_m(A, R_0)$ est invariant et ainsi $X_m = X$ est la solution du système (4.1).*
2. *Comme $m \leq N$, l'algorithme converge en au plus N itérations.*

Dans ce qui va suivre, nous allons donner quelques-unes des méthodes de type Lanczos globales pour la résolution du système linéaire (4.1).

4.3 Méthodes de type Lanczos globales

Comme nous l'avons mentionné auparavant, de nouvelles méthodes globales seront introduites. Il s'agit des méthodes Gradient Biconjugué global et BiCGSTAB global.

4.3.1 L'algorithme du Gradient Biconjugué global

L'algorithme du Gradient Biconjugé Global (Gl-BICG) peut être développé à partir de l'algorithme 4.2.1 de la même manière que dans le cas du BICG classique donné dans [29]. À la $k^{\text{ème}}$ itération, nous obtenons le résidu R_k tel que $R_k - R_0$ appartient au sous-espace de Krylov matriciel $\mathcal{K}_k(A, AR_0) = span\{AR_0, A^2R_0, \ldots, A^kR_0\}$ avec R_k F-orthognal par rapport au sous-espace de Krylov matriciel $\mathcal{K}_k(A^T, \tilde{R}_0) = span\{\tilde{R}_0, A^T\tilde{R}_0, \ldots, A^{T^{k-1}}\tilde{R}_0\}$, où $\tilde{R}_0$ est une matrice choisie de dimension $N \times s$. L'algorithme obtenu est donné ci-dessous

Algorithme 4.3.1 *:Gradient Biconjugué global.*

1. Initialisation
   Choisir X_0, calculer $R_0 = B - AX_0$;
   Choisir $\tilde{R}_0$ tel que $< R_0, \tilde{R}_0 >_F \neq 0$;
   $P_0 = R_0$ et $\tilde{P}_0 = \tilde{R}_0$;
2. Itérations
   Pour $j = 0, 1, \ldots$
   $\alpha_j =< R_j, \tilde{R}_j >_F / < AP_j, \tilde{P}_j >_F$;
   $X_{j+1} = X_j + \alpha_j P_j$;
   $R_{j+1} = R_j - \alpha_j AP_j$;
   $\tilde{R}_{j+1} = \tilde{R}_j - \alpha_j A^T \tilde{P}_j$;
   $\beta_j =< R_{j+1}, \tilde{R}_{j+1} >_F / < R_j, \tilde{R}_j >_F$;
   $P_{j+1} = R_{j+1} + \beta_j P_j$;
   $\tilde{P}_{j+1} = \tilde{R}_{j+1} + \beta_j \tilde{P}_j$;
   Fin.

Comme pour l'algorithme du BICG classique, nous avons les résultats suivants

Propriété 4.3.1
Les matrices produites par l'algorithme Gl-BICG vérifient les relations suivantes

1. $< R_k, \tilde{R}_l >_F = 0$ *et* $< AP_k, \tilde{P}_l >_F = 0$ *pour tout* $k \neq l$.
2. $span\{P_0, \ldots, P_k\} = span\{R_0, \ldots, A^k R_0\}$.
3. $span\{\tilde{P}_0, \ldots, \tilde{P}_k\} = span\{\tilde{R}_0, \ldots, A^{Tk} \tilde{R}_0\}$.
4. $R_k - R_0 \in A\mathcal{K}_k(A, R_0)$ *et* R_k *est orthogonal à* $\mathcal{K}_k(A^T, \tilde{R}_0)$.

Le résidu R_k produit par l'algorithme du Gl-BICG peut être exprimé par la relation

$$R_k = \mathcal{P}_k(A) R_0,$$

où $\mathcal{P}_k$ est un polynôme de degré k ayant des coefficients scalaires et vérifiant $\mathcal{P}_k(0) = 1$. La matrice de direction P_k peut aussi s'écrire sous la forme

$$P_k = \phi_k(A) R_0$$

où ϕ est un polynôme ayant des coefficients scalaires. Notons que $\tilde{R}_k$ et $\tilde{P}_k$ peuvent aussi s'écrire

$$\tilde{R}_k = \mathcal{P}_k(A^T) \tilde{R}_0 \quad \text{et} \quad \tilde{P}_k = \phi_k(A^T) \tilde{R}_0.$$

Dans les méthodes par bloc, les itérés sont exprimés par des polynômes matriciels. Comme dans l'algorithme du BICG classique, des problèmes de *breakdown* peuvent survenir dans l'algorithme Gl-BICG. Dans ce qui va suivre, nous allons appliquer la stratégie du *look-ahead* et donner des algorithmes qui permettent d'éviter ce problème.

4.3.2 L'algorithme du BiCGSTAB global

Nous avons vu que le résidu R_k^{gb} et la matrice de direction P_k^{gb}, produits par l'algorithme du Gl-BICG à la $k^{\text{ème}}$ itération, vérifient les relations suivantes

$$R_k^{gb} = R_{k-1}^{gb} - \alpha_k A P_{k-1}^{gb} \tag{4.14}$$

$$P_k^{gb} = R_k^{gb} + \beta_k P_{k-1}^{gb}. \tag{4.15}$$

Le résidu R_k de l'algorithme du BiCGSTAB global est défini par

$$R_k = (I - \omega_k A) S_k,$$

où

$$S_k = (I - \omega_{k-1} A) \ldots (I - \omega_1 A) R_k^{gb}. \tag{4.16}$$

Le paramètre ω_k est choisi tel que la norme de Frobenius du résidu R_k soit minimale. Nous avons donc

$$\omega_k = \frac{< S_k, AS_k >_F}{< AS_k, AS_k >_F}.$$

En utilisant les relations (4.14) et (4.15), nous obtenons

$$S_k = R_{k-1} - \alpha_k A P_{k-1}$$

avec

$$P_{k-1} = (I - \omega_{k-1} A) \ldots (I - \omega_1 A) P_{k-1}^{gb}.$$

Compte tenu du fait que le résidu R_k^{gb} $(k \geq 1)$ est F-orthogonal par rapport au sous-espace de Krylov matriciel $\mathcal{K}_k(A^T, \tilde{R}_0)$, la relation (4.16) nous donne

$$< \tilde{R}_0, S_k >_F = 0 \quad k \geq 1.$$

En utilisant cette condition d'orthogonalité, nous obtenons

$$\alpha_k = \frac{< \tilde{R}_0, R_{k-1} >_F}{< \tilde{R}_0, AP_{k-1} >_F}.$$

La matrice de direction P_k du BiCGSTAB global est donnée par

$$P_k = (I - \omega_k A) \ldots (I - \omega_1 A) [R_k^{gb} + \beta_k P_{k-1}^{gb}],$$

et ainsi

$$P_k = (I - \omega_k A)(S_k + \beta_k P_{k-1}).$$

Nous pouvons réécrire cette relation sous la forme

$$P_k = (I - \omega_k A)\, Q_k,$$

avec

$$\begin{aligned} Q_k &= S_k + \beta_k P_{k-1} \\ &= (I - \omega_{k-1} A) \ldots (I - \omega_1 A)\, P_k^{gb}. \end{aligned} \tag{4.17}$$

Comme les matrices de direction P_k^{gb} sont F-orthogonales par rapport au sous-espace de Krylov matriciel $\mathcal{K}_k(A^T, \tilde{R}_0)$, nous avons les relations

$$< \tilde{R}_0, A\, Q_k >_F = 0; \quad k \geq 1.$$

En appliquant ces orthogonalités à la relation (4.17), nous obtenons l'expression du coefficient β_k

$$\beta_k = -\frac{< \tilde{R}_0, AS_k >_F}{< \tilde{R}_0, AP_{k-1} >_F}.$$

L'algorithme BiCGSTAB global est représenté comme suit

Algorithme 4.3.2 *: Gl-BiCGSTAB.*

1. Initialisation
   Choisir X_0, $\tilde{R}_0$;
   $R_0 = B - AX_0$; $P_0 = R_0$
2. Itérations
   Pour $k = 1, 2, \ldots$
   $V_{k-1} = AP_{k-1}$;
   $\alpha_k = < \tilde{R}_0, R_{k-1} >_F \; / \; < \tilde{R}_0, V_{k-1} >_F$;
   $S_k = R_{k-1} - \alpha_k V_{k-1}$;
   $T_k = AS_k$;
   $\omega_k = < T_k, S_k >_F \; / \; < T_k, T_k >_F$;
   $R_k = S_k - \omega_k T_k$;
   $\beta_k = -\dfrac{< \tilde{R}_0, T_k >_F}{< \tilde{R}_0, V_{k-1} >_F}$;
   $P_k = R_k + \beta_k (P_{k-1} - \omega_k V_{k-1})$;
Fin.

Dans le cas où $s = 1$, cet algorithme coincide exactement avec la méthode BiCGSTAB de van der Vorst [77].

Remarque 4.3.1 *Dans les méthodes globales nous n'avons pas les problèmes de dépendances linéaires qui affectent les méthodes par bloc. Par contre, des situations de breakdown peuvent survenir dans ces algorithmes.*

4.4 Méthodes de type Lanczos globales avec look-ahead

Dans ce paragraphe, nous appliquons la technique du *look-ahead* aux méthodes de type Lanczos afin de construire des algorithmes qui permettent d'éviter les situations de *breakdown*. Pour cela nous commençons par établir le rapport entre les polynômes orthogonaux formels et la méthode de Lanczos globale.

Dans la méthode de Lanczos globale, nous construisons une suite (X_k), $k = 1, 2, \ldots$ telle que

$$X_k - X_0 \in \mathcal{K}_k(A, R_0)$$

et

$$R_k \perp_F \mathcal{K}_k(A^T, \tilde{R}_0)$$

avec $\tilde{R}_0$ et X_0 deux matrices de dimension $N \times s$ choisies arbitrairement. Le résidu R_k peut s'écrire sous la forme

$$R_k = \mathcal{P}_k(A)R_0$$

où $\mathcal{P}_k$ est un polynôme de degré au plus égal à k avec $\mathcal{P}_k(0) = 1$. Les conditions d'orthogonalité donnent

$$< A^{T^i}\tilde{R}_0, R_k >_F = 0 \quad \text{pour} \quad i = 0, 1, \ldots, k-1.$$

Si, de plus, nous définissons la fonctionnelle c sur l'espace des polynômes réels par

$$c(\xi^i) = c_i = < \tilde{R}_0, A^i R_0 >_F \qquad \text{pour} \quad i = 0, 1, \ldots$$

alors nous obtenons

$$c(\xi^i \mathcal{P}_k(\xi)) = 0 \quad \text{pour} \quad i = 0, 1, \ldots, k-1. \tag{4.18}$$

Ces conditions montrent que $\mathcal{P}_k$ est le polynôme de degré au plus k appartenant à la famille des polynômes orthogonaux formels par rapport à c normalisés par la condition $\mathcal{P}_k(0) = 1$.

Les conditions d'orthogonalité (4.18) prouvent que $\mathcal{P}_k$ existe et est unique si et seulement si le déterminant de Hankel

$$H_k^{(1)} = \begin{vmatrix} c_1 & \cdots & c_k \\ \vdots & & \vdots \\ c_k & \cdots & c_{2k-1} \end{vmatrix} \neq 0.$$

Nous considérons $c^{(1)}$ la fonctionnelle définie sur l'espace polynômial par $c^{(1)}(t^i) = c(\xi^{i+1}) = c_{i+1}$ et $\mathcal{P}_k^{(1)}$ le polynôme unitaire de degré k appartenant à la famille des polynômes orthogonaux formels par rapport à la fonctionnelle $c^{(1)}$. $\mathcal{P}_k$ et $\mathcal{P}_k^{(1)}$ existent sous l'unique condition $H_k^{(1)} \neq 0$.

Comme nous l'avons déjà vu dans les chapitres précédents, nous pouvons calculer ces polynômes par des relations de récurrence. Les différentes relations de récurrence permettent la mise en œuvre des différentes méthodes de type Lanczos globales. Par exemple, dans le Gl-Lanczos/Ortores, $\mathcal{P}_{k+1}$ est calculé à partir de $\mathcal{P}_k$ et $\mathcal{P}_{k-1}$. Dans le Gl-Lanczos/Orthomin (Gl-BICG), $\mathcal{P}_{k+1}$ est calculé à partir de $\mathcal{P}_k$ et $\mathcal{P}_k^{(1)}$ et le polynôme $\mathcal{P}_{k+1}^{(1)}$ est obtenu à partir de $\mathcal{P}_{k+1}$ et $\mathcal{P}_k^{(1)}$, (ici on exige que le polynôme $\mathcal{P}_k$ soit de degré égal à k). Enfin, dans le Gl-Lanczos/Orthodir $\mathcal{P}_{k+1}$ est calculé à partir de $\mathcal{P}_k$ et $\mathcal{P}_k^{(1)}$ et le polynôme $\mathcal{P}_{k+1}^{(1)}$ est obtenu à partir de $\mathcal{P}_k^{(1)}$ et $\mathcal{P}_{k-1}^{(1)}$. Nous savons que le calcul de ces polynômes, par des relations de récurrence, fait appel à des divisions par des coefficients qui peuvent être nuls. Cela provoque des situations de *true* et *ghost-breakdown* dans les algorithmes de type Lanczos globales. Ces problèmes peuvent être évités si nous utilisons les généralisations des relations de récurrence obtenues en appliquant le *look-ahead*.

Dans la suite de cette section, nous donnerons deux variantes de la méthode de Lanczos globale avec *look-ahead*. Mais avant cela, nous commençons par donner quelques notations. Notons n_k le degré du polynôme régulier $\mathcal{P}_k^{(1)}$ et m_k la longueur du saut entre deux polynômes réguliers successifs des familles $\{\mathcal{P}_k\}$ et $\{\mathcal{P}_k^{(1)}\}$. D'après les relations données dans le premier chapitre, m_k est défini par

$$\begin{aligned} c^{(1)}(\xi^i \mathcal{P}_k^{(1)}) &= 0 \quad \text{pour} \quad i = 0, \ldots, n_k + m_k - 2, \\ c^{(1)}(\xi^{n_k+m_k-1} \mathcal{P}_k^{(1)}) &\neq 0. \end{aligned} \tag{4.19}$$

4.4.1 Algorithme HMRZ-stab global

Les polynômes $\mathcal{P}_{k+1}$ et $\mathcal{P}_{k+1}^{(1)}$ peuvent être calculés à partir des relations

$$\begin{aligned} \mathcal{P}_{k+1}(\xi) &= \mathcal{P}_k(\xi) - \xi w_k(\xi) \mathcal{P}_k^{(1)}(\xi) \\ \mathcal{P}_{k+1}^{(1)}(\xi) &= q_k(\xi) \mathcal{P}_k^{(1)}(\xi) - C_{k+1} \mathcal{P}_{k-1}^{(1)}(\xi) \end{aligned}$$

avec $\mathcal{P}_{-1}^{(1)}(\xi) = 1$, $C_1 = 0$ et $\mathcal{P}_0^{(1)}(\xi) = 1$.

q_k est un polynôme unitaire de degré m_k et w_k est un polynôme de degré au plus $m_k - 1$.

En tenant compte du fait que $\mathcal{P}_k^{(1)}$ est exactement de degré n_k, les conditions d'orthogonalité (4.19) s'écrivent

$$\begin{aligned} c(\xi^i \mathcal{P}_k^{(1)} \mathcal{P}_{k+1}) &= 0 \quad \text{pour } i = 0, \ldots, m_k - 1 \\ c^{(1)}(\xi^i \mathcal{P}_k^{(1)} \mathcal{P}_{k+1}^{(1)}) &= 0 \quad \text{pour } i = 0, \ldots, m_k - 1 \end{aligned}$$

et

$$\begin{aligned} c^{(1)}(\xi^i {\mathcal{P}_k^{(1)}}^2) &= 0 \quad \text{pour } i = 0, \ldots, m_k - 2 \\ c^{(1)}(\xi^{m_k-1} {\mathcal{P}_k^{(1)}}^2) &\neq 0. \end{aligned}$$

Les coefficients de q_k et w_k ainsi que C_{k+1} peuvent être calculés en imposant ces conditions d'orthogonalité aux polynômes $\mathcal{P}_{k+1}$ et $\mathcal{P}_{k+1}^{(1)}$. Les expressions de ces coefficients ont déjà été données dans la section 2.3.4.

En posant

$$R_k = \mathcal{P}_k(A)R_0 \qquad \text{et} \qquad Z_k = \mathcal{P}_k^{(1)}(A)R_0,$$

nous obtenons les relations suivantes

$$\begin{aligned} R_{k+1} &= R_k - A w_k(A) Z_k, \\ X_{k+1} &= X_k - w_k(A) Z_k, \\ Z_{k+1} &= q_k(A) Z_k - C_{k+1} Z_{k-1}, \end{aligned}$$

avec les initialisation $Z_0 = R_0$, $Z_{-1} = 0$ et $C_1 = 0$.
Si nous prenons

$$\tilde{Z}_k = \mathcal{P}_k^{(1)}(A^T)\tilde{R}_0,$$

alors, nous avons la récurrence

$$\tilde{Z}_{k+1} = q_k(A^T)\tilde{Z}_k - C_{k+1}\tilde{Z}_{k-1},$$

avec $\tilde{Z}_0 = \tilde{R}_0$ et $\tilde{Z}_{-1} = 0$. Le saut m_k est donc déterminé par

$$\begin{aligned} &< \tilde{Z}_k, A^{i+1} Z_k >_F \;=\; 0 \quad \text{pour } i = 0, \ldots, m_k - 2 \\ \text{et} \quad &< \tilde{Z}_k, A^{m_k} Z_k >_F \;\neq\; 0. \end{aligned}$$

Prenons $w_k(t) = \sum_{i=0}^{m_k-1} \beta_i^{(k)} t^i$ et $q_k(t) = \sum_{i=0}^{m_k} \alpha_i^{(k)} t^i$ avec $\alpha_{m_k}^{(k)} = 1$.
En notant

$$\begin{aligned} d_i^{(k)} &= < \tilde{Z}_k, A^i R_k >_F \quad \text{pour } i = 0, \ldots, m_k - 1 \\ b_i^{(k)} &= < \tilde{Z}_k, A^{m_k+i} Z_k >_F \quad \text{pour } i = 0, \ldots, m_k, \end{aligned}$$

les coefficients $(\alpha_i^{(k)})_{0\leq i\leq m_k-1}$ et $(\beta_i^{(k)})_{0\leq i\leq m_k-1}$ sont les solutions des deux systèmes linéaires suivants

$$\begin{pmatrix} b_0^{(k)} & & & & & \\ b_1^{(k)} & b_0^{(k)} & & & 0 & \\ \vdots & b_1^{(k)} & \ddots & & & \\ \vdots & \vdots & \ddots & \ddots & & \\ b_{m_k-2}^{(k)} & b_{m_k-1}^{(k)} & \cdots & \ddots & b_0^{(k)} & \\ b_{m_k-1}^{(k)} & b_{m_k-2}^{(k)} & \cdots & \cdots & b_1^{(k)} & b_0^{(k)} \end{pmatrix} \begin{pmatrix} \beta_{m_k-1}^{(k)} & \alpha_{m_k-1}^{(k)} \\ \beta_{m_k-2}^{(k)} & \alpha_{m_k-2}^{(k)} \\ \vdots & \vdots \\ \vdots & \vdots \\ \beta_1^{(k)} & \alpha_1^{(k)} \\ \beta_0^{(k)} & \alpha_0^{(k)} \end{pmatrix} = \begin{pmatrix} d_0^{(k)} & -b_1^{(k)} \\ d_1^{(k)} & -b_2^{(k)} \\ \vdots & \vdots \\ \vdots & \vdots \\ d_{m_k-2}^{(k)} & -b_{m_k-1}^{(k)} \\ d_{m_k-1}^{(k)} & -b_{m_k}^{(k)} \end{pmatrix}.$$

Ce système est non singulier puisque m_k est déterminé par la condition

$$b_0^{(k)} =< \tilde{Z}_k, A^{m_k} Z_k >_F \neq 0.$$

Nous avons aussi $C_{k+1} = b_0^{(k)}/b_0^{(k-1)}$.
En rassemblant toutes les formules précédentes, nous obtenons l'algorithme MRZ-stab global.

Algorithme 4.4.1 : *MRZ-stab global.*

1. Initialisation
   $Z_{-1} = 0, \quad \tilde{Z}_{-1} = 0, \quad R_0 = B - AX_0;$
   $Z_0 = R_0, \quad \tilde{Z}_0 = \tilde{R}_0, \quad C_1 = 0;$
   $n_0 = 0, \quad m_{-1} = 0, \quad k = 0;$

2. Tant que $R_k \neq 0$
   $Z_{k,0} = Z_k, \quad \tilde{Z}_{k,0} = \tilde{Z}_k$
   $d_0^{(k)} =< \tilde{Z}_{k,0}, R_k >_F;$
   $Z_{k,1} = AZ_{k,0}, \quad m_k = 1;$
   $b_0^{(k)} =< \tilde{Z}_{k,0}, Z_{k,1} >_F;$

3. Tant que $b_0^{(k)} = 0$
   $m_k = m_k + 1;$
   $Z_{k,m_k} = AZ_{k,m_k-1};$
   $b_0^{(k)} =< \tilde{Z}_{k,0}, Z_{k,m_k} >_F;$
   Fin(tant que)

4. $\beta_{m_k-1}^{(k)} = d_0^{(k)}/b_0^{(k)}$;
Si $k \neq 0$
$C_{k+1} = b_0^{(k)}/b_0^{(k-1)}$;
Fin(si)

5. Pour $i = 1, \ldots, m_k$
$\tilde{Z}_{k,i} = A^T \tilde{Z}_{k,i-1}$;
$b_i^{(k)} =< \tilde{Z}_{k,i}, Z_{k,m_k} >_F$;
Si $i \neq m_k$
$d_i^{(k)} =< \tilde{Z}_{k,i}, R_k >_F$;
Calculer $\beta_{m_k-i-1}^{(k)}$;
Fin (si)
Calculer $\alpha_{m_k-i}^{(k)}$;
Fin(pour)

6. $X_{k+1} = X_k + \beta_0^{(k)} Z_{k,0} + \beta_1^{(k)} Z_{k,1} + \cdots + \beta_{m_k-1}^{(k)} Z_{k,m_k-1}$;
$R_{k+1} = R_k - [\beta_0^{(k)} Z_{k,1} + \beta_1 Z_{k,2} + \cdots + \beta_{m_k-1}^{(k)} Z_{k,m_k}]$;
$Z_{k+1} = \alpha_0^{(k)} Z_{k,0} + \alpha_1^{(k)} Z_{k,1} + \cdots + \alpha_{m_k-1}^{(k)} Z_{k,m_k-1} + Z_{k,m_k} - C_{k+1} Z_{k-1}$;
$\tilde{Z}_{k+1} = \alpha_0^{(k)} \tilde{Z}_{k,0} + \alpha_1^{(k)} \tilde{Z}_{k,1} + \cdots + \alpha_{m_k-1}^{(k)} \tilde{Z}_{k,m_k-1} + \tilde{Z}_{k,m_k} - C_{k+1} \tilde{Z}_{k-1}$;

7. $n_{k+1} = n_k + m_k$;
$k = k + 1$;
Fin

La méthode MRZ-stab globale nécessite le stockage d'un nombre important de matrices de dimension $N \times s$. Pour éviter ce problème, nous allons utiliser le schéma de Horner pour calculer $w_k(A)R_k$ et $q_k(A)Z_k$.
Commençons par prendre le polynôme unitaire donné par

$$h_k(\xi) = \gamma_0^{(k)} + \cdots + \gamma_{m_k-1}^{(k)} \xi^{m_k-1} + \xi^{m_k}.$$

Le schéma de Horner nous permet de calculer le polynôme h_k comme suit

$$\begin{aligned} h_k^{(0)}(\xi) &= 1 \\ h_k^{(i)}(\xi) &= \xi h_k^{(i-1)}(\xi) + \gamma_{m_k-i}^{(k)} \quad \text{pour} \quad i = 1, \ldots, m_k \\ h_k(\xi) &= h_k^{(m_k)}(\xi). \end{aligned}$$

Si nous considérons que les coefficients $\gamma_i^{(k)}$ sont solution du système linéaire

$$\begin{pmatrix} b_0^{(k)} & & & & & \\ b_1^{(k)} & b_0^{(k)} & & & 0 & \\ \vdots & b_1^{(k)} & \ddots & & & \\ \vdots & \vdots & \ddots & \ddots & & \\ b_{m_k-2}^{(k)} & b_{m_k-1}^{(k)} & \cdots & \ddots & b_0^{(k)} & \\ b_{m_k-1}^{(k)} & b_{m_k-2}^{(k)} & \cdots & \cdots & b_1^{(k)} & b_0^{(k)} \end{pmatrix} \begin{pmatrix} \gamma_{m_k-1}^{(k)} \\ \gamma_{m_k-2}^{(k)} \\ \vdots \\ \vdots \\ \gamma_1^{(k)} \\ \gamma_0^{(k)} \end{pmatrix} = - \begin{pmatrix} b_1^{(k)} \\ b_2^{(k)} \\ \vdots \\ \vdots \\ b_{m_k-1}^{(k)} \\ b_{m_k}^{(k)} \end{pmatrix},$$

alors, en tenant compte des expressions des coefficients $\beta_i^{(k)}$ et $\alpha_i^{(k)}$, nous obtenons

$$\begin{aligned} w_k(\xi) &= \frac{1}{b_0^{(k)}} \sum_{j=0}^{m_k-1} d_{m_k-j-1}^{(k)} h_k^{(j)}(\xi), \\ q_k(\xi) &= h_k(\xi) = h_k^{(m_k)}(\xi). \end{aligned}$$

En utilisant cette nouvelle approche, nous n'aurons plus à stocker tout les $b_i^{(k)}$, mais seulement $b_0^{(k)}$, et les coefficients des polynômes w_k et q_k ne seront pas stockés non plus. En plus, le nombre des matrices de dimension $\mathbb{R}^{N\times s}$ à stocker est indépendant de m_k, il sera toujours égal à 12. L'algorithme obtenu est appelé HMRZ-stab global.

Algorithme 4.4.2 : *HMRZ-stab global.*

1. Initialisation
   $Z_{-1} = 0, \quad \tilde{Z}_{-1} = 0, \quad R_0 = B - AX_0$;
   $\tilde{Z}_0 = \tilde{R}_0, \quad Z_0 = R_0, \quad C_1 = 0$;
   $n_0 = 0, \quad b_0^{(-1)} = 0, \quad k = 0$;

2. Tant que $R_k \neq 0$
   $d_0^{(k)} =< \tilde{Z}_k, R_k >_F$;
   $m_k = 1$;
   $Y_k = A^T \tilde{Z}_k$;
   $V_k = Y_k$;
   $b_0^{(k)} =< Y_k, Z_k >_F$;

3. Tant que $b_0^{(k)} = 0$
   $m_k = m_k + 1$;
   $d_{m_k-1}^{(k)} =< Y_k, R_k >_F$;
   $Y_k = A^T Y_k$;
   $b_0^{(k)} =< Y_k, Z_k >_F$;
   Fin(tant que)

4. Si $k \neq 0$
   $C_{k+1} = b_0^{(k)} / b_0^{(k-1)}$;
   Fin(si)
   $T_k = Z_k$;
   $Z_{k+1} = -C_{k+1} Z_{k-1}$;
   $\tilde{T}_k = \tilde{Z}_k$;
   $\tilde{Z}_{k+1} = -C_{k+1} \tilde{Z}_{k-1}$;
   $X_{k+1} = X_k$;
   $R_{k+1} = R_k$;

5. Pour $i = 1, \ldots, m_k$
   $U_k = AT_k$;
   $\beta = d_{m_k-i}^{(k)} / b_0^{(k)}$;
   $X_{k+1} = X_{k+1} + \beta T_k$;
   $R_{k+1} = R_{k+1} - \beta U_k$;
   $\gamma = - < Y_k, U_k >_F / b_0^{(k)}$;
   $T_k = U_k + \gamma Z_k$;
   Si $i \neq 1$
   $V_k = A^T \tilde{T}_k$;
   Fin(si)
   $\tilde{T}_k = V_k + \gamma \tilde{Z}_k$;
   Fin(pour)

6. $Z_{k+1} = Z_{k+1} + T_k$;
   $\tilde{Z}_{k+1} = \tilde{Z}_{k+1} + \tilde{T}_k$;

7. $n_{k+1} = n_k + m_k$;
   $k = k + 1$;

Fin

4.4.2 L'algorithme du BiCGSTAB global avec look-ahead

Dans l'algorithme du BiCGSTAB global, le résidu R_k peut être défini comme suit

$$R_k = \mathcal{Q}_k(A)\mathcal{P}_k(A)R_0$$

où $\mathcal{Q}_k$ est un polynôme scalaire vérifiant la récurrence suivante

$$\mathcal{Q}_k(\xi) = (1 - w_{k-1}\xi)\mathcal{Q}_{k-1}(\xi) \quad \text{et} \quad \mathcal{Q}_0(\xi) = 1,$$

et w_{k-1} choisi tel que $< R_k, R_k >_F = \|R_k\|_F^2$ soit minimale.
Notons

$$\tilde{\mathcal{P}}_k^{(1)}(\xi) = (-1)^k \frac{H_k^{(0)}}{H_k^{(1)}} \mathcal{P}_k^{(1)}(\xi).$$

$\tilde{\mathcal{P}}_k^{(1)}$ appartient à la famille des polynômes orthogonaux formels par rapport à la fonctionnelle $c^{(1)}$. $\mathcal{P}_k$ et $\tilde{\mathcal{P}}_k^{(1)}$ ont le même coefficient de plus haut degré, et ils vérifient les relations de récurrence suivantes

$$\begin{aligned} \mathcal{P}_{k+1}(\xi) &= \mathcal{P}_k(\xi) - \alpha_{k+1}\xi\tilde{\mathcal{P}}_k^{(1)}(\xi) \\ \tilde{\mathcal{P}}_{k+1}^{(1)}(\xi) &= \mathcal{P}_{k+1}(\xi) + \beta_{k+1}\tilde{\mathcal{P}}_k^{(1)}(\xi) \end{aligned} \tag{4.20}$$

avec $\mathcal{P}_0(\xi) = \tilde{\mathcal{P}}_0(\xi) = 1$.
Ces relations de récurrence ont déjà été utilisées dans la section 2.2.1. Nous rappelons que si le déterminant de Hankel $H_k^{(1)} = 0$, alors, les polynômes $\mathcal{P}_k$ et $\tilde{\mathcal{P}}_k^{(1)}$ n'existent pas, et il se produit un *true-breakdown* dans les récurrences (4.20). Dans le cas où $H_k^{(1)} \neq 0$, l'algorithme peut être sujet au *ghost-breakdown*, ce dernier est dû au fait que les polynômes $\mathcal{P}_k$ et $\tilde{\mathcal{P}}_k^{(1)}$ ne sont pas de degrés exactement égal à k, ce qui ce traduit aussi par $H_k^{(0)} = 0$. Et dans ce cas nous ne pouvons utiliser les relations (4.20).
Dans la section 2.2, nous avons donné une généralisation des relations (4.20) dans laquelle nous évitons le *true* et le *ghost-breakdown*. Ici, nous allons traiter seulement le cas où $c(\mathcal{P}_k\mathcal{P}_k) \neq 0$, $\forall k$.
Dans ce cas, les polynômes $\mathcal{P}_{k+1}$ et $\tilde{\mathcal{P}}_{k+1}^{(1)}$ peuvent être calculés par les relations

$$\begin{aligned} \mathcal{P}_{k+1}(\xi) &= \mathcal{P}_k(\xi) - \xi\eta_k(\xi)\tilde{\mathcal{P}}_k^{(1)}(\xi) \\ \tilde{\mathcal{P}}_{k+1}^{(1)}(\xi) &= \mathcal{P}_{k+1}(\xi) + \beta_{k+1}\tilde{\mathcal{P}}_k^{(1)}(\xi) \end{aligned} \tag{4.21}$$

avec $\mathcal{P}_0(\xi) = \tilde{\mathcal{P}}_0(\xi) = 1$ et η_k est un polynôme de degré au plus égal à $m_k - 1$.
Les relations (4.19) nous permettent de déterminer m_k, et s'écrivent aussi sous la forme

$$\begin{aligned} c^{(1)}(\xi^i\tilde{\mathcal{P}}_k^{(1)}) &= 0 \quad \text{pour} \quad i = 0, \ldots, n_k + m_k - 2, \\ \text{et} \qquad c^{(1)}(\xi^{n_k+m_k-1}\tilde{\mathcal{P}}_k^{(1)}) &\neq 0. \end{aligned} \tag{4.22}$$

Considérons maintenant le polynôme auxiliaire $\mathcal{Q}_k$ de degré au plus n_k et qui vérifie la relation de récurrence

$$\mathcal{Q}_{k+1}(\xi) = (1 + w_1^{(k)}\xi + w_2^{(k)}\xi^2 + \cdots + w_{m_k}^{(k)}\xi^{m_k})\mathcal{Q}_k(\xi). \tag{4.23}$$

Les coefficients $(w_i{}^{(k)})_{1\leq i\leq m_k}$ sont choisis tels que $< R_k, R_k >_F$ soit minimal.
Les relations (4.22) s'écrivent aussi

$$\begin{aligned} & c^{(1)}(\xi^i \mathcal{Q}_k \tilde{\mathcal{P}}_k^{(1)}) = 0 \quad \text{pour } i = 0, \ldots, m_k - 2, \\ \text{et} \quad & c^{(1)}(\xi^{m_k-1} \mathcal{Q}_k \tilde{\mathcal{P}}_k^{(1)}) \neq 0. \end{aligned} \tag{4.24}$$

Posons

$$\eta_k(\xi) = \sum_{i=0}^{m_k-1} \gamma_i^{(k)} \xi^i$$

Pour calculer les coefficients $(\gamma_i^{(k)})_{0\leq i\leq m_k-1}$, nous multiplions dans (4.21) $\mathcal{P}_{k+1}$ par $\xi^i\mathcal{Q}_k$ pour $i = 0, \ldots, m_k - 1$, et après avoir appliqué c nous imposons les conditions d'orthogonalité (4.24), ce qui nous donne

$$\begin{pmatrix} b_0^{(k)} & & & & & \\ b_1^{(k)} & b_0^{(k)} & & & 0 & \\ \vdots & b_1^{(k)} & \ddots & & & \\ \vdots & \vdots & \ddots & \ddots & & \\ b_{m_k-2}^{(k)} & b_{m_k-1}^{(k)} & \cdots & \ddots & b_0^{(k)} & \\ b_{m_k-1}^{(k)} & b_{m_k-2}^{(k)} & \cdots & \cdots & b_1^{(k)} & b_0^{(k)} \end{pmatrix} \begin{pmatrix} \gamma_{m_k-1}^{(k)} \\ \gamma_{m_k-2}^{(k)} \\ \vdots \\ \vdots \\ \gamma_1^{(k)} \\ \gamma_0^{(k)} \end{pmatrix} = \begin{pmatrix} d_0^{(k)} \\ d_1^{(k)} \\ \vdots \\ \vdots \\ d_{m_k-2}^{(k)} \\ d_{m_k-1}^{(k)} \end{pmatrix}, \tag{4.25}$$

avec

$$b_i^{(k)} = c^{(1)}(\xi^{m_k-1+i}\mathcal{Q}_k\tilde{\mathcal{P}}_k^{(1)}) \qquad \text{pour } i = 0, \ldots, m_k - 1$$

$$d_i^{(k)} = c(\xi^i \mathcal{Q}_k \mathcal{P}_k) \qquad \text{pour } i = 0, \ldots, m_k - 1.$$

Puisque m_k est déterminé sous la condition $c^{(1)}(\xi^{m_k-1}\mathcal{Q}_k\tilde{\mathcal{P}}_k^{(1)}) = b_0^{(k)} \neq 0$, alors la matrice du système (4.25) est non-singulière.
Pour calculer β_{k+1}, nous multiplions dans (4.21) la récurrence de $\tilde{\mathcal{P}}_{k+1}^{(1)}$ par $\xi^{m_k-1}\mathcal{Q}_k$ et nous appliquons $c^{(1)}$. Alors, en utilisant les conditions d'orthogonalité (4.22), nous obtenons

$$\beta_{k+1} = -\frac{c(\xi^{m_k}\mathcal{Q}_k\mathcal{P}_{k+1})}{c^{(1)}(\xi^{m_k-1}\mathcal{Q}_k\tilde{\mathcal{P}}_k^{(1)})}.$$

En notant $p_0^{(k)} = c(\xi^{m_k}\mathcal{Q}_k\mathcal{P}_{k+1})$, nous pouvons écrire

$$\beta_{k+1} = -\frac{p_0^{(k)}}{b_0^{(k)}}.$$

Si nous posons

$$Z_k = \mathcal{Q}_k(A)\tilde{\mathcal{P}}_k^{(1)}(A)R_0 \qquad \text{et} \qquad S_k = \mathcal{Q}_{k-1}(A)\mathcal{P}_k(A)R_0$$

alors les relations (4.21) et (4.23) nous donnent

$$\begin{aligned} S_{k+1} &= R_k - A\eta_k(A)Z_k \\ R_{k+1} &= S_{k+1} + w_1^{(k)}AS_{k+1} + w_2^{(k)}A^2S_{k+1} + \ldots + w_{m_k}^{(k)}A^{m_k}S_{k+1} \\ Z_{k+1} &= R_{k+1} + \beta_{k+1}(Z_k + w_1^{(k)}AZ_k + w_2^{(k)}A^2Z_k + \ldots + w_{m_k}^{(k)}A^{m_k}Z_k). \end{aligned}$$

Les approximations X_k sont calculées par la relation

$$X_{k+1} = X_k + \eta_k(A)Z_k - (w_1^{(k)}S_{k+1} + w_2^{(k)}AS_{k+1} + \ldots + w_{m_k}^{(k)}A^{m_k-1}S_{k+1}).$$

Les coefficients $(w_i^{(k)})_{1\leq i\leq m_k}$ minimisent $< R_{k+1}, R_{k+1} >_F$ et nous pouvons les calculer en résolvant le système linéaire

$$w_1^{(k)} < A^iS_{k+1}, AS_{k+1} >_F + \cdots + w_{m_k}^{(k)} < A^iS_{k+1}, A^{m_k}S_{k+1} >_F = - < A^iS_{k+1}, S_{k+1} >_F,$$

pour $i = 1, \ldots, m_k$.
Puisque $c(\xi^i) = < \tilde{R}_0, A^iR_0 >_F$ alors les coefficient $(b_i^{(k)})$, $(d_i^{(k)})$ et $(p_0^{(k)})$ sont donnés par

$$\begin{aligned} b_i^{(k)} &= < \tilde{R}_0, A^{i+m_k}R_k >_F \qquad \text{pour } i = 0, \ldots, m_k - 1, \\ d_i^{(k)} &= < \tilde{R}_0, A^iR_k >_F \qquad \text{pour } i = 0, \ldots, m_k - 1. \\ p_0^{(k)} &= < \tilde{R}_0, A^{m_k}S_{k+1} >_F . \end{aligned}$$

Le saut m_k est déterminé par les conditions

$$< \tilde{R}_0, A^iZ_k >_F = 0 \quad \text{pour } i = 1, \ldots, m_k - 1, \quad \text{et} \quad < \tilde{R}_0, A^{m_k}Z_k >_F \neq 0.$$

En rassemblant toutes les relations précédentes, nous obtenons l'algorithme suivant

Algorithme 4.4.3 *:Gl-BiCGSTAB avec look-ahead.*

1. Initialisation
   $Z_{-1} = 0, \quad R_0 = B - AX_0, \quad Z_0 = R_0$;
   $n_0 = 0, \quad k = 0$;

2. Tant que $R_k \neq 0$
   $d_0^{(k)} =< \tilde{R}_0, R_k >_F$
   Si $d_0^{(0)} = 0$ On arrête l'algorithme
   Fin(si).
   $Z_{k,0} = Z_k, \quad Z_{k,1} = AZ_{k,0};$
   $m_k = 1$
   $b_0^{(k)} =< \tilde{R}_0, Z_{k,1} >_F$

3. Tant que $b_0^{(k)} = 0$
   $m_k = m_k + 1$
   $Z_{k,m_k} = AZ_{k,m_{k-1}}$
   $b_0^{(k)} =< \tilde{R}_0, Z_{k,m_k} >_F$
   Fin(tant que)

4. $\gamma_{m_k-1}^{(k)} = d_0^{(k)}/b_0^{(k)}$;
   $U_k = Z_{k,m_k}$;
   $Y_k = R_k$;
   Pour $i = 1, \ldots, m_k - 1$
   $U_k = AU_k$
   $b_i^{(k)} =< \tilde{R}_0, U_k >_F$
   $Y_k = AR_k$
   $d_i^{(k)} =< \tilde{R}_0, Y_k >_F$
   Calculer $\gamma_{m_k-i-1}^{(k)}$ à partir du système (4.25)
   Fin(pour)

5. $S_{k+1} = R_k - \gamma_0^{(k)} Z_{k,1} - \gamma_1^{(k)} Z_{k,2} - \cdots - \gamma_{m_k-1}^{(k)} Z_{k,m_k}$

6. $M_k = [AS_{k+1}, \ldots, A^{m_k} S_{k+1}]^T [AS_{k+1}, \ldots, A^{m_k} S_{k+1}]$
   $N_k = [AS_{k+1}, \ldots, A^{m_k} S_{k+1}]^T S_{k+1}$

7. Résoudre $M_k w_k = -N_k$; avec $[w_1^{(k)}, w_2^{(k)}, \ldots, w_{m_k}^{(k)}]^T = w_k$;

8. $X_{k+1} = X_k + \gamma_0^{(k)} Z_{k,0} + \cdots + \gamma_{m_k-1}^{(k)} Z_{k,m_k-1} - w_1^{(k)} S_{k+1} - \cdots - w_{m_k}^{(k)} A^{m_k-1} S_{k+1}$;
   $R_{k+1} = S_{k+1} + w_1^{(k)} AS_{k+1} + \cdots + w_{m_k}^{(k)} A^{m_k} S_{k+1}$;
   $p_0^{(k)} =< \tilde{R}_0, A^{m_k} S_{k+1} >_F$
   $\beta_{k+1} = -\dfrac{p_0^{(k)}}{b_0^{(k)}}$

$$Z_{k+1} = R_{k+1} + \beta_{k+1}(Z_{k,0} + w_1^{(k)} Z_{k,1} + \cdots + w_{m_k}^{(k)} Z_{k,m_k})$$

9. $n_{k+1} = n_k + m_k$
   $k = k + 1$

Fin

Nous pouvons éviter le stockage des vecteurs $Z_{k,i}$ en utilisant le schéma de Horner comme cela a été fait pour l'algorithme 4.4.2.

4.5 Résultats numériques

Dans cette section, nous allons donner quelques résultats numériques. Nos tests ont été réalisés à l'aide du logiciel MATLAB (version 7). Les tests sont arrêtés lorsque

$$\max_{j=1,\ldots,s} (\| R_k^{(j)} \|_2 \, / \, \| R_0^{(j)} \|_2) \leq tol,$$

où tol est la tolérance choisie.
Pour détecter le *breakdown* dans les méthodes avec *look-ahead* étudiées, nous allons considérer qu'une quantité est nulle si sa valeur absolue est inférieure à une tolérance donnée ϵ. Les valeurs de tol et ϵ seront proposées dans les exemples.

Exemple 4.5.1

Dans cet exemple nous comparons les performances numériques des méthodes BICG global, BiCGSTAB global et le Bl-BICG. Nous prenons les matrice A_4=PDE2961 et A_5=SHERMAN4 de la collection **Harwell-Boeing**. Ces matrices contiennent $nnz(A_4) = 14585$ et $nnz(A_5) = 3786$ éléments non nuls. L'approximation initiale X_0 est choisie égale à zéro et $B = rand(N, s)$.
Dans le tableau 4.5.1, nous donnons le temps CPU (en secondes) des trois algorithmes. Nous mettons entre parenthèses la quantité $s * t(1)/t(s)$ où $t(s)$ est le temps CPU obtenu par la méthode, globale ou par bloc, étudiée et $t(1)$ est le temps CPU obtenu par cette méthode en résolvant le système avec un seul second membre. Remarquons que le temps obtenu avec un seul second membre dépend du second membre utilisé. Ici, $t(1)$ est obtenu en divisant le temps total nécessaire pour les s systèmes résolus séparément par s.
Notons aussi qu'une méthode globale est efficace si $(s * t(1)/t(s)) > 1$.
Nous prenons le nombre maximal d'itérations égal à 500 pour toutes les méthodes. Comme cela a été cité dans [71], nous prenons $\tilde{R}_0 = AR_0$ dans l'algorithme Bl-BICG ; cela permet

d'améliorer ses résultats.
Le Tableau 4.5.1 montre que la méthode Gl-BiCGSTAB donne les meilleurs résultats. Nous remarquons que le Bl-BICG est meilleur que le Gl-BICG dans le cas de la matrice Sherman4 avec $s = 20$. Pour la matrice PDE2961, la méthode Bl-BICG ne converge pas, c'est le cas aussi pour la matrice Sherman4 avec $s = 10$.

Tableau 4.5.1
Durée de convergence pour Gl-BICG, Gl-BiCGSTAB et Bl-BICG.
Matrice A_4 et A_5 ; $s = 10$ et $s = 20$.

Matrices	s	Gl-BICG	Gl-BiCGSTAB	Bl-BICG
PDE2961	10	98	40	-
(N=2961)		(1.39)	(1.43)	-
	20	201	81	-
		(1.43)	(1.45)	
SHERMAN4	10	17	10	-
(N=1140)		(1.37)	(1.34)	-
	20	36	19	24
		(1.41)	(1.39)	(1.35)

Entre parenthèses, nous avons la quantité $s * t(1)/t(s)$ pour chaque méthode.

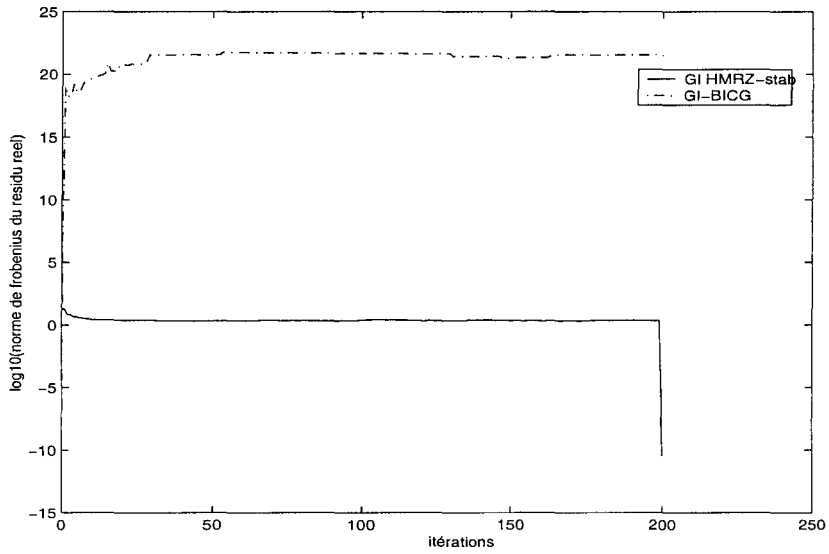
Exemple 4.5.2

Considérons la matrice suivante

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

avec $B = I_{N,s}$, $X_0 = rand(N, s)$, $\tilde{R}_0 = R_0$, $N = 200$, $s = 6$ et $a = 0$.
Dans cet exemple, la méthode du BICG global ne converge pas. En prenant $\epsilon = 10^{-8}$ pour détecter le *breakdown*, le HMRZ-stab fait des sauts de longueur $m_k = 2$ depuis la première itération, et nous obtenons $n_{100} = 200$ avec $\|R_{n_{100}}\|_F = 3.14\ 10^{-11}$.
Les résultats obtenus sont donnés dans la figure 4.1.

FIG. 4.1 - $N = 200$, $s = 6$, $\epsilon = 10^{-8}$



Exemple 4.5.3

Nous reprenons la matrice A utilisée dans l'exemple 4.5.2, avec $N = 1000$, $B = rand(N, s)$, $s = 6$, $a = 1.1$, $X_0 = 0_{N,s}$ et $\tilde{R}_0 = R_0$.

La figure 4.2 représente les résultats obtenus par le Gl-BiCGSTAB et le Gl-BiCGSTAB avec look-ahead. Ce dernier effectue trois sauts pour $\epsilon = 10^{-10}$, le premier de $n_{16} = 16$ à $n_{17} = 18$, le deuxième de $n_{23} = 24$ à $n_{24} = 26$, et le dernier de longueur $m_k = 11$ de $n_{36} = 51$ à $n_{37} = 62$.

Exemple 4.5.4 *A= PDE2961*

C'est une matrice réelle non symétrique de dimension 2961×2961 de la collection **NEP** générée par **MATPDE**. Elle contient 14585 éléments non nuls, son conditionnement est estimé à $\kappa(A) \simeq 9.49\ 10^2$.
Si nous prenons $B = I_{N,s}, X_0 = 0$ et $\tilde{R}_0 = R_0$, avec $N = 2961$ et $s = 10$, alors pour $\epsilon = 10^{-10}$ et $tol = 10^{-7}$ nous obtenons les résultats donnés dans la figure 4.3.

Exemple 4.5.5

FIG. 4.2 - $N = 1000$, $s = 6$ and $\epsilon = 10^{-10}$

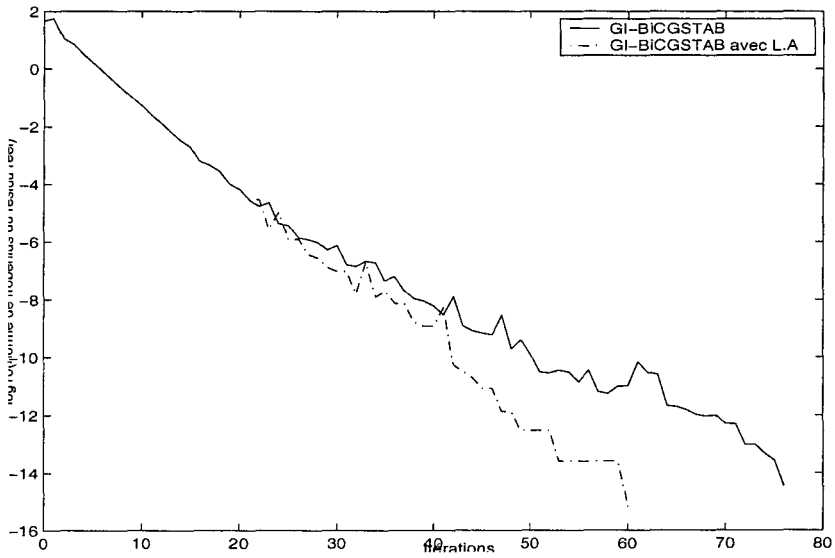


FIG. 4.3 - $N = 2961, s = 10, \epsilon = 10^{-10}$

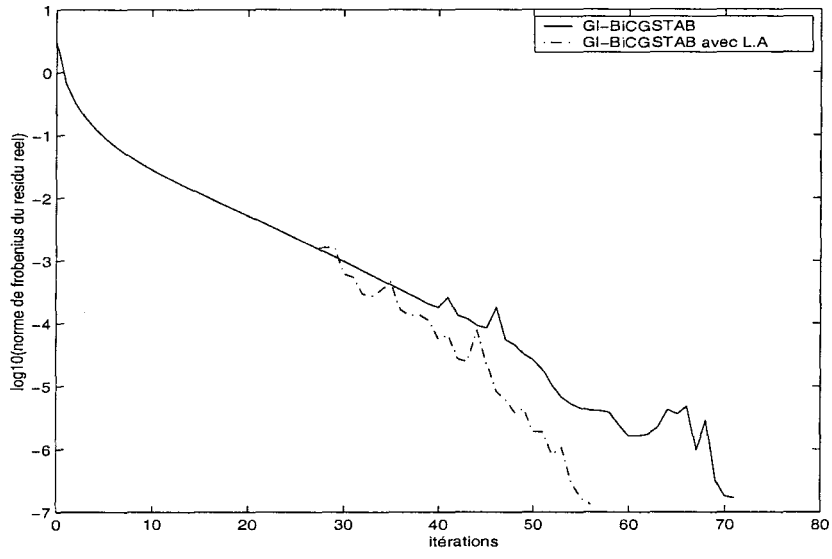
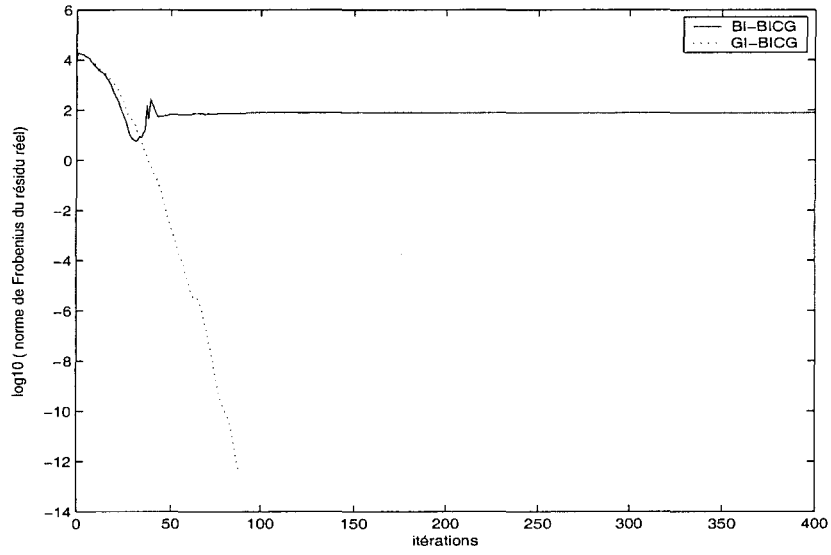


FIG. 4.4 – $N = 400, \quad s = 5$

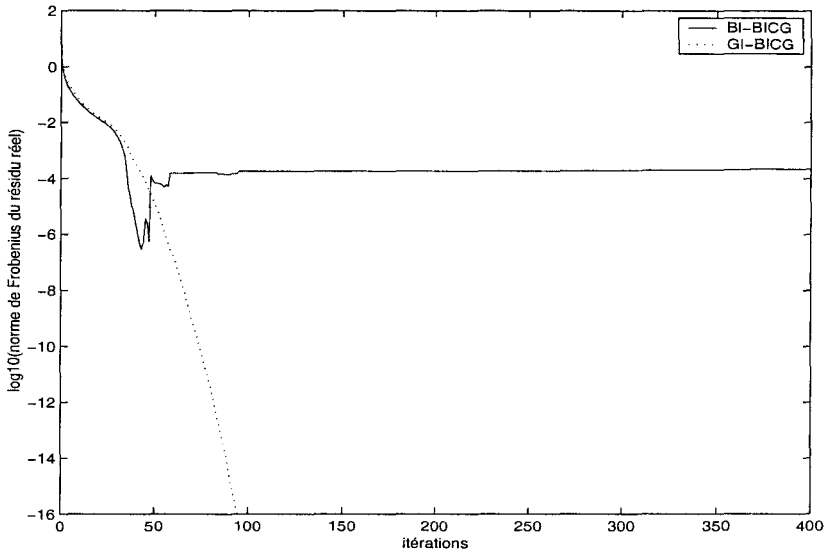


Nous proposons ici un exemple très simple qui illustre le problème de dépendance linéaire que subissent les méthodes par bloc.
Considérons la matrice obtenue par discrétisation de l'équation du Laplacien, en utilisant un schéma à cinq points sur une grille uniforme 20×20 avec un pas $h = 1/21$. Ceci nous donne une matrice creuse symétrique de dimension $N = 400$.
Prenons $s = 5$ avec $B = ones(N, s) + (10^{-3} \times rand(N, s))$, $X_0 = 0_{N,s}$ et $\tilde{R}_0 = R_0$.
Les résultats obtenus sont donnés dans la figure 4.4. Dans cet exemple, l'algorithme Gl-BICG converge alors que, dans l'algorithme du Bl-BICG, nous inversons des matrices qui sont presque singulières ce qui empêche la convergence de la méthode.
Prenons maintenant $B = I_{N,5}$, il se produit le même phénomène dans l'algorithme Bl-BICG alors que le Gl-BICG converge sans problème. Les résultats obtenus sont donnés dans la figure 4.5.

4.6 Conclusion

Dans ce chapitre nous avons défini la méthode de Lanczos globale pour la résolution des systèmes d'équations linéaires avec plusieurs second membres. Cette méthode consiste à projeter la matrice résidu sur un sous-espace de Krylov matriciel. Nous avons commencé par donner le processus de Lanczos global, ensuite nous avons introduit de nouvelles méthodes de type Lanczos globales, telles que le BICG global, le BiCGSTAB global et enfin nous

FIG. 4.5 – $N = 400, \quad s = 5$



avons développé des versions avec *look-ahead* de ces algorithmes. Les résultats numériques qui ont été étudiés dans le présent chapitre montrent que les méthodes globales peuvent donner de meilleurs résultats que les méthodes par bloc. Notons aussi que les méthodes par bloc nécessitent l'application de procédure de déflation pour éliminer les dépendances linéaires qui peuvent exister entre les colonnes des itérés, alors que les méthodes globales sont à l'abri de ce problème.

Bibliographie

[1] J. I Aliaga, D. L. Boley, R. Freund, V. Hernandez, *A Lanczos-type algorithm for multiple starting vectors*, revised vesrsion, Tech Numerical Analysis Manuscript, No, 98-3-05, Bell Laboratories, Murray Hill, New Jersey ; USA, (1998).

[2] O. Axelsson, *Conjugate gradient type methods for unsymmetric and inconsistent systems of linear equations*, Linear. Alg. Appl., 29 (1980) 1-16.

[3] E. H. Ayachour, *Avoiding look-ahead in the Lanczos method and Padé approximation*, Appl. Math., 26 (1999) 33-62.

[4] C. Baheux, *New implementations of Lanczos method*, J. Comput. Appl. Math, 57 (1995) 3-15.

[5] C. Brezinski, *Padé-Type approximation and general orthogonal polynomials.* ISNM, Vol 50, *Birkhäuser*, Basel, (1980).

[6] C. Brezinski, *CGM : a whole class of Lanczos type solvers for linear systems.* Pub. ANO-253, Université des Sciences et Technologies de Lille (1991).

[7] C. Brezinski, M. Redivo Zaglia, *A new presentation of orthogonal polynomials with applications to their computation*, Numer. Algorithms, 1 (1991) 207-222.

[8] C. Brezinski, M. Redivo-Zaglia, *Extrapolation Methods. Theory and Practice*, North Holland, Amsterdam, (1991).

[9] C. Brezinski, M. Redivo-Zaglia, *Breakdowns in the computation of orthogonal polynomials*, dans Nonlinear Numerical Methods and Rational Approximation, A. Cuyt ed., Kluwer, Dordrecht, (1994) 49-59.

[10] C. Brezinski, M. Redivo-Zaglia, *Look-ahead in BiCGSTAB and other methods for linear systems*, BIT, 35 (1995) 169-201.

[11] C. Brezinski, M. Redivo-Zaglia, *Transpose-free Lanczos-type algorithms for nonsymmetric linear systems*, Numer. Algorithms 17 (1998) 67-103.

[12] C. Brezinski, M. Redivo-Zaglia, *Variations on Lanczos's tridiagonalization process*, Calcolo 37 (2000) 159-179.

[13] C. Brezinski, M. Redivo Zaglia, H. Sadok, *Avoiding breakdown and near-breakdown in Lanczos-type algorithms*, Numer. Algorithms, 1 (1991) 261-284.

[14] C. Brezinski, M. Redivo-Zaglia, H. Sadok, *Addendum to "Avoiding breakdown and near-breakdown in Lanczos type algorithms"*, Numer. Algorithms, 2 (1992) 133-136.

[15] C. Brezinski, M. Redivo-Zaglia, H. Sadok, *A breakdown-free Lanczos-type algorithm for solving linear systems*, Numer. Math., 63 (1992) 29-38.

[16] C. Brezinski, M. Redivo-Zaglia, H. Sadok, *Breakdowns in the implementation of the Lanczos method for solving linear systems*, Comput. & Math. with Applics., 33 (1997) 31-44.

[17] C. Brezinski, M. Redivo-Zaglia, H. Sadok, *New look-ahead Lanczos-type algorithms for linear systems*, Numer. Math., 83 (1999) 53-83.

[18] C. Brezinski, M. Redivo-Zaglia, H. Sadok, *A review of formal orthogonality in Lanczos-based methods*, J. Comput. Appl. Math., 140 (2002) 81-98.

[19] C. Brezinski, H. Sadok, *Lanczos-type algorithms for solving systems of linear equations*, Appl. Numer. Math., 11 (1993) 443-473.

[20] P. N. Brown, *A theoritical comparaison of the Arnoldi and GMRES algorithms*, SIAM J. Sci. Stat, Comput. 12 (1991) 58-78.

[21] T. F. Chan, R. E. Bank, *A composite step bi-conjugate gradient algorithm for solving nonsymmetric systems*, Numer. Algorithms, 7 (1994) 1-16.

[22] T. F. Chan, R. E. Bank *An analysis of the composite step bi-conjugate gradient method* , Numer. Math., 66 (1993) 295-319.

[23] T. F. Chan, W. L. Wan, *Analysis of projection methods for solving linear systems with multiple right-hand sides*, SIAM J. Sci. Comput., 18 (1997) 1698-1721.

[24] E. J. Craig, *The N-step iteration procedures*, J. Math. Physics., 34 (1995) 64-73.

[25] A. Draux, *Polynômes orthogonaux formels*, LNM. vol 974. Springer-Verlag, Berlin (1983).

[26] S. C. Eisenstat, H. C. Elman, M. H. Schultz, A. Sherman, *Solving approximations to the convection diffusion equation* , Proc. Fifth SPE Symposium on Reservoir Simulation, Denver, Colorado, (1979) 127-132.

[27] S. C. Eisenstat, H. C. Elman, M. H. Schultz, A. Sherman, *Variational iterative method for nonsymmetric systems of linear equation*, Siam J. Numer. Anal., 20 (1989) 345-357.

[28] A. El Guennouni, K. Jbilou, H. Sadok, *A block version of BiCGSTAB for linear systems with multiple right-hand sides*, Elec. Trans. Numer. Anal., 16 (2003) 129-142.

[29] R. Fletcher, *Conjugate gradient methods for indefinite systems*, in Numerical Analysis, G. A. Waston editor, LNM 506, Springer-Verlag, New York, (1976) 73-89.

[30] R. W. Freund, *Conjugate gradient-type methods for linear systems with complex symmetric coefficient matrices.* SIAM J. Sci. Stat. Comput., 13, No. 1 (1992) 425-448.

[31] R. W. Freund, M. H. Gutknecht, N. M. Nachtigal, *An implementation of the look-ahead Lanczos algorithm for non-Hermitian matrices*, SIAM J. Sci. Comp. 14 (1993) 137-158.

[32] R. W. Freund, M. Hochbruck, *On the use of two QMR algorithms for solving singular systems and applications in Markov chain modeling* , Technical Report 91.25 RIACS, NASA Ames Research Center, Moffet Field, CA, December (1991).

[33] R. W. Freund, M. Malhotra, *A Block-QMR algorithm for non-Hermitian linear systems with multiple right-hand sides*, Linear Alg. Appl., 254 (1997) 119-157.

[34] R. W. Freund, N. M. Nachtigal, *QMR : a quasi-minimal residual method for non-Hermitian linear systems* , Numer. Math., 60 (1991) 315-339.

[35] R. W. Freund, N. M. Nachtigal, *An implementation of the QMR method based on coupled two-term recurrences*, SIAM J. Sci. Statist. Comput., 15 (1994) 313-337.

[36] G. H. Golub, C. F. Van Loan, *Matrix Computations*, Johns Hopkins Studies in the Mathematical Sciences. The Johns Hopkins University Press, Baltimore, MD, USA, third edition, (1996).

[37] G. H. Golub, R. R. Underwood, *A block Lanczos algorithm for computing the q algebraically largest eigenvalues and a corresponding eigenspace of large, sparse, real symmetric matrices*, Proc. of 1974 IEEE conf. on Decision avd Control, Phoenix.

[38] G. H. Golub, R. R. Underwood, *The block Lanczos method for computing eigenvalues*, in Mathematical Software 3, J. R. Rice ed., Academic Press, New York, (1977) 364-377.

[39] P. R. Graves-Morris, *A "Look-around Lanczos" algorithm for solving a system of linear equation*, Numer. Algorithms, 15 (1997) 247-274.

[40] M. H. Gutknecht, *A completed theory of the unsymmetric lanczos process and related algorithms*, Part I, SIAM J. Matrix Anal. Appl., 13 (1992) 594-639.

[41] M. H. Gutknecht, *A completed theory of the unsymmetric Lanczos process and related algorithms*, Part II, SIAM J. Matrix Anal. Appl., 15 (1994) 15-58.

[42] M. H. Gutknecht, *The unsymmetric Lanczos algorithms and their relations to Padé approximation, continued fractions and the qd algorithm*, dans Proceedings of the Copper Mountain Conference on Iterative Methods, 1990, unpublished.

[43] M. H. Gutknecht, *Variants of Bi-CGSTAB for matrices with complex spectrum*, SIAM. J. Sci. Comput., 193 (1993) 1020-1033.

[44] M. H. Hestenes, E. Stiefel *Methods of conjugate gradient for solving linear systems*, J. Res. Natl. Bur. Stand., 49 (1952) 409-435.

[45] K. Jbilou, H. Sadok, A. Tinzefte, *Oblique projection methods for linear systems with multiple right-hand sides*, Elect. Trans. Numer. Anal., 20 (2005) 119-138.

[46] K. Jbilou, A. Messaoudi, H. Sadok, *Global FOM and GMRES algorithms for matrix equations*, Appl. Numer. Math., 31 (1999) 49-63.

[47] K. C. Jea, D. M. Young, *On the simplification of generalized conjugate gradient methods for nonsymmetrizable linear systems*, Linear Alg. Appl., 52/53 (1983) 399-417.

[48] P. Joly, *Résolution de systèmes linéaires avec plusieurs seconds membres par la méthode du gradient conjugué*, Tech. Rep. R-91012, Publication du Laboratoire d'Analyse Numérique, Université Pierre Marie Curie, Paris, (1990).

[49] W. Joubert, *Generalized Conjugate Gradient and Lanczos methods for the solution of nonsymmetric systems of linear equations*, Ph.D. Thesis, University of Texas at Austin, Austin, (1990).

[50] C. Lanczos, *An iteration method for the solution of the eigenvalue problem of linear differential and integral operators*, J. Res. Nat. Bur. Stand., 45 (1950) 255-288.

[51] C. Lanczos, *Solution of systems of linear equations by minimized iterations*, J. Res. Nat. Bur. Stand., 49(1952) 33-53.

[52] M. Malhotra, R. Freund, P. M. Pinsky, *Iterative solution of multiple radiation and scattering problems in structural acoustics using a block Quasi-Minimal residual algorithm*, Comp. Meth. Appl. Mech. Eng., 146 (1997) 173-196.

[53] G. Meurant, *Computer Solution of Large Linear Systems*, Volume 28 of Studies in Mathematics and its Applications. North-Holland Publishing Co., Amsterdam, (1999).

[54] A. Nikishin, A. Yermin, *Variable block CG Algorithms for Solving Large Sparse Symmetric Positive Definite Linear Systems on Parallel Computers, I : General Iterative Scheme*, SIAM J. Matrix Anal. Appl., 16 (1995) 1135-1153.

[55] D. O'Leary, *The block conjugate gradient algorithm and related methods*, Linear Algebra Appl., 29 (1980) 293-322.

[56] C. C. Paige, M. A. Saunders, *Solution of sparse indefinite systems of linear equations*, SIAM. J. Numer. Anal., 12 (1975) 617-629.

[57] C. C. Paige, M. A. Saunders, *LSQR : An algorithm for sparse linear equations and sparse least squares*, ACM Trans. Math. Soft., 8 (1982) 43-71.

[58] B. N. Parlett, D. R. Taylor, Z. S. Liu, *A look-ahead Lanczos algorithm for nonsymmetric matrices*, Math. Comp., 44 (1985) 105-124.

[59] Y. Saad, *Variations on Arnoldi's method for computing eigenelements of large unsymmetric matrices*, Linear. Alg. Appl., 34 (1980) 269-295.

[60] Y. Saad, K. Wu, *DQGmres : A Direct Quasi-Minimal Residual algorithm based on Incomplete Orthogonalization*, Numer. Linear. Algebra Appl. 3 (1996) 329-343.

[61] Y. Saad, *Krylov subspace Methods for Solving unsymmetric Linear Systems* , Math. Comp., 37 (1981), 105-126.

[62] Y. Saad, *On the Lanczos method for solving symmetric linear systems with several right-hand sides*, Math. Comp., 48 (1987) 651-662.

[63] Y. Saad, M. H. Schultz, *GMRES : a generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Stat. Comp., 7 (1986) 856-869.

[64] Y. Saad, *Iterative Methods for Sparse Linear Systems*, PWS publishing, New York, (1995).

[65] M. Sadkane, *Block Arnoldi and Davidson methods for unsymmetric large eigenvalue problems*, Numer. Math., 64 (1993) 687-706.

[66] H. Sadok, *Communication privée.*

[67] H. Sadok, *CMRH : A new method for solving nonsymmetric linear systems based on the Hessenberg reduction algorithm*, Numer. Algo., 20 (1999) 303-321.

[68] M. A. Saunders, H. D. Simon, E. L. Yip *Two conjugate-gradient type methods for unsymmetric linear equations*, SIAM. J. Numer. Anal., 25 (1988) 927-940.

[69] W. Schonauer, *Scientific Computing on Vector Computers*, North-Holland, Amsterdam, New York, Tokyo, (1997).

[70] V. Simoncini, *A stabilized QMR version of block BICG*, SIAM J. Matrix Analysis and Appl., 18 (1997) 419-434.

[71] V. Simoncini, E. Gallopoulos, *An Iterative Method for Nonsymmetric Systems with Multiple Right-hand Sides*, SIAM J. Sci. Comp., 16 (1995) 917-933.

[72] C. F. Smith, A. F. Peterson, R. Mittra, *A Conjugate Gradient algorithm for treatment of multiple incident electromagnetic fields*, IEEE Transactions On Antennas and Propagation, 37 (1989) 1490-1493.

[73] P. Sonneveld, *CGS, a fast Lanczos-type solver for nonsymetric linear systems*, SIAM J. Sci. Stat. Comp., 10 (1989) 36-52.

[74] G. W. Struble, *Orthogonal polynomials : variable-signed Weight Functions*, Numer. Math., 5 (1963) 88-94.

[75] H. Van Rossum, *A theory of orthogonal polynomials based on the Padé table*, Thesis, University of Utrecht, Van Gorcum, Assen, (1953).

[76] H. A. Van der Vorst, *An Iterative Solution Method for Solving $f(A) = b$, using Krylov Subspace Information Obtained for the Symmetric Positive Definite Matrix A*, J. Comp. Appl. Math., 18 (1987) 249-263.

[77] H. A. Van der Vorst, *Bi-CGSTAB : A fast and smoothly converging variant of Bi-CG for the solution of non-symmetric linear systems*, SIAM J. Sci. Stat. Comp., 12 (1992) 631-644.

[78] P. K. W. Vinsome, *ORTHOMIN : an iterative method for solving sparse sets of simultaneous linear equations*, Proc. Fourth. SPE. Symposium on Reservoir Simulation, Los Angeles, (1976) 149-160,

[79] B. Vital, *Etude de Quelques Méthodes de Résolution de Problèmes Linéaires de Grande Taille sur Multiprocesseur*, thèse, Université de Rennes, Rennes, France, (1990).

[80] J. H Wilkinson, *The Algebraic Eigenvalue Problem*, Clarendon Press, Oxford, (1965).

[81] D. M. Young, K. C. Jea, *Generalized conjugate-gradient acceleration of nonsymmetrizable iterative methods*, Linear Algebra Appl., 34 (1980) 159-194.