

Calcul des valeurs propres

par **Bernard PHILIPPE**

INRIA Rennes-Bretagne Atlantique

et **Yousef SAAD**

Department of computer science and engineering, university of Minnesota

1. Principes de calcul des valeurs propres	AF 1 224 - 2
2. Algorithme QR pour le cas non symétrique	— 5
3. Algorithmes pour le cas d'une matrice pleine symétrique	— 8
4. Bibliothèque LAPACK	— 9
5. Méthodes pour les matrices de grande taille	— 9
6. Problème généralisé aux valeurs propres	— 17
7. Décomposition aux valeurs singulières	— 20
8. Conclusion	— 22
Pour en savoir plus	Doc. AF 1 224

Calculer les valeurs propres et les vecteurs propres de matrices est un des problèmes les plus importants en **analyse numérique linéaire**. Les techniques requérant la connaissance du spectre de matrices sont utilisées dans des domaines aussi variés que la mécanique quantique, l'analyse des structures, la théorie des graphes, les modèles de l'économie et le classement des pages de la Toile informatique par les moteurs de recherche.

Par exemple, en mécanique des structures, les problèmes de « résonances » ou de « vibrations » de structures mécaniques, décrits par l'analyse spectrale, se ramènent à des calculs de valeurs et de vecteurs propres.

Les problèmes non symétriques de valeurs propres apparaissent dans l'analyse de la stabilité de systèmes dynamiques. Dans un tout autre domaine, la chimie quantique donne lieu à des problèmes symétriques aux valeurs propres qui peuvent être gigantesques, tant par leur taille que par le nombre de valeurs et de vecteurs propres à extraire. On peut également mentionner que la décomposition aux valeurs singulières, qui est une sorte de généralisation de la décomposition spectrale classique, est primordiale en statistique et dans les problèmes de la « nouvelle économie » (reconnaissance de formes, fouille de données, traitement du signal, exploitation de données, etc.).

Les problèmes de valeurs propres sont très riches, tant par leur variété que par le type de matrices que l'on doit traiter et par les méthodes et algorithmes de calcul à utiliser : les matrices peuvent être symétriques ou non symétriques, creuses ou pleines, et les problèmes peuvent être classiques ou généralisés ou même quadratiques. Il existe des applications qui requièrent le calcul d'un très petit nombre de valeurs propres, d'autres au contraire un grand nombre de valeurs propres ou même tout le spectre.

On essaiera donc dans cet article de survoler les outils permettant de résoudre ces différents cas.

1. Principes de calcul des valeurs propres

1.1 Applications du calcul des valeurs propres

Une application linéaire \mathcal{L} d'un espace vectoriel E de dimension n dans lui-même est caractérisée par une matrice A . Celle-ci dépend de la base de référence \mathcal{B} dans l'espace vectoriel, ce que l'on note $A = M_{\mathcal{B}}(\mathcal{L})$. En changeant la base de référence, on change la matrice A en une matrice \tilde{A} semblable à A : $\tilde{A} = X^{-1}AX$ où les colonnes de X sont les vecteurs de la nouvelle base exprimée dans l'ancienne base. Une question naturelle consiste alors à se demander s'il est possible de choisir la nouvelle base de manière que la matrice \tilde{A} ait la forme la plus simple c'est-à-dire la forme diagonale. En effet, y parvenir revient à dire que l'on a pu *découpler* l'action de l'application linéaire en n applications scalaires.

Supposons qu'il existe une matrice inversible X telle que $D = X^{-1}AX$ soit diagonale. En notant $D = \text{diag}(\lambda_1, \dots, \lambda_n)$ et $X = [x_1, \dots, x_n]$ où les x_i représentent les colonnes de X , on en déduit que :

$$Ax_i = \lambda_i x_i, \text{ pour } i = 1, \dots, n \quad (1)$$

Cela entraîne que les valeurs (λ_i) sont telles que $(A - \lambda_i I)$ n'est pas inversible. Ce sont donc les racines du polynôme $p(\lambda) = \det(A - \lambda I)$, polynôme qui ne dépend pas de la base choisie. On les appelle valeurs propres de A . Tout vecteur $x \in \ker(A - \lambda_i I) \setminus \{0\}$ est appelé vecteur propre associé à λ_i . Nous étudierons dans le paragraphe suivant l'existence de ces éléments propres mais voyons d'abord deux exemples d'utilisation des valeurs propres.

1.1.1 Première application : récurrences linéaires

Supposons qu'un phénomène soit décrit par des effectifs $x_1(k), x_2(k), \dots, x_n(k)$ de n classes à intervalles de temps réguliers numérotés par l'indice k et que le passage d'un instant au suivant soit régi par une multiplication par la matrice A :

$$X(k+1) = AX(k)$$

où $X(k)$ est le vecteur des effectifs. On peut donc naturellement écrire que $X(k) = A^k X(0)$, où $X(0)$ est le vecteur des effectifs initiaux. Si la matrice A est diagonale, alors il est immédiat d'en calculer n'importe quelle puissance, puisqu'il suffit de le faire sur chaque coefficient diagonal. Si la matrice est diagonalisable, c'est-à-dire s'il existe une matrice de changement de base $P \in \mathbb{R}^{n \times n}$ telle que la matrice A puisse s'écrire $A = PDP^{-1}$ où D est une matrice diagonale $D = \text{diag}(\lambda_1, \dots, \lambda_n)$, alors on peut aussi calculer sa puissance k -ème par la relation $A^k = PD^k P^{-1}$. On en déduit que pour tout $i = 1, \dots, n$, l'effectif $x_i(k)$ est une combinaison linéaire des puissances k -èmes des valeurs propres $\lambda_1, \dots, \lambda_n$. Si toutes les valeurs propres sont de valeurs absolues inférieures à 1, alors les effectifs tendent vers 0, tandis que si l'une d'entre elles au moins est supérieure à 1, le processus va exploser pour presque tous les vecteurs initiaux.

Comme exemple, considérons la **suite de Fibonacci** qui est définie par la récurrence :

$$\alpha_{k+1} = \alpha_k + \alpha_{k-1}, \text{ pour } k \geq 1$$

$$\text{et } \alpha_0 = 1, \alpha_1 = 1$$

Elle peut s'écrire $X_k = \begin{pmatrix} \alpha_{k+1} \\ \alpha_k \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} X_{k-1}$. Les valeurs propres de

la matrice sont $\lambda_1 = \frac{1 + \sqrt{5}}{2}$ et $\lambda_2 = \frac{1 - \sqrt{5}}{2}$. Les valeurs initiales permettent alors d'en déduire que $\alpha_k = \frac{\sqrt{5}}{5} (\lambda_1^k - \lambda_2^k)$. Nous sommes dans le cas d'une suite qui tend vers l'infini quand k tend vers l'infini.

Pour ces récurrences, voir l'article *Équations aux différences* [AF 104].

1.1.2 Deuxième application : stabilité des systèmes différentiels

Dans le deuxième exemple, nous considérons le modèle d'un processus qui évolue en fonction du temps de manière continue. On suppose que le modèle est celui d'un système différentiel linéaire homogène à coefficients constants :

$$\begin{cases} \frac{dx}{dt} = Ax(t) \text{ pour } t \geq 0 \\ x(0) = x_0 \end{cases}$$

où $x(t) \in \mathbb{R}^n$. Une question classique est alors de se demander si le système est asymptotiquement stable ou non : est-ce que la solution $x(t)$ tend vers 0 quand t tend vers l'infini ? Pour simplifier, supposons que la matrice A est diagonalisable au sens défini plus haut. Par un raisonnement du même type que pour le cas des récurrences linéaires, on montre que toutes les composantes du vecteur $x(t)$ sont des combinaisons linéaires fixes des exponentielles $e^{\lambda_i t}$ où les valeurs propres (éventuellement complexes) de la matrice A sont $\{\lambda_i, i = 1, \dots, n\}$. Le système sera stable si toutes les valeurs propres sont à parties réelles strictement négatives.

Sur la stabilité des systèmes différentiels, voir l'article *Aspects numériques du contrôle linéaire* [AF 1400].

1.2 Décomposition spectrale d'une matrice

On précise maintenant les notions introduites dans le paragraphe précédent. Dans tout l'article, le corps \mathbb{K} peut être soit le corps \mathbb{C} des complexes soit le corps des réels \mathbb{R} .

Définition 1

Soit A une matrice carrée d'ordre n : $A \in \mathbb{K}^{n \times n}$.

Le polynôme caractéristique de A est le polynôme défini par $p(z) = \det(A - zI)$. Les valeurs propres de A sont ses racines et leur ensemble $\lambda(A)$ forme le spectre de la matrice. Toutes les matrices semblables à A ont le même polynôme caractéristique.

Un vecteur $x \in \mathbb{K}^n$ est un vecteur propre de A associé à la valeur propre $\lambda \in \mathbb{K}$ si et seulement si c'est un vecteur non nul qui vérifie $Ax = \lambda x$. Le noyau $\ker(A - \lambda I)$ des vecteurs propres associés à une valeur propre λ , ensemble complété du vecteur nul, forme un sous-espace invariant par A appelé sous-espace propre associé à λ .

La matrice A est dite diagonalisable dans \mathbb{K} s'il existe une base $X = [x_1, \dots, x_n] \in \mathbb{K}^{n \times n}$ telle que la matrice $D = X^{-1}AX$ soit diagonale. On dit alors que X diagonalise A .

Si λ est une valeur propre alors par définition il existe au moins un vecteur propre u associé ou autrement dit, le sous-espace propre $\ker(\lambda I - A)$ est au moins de dimension 1. Cela permet d'affirmer la proposition suivante.

Proposition 1

Si le polynôme caractéristique d'une matrice a n racines distinctes dans \mathbb{K} alors la matrice est diagonalisable dans \mathbb{K} .

Remarque 1

Il existe des matrices non diagonalisables. Si on choisit $\mathbb{K} = \mathbb{R}$, alors il peut y avoir des valeurs propres complexes. Même lorsque l'on se place dans le corps $\mathbb{K} = \mathbb{C}$, certaines matrices ne sont pas diagonalisables (elles ont donc des valeurs propres multiples). Par exemple la matrice $A = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}$ n'est diagonalisable ni dans \mathbb{R} ni dans \mathbb{C} .

On énonce maintenant un théorème fondamental de décomposition. La démonstration du théorème est constructive. Certains algorithmes sont construits sur ce principe. Dans l'énoncé, l'exposant H appliqué à une matrice dénote son adjoint, c'est-à-dire la matrice conjuguée de sa transposée.

Théorème 1 : forme de Schur complexe

Dans \mathbb{C} , toute matrice A est unitairement semblable à une matrice triangulaire supérieure : il existe une matrice $U \in \mathbb{C}^{n \times n}$ telle que $U^H U = I$ (donc $U^H = U^{-1}$) et la matrice $T = U^H A U$ est triangulaire supérieure.

Preuve

♦ Voir par exemple [7] p. 9-10 ou [6] p. 33.34. ♦

Remarque 2

Puisque le déterminant d'une matrice triangulaire est égal au produit de ses éléments diagonaux, il est évident que les valeurs propres de A sont obtenues sur la diagonale de la matrice T de la forme de Schur de A .

Pour tout $\mu \in \mathbb{R}$, on a $A - \mu I = U(T - \mu I)U^H$ et $(A - \mu I)^{-1} = U(T - \mu I)^{-1}U^H$, ce qui prouve que les deux matrices sont réduites sous la forme de Schur par la même matrice unitaire que celle qui réduit A . Si λ est valeur propre de A alors $\lambda - \mu$ et $\frac{1}{\lambda - \mu}$ (on suppose dans ce cas que $\lambda \neq \mu$) sont respectivement valeurs propres de $A - \mu I$ et de $(A - \mu I)^{-1}$.

La forme de Schur n'est pas unique puisque l'on peut choisir un ordre arbitraire d'énumération des valeurs propres sur la diagonale de T .

On énonce maintenant un théorème qui est à la base de pratiquement tous les traités sur la diagonalisation des matrices car il décrit une réduction de matrice sous la forme la plus simple. Par contre, la réduction qu'il décrit n'est pas numériquement calculable en arithmétique puisqu'elle correspond à un problème mal posé car l'ensemble des matrices complexes diagonalisables à valeurs propres distinctes est dense dans l'ensemble des matrices.

Théorème 2 : forme de Jordan

Toute matrice carrée complexe est semblable à une matrice J diagonale par blocs où tout bloc diagonal B peut être soit :
 - de dimension 1 : $B = \mu$: il est alors une valeur propre de la matrice ;
 - de dimension $r > 1$: il est alors une matrice bidiagonale de la forme $B = \mu I + L$ où μ est une valeur propre et L la matrice carrée de dimension r dont tous les coefficients sont nuls sauf ceux de la première surdiagonale qui sont égaux à 1.

On appelle blocs de Jordan ces blocs, même dans le cas de la dimension 1.

Preuve

♦ Voir par exemple [6] p. 34-37. ♦

Ainsi chaque bloc de Jordan est associé à une valeur propre. Par contre, plusieurs blocs peuvent être associés à la même valeur propre. On peut montrer que l'exposant du facteur $(\lambda - \mu)$ dans la forme factorisée du polynôme caractéristique $p(\lambda)$ est égal à la somme des dimensions des blocs associés à la valeur propre μ .

Remarque 3

La décomposition de Jordan est plus complète que la décomposition de Schur puisque cette dernière s'obtient facilement à partir de la forme canonique de Jordan et de la factorisation QR (pour la définition de cette dernière, voir l'article *Méthodes numériques de base. Algèbre numérique* [AF 1 221]). En effet, de la forme de Jordan $A = XJX^{-1}$, et de la factorisation QR de X qui assure que $X = QR$ où Q est une matrice orthogonale et R une matrice triangulaire supérieure, inversible puisque X est inversible, alors la matrice A se décompose en $A = Q(RJR^{-1})Q^H$. Comme RJR^{-1} est triangulaire supérieure car J est bidiagonale supérieure, on a bien le résultat.

Corollaire 1 : développement du polynôme caractéristique

Le polynôme caractéristique $p(\lambda) = \det(A - \lambda I)$ s'écrit sous la forme :

$$p(\lambda) = (-1)^n \lambda^n + (-1)^{n-1} \gamma_1 \lambda^{n-1} + (-1)^{n-2} \gamma_2 \lambda^{n-2} + \dots - \gamma_{n-1} \lambda + \gamma_n$$

où les (γ_i) sont les polynômes symétriques élémentaires en les valeurs propres (λ_i) de A . En particulier :

$$\gamma_1 = \sum_{i=1}^n \lambda_i = \text{trace}(A) \tag{2}$$

$$\text{et } \gamma_n = \prod_{i=1}^n \lambda_i = \det(A) \tag{3}$$

Preuve

♦ Les formules (2) et (3) sont faciles à montrer. Pour calculer les autres coefficients du polynôme caractéristique, on a recours aux identités de Newton (voir [15] p. 166-168). ♦

Corollaire 2 : cas des matrices hermitiennes

Une matrice $A \in \mathbb{C}^{n \times n}$ hermitienne est diagonalisable par une matrice unitaire et toutes ses valeurs propres sont réelles : il existe une matrice $U \in \mathbb{C}^{n \times n}$ telle que $U^H U = I$ et la matrice $D = U^H A U$ est diagonale réelle.

Dans le cas particulier où la matrice A est réelle symétrique, alors la matrice U est une matrice réelle orthogonale ($U^T U = I$).

Preuve

♦ On suppose que A est hermitienne : $A^H = A$. On en déduit qu'une décomposition de Schur de cette matrice vérifie : $T^H = (U^H AU)^H = U^H AU = T$. Une matrice triangulaire supérieure hermitienne ne peut être que diagonale. De plus ses éléments diagonaux doivent être égaux à leurs conjugués ; ils sont donc réels. ♦

Théorème 3 : forme de Schur réelle

Toute matrice $A \in \mathbb{R}^{n \times n}$ est orthogonalement semblable à une matrice quasi-triangulaire supérieure : il existe une matrice $Q \in \mathbb{R}^{n \times n}$ telle que $Q^T Q = I$ et la matrice $T = Q^T A Q$ est triangulaire supérieure par bloc, les blocs diagonaux étant de dimension 1 ou 2 (les blocs de dimension 2 correspondent à des valeurs propres complexes conjuguées).

Preuve

♦ Il suffit d'adapter la démonstration du théorème de la forme de Schur complexe. On peut avancer la récurrence d'une ou deux unités suivant que l'on considère une valeur propre réelle λ ou un couple de valeurs propres conjuguées λ et $\bar{\lambda}$. ♦

Il existe des inégalités sur les modules des valeurs propres qui permettent de les localiser dans des parties bornées du plan complexe. La plus simple (mais la plus lâche) est donnée par toute norme matricielle subordonnée à une norme vectorielle.

Proposition 2

Pour toute norme matricielle de $\mathbb{C}^{n \times n}$ subordonnée à une norme de \mathbb{C}^n , on est assuré de l'inégalité :

$$\rho(A) \leq \|A\|$$

où $\rho(A) = \max\{|\lambda| \mid \lambda \text{ est valeur propre de } A\}$ est appelé le rayon spectral de A .

Preuve

♦ Soit λ une valeur propre de module maximal et u un vecteur propre associé normé dans la norme vectorielle considérée. On a alors l'inégalité suivante :

$$\rho(A) = |\lambda| \|u\| = \|Au\| \leq \|A\| \quad \blacklozenge$$

Cela entraîne donc que le spectre de A est inclus dans le disque centré en 0 et de rayon $\|A\|$. Le théorème suivant permet de définir une région plus restreinte que ce disque.

Théorème 4 : Gershgorin

Pour toute matrice $A = (a_{ij}) \in \mathbb{C}^{n \times n}$, l'ensemble des valeurs propres de A est inclus dans l'ensemble $(\cup_{i=1}^n \mathcal{D}_i)$ où \mathcal{D}_i est le disque fermé du plan complexe de centre a_{ii} et de rayon $\sum_{j \neq i} |a_{ij}|$.

Théorème 5 : Cayley-Hamilton

Le polynôme caractéristique p de la matrice $A \in \mathbb{C}^{n \times n}$ vérifie $p(A) = 0$.

Preuve

♦ Voir [13]. ♦

Dans l'anneau des polynômes, l'ensemble des polynômes qui sont nuls en A forme un idéal. Le polynôme caractéristique p y appartient.

Définition 2

Le polynôme minimal est le polynôme monique q qui engendre l'idéal :

$$\mathcal{I} = \{q \in \mathbb{C}[X] \mid q(A) = 0\}$$

Le polynôme minimal divise donc le polynôme caractéristique et c'est donc celui de plus petit degré qui vérifie $q(A) = 0$.

Remarque 4

On peut montrer que l'exposant du facteur $(\lambda - \mu)$ dans la forme factorisée du polynôme caractéristique $p(\lambda)$ est égal à la plus grande dimension des blocs de Jordan associés à la valeur propre μ .

On termine les rappels de propriétés mathématiques par le théorème suivant qui exprime une propriété fondamentale du spectre des matrices symétriques.

Théorème 6 : Cauchy

Soit B la matrice principale supérieure de $A \in \mathbb{R}^{n \times n}$ de dimension $n-1$. En numérotant les valeurs propres en ordre croissant : $\lambda_1 \leq \dots \leq \lambda_n$ pour A et $\mu_1 \leq \dots \leq \mu_{n-1}$ pour B , on obtient l'entrelacement suivant :

$$\lambda_1 \leq \mu_1 \leq \lambda_2 \leq \mu_2 \leq \dots \leq \mu_{n-1} \leq \lambda_n$$

Preuve

♦ Voir [30] p. 197. ♦

1.3 Algorithme de la puissance itérée et ses dérivés

Pour simplifier l'exposé, on se restreint ici au cas d'une matrice réelle, mais la généralisation au cas complexe est triviale. Si on suppose que la matrice réelle A a pour valeur propre de plus grand module une seule valeur propre et simple λ (donc λ est égale au rayon spectral de A ou à son opposé), alors l'algorithme 1 permet de calculer cette valeur propre ainsi que son vecteur propre x associé.

Algorithme 1 – Algorithme de la puissance $[\lambda, x] = \text{puissance}(A, \text{tol})$

```
x=rand(n,1) ; x = x/norm(x) ;
y=A*x ; lambda = x'*y ;
r= y - lambda *x ;
while norm(r) > tol
    x = y / norm(y) ;
    y = A*x ; lambda = x'*y ;
    r = y - lambda*x ;
end ;
```

Nota : l'algorithme est écrit sous une forme simplifiée ; pour une version de bibliothèque, il serait nécessaire de tester le nombre d'itérations afin d'arrêter le procédé en cas de non-convergence ; il faudrait aussi définir un paramètre tol qui soit proportionnel à $|\lambda|$ ou à la norme de la matrice A .

Proposition 3

L'algorithme 1 converge presque sûrement au taux de convergence $\left| \frac{\lambda_2}{\lambda} \right|$ où λ_2 est une valeur propre de plus grand module inférieur au rayon spectral de A .

Preuve

♦ La suite des itérés est indicée par $k=0, 1, \dots$ où x_0 est un vecteur initial. Supposons que $A = QDQ^T$ où Q est la matrice orthogonale des vecteurs propres et $D = \text{diag}(\lambda, \lambda_2, \dots, \lambda_n)$. Soit $u = Q^T x_0$. Alors $x_k = \alpha_k A^k x_0$ où α_k est un certain réel qui rend normé le vecteur x_k . On en déduit que $x_k = \alpha_k Q D^k u$ et donc que :

$$x_k = \lambda^k \alpha_k Q \begin{pmatrix} u_1 \\ \left(\frac{\lambda_2}{\lambda}\right)^k u_2 \\ \vdots \\ \left(\frac{\lambda_n}{\lambda}\right)^k u_n \end{pmatrix}$$

où $u_i (i = 1, \dots, n)$ sont les composantes de u . La convergence de la direction du vecteur x_k vers celle du vecteur q_1 , première colonne de Q est donc obtenue dès que $u_1 \neq 0$ (si la valeur propre λ est négative, la suite des itérés sera alternée à la limite). ♦

Cet algorithme a eu beaucoup de succès à cause de sa grande simplicité et du fait qu'il n'accède à la matrice A qu'à travers sa multiplication par des vecteurs. Cette propriété est spécialement intéressante dans le cas des matrices de grande taille. La meilleure illustration est sans doute son utilisation dans le **moteur de recherche Google** qui recherche le vecteur propre dominant d'une matrice stochastique de dimension supérieure à 25 milliards [17]. Cependant, ces avantages sont souvent un peu illusoire car la méthode est à convergence lente pour les grandes matrices où le

plus couramment, le rapport $\left| \frac{\lambda_2}{\lambda} \right|$ est très proche de l'unité.

Par contre, une adaptation de cet algorithme est couramment utilisée pour calculer le vecteur propre d'une valeur propre simple μ déjà estimée par une bonne approximation μ . Supposons que l'erreur sur la valeur propre soit $\varepsilon = |\lambda - \mu|$ et que $\varepsilon \ll |\lambda_2 - \mu|$ où λ_2 est la valeur propre de A différente de λ mais la plus proche. Si on applique l'algorithme de la puissance à la matrice $C = (A - \mu I)^{-1}$ alors

le taux de convergence sera de $\left| \frac{\varepsilon}{\lambda_2 - \mu} \right| \ll 1$. En remarquant que la

relation $y = Cx$ entraîne que $\frac{x}{\|x\|} = (A - \mu I) \frac{y}{\|y\|}$ est égal au résidu du

couple $\left(\mu, \frac{y}{\|y\|} \right)$, on en déduit que la norme du résidu est $\frac{1}{\|y\|}$. On peut donc utiliser cette quantité pour détecter la convergence.

Algorithme 2 – Algorithme de la puissance inverse
 $x = \text{puissance_inverse}(A, \mu, \text{tol})$

```
B=A - mu*eye(n) ;
x=rand(n,1) ; x = x/norm(x) ;
y=B \ x ; alpha = norm(y) ;
while 1/alpha >= tol
    x = y / alpha ;
    y = B \ x ;
    alpha = norm(y) ;
end ;
```

L'algorithme 2 est utilisé dans les bibliothèques pour calculer le vecteur propre associé à une valeur propre calculée à la précision machine. On peut en être surpris car alors la résolution du système linéaire de chaque itération est singulier à la précision machine. Il produit donc une erreur très grande sur la solution. Cependant, un miracle se produit puisque l'erreur est elle-même pratiquement colinéaire au vecteur propre cherché (pour plus de précision, voir [24] p. 65-68). Avec une telle procédure, lorsque la valeur propre est assez bien isolée du reste du spectre, une seule itération suffit pour obtenir la convergence.

On peut aussi adapter la méthode de la puissance inverse en réestimant à chaque itération la valeur propre par le quotient de Rayleigh : pour chaque itéré normalisé x du vecteur propre, on calcule le nombre $\lambda = \rho(x) = x^T A x$. Cela permet de partir d'une approximation plus grossière de la valeur propre. L'inconvénient est que la procédure entraîne une factorisation de matrice à chaque itération. On obtient alors l'algorithme suivant.

Algorithme 3 – Algorithme du quotient de Rayleigh
 $[\lambda, x] = \text{quotient_rayleigh}(A, \mu, \text{tol})$

```
x=rand(n,1) ; x = x/norm(x) ;
y=(A - mu*eye(n)) \ x ; alpha = norm(y) ;
while 1/alpha >= tol
    x = y / alpha ;
    lambda = x'*A*x ;
    y = (A - lambda*eye(n)) \ x ;
    alpha = norm(y) ;
end ;
```

Lorsque la matrice est réelle symétrique (ou hermitienne complexe), la convergence de l'algorithme est cubique (voir [24] p. 72). Sinon, la convergence est quadratique.

2. Algorithme QR pour le cas non symétrique

Dans ce paragraphe, on suppose que la matrice A , dont on recherche les valeurs propres, est réelle et qu'on maintient les calculs réels aussi longtemps que possible. C'est en effet le cas le plus courant. Même si le recours à l'arithmétique complexe dans le cas des matrices non symétriques est conceptuellement plus simple que la restriction aux calculs réels puisqu'il supprime le traitement de cas particuliers, du point de vue informatique, on préfère éviter les calculs complexes car ils ralentissent généralement l'exécution. Cependant, tous les calculs décrits peuvent être exécutés en arithmétique complexe en faisant attention à utiliser le produit scalaire hermitien de deux vecteurs complexes x et y défini par $x^H y = \overline{x}^T y$.

Puisque la **forme de Hessenberg** est préservée par l'algorithme QR qui sera décrit ensuite, il est important de réduire d'abord la matrice donnée sous cette forme par des transformations de similarité.

2.1 Réduction à la forme Hessenberg supérieure

On rappelle d'abord les **transformations de Householder** pour transformer une matrice non symétrique à la forme de Hessenberg supérieure. Les matrices de symétrie de Householder sont des matrices de la forme :

$$P = I - 2ww^T$$

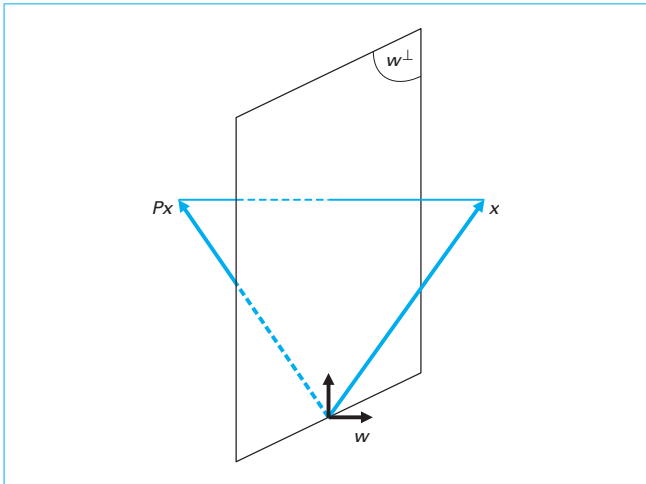


Figure 1 – Illustration des transformations de Householder

où w est un vecteur de norme unité. Une telle matrice correspond à une transformation linéaire qui envoie un vecteur x vers son image symétrique par rapport à l'hyperplan orthogonal à w , noté $\text{sev}\{w\}^\perp$.

On remarque d'abord que P est une matrice réelle symétrique orthogonale lorsque w est réel. Dans le cas complexe, la matrice de Householder $P = I - 2ww^H$ est hermitienne et unitaire. Pour simplifier l'exposé du paragraphe, on se limite au cas réel.

En pratique, on réécrit P sous la forme $P = I - \beta vv^T$ avec $\beta = 2/\|v\|^2$, où v est un multiple de w . Pour la stocker, il suffit de garder v et le scalaire β . Pour évaluer Px , on évalue $x - \beta(x^T v) \times v$. Dans le cas des matrices, on peut de la même manière évaluer PA par $PA = A - \beta v z^T$ où $z^T = \beta v^T A$.

Le premier problème que l'on doit résoudre est celui de trouver une transformation de Householder (figure 1) qui transforme un vecteur donné $x \neq 0$, en un vecteur de base (à une normalisation près). Donc, en premier lieu, on voudrait trouver w tel que :

$$(I - 2ww^T)x = \alpha e_1$$

où α est un scalaire quelconque qui reste à choisir.

En écrivant $(I - \beta vv^T)x = \alpha e_1$, on a $\beta(v^T x)v = x - \alpha e_1$. Donc la solution de ce problème est donnée par $v = x - \alpha e_1$ et il suffit de trouver α , qui est déterminé en se rappelant que $\|(I - 2ww^T)x\|_2 = \|x\|_2$ puisque $I - 2ww^T$ est unitaire. Le résultat est alors $\alpha = \pm \|x\|_2$. On choisira le signe opposé à celui de la première composante ξ_1 de x de manière à éviter les éliminations catastrophiques. Ainsi $\alpha = -\text{sign}(\xi_1)\|x\|_2$:

$$v = x + \text{sign}(\xi_1)\|x\|_2 e_1 \quad \text{et} \quad \beta = 2/\|v\|_2^2$$

$$v = \begin{pmatrix} \widehat{\xi}_1 \\ \widehat{\xi}_2 \\ \vdots \\ \widehat{\xi}_{n-1} \\ \widehat{\xi}_n \end{pmatrix} \quad \text{avec} \quad \widehat{\xi}_1 = \begin{cases} \xi_1 + \|x\|_2 & \text{if } \xi_1 > 0 \\ \xi_1 - \|x\|_2 & \text{if } \xi_1 \leq 0 \end{cases}$$

On peut généraliser le résultat ci-avant et se demander si on peut trouver une transformation qui laisse les $k - 1$ premières positions de x inchangées et qui mette des zéros aux composantes d'indices $k + 1 : n$. Pour cela, il suffit que w ait des zéros dans les positions numérotées de 1 à $k - 1$. Donc, $P_k = I - \beta vv^T$, (avec $\beta = 2/\|v\|^2$), ou, en utilisant les notations MATLAB, le vecteur v est

tel que $v(1 : k - 1) = 0$ et le reste est un vecteur de Householder déterminé comme ci-avant pour transformer le sous-vecteur $v(k : n)$, qui est de dimension $n - k + 1$, en la forme αe_1 .

Les transformations de Householder permettent de réduire la matrice de départ A en une matrice orthogonalement semblable $H = Q^T A Q$ qui soit Hessenberg supérieure de manière numériquement stable. Considérons la première étape, pour une matrice de taille 6×6 . On voudrait que $P_1 A P_1^T = P_1 A P_1$ ait la forme :

$$\begin{pmatrix} \star & \star & \star & \star & \star & \star \\ \star & \star & \star & \star & \star & \star \\ 0 & \star & \star & \star & \star & \star \\ 0 & \star & \star & \star & \star & \star \\ 0 & \star & \star & \star & \star & \star \\ 0 & \star & \star & \star & \star & \star \end{pmatrix}$$

On sélectionne un w pour $P_1 = I - 2ww^T$ qui assure que la première colonne ait des zéros pour les composantes numérotées de 2 à n . Donc en particulier, on choisira $w_1 = 0$.

On applique la transformation à gauche : $B = P_1 A$ puis à droite : $A_1 = B P_1$. On remarque alors que :

la matrice Householder P_1 qui transforme la colonne $A(2 : n, 1)$ en e_1 n'altère que les lignes 2 à n . Quand on applique P_1 (donc son inverse puisqu'elle est symétrique orthogonale, donc involutive) à droite de $B = P_1 A$, on observe que seules les colonnes 2 à n sont modifiées. Ainsi la première colonne est mise sous la forme voulue (avec des zéros en dessous de la deuxième ligne).

L'algorithme continue de la même manière pour les colonnes 2, ..., $n - 2$, en réduisant la dimension des vecteurs de Householder d'une unité à chaque étape. En sortie, on obtient donc une matrice A_{n-2} telle que $A = Q A_{n-2} Q^T$ où Q est la matrice orthogonale $Q = P_1 P_2 \dots P_{n-2}$.

Suivant le problème traité, on peut ou non avoir besoin de la matrice Q . Pour la conserver, on peut le faire en gardant les vecteurs w des transformations de Householder successives ou bien en calculant explicitement la matrice. Dans ce dernier cas, on le fait à partir de la dernière transformation qui est définie par le plus court vecteur car cela réduit les calculs.

2.2 Algorithme QR pour les matrices de Hessenberg

L'algorithme QR sur une matrice quelconque consiste mathématiquement à appliquer l'itération suivante :

$$\begin{cases} H_0 = H \text{ et } H_0 = I \\ \text{Pour } k \geq 0, \\ (Q_{k+1}, R_{k+1}) = qr(H_k) \text{ (factorisation QR)} \\ H_{k+1} = R_{k+1} Q_{k+1} \end{cases} \quad (4)$$

(pour la définition de la factorisation QR, voir l'article *Méthodes numériques de base. Algèbre numérique* [AF 1 221]). On remarque donc que $H_{k+1} = Q_{k+1}^T R_{k+1} Q_{k+1}$ est bien obtenue par une transformation de similitude orthogonale de H_k . Il est assez facile de voir que si la matrice H_k est Hessenberg supérieure alors il en est de même pour la matrice H_{k+1} .

On pourrait mettre en œuvre QR de manière naïve en programmant exactement les étapes de (4). Pour une matrice de Hessenberg, le coût est un peu moins important que pour une matrice

quelconque mais l'itération reste de complexité $O(n^3)$, où n est l'ordre de la matrice. Il existe une approche qui consiste à utiliser des matrices de Givens et à les appliquer de manière judicieuse. La complexité de l'itération sera alors en $O(n^2)$.

Les matrices de Givens sont des matrices de la forme :

$$G(i, k, \theta) = \begin{pmatrix} 1 & \dots & 0 & \dots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & c & \dots & s & 0 \\ \vdots & \dots & \vdots & \ddots & \vdots & \vdots \\ 0 & \dots & -s & \dots & c & 0 \\ \vdots & \dots & \vdots & \dots & \vdots & \vdots \\ 0 & \dots & 0 & \dots & \dots & 1 \end{pmatrix} \begin{matrix} i \\ k \end{matrix}$$

où $c = \cos \theta$ et $s = \sin \theta$. Cette matrice représente une rotation d'angle $-\theta$ dans le plan engendré par les vecteurs e_i et e_k .

L'idée principale des rotations de Givens est que l'on peut mettre une matrice sous forme triangulaire en la multipliant successivement à gauche par des matrices de Givens bien choisies. Considérons le vecteur $y = Gx$. Alors :

$$\begin{aligned} y_i &= cx_i + sx_k \\ y_k &= -sx_i + cx_k \\ y_j &= x_j \text{ pour } j \neq i, k \end{aligned}$$

Ainsi, on peut faire en sorte que $y_k = 0$ en sélectionnant $s = x_k/t$;

$c = x_i/t$; $t = \sqrt{x_i^2 + x_k^2}$. Cela peut être utilisé pour introduire des zéros dans les premières colonnes de A (par exemple, $G(n-1, n)$, $G(n-2, n-1)$ etc... $G(1, 2)$).

Pour l'algorithme QR , les matrices de Givens sont cependant utilisées de manière différente. Elles sont combinées grâce au théorème Q -implicite :

Théorème 7 : Q implicite [14]

Supposons que $Q^T A Q$ soit une matrice de Hessenberg supérieure irréductible. Alors les colonnes 2 à n de Q sont déterminées de manière unique (au signe près) par la première colonne de Q .

L'implication de ce théorème est importante en pratique : on pourra obtenir $A_{i+1} = Q_i^T A Q_i$, en calculant la première colonne de Q_i (scalaire $\times A(:, 1)$), et ensuite choisir les autres colonnes en s'arrangeant pour que Q_i soit orthogonale, et que A_{i+1} soit de forme de Hessenberg.

Par exemple, dans le cas $n = 6$:

$$A = \begin{pmatrix} * & * & * & * & * \\ * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & 0 & * & * & * \\ 0 & 0 & 0 & * & * \end{pmatrix}$$

■ **Étape 1** : choisir $G_1 = G(1, 2, \theta_1)$ tel que $Q(:, 1) = \text{scal} * A(:, 1)$. Remarque le terme de remplissage en position (3,1) :

$$A_1 = G_1^T A G_1 = \begin{pmatrix} * & * & * & * & * \\ * & * & * & * & * \\ + & * & * & * & * \\ 0 & 0 & * & * & * \\ 0 & 0 & 0 & * & * \end{pmatrix}$$

■ **Étape 2** : choisir $G_2 = G(2, 3, \theta_2)$ pour éliminer le terme (3,1) de la matrice $(G_2 A_1)$, introduit dans l'étape précédente. Noter alors le terme non nul qui apparaît en position (4,2) :

$$A_2 = G_2^T A_1 G_2 = \begin{pmatrix} * & * & * & * & * \\ * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & + & * & * & * \\ 0 & 0 & 0 & * & * \end{pmatrix}$$

■ **Étape 3** : choisir $G_3 = G(3, 4, \theta_3)$ pour éliminer le terme (4,2) de la matrice $(G_3 A_2)$. Noter le terme de remplissage en position (5,3) :

$$A_3 = G_3^T A_2 G_3 = \begin{pmatrix} * & * & * & * & * \\ * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & 0 & * & * & * \\ 0 & 0 & + & * & * \end{pmatrix}$$

■ **Étape 4** : choisir $G_4 = G(4, 5, \theta_4)$ tel que $(G_4 A_3)_{53} = 0$ pour éliminer le terme (5,3). Noter que la matrice finale est maintenant de forme Hessenberg :

$$A_4 = G_4^T A_3 G_4 = \begin{pmatrix} * & * & * & * & * \\ * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & 0 & * & * & * \\ 0 & 0 & 0 & * & * \end{pmatrix}$$

Ce procédé est parfois appelé en anglais « la chasse à la grosseur » (*bulge chasing*). Une approche similaire est suivie dans le cas symétrique (cf. § 3.1).

2.3 Convergence de la méthode QR et décalages

On a vu dans (4) que la méthode QR construit une suite de matrices de Hessenberg supérieures $(H_k)_{k \geq 0}$ orthogonalement semblables. On peut montrer qu'en général, une partie des éléments sous-diagonaux de H_k tendent vers zéro (il peut y avoir des cas particuliers dans le cas de valeurs propres de même module comme par exemple une matrice orthogonale qui définit une suite constante). À la limite, les éléments non nuls sont toujours isolés (c'est-à-dire séparés par au moins un zéro). Les points d'accumulation de la suite de matrices sont donc des matrices quasi triangulaires supérieures avec des blocs diagonaux de dimension 1 ou 2. Il s'agit d'une forme de Schur réelle. Un bloc de dimension 1 correspond à une valeur propre réelle tandis que les blocs de dimension 2 correspondent à des couples de valeurs propres complexes conjuguées (pour une démonstration de convergence, voir [7] p. 124-129).

En fait, la convergence peut être accélérée de manière importante par des décalages $u_k I$:

$$\begin{aligned} H_0 &= H \text{ et } H_0 = I \\ \left\{ \begin{aligned} &\text{Pour } k \geq 0, \\ &(Q_{k+1}, R_{k+1}) = qr(H_k - u_k I) \text{ (factorisation QR)} \\ &H_{k+1} = R_{k+1} Q_{k+1} + u_k I \end{aligned} \right. \end{aligned} \tag{5}$$

Le paramètre u_k est appelé « décalage » ou *shift* en anglais. Si on choisit pour u_k une valeur propre réelle de H_k , supposée irréductible, alors on peut montrer que le dernier bloc diagonal 2×2 de H_{k+1} est de la forme $\begin{pmatrix} * & * \\ 0 & \mu_k \end{pmatrix}$. Dans ce cas, on a « déflaté »

d'une unité la matrice en l'isolant à la dernière ligne. Cette remarque est théorique puisque c'est justement la valeur propre que l'on cherche à calculer. L'idée est de chercher à se rapprocher d'une valeur propre au fur et à mesure de la convergence et quand une valeur propre a convergé, on réduit la matrice par le bas d'une unité. Pour définir le *shift* μ_k , on peut choisir le dernier élément de la dernière ligne (stratégie du simple *shift*). En fait, les valeurs propres pouvant être complexes, on aimerait décaler avec les valeurs propres (λ_1, λ_2) du dernier bloc 2×2 (noté B) de la matrice H_k en les utilisant comme *shifts* dans deux par successifs :

$$\begin{aligned} (Q_{k+1}, R_{k+1}) &= qr (H_k - \lambda_1 I) \text{ (factorisation QR)} \\ H_{k+1} &= R_{k+1} Q_{k+1} + \lambda_1 I \\ (Q_{k+2}, R_{k+2}) &= qr (H_{k+1} - \lambda_2 I) \text{ (factorisation QR)} \\ H_{k+2} &= R_{k+2} Q_{k+2} + \lambda_2 I \end{aligned}$$

Si les valeurs propres sont complexes, pour garder les calculs en arithmétique réelle, on applique implicitement les deux *shifts* conjugués grâce aux relations suivantes :

$$(H_k - \lambda_2 I) (H_k - \lambda_1 I) = Q_{k+1} Q_{k+2} R_{k+2} R_{k+1} \quad (6)$$

et
$$(H_k - \lambda_2 I) (H_k - \lambda_1 I) = H_k^2 - \tau H_k + \delta I \quad (7)$$

où τ et δ sont respectivement la trace et le déterminant du bloc B . Autrement dit, la transformation orthogonale $Q_{k+1} Q_{k+2}$ qui permet de passer directement de H_k à H_{k+2} pourrait être calculée à partir de la factorisation QR de la matrice réelle $H_k^2 - \tau H_k + \delta I$. En fait, une application astucieuse du théorème 7 permet de faire un calcul équivalent avec une complexité de calcul $O(n^2)$ au lieu de $O(n^3)$ [14].

3. Algorithmes pour le cas d'une matrice pleine symétrique

3.1 Réduction à la forme tridiagonale

Dans ce paragraphe, on adapte la transformation orthogonale, introduite dans le paragraphe 2.1 dans le cas où la matrice de départ est symétrique. Toute transformation orthogonale $Q^T A Q$ de la matrice A reste alors symétrique. La matrice finale obtenue par le procédé est donc à la fois Hessenberg supérieure et symétrique : c'est une matrice tridiagonale symétrique.

Tout au long de l'application des transformations de Householder, on peut tenir compte de la symétrie pour diminuer le nombre d'opérations à exécuter. On peut en particulier le faire en utilisant la procédure de la bibliothèque BLAS qui met à jour une matrice avec une correction de rang 2. C'est ce qui est programmé dans la procédure DSYTRD de la bibliothèque LAPACK [1].

Lorsque la matrice de transformation Q qui accumule les transformations de Householder successives est nécessaire, on procède exactement comme dans le cas non symétrique.

3.2 Algorithmes pour le problème tridiagonal symétrique

3.2.1 Matrices irréductibles et suites de Sturm

Dans ce paragraphe, on considère la matrice tridiagonale symétrique d'ordre n notée $T = [\beta_j, \alpha_j, \beta_{j+1}]_{1,n}$ c'est-à-dire telle que

$T_{i,i} = \alpha_i$ pour $i = 1, \dots, n$ et $T_{i,i-1} = \beta_i$ pour $i = 2, \dots, n$. Pour tout $k = 1, \dots, n$, on note $p_k = \det(T_k - \lambda I_k)$ le polynôme caractéristique de la matrice principale T_k d'ordre k de la matrice T .

On suppose que la matrice T est irréductible, c'est-à-dire que $\prod_{k=2}^n \beta_k \neq 0$. En effet, si elle ne l'est pas alors la matrice T est constituée d'au moins deux blocs diagonaux qui sont des matrices tridiagonales symétriques et le problème aux valeurs propres se réduit à l'étude des problèmes aux valeurs propres des blocs diagonaux.

Proposition 4

La suite des polynômes caractéristiques des matrices principales de la matrice T vérifie la relation de récurrence suivante :

$$\begin{cases} p_0(\lambda) = 1, p_1(\lambda) = \alpha_1 - \lambda \text{ et pour } k \geq 1 \\ p_{k+1}(\lambda) = (\alpha_{k+1} - \lambda) p_k(\lambda) - \beta_{k+1}^2 p_{k-1}(\lambda) \end{cases} \quad (8)$$

Preuve

♦ On démontre la propriété par récurrence. Elle est évidemment vraie pour $k = 1$. Pour le passage de l'ordre k à l'ordre $k + 1$, il suffit de développer $p_{k+1}(\lambda) = \det(T_{k+1} - \lambda I_{k+1})$ par rapport à la dernière ligne de la matrice. ♦

Corollaire 3

Toutes les valeurs propres d'une matrice tridiagonale symétrique irréductible sont simples.

On énonce maintenant le théorème qui va permettre effectivement de localiser les valeurs propres d'une matrice tridiagonale symétrique.

Théorème 8 : suites de Sturm

Pour tout $k = 1, \dots, n$ et tout $\lambda \in \mathbb{R}$, on note :

$$\text{sgn } p_k(\lambda) = \begin{cases} \text{signe de } p_k(\lambda) \text{ si } p_k(\lambda) \neq 0 \\ \text{signe de } p_{k-1}(\lambda) \text{ si } p_k(\lambda) = 0 \end{cases} \quad (9)$$

Alors le nombre $N(n, \lambda)$ de changements de signes entre éléments consécutifs de la suite :

$$(\text{sgn } p_k(\lambda))_{(k=0, \dots, n)}$$

est égal au nombre de valeurs propres de la matrice T qui sont strictement inférieures à λ .

Preuve

♦ Voir par exemple [7] p. 122. ♦

On a donc ainsi un moyen de déterminer le nombre de valeurs propres qui sont incluses dans un intervalle donné $[a, b[$ en calculant le nombre $N(n, b) - N(n, a)$. Par bisection, on peut ensuite encadrer toutes les valeurs propres de l'intervalle. Cet algorithme est très peu coûteux en opérations, puisque l'évaluation de $N(n, \lambda)$ se fait en $O(n)$ opérations. Afin d'éviter les phénomènes d'*underflow* ou d'*overflow*, on préfère utiliser la suite des quotients

$q_k(\lambda) = \frac{p_k(\lambda)}{p_{k-1}(\lambda)}$ définie de $k = 1$ à $k = n$. Le nombre $N(n, \lambda)$ est alors égal aux nombres de termes négatifs de la suite. On peut montrer que la suite des $q_k(\lambda)$ est en fait la suite des termes diagonaux de la matrice U de la factorisation LU de la matrice $T - \lambda I$ et que $p_k(\lambda) = \prod_{i=1}^k q_i(\lambda)$.

Pour calculer les vecteurs propres associés aux valeurs propres calculées par les suites de Sturm, on utilise l'algorithme 2, algo-

rithme de la puissance inverse. Dans le cas de valeurs propres voisines, il est nécessaire d'assurer l'orthogonalité des vecteurs propres entre eux. Cela est fait par l'utilisation du **procédé de Gram-Schmidt modifié** (MGS). Une technique plus sophistiquée a été introduite par Dhillon et Parlett [11].

3.2.2 Algorithme QR

On décrit dans ce paragraphe l'adaptation de la méthode QR au cas particulier d'une matrice tridiagonale symétrique.

Proposition 5

Soit $T \in \mathbb{R}^{n \times n}$ une matrice tridiagonale symétrique et $\lambda \in \mathbb{R}$. Soit Q et R des matrices carrées d'ordre n , respectivement orthogonale et triangulaire supérieure, telles que $T - \lambda I = QR$, où I est la matrice de l'identité d'ordre n . Alors la matrice Q est une matrice Hessenberg supérieure et la matrice $T_1 = RQ + \lambda I$ est une matrice tridiagonale symétrique orthogonalement semblable à la matrice T .

L'algorithme QR consiste à enchaîner itérativement la transformation qui est décrite dans l'itération (5), où la matrice de Hessenberg est ici tridiagonale symétrique. Pour accélérer la convergence, on procède comme dans le cas non symétrique par un décalage. Les choix classiques sont de choisir $\lambda_k = (T_k)_{nn}$ ou plus souvent la valeur propre du dernier bloc diagonal de dimension 2 de la matrice T_k qui est la plus proche de $(T_k)_{nn}$.

La convergence de l'algorithme signifie que la matrice T_k devient diagonale à la limite, c'est-à-dire que $\lim_{k \rightarrow \infty} \text{Off}(T_k) = 0$. Il a été prouvé que les deux décalages indiqués précédemment entraînaient une convergence cubique à la limite.

Comme dans le cas non symétrique, la mise en œuvre se fait de manière implicite. Pour plus de détail, on peut consulter [14] p. 421.

4. Bibliothèque LAPACK

La diagonalisation des matrices pleines est maintenant complètement maîtrisée, même si des améliorations sont encore possibles dans la garantie de la précision, ou dans la parallélisation des méthodes.

La bibliothèque LAPACK du site Netlib est un **logiciel libre** qui rassemble les procédures pour résoudre la plupart des problèmes d'algèbre linéaire sur matrices pleines ou bande. Elle contient donc des procédures sur toutes les méthodes décrites ici.

Un soin particulier a été apporté à sa vitesse d'exécution puisqu'elle a systématiquement recours aux versions blocs afin d'optimiser la gestion de la mémoire. Son deuxième point fort est la qualité numérique, puisque la stabilité des méthodes est prouvée et l'erreur rétrograde souvent disponible comme l'est aussi le conditionnement du problème. Le manuel de l'utilisateur est disponible en ligne [1]. Cette bibliothèque devrait être systématiquement installée sur tous les postes dédiés au calcul numérique.

5. Méthodes pour les matrices de grande taille

La plupart des méthodes pratiques utilisées pour résoudre les problèmes de grande taille sont souvent une combinaison de techniques de projection, méthodes de déflation et de redémarrage et techniques de préconditionnement. Parmi les méthodes les plus utilisées, on peut citer :

– la **méthode des itérations simultanées**, qui est une méthode de base, robuste, et relativement lente, mais qui peut jouer un rôle par exemple pour valider les résultats des autres approches ;

– la **méthode d'Arnoldi (ou Lanczos)** avec accélération polynomiale, comme par exemple, la méthode implémentée dans le code ARPACK [18] ;

– les **méthodes du type décalage et inversion** (*shift-and-invert*) qui en général utilisent les méthodes directes de factorisation pour accélérer la convergence. Ce genre de méthodes est implémenté par exemple dans le code NASTRAN [16] ;

– les **variantes des méthodes du type Davidson et Davidson généralisé** qui exploitent les préconditionnements.

Parmi les méthodes émergentes, on pourra mentionner la méthode AMLS (*Automatic Multilevel Substructuring*) qui a été spécifiquement développée pour les problèmes de structure [3].

5.1 Méthodes de projection pour les problèmes de valeurs propres

On se place dans ce paragraphe dans le corps des complexes, mais les procédés décrits ont leur version réelle avec des techniques particulières lorsque la matrice réelle a des couples de valeurs propres conjuguées.

La formulation générale des techniques de projection part de la donnée de deux sous-espaces K et L de même dimension m . On cherche alors une valeur propre approchée $\tilde{\lambda} \in \mathbb{C}$ et un vecteur approché $\tilde{u} \in K$ comme solutions du problème projeté :

$$(\tilde{\lambda}I - A)\tilde{u} \perp L \quad (10)$$

Comme il y a m degrés de liberté et que la condition d'orthogonalité (10) impose m contraintes, on voit facilement que l'équation (10) mène à un problème de valeurs propres de taille m .

On distingue deux types de méthodes. Les méthodes de type projection orthogonale qui consistent à prendre $L = K$ et les méthodes de type projection oblique où $K \perp L$.

5.2 Projection de Rayleigh-Ritz

On se donne un sous-espace \mathcal{X} engendré par les colonnes de la matrice X (de taille $n \times m$) et on suppose que ce sous-espace contient de bonnes approximations des vecteurs propres que l'on cherche à calculer. La question est alors de savoir comment extraire ces approximations. La réponse est le procédé de projection de Rayleigh-Ritz, qui réalise une méthode de projection orthogonale sur l'espace \mathcal{X} . Ce procédé construit d'abord une base orthonormale $Q = [q_1, \dots, q_m]$ de \mathcal{X} avec un algorithme d'orthogonalisation appliqué à X . On calcule ensuite une approximation d'un vecteur propre donné sous la forme $\tilde{u} = Q\gamma$ où γ est un vecteur de \mathbb{C}^m . On obtient le vecteur γ en écrivant la **condition de Galerkin** (10) qui se traduit par :

$$Q^H (A - \tilde{\lambda}I)\tilde{u} = 0$$

ce qui donne, en se rappelant que $\tilde{u} = Q\gamma$, le problème aux valeurs propres :

$$Q^H A Q \gamma = \tilde{\lambda} \gamma$$

Algorithme 4 – Procédure de Rayleigh-Ritz

Calculer une base orthonormale $Q = [q_1, \dots, q_m]$ de \mathcal{X}
 Calculer $C = Q^H A Q$ (une matrice $m \times m$)
 Obtenir la factorisation de Schur de $C : C = Y R Y^H$
 Calculer $\tilde{U} = Q Y$

Théorème 9

Si le sous-espace \mathcal{X} est (exactement) invariant alors les valeurs propres et vecteurs propres calculés par le procédé de Rayleigh-Ritz sont exacts.

Preuve

◆ Puisque \mathcal{X} est invariant, et puisque $\tilde{u} \in K, (A - \tilde{\lambda}I)\tilde{u} = Qz$ pour un certain $z \in \mathbb{C}^m$. La contrainte de Galerkin (10) implique que $Q^H(A - \tilde{\lambda}I)\tilde{u} = 0$ et donc $Q^H Qz = 0$, ce qui donne $z = 0$. Par conséquent, $(A - \tilde{\lambda}I)\tilde{u} = 0$. ◆

La procédure de Rayleigh-Ritz joue un rôle primordial dans les méthodes de calcul de valeurs propres et vecteurs propres de matrices de grande taille. Par exemple, c'est l'outil principal utilisé par la méthode des itérations simultanées. Avant de la définir, nous considérons le cas d'un sous-espace invariant approché.

Supposons que l'on connaisse une base orthonormée $Q \in \mathbb{C}^{n \times k}$ d'un sous-espace invariant \mathcal{X} de \mathbb{R}^n de dimension k . On a donc les relations :

$$\forall x \in \mathcal{X}, \exists y \in \mathbb{C}^k, \text{ tel que } x = Qy \text{ et } Q^H Q = I_k$$

On complète la base Q par $\tilde{Q} \in \mathbb{C}^{n \times (n-k)}$ tel que la matrice $U = [Q, \tilde{Q}]$ forme une base orthonormée de l'espace \mathbb{C}^n . La matrice $U \in \mathbb{C}^{n \times n}$ est donc une matrice unitaire. La matrice de l'opérateur exprimé dans la base U est alors :

$$U^H A U = \begin{pmatrix} Q^H A Q & Q^H A \tilde{Q} \\ 0 & \tilde{Q}^H A \tilde{Q} \end{pmatrix} \quad (11)$$

car le bloc $\tilde{Q}^H A Q$ est nul puisque les colonnes de AQ appartenant au sous-espace invariant \mathcal{X} sont orthogonales aux colonnes de \tilde{Q} (lorsque la matrice A est hermitienne alors le bloc $Q^H A \tilde{Q}$ est aussi nul). La forme triangulaire par blocs de la matrice obtenue dans (11) montre que l'ensemble des valeurs propres de A est égal à la réunion des k valeurs propres de $Q^H A Q$ et des $n - k$ valeurs propres de $\tilde{Q}^H A \tilde{Q}$. La matrice $H = Q^H A Q \in \mathbb{C}^{k \times k}$ peut être vue comme la matrice de la restriction de l'opérateur au sous-espace \mathcal{X} et exprimée dans la base Q .

On suppose maintenant que le sous-espace \mathcal{X} n'est *a priori* plus invariant. Dans ce cas, le bloc $\tilde{Q}^H A Q$ n'est plus nul. On peut alors décomposer la matrice AQ sur la somme directe du sous-espace \mathcal{X} et de son orthogonal \mathcal{X}^\perp par :

$$AV = QQ^H(AQ) + (I - QQ^H)(AQ) \quad (12)$$

$$= QH + R \quad (13)$$

où $R = AQ - QH = (I - QQ^H)(AQ)$. On remarque que QQ^H et $(I - QQ^H)$ sont respectivement les matrices des projections orthogonales sur \mathcal{X} et sur \mathcal{X}^\perp . Pour un sous-espace \mathcal{X} presque invariant, la norme $\|R\|_2$ est presque nulle. On a ainsi une mesure de la qualité de l'approximation du sous-espace invariant calculé. Dans le cas d'une matrice hermitienne, le théorème suivant donne même des encadrements des valeurs propres calculées.

Théorème 10

Soit $A \in \mathbb{C}^{n \times n}$ une matrice hermitienne, de valeurs propres $(\lambda_j)_{j=1, \dots, n}$. Soit $Q \in \mathbb{C}^{n \times k}$ tel que $Q^H Q = I_k$. Si on note $(\mu_j)_{j=1, \dots, k}$ les valeurs propres de la matrice $H = Q^H A Q$, alors il existe k valeurs propres $(\lambda_{i_j})_{j=1, \dots, k}$ de A telles que :

$$|\lambda_{i_j} - \mu_j| \leq \|R\|_2$$

où $R = AQ - QH$.

Lorsque l'on a obtenu un sous-espace approximativement invariant, on calcule alors des approximations des valeurs et vecteurs propres correspondants à partir des valeurs et vecteurs de Ritz.

Définition 3

Soit $A \in \mathbb{R}^{n \times n}$ et $Q \in \mathbb{R}^{n \times k}$ une base orthonormée du sous-espace \mathcal{X} de dimension k . Les valeurs de Ritz de la matrice A correspondant au sous-espace \mathcal{X} sont les valeurs propres de la matrice $H = Q^T A Q$. Soit $y \in \mathbb{R}^k$ un vecteur propre normé correspondant à une valeur propre μ de H . Alors le vecteur normé $x = Qy$ est un vecteur de Ritz associé à la valeur de Ritz μ qui correspond à des quotients de Rayleigh :

$$\mu = \rho_A(x) = \rho_H(y)$$

Le calcul du résidu d'une paire de Ritz s'écrit facilement :

$$\begin{aligned} r &= Ax - \mu x \\ &= AQy - \mu Qy \\ &= Q(Hy - \mu y) + Ry \\ &= Ry \end{aligned}$$

et sa norme vérifie donc $\|r\|_2 = \|Ry\|_2 \leq \|R\|_2$.

5.3 Méthodes des itérations simultanées

La méthode des itérations simultanées (ou méthode d'itérations de sous-espaces) est une méthode qui converge relativement lentement mais qui est caractérisée par une excellente robustesse. Pour cette raison, elle est souvent utilisée pour valider les résultats obtenus par d'autres procédés. Le principe de la méthode est de générer une suite de sous-espaces qui deviennent progressivement invariants. Sous sa plus simple forme, la méthode peut être vue comme une généralisation de la méthode de la puissance.

L'idée originale est simplement d'effectuer un procédé de Rayleigh-Ritz sur le sous-espace engendré par $X_k = A^k X_0$ où X_0 est une base d'un certain sous-espace de départ. Cependant, il est clair qu'il n'y a aucune raison de se limiter au choix du polynôme $p_k(t) = t^k$ pour générer un bon espace pour la projection. En pratique et dans le cas symétrique, ce polynôme est remplacé par un polynôme de type Tchebychev, donc A^k est remplacé par $C_k(\alpha A + \beta I)$, où C_k est le **polynôme de Tchebychev** de première espèce de degré k et où α, β sont deux scalaires qui ont pour rôle de transformer les valeurs propres que l'on ne veut pas calculer dans l'intervalle $[-1, 1]$. Dans la description de l'algorithme, on ne précise pas le choix de ce polynôme.

Un des avantages principaux de l'algorithme des itérations simultanées est la simplicité de son implémentation, surtout dans le cas hermitien. Il est également relativement facile à analyser. Son inconvénient principal est la lenteur de sa convergence. Cette convergence est accélérée si on choisit bien le polynôme p_k .

Le théorème suivant suppose que les valeurs propres sont ordonnées par module décroissant et que :

$$|\lambda_1| \geq |\lambda_2| \geq \dots \geq |\lambda_{m-1}| \geq |\lambda_m| > |\lambda_{m+1}| \geq |\lambda_{m+2}| \geq \dots \geq |\lambda_n|$$

Algorithme 5 – Itérations simultanées avec projection

- 1. Initialisation :** Choisir un système initial de vecteurs $X = [x_0, \dots, x_m]$ et un polynôme initial p_k .
- 2. Itération :**
Calculer $\hat{Z} = p_k(A)X$
- 3. Projection :** Rayleigh-Ritz
Orthonormaliser \hat{Z} pour obtenir Z .
Calculer $B = Z^H AZ$ et obtenir les vecteurs de Schur $Y = [y_1, \dots, y_m]$ de B .
Calculer $X := ZY$.
Tester la convergence.
Si la convergence est atteinte - arrêter
sinon sélectionner un nouveau polynôme p_k .
aller à 2.

Théorème 11

Soit $S_0 = \text{sev}(X_0)$, sous-espace vectoriel engendré par les colonnes de $X_0 = \{x_1, x_2, \dots, x_m\}$, et supposons que X_0 est tel que les vecteurs $\{P_X\}_{i=1, \dots, m}$ sont linéairement indépendants où P est le projecteur spectral associé à $\lambda_1, \dots, \lambda_m$. Soit, \mathcal{P}_k le projecteur orthogonal sur l'espace $S_k = \text{sev}(X_k)$. Alors, pour chaque vecteur propre u_i de A , $i = 1, \dots, m$, il existe un vecteur unique s_j dans le sous-espace S_0 tel que $Ps_j = u_i$. De plus, on a l'inégalité suivante :

$$\|(I - \mathcal{P}_k)u_i\|_2 \leq \|u_i - s_j\|_2 \left(\left| \frac{\lambda_{m+1}}{\lambda_i} \right| + \varepsilon_k \right)^k \tag{14}$$

où ε_k tend vers zéro quand k tend vers l'infini.

Pour une démonstration, on consultera [26].

Le résultat établit simplement que sous certaines conditions assez faibles il existera une suite de vecteurs propres approchés dans l'espace S_k qui est l'espace utilisé par le procédé de Rayleigh-Ritz.

5.4 Algorithme d'Arnoldi

Dans beaucoup de méthodes itératives qui s'appliquent à une matrice carrée A de très grande taille n et creuse, on a recours aux espaces de Krylov. Ces espaces sont définis par, outre la matrice, un vecteur initial v et un degré k ; l'**espace de Krylov** :

$$K_k(A, v) = \text{sev}(v, Av, A^2v, \dots, A^{k-1}v)$$

est le sous-espace engendré par les puissances successives de A appliquées au vecteur v . Ce sous-espace est donc celui des images des polynômes de A de degré inférieur ou égal à $k - 1$ appliqués à v . On démontre facilement que si la dimension de ce sous-espace est strictement inférieure à k alors le sous-espace est invariant par A . Inversement, si k est supérieur au degré du polynôme minimal, alors le sous-espace est invariant par A . Nous nous plaçons maintenant dans le cas où $\dim(K_k(A, v)) = k$. On exhibe un algorithme qui va construire une base orthonormée de $K_k(A, v)$. À cette fin, on applique l'algorithme de Gram-Schmidt modifié, non au système $(v, Av, A^2v, \dots, A^{k-1}v)$ mais au système $(v_1, Av_1, Av_2, \dots, Av_{k-1})$ où les vecteurs $(v_i)_{i=1, k-1}$ sont les vecteurs du système orthonormé trouvé à l'étape $k - 1$. Ces vecteurs sont donc calculés de proche en proche. Il est évident que si (v_1, \dots, v_{k-1}) engendrent toutes les images par des polynômes de degré inférieur ou égal à $k - 2$, le système $(v_1, Av_1, Av_2, \dots, Av_{k-1})$ engendre toutes les images par

des polynômes de degré inférieur ou égal à $k - 1$. On obtient donc bien une base orthonormée de $K_k(A, v)$. La factorisation QR réalisée par l'algorithme de Gram-Schmidt peut s'écrire matriciellement :

$$[v_1; Av_1; \dots; Av_{k-1}] = [v_1; v_2; \dots; v_k] [e_1; \overline{H}_{k-1}]$$

où e_1 est le premier vecteur de la base canonique de \mathbb{R}^k et \overline{H}_{k-1} est une matrice de Hessenberg à k lignes et $k - 1$ colonnes. En notant $V_k = [v_1; v_2; \dots; v_k]$ on obtient la **relation fondamentale d'Arnoldi** :

$$\begin{aligned} AV_{k-1} &= V_k \overline{H}_{k-1} \\ &= V_{k-1} H_{k-1} + h_{k, k-1} v_k e_{k-1}^T \end{aligned}$$

où H_{k-1} est la matrice carrée constituée des $k - 1$ premières lignes de \overline{H}_{k-1} . On en déduit aussi la relation :

$$V_{k-1}^T AV_{k-1} = H_{k-1} \tag{15}$$

L'algorithme d'Arnoldi qui construit le couple $(V_m, \overline{H}_{m-1})$ s'écrit :

Algorithme 6 – Arnoldi

```

v1 = (1/||v||)v ;
for k = 1 : m - 1,
    w = Avk ;
    for i = 1 : k
        h_i,k = v_i^T w ;
        w = w - h_i,k v_i ;
    end ;
    h_{k+1,k} = ||w|| ;
    v_{k+1} = (1/h_{k+1,k})w ;
end
    
```

Il met en œuvre du calcul vectoriel (sur des vecteurs de longueur n) et des multiplications de la matrice (supposée creuse) par des vecteurs.

Soit \overline{H}_m la matrice de taille $(m + 1) \times m$ qui contient les éléments h_{ij} générés par l'algorithme, complétée par des zéros (en position i, j avec $i > j + 1$). On définit également la matrice H_m obtenue de \overline{H}_m en supprimant sa dernière ligne. Ces matrices ont les structures suivantes quand $m = 5$:

$$\overline{H}_m = \begin{pmatrix} \star & \star & \star & \star & \star \\ \star & \star & \star & \star & \star \\ & \star & \star & \star & \star \\ & & \star & \star & \star \\ & & & \star & \star \\ & & & & \star \end{pmatrix} \quad H_m = \begin{pmatrix} \star & \star & \star & \star & \star \\ \star & \star & \star & \star & \star \\ & \star & \star & \star & \star \\ & & \star & \star & \star \\ & & & \star & \star \end{pmatrix}$$

On a ainsi trois propriétés immédiates :

- $V_m = [v_1, v_2, \dots, v_m]$ est une base orthonormée de \mathbb{K}_m ;
- $AV_m = V_{m+1} \overline{H}_m = V_m H_m + h_{m+1, m} v_{m+1} e_m^T$;
- $V_m^T AV_m = H_m$.

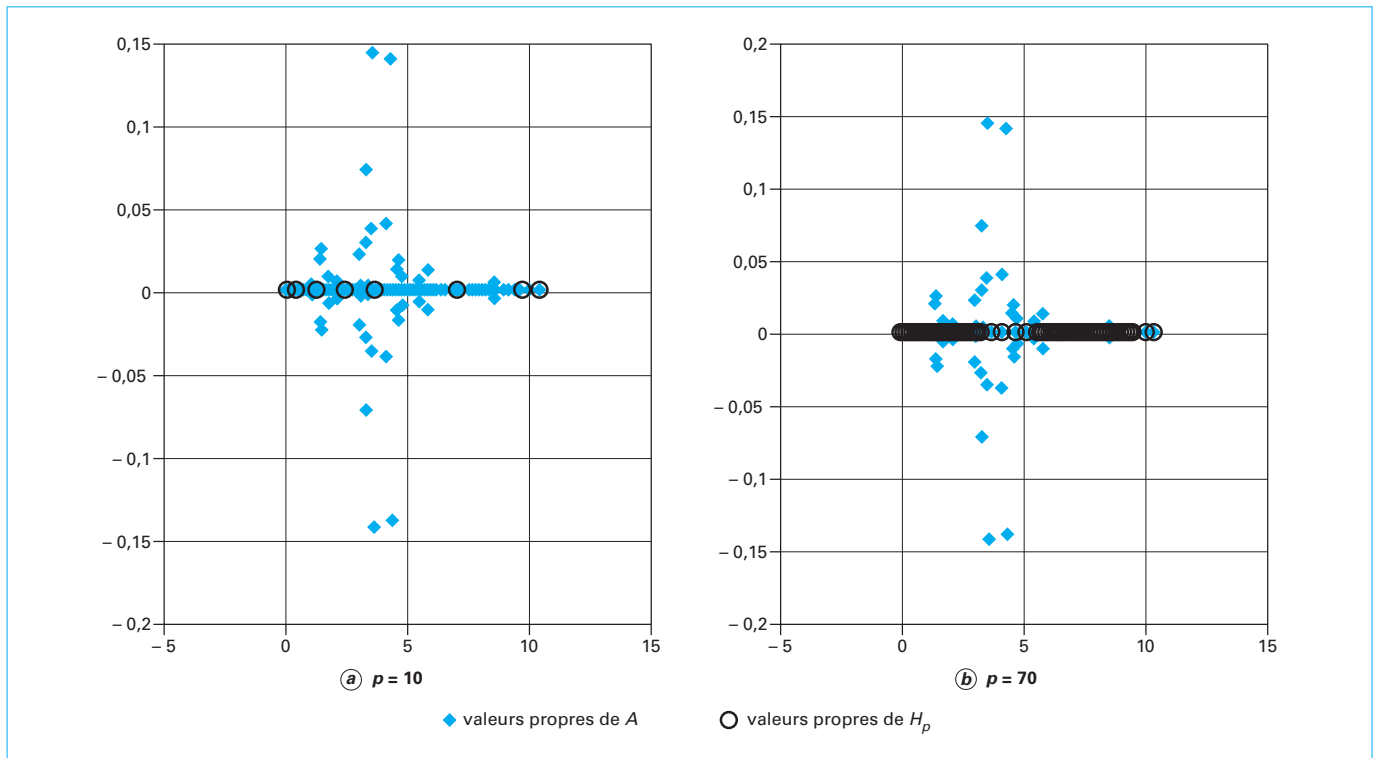


Figure 2 – Approximation des valeurs propres de la matrice BFW398A par le procédé d’Arnoldi pour des dimensions de sous-espaces de 10 et 70

Exprimons le vecteur propre approché dans la base V_m comme $\tilde{u} = V_m \gamma$ et écrivons la **condition de Galerkin** :

$$(A - \tilde{\lambda}I)V_m \gamma \perp \mathbb{K}_m \rightarrow V_m^H (A - \tilde{\lambda}I)V_m \gamma = 0$$

Les valeurs propres sont des valeurs propres de H_m :

$$H_m \gamma_j = \tilde{\lambda}_j \gamma_j$$

Les vecteurs propres approchés associés sont donnés par :

$$\tilde{u}_j = V_m \gamma_j$$

On observe en pratique que quelques-unes des valeurs propres les plus à l’extérieur du spectre convergeront en premier. Pour l’illustrer, considérons l’application du procédé d’Arnoldi sur la matrice BFW398A, matrice de l’ensemble Matrix Market [4]. Sur la figure 2, on a reporté les valeurs propres et les valeurs de Ritz pour deux dimensions de sous-espace.

5.5 Algorithme d’Arnoldi avec redémarrage

En pratique, les exigences en mémoire de l’algorithme font qu’il ne peut pas être déroulé sur un grand nombre de pas m et il faudra donc le *redémarrer*. L’algorithme 7 a pour but de calculer la valeur propre qui a la plus petite partie réelle (au sens algébrique), donc $\text{Re}(\lambda_1) = \min\{\text{Re}(\lambda_j), j = 1, 2, \dots, n\}$.

Algorithme 7 – Arnoldi avec redémarrage

1. **Init.** : sélectionner un vecteur initial v_1 et une dimension m .
2. **Itération** : Réaliser m pas de l’algorithme d’Arnoldi.
3. **Redémarrer** :
Calculer le vecteur propre approché $u_1^{(m)}$ associé à la valeur propre $\lambda_1^{(m)}$.
4. Si la paire $\lambda_1^{(m)}, u_1^{(m)}$ a convergé, arrêter,
sinon définir $v_1 \equiv u_1^{(m)}$
aller à 2.

Tableau 1 – Algorithme d'Arnoldi avec redémarrage pour calculer le vecteur propre associé à la valeur propre de plus grande partie réelle, qui est égale à 1 dans ce cas

Itér.	Re (λ)	Im (λ)	Norme résiduelle
1	0,998 743 589 9	0,0	$0,246 \cdot 10^{-1}$
2	0,999 952 332 4	0,0	$0,144 \cdot 10^{-2}$
3	$0,100 000 036 8 \cdot 10^1$	0,0	$0,221 \cdot 10^{-4}$
4	$0,100 000 002 5 \cdot 10^1$	0,0	$0,508 \cdot 10^{-6}$
5	0,999 999 999 6	0,0	$0,138 \cdot 10^{-7}$

Pour illustration, le tableau 1 montre un petit exemple correspondant à une matrice de chaîne de Markov qui simule une marche aléatoire sur une grille triangulaire de 55 nœuds (disposés sur une grille de 10×10).

L'algorithme d'Arnoldi avec redémarrage est utilisé pour calculer le vecteur propre associé à la valeur propre de plus grande partie réelle, qui est égale à 1 dans ce cas. La dimension de l'espace de Krylov est $m = 10$. Dans le tableau, la colonne Itér. correspond au nombre de redémarrage (passage au pas 2 de l'algorithme).

On peut également généraliser l'algorithme pour calculer plus d'un vecteur à la fois en prenant par exemple pour nouveau v_1 en ligne 4, une combinaison linéaire de vecteurs propres sélectionnés. Cependant, cette approche ne fonctionne pas bien en pratique car il est difficile de trouver une bonne combinaison linéaire. Il est bien meilleur de calculer les vecteurs propres (ou vecteurs de Schur) un par un et d'exploiter certaines formes de redémarrage et de déflation. On reviendra là-dessus dans les paragraphes 5.10 et 5.11.

5.6 Cas hermitien : algorithme de Lanczos

Lorsque A est hermitienne complexe (ou symétrique réelle), la matrice de Hessenberg construite par l'algorithme d'Arnoldi devient tridiagonale. En effet, puisque $V_m^H A V_m = H_m$ et que $A = A^H$ alors on a bien $H_m = H_m^H$ et puisque H_m est de Hessenberg, elle est nécessairement tridiagonale. Elle est même réelle (même dans le cas où A est complexe hermitienne) puisque les termes diagonaux d'une matrice hermitienne sont réels et que les termes $h_{i,i+1}$ sont des normes et donc réels aussi.

On changera de notation et on écrira H_m sous la forme :

$$H_m = \begin{pmatrix} \alpha_1 & \beta_2 & & & & \\ \beta_2 & \alpha_2 & \beta_3 & & & \\ & \beta_3 & \alpha_3 & \beta_4 & & \\ & & & \ddots & \ddots & \\ & & & & \beta_m & \alpha_m \\ & & & & \beta_m & \alpha_m \end{pmatrix} \quad (16)$$

Une conséquence importante de l'observation ci-avant est que les vecteurs v_j obéissent à une récurrence à trois termes :

$$\beta_{j+1} v_{j+1} = A v_j - \alpha_j v_j - \beta_j v_{j-1}$$

L'algorithme de Lanczos prend alors la forme :

Algorithme 8 – Lanczos

```
1. Choisir un vecteur initial  $v_1$  de norme unité.
Poser  $\beta_1 \int 0$ ,  $v_0 \int 0$ 
2. For  $j = 1, 2, \dots, m$  Do:
3.    $w_j := A v_j - \beta_j v_{j-1}$ 
4.    $\alpha_j := (w_j, v_j)$ 
5.    $w_j := w_j - \alpha_j v_j$ 
6.    $\beta_{j+1} := \|w_j\|_2$ . Si  $\beta_{j+1} == 0$  Arrêt
7.    $v_{j+1} := w_j / \beta_{j+1}$ 
8. EndDo
```

Les vecteurs construits par l'algorithme sont théoriquement orthogonaux puisque l'algorithme n'est rien d'autre qu'une réécriture (simplifiée certes, mais mathématiquement équivalente) de la méthode d'Arnoldi. En pratique, une perte d'orthogonalité très rapide prend place dès qu'un ou plusieurs vecteurs propres commencent à converger. Cette perte d'orthogonalité, constatée et analysée par Paige [21] [22] [23] est un signe de perte d'indépendance linéaire des vecteurs propres. Une fois que cette indépendance est perdue, il commence à apparaître plusieurs copies de la même valeur propre (et du vecteur propre correspondant), qui a convergé.

Plusieurs méthodes ont été développées pour remédier au problème de la perte d'orthogonalité. Ces méthodes de réorthogonalisation sont classées en plusieurs catégories :

- **méthodes de réorthogonalisation complète** : on réorthogonalise chaque v_{j+1} par rapport à tous les v_i précédents ;
- **méthodes de réorthogonalisation partielle** : on réorthogonalise v_{j+1} par rapport à tous les v_i précédents mais seulement quand c'est nécessaire ;
- **méthodes de réorthogonalisation sélective** : on réorthogonalise v_{j+1} par rapport aux vecteurs propres déjà calculés ;
- **méthodes sans réorthogonalisation** : au lieu de réorthogonaliser on prend des mesures pour trier et retirer les valeurs propres superflues qui apparaissent [9] [10].

5.7 Algorithme de biorthogonalisation de Lanczos

L'algorithme de Lanczos symétrique peut être généralisé au cas non symétrique appelé algorithme de biorthogonalisation de Lanczos.

Algorithme 9 – Procédure de biorthogonalisation de Lanczos

```
1. Choisir  $v_1$ ,  $w_1$  tel que  $(v_1, w_1) = 1$ .
Set  $\beta_1 = \delta_1 \int 0$ ,  $w_0 = v_0 \int 0$ 
2. For  $j = 1, 2, \dots, m$  Do:
3.    $\alpha_j = (A v_j, w_j)$ 
4.    $\hat{v}_{j+1} = A v_j - \alpha_j v_j - \beta_j v_{j-1}$ 
5.    $\hat{w}_{j+1} = A^H w_j - \alpha_j w_j - \delta_j w_{j-1}$ 
6.    $\delta_{j+1} = |(\hat{v}_{j+1}, \hat{w}_{j+1})|^{1/2}$ . Si  $\delta_{j+1} = 0$  Stop
7.    $\beta_{j+1} = (\hat{v}_{j+1}, \hat{w}_{j+1}) / \delta_{j+1}$ 
8.    $w_{j+1} = \hat{w}_{j+1} / \beta_{j+1}$ 
9.    $v_{j+1} = \hat{v}_{j+1} / \delta_{j+1}$ 
10. EndDo
```

En pratique, on préfère définir α_j en ligne 3 par $\alpha_j = (Av_j - \beta_j v_{j-1}, w_j)$ et de même, les lignes 4 et 5 peuvent être remplacées par $\hat{v}_{j+1} = (Av_j - \beta_j v_{j-1}) - \alpha_j v_j$ et $\hat{w}_{j+1} = (A^H w_j - \delta_j w_{j-1}) - \bar{\alpha}_j w_j$.

L'algorithme construit une paire de bases biorthogonales pour les deux sous-espaces :

$$\mathbb{K}_m(A, v_1) \text{ et } \mathbb{K}_m(A^H, w_1)$$

Il y a d'autres choix possibles pour $\delta_{j+1}, \beta_{j+1}$ en lignes 7 et 8, la seule contrainte étant que l'on maintienne la relation :

$$\delta_{j+1} \beta_{j+1} = (\hat{v}_{j+1}, \hat{w}_{j+1}).$$

Soit :

$$T_m = \begin{pmatrix} \alpha_1 & \beta_2 & & & & \\ \delta_2 & \alpha_2 & \beta_3 & & & \\ & & & \ddots & & \\ & & & & \delta_{m-1} & \alpha_{m-1} & \beta_m \\ & & & & & & \delta_m & \alpha_m \end{pmatrix}$$

Il est facile de démontrer que $v_i \in \mathbb{K}_m(A, v_1)$ et $w_j \in \mathbb{K}_m(A^H, w_1)$. Si l'algorithme ne s'arrête pas avant le pas m , alors les vecteurs $v_i, i = 1, \dots, m$, et $w_j, j = 1, \dots, m$, sont biorthogonaux, c'est-à-dire :

$$(v_j, w_i) = \delta_{ij} \quad 1 \leq i, j \leq m$$

De plus, $\{v_i\}_{i=1,2,\dots,m}$ forment une base de $\mathbb{K}_m(A, v_1)$ et $\{w_i\}_{i=1,2,\dots,m}$ forment une base de $\mathbb{K}_m(A^H, w_1)$; on a les relations suivantes :

$$\begin{aligned} AV_m &= V_m T_m + \delta_{m+1} v_{m+1} e_m^H \\ A^H W_m &= W_m T_m^H + \bar{\beta}_{m+1} w_{m+1} e_m^H \\ W_m^H AV_m &= T_m \end{aligned}$$

Si θ_j, y_j et z_j sont respectivement une valeur propre de T_m , et les vecteurs propres associés à droite et à gauche, alors les approximations correspondantes pour A sont :

- valeur de Ritz : θ_j ;
- vecteur de Ritz à droite : $V_m y_j$;
- vecteur de Ritz à gauche : $W_m z_j$.

Les expressions *valeur de Ritz* et *vecteur de Ritz* sont utilisées ici par abus de langage. En effet, dans le cas hermitien, les valeurs de Ritz appartiennent à l'enveloppe convexe (i.e. l'intervalle) des valeurs propres, alors qu'il n'en est rien ici puisque les valeurs de Ritz peuvent être arbitrairement éloignées des valeurs propres.

On peut comparer certaines caractéristiques de l'algorithme de biorthogonalisation décrit ci-avant avec la méthode d'Arnoldi. L'avantage principal de l'algorithme de biorthogonalisation est qu'il est fondé sur une récurrence à trois termes. Donc ce procédé est beaucoup moins gourmand en mémoire et en calcul. Un autre avantage est qu'il permet de calculer simultanément les vecteurs propres à gauche et à droite. Par contre, son inconvénient majeur est que la méthode peut mener à des divisions par zéro et donc à un arrêt prématuré de l'algorithme. Même si l'algorithme ne s'arrête pas, il se peut qu'il y ait des divisions par de petits nombres menant à des pertes de précision. Un autre inconvénient est que la convergence peut être assez irrégulière. Dernier inconvénient, il n'est pas facile d'exploiter la forme tridiagonale puisque l'algorithme QR , qui est l'algorithme stable à utiliser, augmentera la matrice jusqu'à la forme de Hessenberg.

Comme mentionné ci-avant, l'algorithme de biorthogonalisation échoue quand $(\hat{v}_{j+1}, \hat{w}_{j+1}) = 0$. On distingue trois situations :

- **arrêt souhaité** (en anglais *lucky breakdown*) : c'est le cas $\hat{v}_{j+1} = 0$ ou $\hat{w}_{j+1} = 0$. Dans ce cas, les valeurs propres de T_m sont des valeurs propres de A ;
- **échec sérieux** (en anglais *serious breakdown*) : $(\hat{v}_{j+1}, \hat{w}_{j+1}) = 0$ mais $\hat{v}_{j+1} \neq 0$, et $\hat{w}_{j+1} \neq 0$. Il est alors possible de contourner le pas qui provoque l'échec et de continuer l'algorithme. Si ceci n'est pas possible, alors on obtient le cas suivant ;
- **échec irréparable** (en anglais *incurable breakdown*). Cette situation est très rare.

L'algorithme appelé *look-ahead* (avec prévision) de Lanczos a été développé pour traiter le second cas. L'idée principale est que si un échec a lieu au pas j , on peut éviter de calculer les vecteurs v_{j+1}, w_{j+1} et chercher à obtenir v_{j+2}, w_{j+2} directement à partir de v_j, w_j . L'idée la plus simple serait simplement d'orthogonaliser le vecteur $A^2 v_j$ par rapport aux vecteurs w_j, w_{j-1}, w_{j-2} . Le vecteur v_{j+1} peut être défini, par exemple, comme $v_{j+1} = Av_j$. On procède de manière similaire pour w_{j+1} .

Si ce calcul échoue à son tour, on peut tenter d'avoir v_{j+3}, w_{j+3} et ainsi de suite. Si ce procédé réussit après quelques pas, on dit que l'échec est réparé. La matrice du problème projeté n'est plus tridiagonale, car il apparaît un bloc diagonal plein, de dimension égale au nombre de pas sautés augmenté d'une unité [5] [12].

5.8 Déflation

Les techniques de déflation sont très utiles en pratique et ont été développées sous des formes variées. Par exemple, une forme de déflation appelée *locking* – ou verrouillage – est utilisée avec l'algorithme des itérations simultanées. Considérons la forme canonique de Schur :

$$A = U R U^H$$

où U est une matrice complexe triangulaire supérieure. Les vecteurs-colonnes u_1, \dots, u_n sont appelés vecteurs de Schur. Les vecteurs de Schur dépendent les uns des autres et ne sont pas définis uniquement, même en direction. Ils dépendent de l'ordre du placement des valeurs propres sur la diagonale de R .

Dans la **déflation de Wiedlandt**, on suppose que l'on a calculé une valeur propre λ_1 et son vecteur propre à droite associé u_1 . On choisit alors un vecteur v tel que $(v, u_1) = 1$ et on considère les valeurs propres et vecteurs propres de la matrice déflatée :

$$A_1 = A - \sigma u_1 v^H$$

On commence par observer que les valeurs propres de la matrice sont :

$$\Lambda(A_1) = \{\lambda_1 - \sigma, \lambda_2, \dots, \lambda_n\}$$

Le procédé de déflation de Wiedlandt préserve u_1 comme vecteur propre à droite aussi bien que le vecteur propre de A à gauche associé à λ_1 .

Un choix intéressant pour v est de simplement prendre $v = u_1$. Dans ce cas, la déflation de Wiedlandt préserve tous les vecteurs de Schur puisque :

$$Q^H (A - \sigma u_1 u_1^T) Q = R - \sigma_1 e_1 e_1^T$$

Le choix de σ_1 dépendra de l'algorithme choisi. Cette déflation a été inventée pour la méthode de la puissance qui ne produit que la valeur propre λ_1 de plus grand module. Supposons que les valeurs propres de A sont numérotées par ordre de modules décroissants. Une fois λ_1 calculée, on opère la déflation avec $\sigma_1 = \lambda_1$, ce qui

remplace la valeur propre calculée par la valeur propre nulle et c'est alors λ_2 qui devient de plus grand module pour la matrice transformée.

On peut appliquer le procédé de Wiedlandt de manière successive, en déflétant de plus en plus de vecteurs calculés. Ainsi, à l'étape $j \geq 1$, on aura à calculer les valeurs propres de la matrice $A_{j+1} = A_j - \sigma_j u_j u_j^H$. Ce procédé est un **procédé de déflation explicite**. Il ne peut être appliqué avec trop de valeurs propres, car il perd en précision et devient coûteux.

Une approche alternative est d'employer un **procédé implicite de déflation**. Ce genre de méthode s'applique à une technique de projection telle que la méthode d'Arnoldi. Quand le premier vecteur propre désiré converge, on le *gèle* et on le place en première colonne (v_1) de la base $V_m = [v_1, v_2, \dots, v_m]$. Le procédé d'Arnoldi démarre maintenant à partir de la colonne v_2 . Les vecteurs calculés par l'algorithme au pas $j \geq 2$ sont alors orthogonalisés par rapport à v_1, \dots, v_j . Chaque fois qu'un vecteur propre converge, on l'ajoute à l'ensemble des vecteurs gelés.

Ainsi pour $k = 2$ on aura la situation suivante :

$$V_m = [v_1, v_2, v_3, \dots, v_m]$$

avec v_1, v_2 gelés et v_3, \dots, v_m actifs.

La matrice de Hessenberg correspondant à la situation décrite ci-avant aura la forme suivante ($m = 6$) :

$$H_m = \begin{pmatrix} \star & \star & \star & \star & \star & \star \\ & \star & \star & \star & \star & \star \\ & & \star & \star & \star & \star \\ & & & \star & \star & \star \\ & & & & \star & \star \\ & & & & & \star & \star \end{pmatrix}$$

Pour illustrer le procédé, on continue l'exemple vu dans le paragraphe 5.5, pour calculer les deux valeurs propres suivantes, dans l'ordre des plus grandes parties réelles. L'illustration précédente (tableau 1) a montré comment la procédure d'Arnoldi avec redémarrage a permis de calculer la valeur propre dominante et le vecteur propre à droite. Le tableau 2 montre comment le procédé se poursuit grâce à la déflation pour calculer les valeurs propres λ_2 et λ_3 . L'indice k dans la table est le numéro de la valeur propre en cours de calcul.

Tableau 2 – Algorithme d'Arnoldi avec redémarrage et déflation pour calculer les premières valeurs propres (ordonnées par partie réelle décroissante) et les vecteurs propres associés				
k	Mat-vects	Re (λ)	Im (λ)	Norme résiduelle
2	60	0,937 050 947 4	0,0	$0,870 \cdot 10^{-3}$
	69	0,937 154 961 7	0,0	$0,175 \cdot 10^{-4}$
	78	0,937 150 144 2	0,0	$0,313 \cdot 10^{-6}$
	87	0,937 150 156 4	0,0	$0,490 \cdot 10^{-8}$
3	96	0,811 224 713 3	0,0	$0,210 \cdot 10^{-2}$
	104	0,809 755 345 0	0,0	$0,538 \cdot 10^{-3}$
	112	0,809 641 948 3	0,0	$0,874 \cdot 10^{-4}$
	⋮	⋮	⋮	⋮
	⋮	⋮	⋮	⋮
	152	0,809 571 716 7	0,0	$0,444 \cdot 10^{-7}$

5.9 Méthodes de décalage et inversion

Le procédé de décalage-inversion (en anglais *shift-and-invert*) transforme le problème initial de valeurs propres de manière à accélérer la convergence de la méthode de calcul utilisée, ou même de rendre possible le calcul lorsque les valeurs propres visées sont internes au spectre, puisque les méthodes de sous-espace déterminent d'abord les valeurs propres périphériques. Ce procédé a déjà été introduit dans la méthode des itérations inverses (cf. § 1.3).

On remplace le problème initial par le problème associé à la matrice :

$$B = (A - \sigma I)^{-1} \tag{17}$$

Les valeurs propres de B sont $(\lambda_j - \sigma)^{-1}$ où les λ_j sont les valeurs propres de A . Les valeurs propres de A qui sont voisines de σ vont devenir très grandes par rapport aux autres et donc rejetées à l'extérieur du spectre.

L'idée est d'utiliser la méthode Arnoldi, de Lanczos, ou des itérations simultanées sur la matrice $B = (A - \sigma I)^{-1}$. Il est clair que l'on n'a pas besoin de calculer la matrice B explicitement mais, à chaque fois que l'on a besoin d'itérer avec B sur un vecteur, il faudra résoudre un système linéaire avec la matrice $A - \sigma I$. En pratique, cela veut dire que pour chaque nouveau décalage σ , on calculera une factorisation LU (voir la définition dans l'article *Méthodes numériques de base. Algèbre numérique* [AF 1 221]) de $A - \sigma I$ (ou LDL^T dans le cas symétrique). Les systèmes $(A - \sigma I)x = b$ successifs se résoudront alors par descente et remontée : $Lz = b$ et $Ux = z$.

Dans le cas non hermitien, il y aura des situations où A est réelle, mais où l'on voudra utiliser un décalage σ qui est complexe. Dans ce cas, on voudrait malgré tout limiter l'emploi de l'arithmétique complexe dans les procédures d'Arnoldi ou de Lanczos. Ceci est possible car, au lieu d'utiliser $B = (A - \sigma I)^{-1}$, on utilise l'une des alternatives suivantes :

$$B_+ = \text{Re}(A - \sigma I)^{-1} = \frac{1}{2} [(A - \sigma I)^{-1} + (A - \bar{\sigma} I)^{-1}] \tag{18}$$

$$B_- = \text{Im}(A - \sigma I)^{-1} = \frac{1}{2i} [(A - \sigma I)^{-1} - (A - \bar{\sigma} I)^{-1}] \tag{19}$$

En pratique, il n'y a pas une grande différence entre ces deux alternatives.

Proposition 6

On a la relation suivante :

$$B_- = \theta (A - \sigma I)^{-1} (A - \bar{\sigma} I)$$

avec $\theta = \text{Im}(\sigma)$.

Les méthodes de déplacement-inversion peuvent être considérées comme des techniques de **préconditionnement** pour les problèmes de valeurs propres. Le preconditionnement décalage-inversion préserve les vecteurs propres. D'autres méthodes considérées plus loin (voir les méthodes de Davidson, § 5.12) utilisent des preconditionneurs plus généraux.

5.10 Filtres polynomiaux

Un simple procédé d'accélération pour la méthode d'Arnoldi consiste à remplacer la définition du nouveau v_1 en ligne 4 de l'algorithme 7 par une affectation plus générale du type :

$$v_{1,\text{nouv}} = q_k(A)v_1$$

où q_k est un certain polynôme de degré k . Il est évident que cela représente une généralisation du cas précédent car l'espace de Krylov n'est autre qu'un sous-espace vectoriel de vecteurs de la forme ci-avant et donc en particulier $u_j^{(m)}$ qui est aussi de cette forme.

Le redémarrage, tel que défini ci-avant, est *explicite*, en ce sens qu'il applique explicitement un filtre polynomial $q_k(A)$ à un certain vecteur. On peut également appliquer le filtre de manière *implicite* dans le cadre d'une méthode de type Arnoldi ou Lanczos. Considérons le filtre polynomial :

$$p(t) = (t - \theta_1)(t - \theta_2) \dots (t - \theta_q)$$

et supposons que l'on a exécuté la procédure d'Arnoldi. En sortie, de l'itération 2, on a :

$$AV_m = V_m H_m + \beta_m v_{m+1} e_m^T \quad (20)$$

On considère d'abord le premier facteur $(t - \theta_1)$:

$$(A - \theta_1 I)V_m = V_m (H_m - \theta_1 I) + \beta_m v_{m+1} e_m^T$$

Soit $H_m - \theta_1 I = V_1 R_1$ la factorisation QR de $H_m - \theta_1 I$. Alors :

$$\begin{aligned} (A - \theta_1 I)V_m &= V_m Q_1 R_1 + \beta_m v_{m+1} e_m^T \Rightarrow \\ (A - \theta_1 I)(V_m Q_1) &= (V_m Q_1) R_1 Q_1 + \beta_m v_{m+1} e_m^T Q_1 \Rightarrow \\ A(V_m Q_1) &= (V_m Q_1)(R_1 Q_1 + \theta_1 I) + \beta_m v_{m+1} e_m^T Q_1 \end{aligned} \quad (21)$$

En utilisant les notations :

$$H_m^{(1)} \equiv R_1 Q_1 + \theta_1 I \quad (b_{m+1}^{(1)})^T \equiv e_m^T Q_1 \quad V_m^{(1)} \equiv V_m Q_1$$

la relation (21) devient alors :

$$AV_m^{(1)} = V_m^{(1)} H_m^{(1)} + v_{m+1} (b_{m+1}^{(1)})^T$$

Comme la matrice $H_m^{(1)}$ est de forme Hessenberg supérieure, on obtient une décomposition semblable à la décomposition d'Arnoldi. Dans cette relation, on a les propriétés suivantes :

- $H_m^{(1)} = R_1 Q_1 + \theta_1 I \equiv$ matrice résultant d'un pas de l'algorithme QR avec le décalage θ_1 appliqué à H_m .
- la première colonne de $V_m^{(1)}$ est un vecteur normé de direction $(A - \theta_1 I)v_1$;
- les colonnes de $V_m^{(1)}$ sont orthonormales.

On peut maintenant appliquer le second décalage et procéder de la même manière :

$$(A - \theta_2 I)V_m^{(1)} = V_m^{(1)} (H_m^{(1)} - \theta_2 I) + v_{m+1} (b_{m+1}^{(1)})^T$$

Le procédé est similaire : $(H_m^{(1)} - \theta_2 I) = Q_2 R_2$ alors on multiplie par Q_2 à droite :

$$(A - \theta_2 I)V_m^{(1)} Q_2 = (V_m^{(1)} Q_2) (R_2 Q_2) + v_{m+1} (b_{m+1}^{(1)})^T Q_2$$

et on aboutit à :

$$AV_m^{(2)} = V_m^{(2)} H_m^{(2)} + v_{m+1} (b_{m+1}^{(2)})^T$$

où la première colonne de $V_m^{(2)} = \text{scalaire} \times (A - \theta_2 I)v_1^{(1)}$
 $= \text{scalaire} \times (A - \theta_2 I)(A - \theta_1 I)v_1$

On remarque aussi que :

$$(b_{m+1}^{(2)})^T = e_m^T Q_1 Q_2 = [0, 0, \dots, 0, q_1, q_2, q_3]$$

Si $\widehat{V}_{m-2} = [\widehat{v}_1, \dots, \widehat{v}_{m-2}]$ rassemble les $m-2$ premières colonnes de $V_m^{(2)}$ et si \widehat{H}_{m-2} est la sous-matrice principale de H_m , alors :

$$\begin{aligned} A\widehat{V}_{m-2} &= \widehat{V}_{m-2} \widehat{H}_{m-2} + \widehat{\beta}_{m-1} \widehat{v}_{m-1} e_m^T \quad \text{avec} \\ \widehat{\beta}_{m-1} \widehat{v}_{m-1} &\equiv q_1 v_{m+1} + h_{m-1, m-2}^{(2)} v_{m-1}^{(2)} \quad \|\widehat{v}_{m-1}\|_2 = 1 \end{aligned}$$

On a donc obtenu une traduction du procédé d'Arnoldi à $m-2$ pas avec le vecteur initial $p(A)v_1$ via une combinaison du procédé d'Arnoldi, combiné avec l'algorithme QR avec décalage appliqué à la matrice de Hessenberg. Nous venons de montrer comment le procédé s'écrit pour un degré 2, mais la généralisation à un degré supérieur est évidente.

5.11 Redémarrage implicite

Le procédé de redémarrage implicite s'est imposé ces dernières années comme la méthode la plus couramment utilisée. Elle a été introduite par Lehoucq et Sorensen et est mise en œuvre dans la bibliothèque ARPACK [18].

On se place dans le cas du filtre polynomial implicite tel qu'il a été décrit dans la section précédente (§ 5.10). Les valeurs propres de la matrice H_m sont censées être des approximations de valeurs propres de A au moins pour certaines d'entre elles. L'objectif du procédé IRAM (*implicitly restarted Arnoldi method*) est de réduire la base V_m à une base \widetilde{V}_p qui engendre un sous-espace de celui

engendré par V_m et de calculer la matrice $\widetilde{H}_p = \widetilde{V}_p^T A \widetilde{V}_p$ correspondante en maintenant sa forme Hessenberg supérieure. Le procédé sera efficace si l'information utile dans V_m se trouve concentrée dans \widetilde{V}_p . L'astuce du procédé, décrit dans la section 5.10, consiste à appliquer des pas de l'algorithme QR avec décalage explicite à la matrice \widetilde{H}_p . Pour déterminer les $m-p$ *shifts*, on calcule toutes les valeurs propres de H_m et on détermine celles que l'on veut éliminer. Ce sont elles qui seront choisies comme *shifts*. L'application des $m-p$ transformations détermine une matrice orthogonale Q et une matrice de Hessenberg supérieure $\widetilde{H}_m = Q^T H_m Q$. On y choisit alors la matrice principale d'ordre p pour définir la matrice \widetilde{H}_p . La base \widetilde{V}_p et la définition de la quantité $\widetilde{\beta}_{p+1}$ et du vecteur \widetilde{v}_{p+1} qui vérifient la relation d'Arnoldi (20) à l'ordre p est obtenue grâce à la matrice Q . On est alors en position pour redémarrer l'algorithme d'Arnoldi pour compléter la base \widetilde{V}_p en une nouvelle base de dimension m .

Le procédé fonctionne bien car la méthode QR appliquée à une matrice de Hessenberg avec un *shift* égal à une valeur propre fait apparaître la valeur propre dans la dernière position diagonale et un zéro en dernière position de la sous-diagonale. Le procédé consiste donc à appliquer une déflation pour purifier le sous-espace des valeurs propres sans intérêt.

5.12 Approche de Davidson

La méthode de Davidson a été développée par des chimistes dans les années 1970 et peut être décrite comme une forme pré-conditionnée de l'algorithme de Lanczos. Le point de vue pris est le suivant. On a à chaque pas de l'algorithme un certain sous-espace V_m que l'on voudrait augmenter en ajoutant un vecteur à la base courante. La question est de savoir quel vecteur ajouter. Si on a une paire approchée $(\tilde{\lambda}, \tilde{u})$, on pourrait penser à ajouter simplement le résidu $r = (A - \tilde{\lambda}I)\tilde{u}$. Ce vecteur est en effet

orthogonal à \tilde{u} et représente en quelque sorte un choix tentant. Malheureusement, ce choix mènerait à un algorithme qui serait mathématiquement équivalent à l'algorithme de Lanczos. L'approche de Davidson consiste à *préconditionner* r avant de l'introduire dans l'espace. Cela veut dire que l'on transforme r en $t = M^{-1}r$ où M est une matrice qui a pour but d'éliminer les composantes de r qui sont indésirables et de garder les autres.

Si l'on avait le choix, le meilleur vecteur à introduire dans l'espace serait le vecteur propre lui-même ! En effet, avec un tel

choix, le vecteur propre cherché u serait calculé au pas suivant. On pourrait aussi penser au choix $M = (A - \tilde{\lambda} I)$ mais il est clair que le choix est sans intérêt puisque $M^{-1}r = \tilde{u}$ et que cela n'apporterait rien de supplémentaire à l'espace. Dans la version originale de l'algorithme, la matrice M était simplement $D - \tilde{\lambda} I$ où D est la diagonale de A . On considère en général des matrices qui possèdent plus d'information sur A . On retrouve là les mêmes questions que dans le choix d'un préconditionneur pour une matrice itérative de résolution de systèmes linéaires.

Algorithme 10 – Méthode de Davidson (cas symétrique réel)

```

1. Choisir un vecteur initial de norme 1  $v_1$ . Set  $V_1 = [v_1]$ .
2. Itérer :
3.   For  $j = 1, \dots, m$  Do:
4.      $w := AV_j$ .
5.     Mettre à jour  $H_j \equiv V_j^T AV_j$ 
6.     Calculer les plus petites valeurs propres et vect. propres  $\mu, y$  de  $H_j$ .
7.      $z := V_j y$   $r := Az - \mu z$ 
8.     Test de convergence. Si convergence atteinte arrêter
9.     si  $j < m$  Calculer  $t := M_j^{-1}r$ 
10.    Calculer  $V_{j+1} := ORTHN([V_j, t])$ 
11.    EndIf
12.    Poser  $v_1 := z$  et aller à 3
13.  EndDo
14 EndDo

```

Pour préciser les choix pour l'étape de preconditionnement $t := M_j^{-1}r$, on examine d'abord ce que l'on recherche. Supposons que l'on ait une paire propre approchée $(\tilde{\lambda}, \tilde{u})$ où \tilde{u} est un vecteur normé et $\tilde{\lambda} = \tilde{u}^T A \tilde{u}$. On recherche une correction v telle que le vecteur $\tilde{u} + v$ soit un vecteur propre exact correspondant à la valeur propre corrigée $\tilde{\lambda} + \delta$, donc :

$$A(\tilde{u} + v) = (\tilde{\lambda} + \delta)(\tilde{u} + v) \quad (22)$$

Pour mieux contraindre le problème, on impose que la correction v soit orthogonale à \tilde{u} . Il faut alors résoudre le système non linéaire suivant, d'inconnues v et δ :

$$\begin{cases} (A - \lambda I)v = -r + \delta(\tilde{u} + v) \\ v \perp \tilde{u} \end{cases} \quad (23)$$

En négligeant tous les termes du deuxième ordre en $\|v\|$ et parce que $\delta = O(\|v\|^2)$, on obtient le système approché suivant à résoudre dans l'espace orthogonal de \tilde{u} :

$$\tilde{Q}(A - \lambda I)\tilde{Q}v = -r \quad (24)$$

où $\tilde{Q} = I - \tilde{u}\tilde{u}^T$ est la projection orthogonale sur \tilde{u}^\perp .

La **méthode de Jacobi-Davidson** [29] résout partiellement à chaque étape l'équation (24). Les méthodes de Davidson [8] [19], généralisation de la méthode d'origine, choisissent un preconditionnement fixe $M_j = M$ à l'étape 9 de l'algorithme 10.

5.13 Autres méthodes de correction

On peut généraliser l'étude faite dans le paragraphe précédent pour corriger un sous-espace propre approché. Le sous-espace

étant représenté par une base orthonormée, il faut donc calculer une correction sur un bloc de vecteurs et non plus sur un seul vecteur.

L'une des premières études dans ce sens a été définie dans la méthode de la minimisation de la trace [27] [28] qui est apparue comme une accélération de la méthode des itérations simultanées. Elle met donc une correction proche de celle de Jacobi-Davidson mais itère sur un sous-espace de dimension fixe. Pour une étude détaillée des corrections de sous-espaces, on peut consulter [25].

6. Problème généralisé aux valeurs propres

6.1 Définition et propriétés du problème

Dans certaines applications, le problème des valeurs propres ne s'exprime pas sous la forme classique $Ax = \lambda x$ mais sous des formes *généralisées*. Par exemple, en mécanique des solides ou des structures, le problème est souvent sous la forme :

$$Ax = \lambda Bx$$

où $A, B \in \mathbb{R}^{n \times n}$ et où B peut être symétrique définie positive par exemple. Le problème quadratique de valeurs propres :

$$(A - \lambda B - \lambda^2 C)x = 0$$

est une autre généralisation qui intervient en mécanique, et qui peut être convertie en un problème classique généralisé. Nous considérons ici seulement le problème classique.

La forme généralisée classique du problème aux valeurs propres est définie pour une paire de matrices $A, B \in \mathbb{C}^{n \times n}$. On recherche les couples $(\lambda, x) \in \mathbb{C} \times \mathbb{C}^n - \{0\}$ tels que :

$$Ax = \lambda Bx \quad (25)$$

Les valeurs propres λ sont donc les racines du polynôme $p(\lambda) = \det(A - \lambda B)$.

Définition 4

Deux matrices A et B carrées d'ordre n définissent le faisceau noté $(A - \lambda B)$. Le faisceau est dit régulier si, par définition, le polynôme $p(\lambda) = \det(A - \lambda B)$ n'est pas identiquement nul. Dans ce cas, le spectre $\Lambda(A, B)$ de $A - \lambda B$ est l'ensemble des racines de $p(\lambda)$.

Par abus de notation, on parlera aussi du faisceau (A, B) . Il est souvent plus intéressant de formuler la valeur propre non comme un scalaire, mais comme une paire. On remplace (25) par :

$$\alpha Ax = \beta Bx \tag{26}$$

où (α, β) n'est pas le couple nul. L'ensemble des (α, β) satisfaisant l'équation (26) est défini à un scalaire multiple près mais on peut les normaliser et considérer un représentant tel que, par exemple, $\alpha + \beta = 1$. Lorsque $\alpha \neq 0$, la valeur propre correspondante λ est égale à $\frac{\beta}{\alpha}$. Dans le cas $\alpha = 0$ et $\beta \neq 0$, on dit que la valeur propre est infinie.

Dans le cas où l'une des matrices est inversible, on peut se ramener au problème classique. Si, par exemple, B est inversible, on voit que (25) donne lieu au problème :

$$B^{-1}Ax = \lambda x$$

La transformation ci-avant est intéressante en théorie, mais elle est à déconseiller pour un but pratique de calcul. En effet, des méthodes spéciales, plus fiables numériquement, ont été développées pour ce problème.

Dans le cas où les deux matrices sont singulières, la transformation ci-avant ne peut être effectuée. Considérons par exemple les matrices suivantes :

$$A_1 = \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}, B_1 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, A_2 = \begin{pmatrix} -1 & 0 \\ 1 & 0 \end{pmatrix}, B_2 = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} \tag{27}$$

Le faisceau (A_1, B_1) est régulier et admet les valeurs propres $\pm i$, ce qui montre que les valeurs propres peuvent être complexes même quand les deux matrices sont symétriques. Ici, aucune des deux matrices n'est définie positive.

Le faisceau (A_1, B_2) est régulier mais n'admet aucune valeur propre λ . Dans la formulation (26), l'ensemble des valeurs propres est l'ensemble $(0, \beta)$ pour tout $\beta \neq 0$. Ici, l'une des matrices, à savoir A , est inversible.

Le faisceau (A_2, B_1) est régulier. La matrice B_1 est inversible et donc le problème peut se ramener au cas classique, qui admet les valeurs propres ± 1 .

Finalement, les deux matrices du faisceau (A_2, B_2) sont singulières, avec le même noyau. Le faisceau (A_2, B_2) n'est pas régulier car n'importe quelle valeur λ est une valeur propre. Dans la formulation (26), toute paire (α, β) est valeur propre. Ce cas est considéré comme *dégénéré* et il surviendra à chaque fois que les deux matrices A et B ont un noyau dont l'intersection est non réduite à zéro. Il est évident que les cas dégénérés ne pourront pas être traités numériquement et sont donc exclus des algorithmes qui traitent du problème généralisé.

On pourra énoncer les quelques premières propriétés simples suivantes :

- si la matrice B du faisceau $A - \lambda B$ est inversible, le faisceau est régulier et $\Lambda(A, B) = \Lambda(B^{-1}A)$. Dans le cas contraire, son spectre est ou bien fini, ou bien est \mathbb{C} tout entier, ou bien est vide ;

- si le faisceau est régulier et alors pour tout $\lambda \neq 0$:

$$\lambda \in \Lambda(A, B) \Rightarrow \frac{1}{\lambda} \in \Lambda(B, A)$$

- soit σ tel que $(A - \sigma B)$ est une matrice inversible [le faisceau $(A - \lambda B)$ est donc régulier]. On a alors l'équivalence suivante :

$$\det(A - \lambda B) = 0 \text{ si et seulement si } \det[(A - \sigma B)^{-1}B - \tau I] = 0 \tag{28}$$

où $\tau = \frac{1}{\lambda - \sigma}$

La troisième propriété permet de transformer le problème généralisé en un problème standard. De plus, les valeurs propres les plus proches du nombre complexe σ sont celles de plus grand module dans le problème standard transformé. Cette transformation est une généralisation de la méthode de décalage et inversion introduite dans le paragraphe 5.9 et elle sera utilisée plus loin en combinaison avec l'algorithme de Lanczos.

Pour le cas **symétrique**, on se restreint au cas réel, mais le traitement avec des matrices hermitiennes complexes est identique.

Théorème 12

Soit $(A - \lambda B)$ un faisceau de matrices réelles symétriques avec la matrice B définie positive. Alors le problème peut se transformer en un problème standard symétrique : soit $B = LL^T$ la factorisation de Cholesky (voir la définition dans l'article *Méthodes numériques de base. Algèbre numérique* [AF 1 221]) de la matrice B ; on a l'équivalence suivante :

$$Ax = \lambda x \text{ si et seulement si } (L^{-1}AL^{-T})y = \lambda y \text{ avec } x = L^{-T}y$$

Il existe donc une matrice diagonale $D \in \mathbb{R}^{n \times n}$ et une matrice inversible $X \in \mathbb{R}^{n \times n}$ qui vérifient :

$$AX = BXD \text{ avec } X^T BX = I \tag{29}$$

Remarque 5

La condition de positivité de l'une des deux matrices est nécessaire comme on l'a vu dans l'exemple avec les matrices (A_1, B_1) dans (27).

La décomposition (29) montre que dans ce cas, le problème généralisé peut être vu comme un problème standard dans lequel on a remplacé le produit scalaire classique $x^T y$ par le produit scalaire $x^T B y$.

6.2 Cas non symétrique et l'algorithme QZ

L'algorithme QZ est une extension de l'algorithme QR (vu au paragraphe 2) au cas du problème généralisé. Tout comme QR, il comporte deux étapes, l'une pour transformer le problème en une forme simple et la deuxième qui est l'itération QZ elle-même appliquée à la paire résultant de la première transformation. Le but est de transformer la paire (A, B) en une paire où les deux éléments sont des matrices triangulaires supérieures.

6.2.1 Transformation en une paire Hessenberg-triangulaire

La première phase consiste à réduire (A, B) en une paire (A', B') qui est telle que A' est de forme Hessenberg supérieure, et B' est triangulaire supérieure. Ceci est réalisé grâce à une combinaison de **transformations de Householder** et de **rotations de Givens**. La matrice B est d'abord mise sous forme triangulaire supérieure. La

même transformation U^T est ensuite appliquée à A et on obtient par exemple :

$$U^T A = \begin{pmatrix} \star & \star & \star & \star & \star \\ \star & \star & \star & \star & \star \\ \star & \star & \star & \star & \star \\ \star & \star & \star & \star & \star \\ \star & \star & \star & \star & \star \end{pmatrix} \quad U^T B = \begin{pmatrix} \star & \star & \star & \star & \star \\ \star & \star & \star & \star & \star \\ & \star & \star & \star & \star \\ & & \star & \star & \star \\ & & & \star & \star \end{pmatrix}$$

La nouvelle paire a bien le même spectre que celle de (A, B) . Ensuite, des transformations de Givens sont appliquées de façon à réduire A à la forme de Hessenberg tout en préservant la forme triangulaire de B . Ainsi la première rotation, qui est $G(4, 5)$, est appliquée à gauche de façon à éliminer le terme $(5, 1)$ de A . Cela introduit un élément en position $(5, 4)$ de B , qui est à son tour éliminé par une rotation $G(4, 5)$ appliquée à droite. On peut ainsi éliminer la première colonne de A jusqu'à $\alpha_{1,2}$. On procède de la même manière pour les colonnes suivantes.

6.2.2 Itération QZ

L'itération QZ revient à une itération de l'algorithme QR sur la matrice AB^{-1} où A et B sont les matrices résultant de l'étape précédente. Grâce au théorème Q-implicite, on arrive ainsi à retrouver les rotations qu'il faut en éliminant des termes et en « chassant les termes non nuls » introduits à une étape donnée. Pour démarrer, il faut se rappeler que la première colonne de AB^{-1} est connue dû à la forme triangulaire de B . Une description détaillée du procédé n'est pas donnée ici mais elle peut être trouvée dans [14].

6.3 Algorithme de Lanczos pour matrices de grande taille

Nous ne traitons ici que du cas, important dans les applications, symétrique défini, c'est-à-dire le cas où A et B sont symétriques réelles, et l'une des deux matrices, par exemple B , est définie positive. Cette situation correspond au cas du problème classique de valeurs propres de matrices symétriques. Ainsi, les valeurs propres sont réelles et les vecteurs propres forment une base orthonormale par rapport au B -produit scalaire défini par :

$$(x, y)_B = (Bx, y) \tag{30}$$

Cela représente bien un produit scalaire puisque B est symétrique définie positive. Ce produit scalaire définit la B -norme :

$$\|x\|_B = (Bx, x)^{1/2}$$

Considérons la matrice $C = B^{-1}A$. Une observation importante pour comprendre ce qui va suivre est que, quoique la matrice C soit non symétrique en général, elle est *auto-adjointe* par rapport au B -produit scalaire en ce sens que :

$$\forall x, y, (Cx, y)_B = (x, Cy)_B \tag{31}$$

Par conséquent, on peut s'attendre à ce que tous les résultats vus pour le problème standard dans le cas symétrique réel, soient encore vrais *pourvu que l'on remplace le produit scalaire euclidien par le B-produit scalaire*.

Il est aussi possible de factoriser la matrice B en facteurs de Cholesky, soit $B = LL^T$, et de convertir le problème généralisé en un problème classique avec la matrice $C = L^{-1}AL^{-T}$ (voir le théorème 12). Une factorisation de B est nécessaire. L'algorithme de Lanczos implique alors la résolution de systèmes triangulaires avec L et L^T à chaque pas.

Une alternative intéressante serait simplement d'appliquer l'algorithme de Lanczos standard sur la matrice $C = B^{-1}A$ en remplaçant le produit scalaire euclidien par le B -produit scalaire à chaque fois qu'un produit scalaire est calculé. Une implémentation simple conduirait aux pas suivants :

$$w := B^{-1}Av_j \tag{32}$$

$$\alpha_j := (w, v_j)_B \tag{33}$$

$$w := w - \alpha_j v_j - \beta_j v_{j-1} \tag{34}$$

$$\begin{aligned} \beta_{j+1} &:= \|w\|_B \\ v_{j+1} &:= w / \beta_{j+1} \end{aligned} \tag{35}$$

On remarque que α_j dans (33) est aussi égal à (Av_j, v_j) et ceci donne un moyen simple de calculer les α_j , en utilisant le produit scalaire euclidien standard. Avant de multiplier Av_j par B^{-1} en (32), α_j est calculé et sauvegardé. Le calcul de β_{j+1} est un peu plus délicat. En effet, si l'on devait utiliser la définition du B -produit scalaire, on aurait normalement à faire un produit matrice-vecteur supplémentaire avec la matrice B , ce qui peut être évité.

Observons que par construction, le vecteur w en (35) est B -orthogonal aux vecteurs v_j et v_{j-1} . Par conséquent :

$$(Bw, w) = (Av_j, w) - \alpha_j (Bv_j, w) - \beta_j (Bv_{j-1}, w) = (Av_j, w)$$

Ainsi, si on sauvegarde le vecteur Av_j calculé en (32) jusqu'à la fin de la boucle, on peut évaluer la B -norme de w avec juste un produit scalaire euclidien.

Une autre alternative est de garder une récurrence à trois termes pour les vecteurs $z_j = Bv_j$. Dans ce cas, Bw est disponible par la relation :

$$Bw = Av_j - \alpha_j z_j - \beta_j z_{j-1}$$

et le produit scalaire (Bw, w) peut alors être évalué. Normalisant Bw par β_{j+1} donne z_{j+1} . Cette façon de faire exige deux vecteurs de stockage de plus et quelques calculs supplémentaires mais elle sera numériquement plus fiable. La version décrite dans l'algorithme 11 implémente cette procédure avec la forme de l'algorithme de Gram-Schmidt modifié.

Algorithme 11 – Algorithme de Lanczos pour $Ax = \lambda Bx$

1. **Init :** Choisir un vecteur initial v_1 tel que $\|v_1\|_B = 1$.
2. Poser $\beta_1 = 0, z_0 = v_0 = 0, z_1 = Bv_1$.
3. **For** $j = 1, 2, \dots, m$, **Do :**
4. $v := Av_j - \beta_j z_{j-1}$,
5. $\alpha_j = (v, v_j)$,
6. $v := v - \alpha_j z_j$,
7. $w := B^{-1}v$,
8. $\beta_{j+1} = \sqrt{(w, v)}$,
9. $v_{j+1} = w / \beta_{j+1}$ **et** $z_{j+1} = v / \beta_{j+1}$.
10. **EndDo**

Il est à observer que la B -norme calculée en ligne 8 est de la forme $\sqrt{(B^{-1}v, v)}$ et puisque B est symétrique définie positive, cela ne devrait pas causer de difficultés numériques.

En pratique, l'algorithme 11 n'est pas applicable dans la situation où B est singulière. Cette situation a été étudiée dans la

littérature. Sans entrer dans les détails, on soulignera seulement que l'idée principale est encore une fois de décaler le problème de façon à rendre $(A - \sigma B)$ inversible et ensuite de travailler dans le sous-espace $\text{Im}(A - \sigma B)^{-1}B$. Une version simplifiée d'un tel algorithme développé en [20] est la suivante, où σ est le décalage utilisé.

Algorithme 12 – Algorithme de Lanczos avec transformation spectrale

1. **Init :** Choisir un vecteur initial w dans $\text{Im}[(A - \sigma B)^{-1}B]$.
2. Calculer $z_1 = Bw$ et $\beta_1 := \sqrt{(w, z_1)}$. Poser $v_0 := 0$.
3. **For** $j = 1, 2, \dots, m$, **Do :**
4. $v_j = w / \beta_j$ et $z_j := z_j / \beta_j$,
5. $z_j = (A - \sigma B)^{-1}w$,
6. $w := w - \beta_j v_{j-1}$,
7. $\alpha_j = (w, z_j)$,
8. $w := w - \alpha_j z_j$,
9. $z_{j+1} = Bw$,
10. $\beta_{j+1} = \sqrt{(z_{j+1}, w)}$.
11. **EndDo**

L'algorithme ne requiert que des produits avec la matrice B . Comme dans l'algorithme précédent, les deux vecteurs z_j les plus récents doivent être sauvegardés. Si une forme de réorthogonalisation est nécessaire, il faudra alors les sauvegarder tous. À l'exception de la précaution prise pour choisir le vecteur initial, l'algorithme 12 est une réformulation de l'algorithme 11, appliqué à la paire (A', B') $((A - \sigma B), B)$.

7. Décomposition aux valeurs singulières

La décomposition aux valeurs singulières, notée SVD (*singular value decomposition*), a des applications importantes dans des domaines scientifiques très variés. Cette décomposition peut être considérée comme une généralisation de la décomposition spectrale d'une matrice hermitienne qui est une décomposition de A en un produit de la forme $A = UDU^H$ où U est unitaire et D diagonale. Cependant, la décomposition SVD existe pour toute matrice, et ce, même dans le cas des matrices rectangulaires.

Théorème 13

Pour toute matrice $A \in \mathbb{R}^{m \times n}$, il existe des matrices orthogonales $U \in \mathbb{R}^{m \times m}$ et $V \in \mathbb{R}^{n \times n}$ telles que :

$$A = U \Sigma V^T$$

où Σ est une matrice diagonale de dimension $m \times n$ avec des éléments diagonaux $\sigma_{ij} \geq 0$.

Pour une démonstration, on consultera [14]. On supposera que les termes diagonaux sont rangés par ordre décroissant, i.e., $\sigma_{11} \geq \sigma_{22} \geq \dots \geq \sigma_{pp} \geq 0$ où $p = \min(n, m)$. Les σ_{ij} sont les *valeurs singulières* de A et sont dénotées par σ_j . On remarque que la décomposition d'une matrice n'est pas unique. Par exemple, en changeant les signes des matrices U et V , on obtient une autre SVD licite de A . La décomposition est illustrée sur la figure 3 suivant le nombre de lignes et de colonnes de la matrice.

7.1 Version « mince » de la SVD

Considérons le cas où $m > n$ et soit la matrice $\Sigma_1 = \text{diag}(\sigma_1, \dots, \sigma_n)$ de dimension $n \times n$. Dans ce cas, la SVD de A peut être réécrite comme :

$$A = [U_1, U_2] \begin{pmatrix} \Sigma_1 \\ 0 \end{pmatrix} V^T$$

où Σ_1 est une matrice carrée de dimension n , $U_1 \in \mathbb{R}^{m \times n}$ et $U_2 \in \mathbb{R}^{m \times (m-n)}$. Il est évident que la partie U_2 ne sert à rien puisqu'elle est multipliée par un bloc de zéros dans Σ . Donc on peut réécrire la relation précédente :

$$A = U_1 \Sigma_1 V^T \tag{36}$$

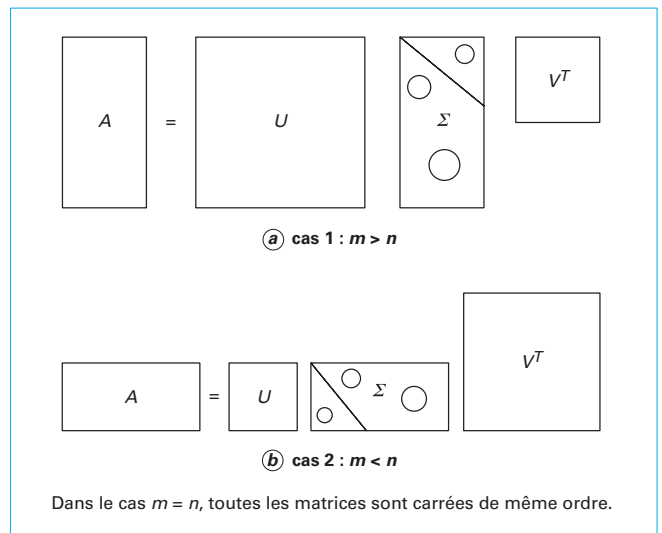


Figure 3 – Illustration de la décomposition aux valeurs singulières

Cette décomposition simplifiée est souvent appelée la version mince de la SVD et elle est importante en pratique.

Dans le second cas, c'est-à-dire quand $n > m$, on a de manière similaire :

$$A = U [\Sigma_1, 0] \begin{pmatrix} V_1^T \\ V_2^T \end{pmatrix}$$

qui mène à la décomposition mince :

$$A = U \Sigma_1 V_1^T \tag{37}$$

où Σ_1 et U sont toutes les deux de dimension $m \times m$, et où V_1 est de dimension $n \times m$.

Il est intéressant de se demander comment obtenir, au moins en théorie, la SVD mince à partir de la factorisation QR de A et de la SVD d'une matrice de plus petite taille. Dans le cas $m > n$, il suffit de commencer par la décomposition QR , $A = QR$ et d'enchaîner par la décomposition SVD de R , soit $R = U_R \Sigma_1 V^T$. Il s'ensuit que si on définit $U_1 = QU_R$ on obtient la SVD mince de A : $A = U_1 \Sigma_1 V^T$.

7.2 Propriétés

Nous allons maintenant énoncer quelques propriétés simples de la SVD. Supposons qu'il y ait r valeurs singulières non nulles, où $r \leq p = \min(m, n)$, c'est-à-dire que :

$$\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_r > \sigma_{r+1} = \dots = \sigma_p = 0 \tag{38}$$

La première propriété est que le rang de A est égal à r , le nombre de valeurs singulières non nulles. Cela provient du fait que l'espace image est simplement :

$$\text{Im}(A) = \text{sev}\{u_1, u_2, \dots, u_r\}$$

comme cela peut être facilement vérifié. De même, on pourra vérifier que le sous-espace noyau de A (ensemble des vecteurs d'image nulle) est :

$$\text{Null}(A) = \text{sev}\{v_{r+1}, v_{r+2}, \dots, v_n\}$$

Une autre expression de la décomposition SVD, qui est importante en pratique, s'obtient par une réécriture de (36) et (37) en forme de somme de matrices de rang 1. Dans les deux cas, on peut en effet écrire que :

$$A = \sum_{i=1}^r \sigma_i u_i v_i^T \tag{39}$$

D'autre part, les valeurs singulières permettent de calculer certaines normes connues de A . Ainsi :

$$\|A\|_2 = \sigma_1, \quad \|A\|_F = \left(\sum_{i=1}^r \sigma_i^2 \right)^{1/2}$$

On peut aussi facilement montrer que quand A est une matrice $n \times n$ inversible, alors $\|A^{-1}\|_2 = 1/\sigma_n$ qui est l'inverse de la valeur singulière la plus petite.

On a vu ci-avant que A , matrice de rang r , est en fait une combinaison linéaire de r matrices de rang 1 de la forme $u_i v_i^T$. On peut démontrer le résultat suivant.

Théorème 14

Soit $k < r$ et :

$$A_k = \sum_{i=1}^k \sigma_i u_i v_i^T$$

où les valeurs singulières vérifient (38), alors :

$$\min_{\text{rang}(B)=k} \|A - B\|_2 = \|A - A_k\|_2 = \sigma_{k+1}$$

Voir [14] pour une démonstration de ce résultat. Il est intéressant d'observer que les vecteurs singuliers et les valeurs singulières de A sont fortement reliés aux valeurs propres et vecteurs propres des matrices AA^T et $A^T A$. Soit $A \in \mathbb{R}^{m \times n}$ avec $m \geq n$. Considérons $A^T A$ (matrice appartenant à $\mathbb{R}^{n \times n}$) ; d'après (36), on a :

$$A^T A = V \Sigma^T \Sigma V^T \Rightarrow A^T A = V \underbrace{\Sigma_1^2}_{n \times n} V^T \tag{40}$$

Ceci donne la décomposition spectrale de $A^T A$. De manière similaire, on peut voir que U donne les vecteurs propres de AA^T :

$$AA^T = U \begin{pmatrix} \Sigma_1^2 & 0 \\ 0 & 0 \end{pmatrix} U^T \tag{41}$$

On note que les diagonalisations de $A^T A$ et de AA^T déterminent respectivement les vecteurs singuliers V et U de A mais *au signe près seulement*. Rappelons en effet que la SVD n'est pas unique.

On peut par exemple calculer la SVD mince de la matrice :

$$A = \begin{pmatrix} 1 & 0 & 2 & 0 \\ 0 & 0 & -2 & 1 \end{pmatrix}$$

à partir de la décomposition spectrale de AA^T qui est $AA^T = U D U^T$ avec :

$$U = \frac{1}{\sqrt{2}} \begin{pmatrix} -1 & 1 \\ 1 & 1 \end{pmatrix}, \quad D = \begin{pmatrix} 9 & 0 \\ 0 & 1 \end{pmatrix}$$

Donc $\Sigma_1 = \text{diag}(3, 1)$. D'après (37), on pourra obtenir V_1 simplement par $V_1^T = \Sigma_1^{-1} U^T A$.

Une autre forme de matrice auxiliaire permet de calculer la SVD d'une matrice :

Théorème 15 : forme augmentée

Soit $A \in \mathbb{R}^{m \times n}$ ($m \geq n$) et les matrices orthogonales :

$$U = [u_1, \dots, u_m] \in \mathbb{R}^{m \times m} \text{ et } V = [v_1, \dots, v_n] \in \mathbb{R}^{n \times n}$$

de la SVD :

$$A = U \Sigma V^T$$

Alors, la matrice symétrique, dite matrice augmentée, d'ordre $n + m$:

$$A_{\text{aug}} = \begin{pmatrix} 0 & A \\ A^T & 0 \end{pmatrix} \tag{42}$$

a pour valeurs propres $\pm \sigma_1, \dots, \pm \sigma_n$, et pour vecteurs propres associés :

$$\frac{1}{\sqrt{2}} \begin{pmatrix} u_i \\ \pm v_i \end{pmatrix}, \quad i = 1, \dots, n$$

les $m - n$ valeurs propres restantes étant nulles.

Le calcul des valeurs singulières de matrice se ramène donc toujours à un calcul de valeurs propres de matrices symétriques soit à partir de l'une des formes normales (40) ou (41), soit avec la matrice augmentée (42). Cela est aussi vrai pour les grandes matrices creuses. Cette manière de calculer la SVD est restreinte à des cas simples et il est évident que dans le cas général, il est préférable d'utiliser des codes performants et adaptés tels que ceux de LAPACK (cf. § 4), codes qui transforment préalablement la matrice pour diminuer la complexité des calculs.

7.3 Applications de la SVD

7.3.1 Pseudo-inverses de matrices

Pour une matrice A rectangulaire quelconque, ou simplement carrée singulière, on ne peut pas définir l'inverse de A , c'est-à-dire une matrice B telle que $AB = BA = I$. Cependant, on peut définir la pseudo-inverse qui satisfait certaines propriétés de l'inverse classique. La pseudo-inverse se définit à partir de l'écriture de la SVD de la manière suivante :

$$A = U \begin{pmatrix} \Sigma_1 & 0 \\ 0 & 0 \end{pmatrix} V^T = \sum_{i=1}^r \sigma_i u_i v_i^T \quad (43)$$

où Σ_1 est carrée et de dimension $r \times r$ et inversible. Si le terme du milieu, qui est en général rectangulaire, était inversible ou saurait inverser A . Il suffit donc de définir la pseudo-inverse de la matrice du milieu. Celle-ci se définit en transposant cette matrice et en inversant les valeurs singulières non nulles.

Le pseudo-inverse de A est donc la matrice :

$$A^+ = V \begin{pmatrix} \Sigma_1^{-1} & 0 \\ 0 & 0 \end{pmatrix} U^T = \sum_{i=1}^r \frac{v_i u_i^T}{\sigma_i} \quad (44)$$

Le pseudo-inverse introduit ci-dessus vérifie les **conditions de Moore-Penrose** suivantes qui caractérisent de manière rigoureuse et compacte les conditions pour qu'une matrice X soit la pseudo-inverse de A :

- (pi.i) $AXA = A$ (pi.ii) $XAX = X$
- (pi.iii) $(AX)^H = AX$ (pi.iv) $(XA)^H = XA$

Le pseudo-inverse d'une matrice A est uniquement déterminé par ces quatre conditions. Dans le cas où A est de plein rang, avec $m \geq n$, on a $A^+ = (A^T A)^{-1} A^T$.

7.3.2 Problèmes aux moindres carrés

La SVD peut apporter beaucoup d'informations dans la résolution de problèmes sur-déterminés ou sous-déterminés. On considère le problème aux moindres carrés sous sa forme générale :

$$\min_{x \in \mathbb{R}^n} \|b - Ax\|_2$$

où A est $m \times n$ et quelconque. Lorsque $m < n$, il y a généralement une infinité de solutions minimales, c'est-à-dire de minimiseurs de $\|b - Ax\|_2$. Dans ce cas, on s'intéresse souvent à la solution de plus petite norme. La situation est semblable lorsque $m > n$ et que A n'est pas de plein rang. En effet, il suffit d'ajouter à une solution minimale x^* un vecteur du noyau de A pour obtenir une autre solution minimale. Dans ce cas aussi, on s'intéresse à la solution de norme euclidienne minimale. Le théorème suivant établit un résultat général pour tous les cas.

Théorème 16

Soit A une matrice $m \times n$ et $A = U \Sigma V^T$ sa décomposition SVD avec $r = \text{rang}(A)$, $V = [v_1, \dots, v_n]$ $U = [u_1, \dots, u_m]$. Alors :

$$x_{LS} = \sum_{i=1}^r \frac{u_i^T b}{\sigma_i} v_i = A^+ b \quad (45)$$

minimise $\|b - Ax\|_2$ et possède la plus petite norme euclidienne parmi toutes les solutions minimales. De plus :

$$\rho_{LS} \equiv \|b - Ax_{LS}\|_2 = \|z\|_2 \text{ avec } z = [u_{r+1}, \dots, u_m]^T b$$

Ainsi, une autre façon d'écrire la solution du problème général aux moindres carrés suivant :

$$\min_{x \in S} \|x\|_2, \quad \text{où } S = \{x \in \mathbb{R}^n \mid \|b - Ax\|_2 \text{ est minimal}\}$$

est :

$$x = A^+ b$$

Régularisation

Soit A une matrice $m \times m$ inversible et $A = U \Sigma V^T$ sa décomposition SVD. La solution du problème $Ax = b$ est :

$$x = A^{-1} b = \sum_{i=1}^m \frac{u_i^T b}{\sigma_i} v_i$$

La matrice A est très mal conditionnée lorsqu'elle a de petites valeurs singulières, petites par rapport à la plus grande. La division par ces petites valeurs va amplifier les petites erreurs dues aux incertitudes sur les données ; si $\tilde{b} = b + \varepsilon$ alors :

$$A^{-1} \tilde{b} = \sum_{i=1}^m \frac{u_i^T b}{\sigma_i} v_i + \underbrace{\sum_{i=1}^m \frac{u_i^T \varepsilon}{\sigma_i} v_i}_{\text{erreur}}$$

Ainsi, la solution calculée peut ne plus avoir de sens. Le remède qui est souvent utilisé est de régulariser la solution à l'aide de la SVD. Il consiste à tronquer la SVD en ne gardant que les $\sigma_i \geq \tau$, où τ est un seuil en dessous duquel les incertitudes deviennent prépondérantes. On obtient alors la solution TSVD (SVD tronquée) :

$$x_{\text{TSVD}(\tau)} = \sum_{\sigma_i \geq \tau} \frac{u_i^T b}{\sigma_i} v_i$$

8. Conclusion

L'exposé de cet article tente de faire un tour d'horizon des méthodes utilisées actuellement. Un exposé du même type mais plus exhaustif est le livre collectif [2] dont des versions libres existent sur le web. Ce livre contient en particulier une liste de références bibliographiques très complète à la date de 2000.