Chapitre V

Valeurs et Vecteurs Propres

Les premiers vecteurs et valeurs propres viennent des équations différentielles (Lagrange 1759, *théorie du son* ; Lagrange 1781, des matrices 6×6 dans le but de calculer les perturbations séculaires des orbites des 6 planètes connues à l'époque, *Oeuvres* V, p. 125-490). Aujourd'hui, le calcul des valeurs et vecteurs propres est indispensable dans toutes les branches de la science, en particulier pour la solution des systèmes des équations différentielles linéaires, en théorie de stabilité, pour les questions de convergence de processus itératifs, et en physique et chimie (mécanique, circuits, cinétique chimique, équation de Schrödinger).

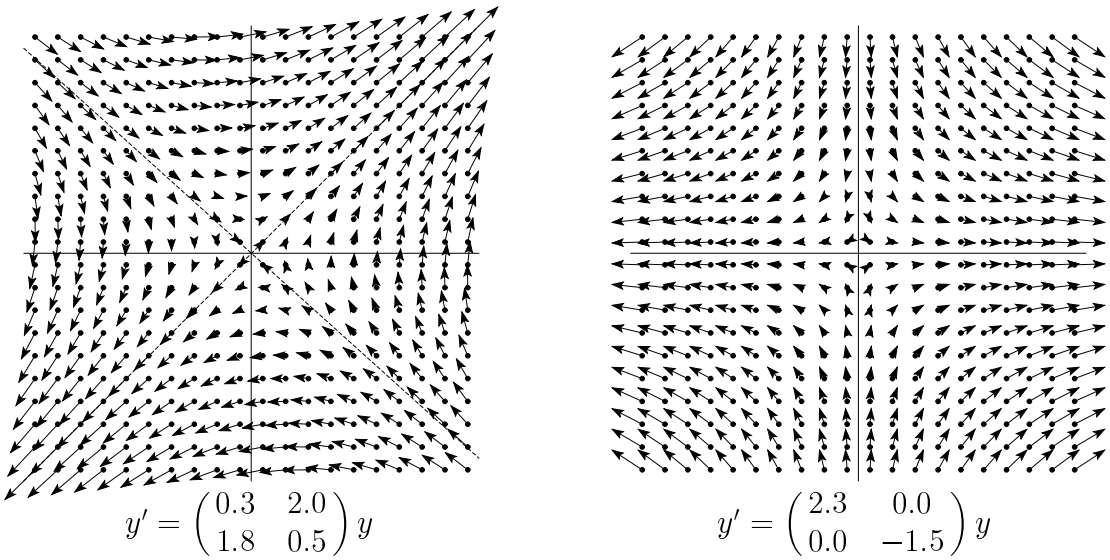


FIG. V.1: Une application linéaire comme champ de vecteurs (à gauche) ; transformée sur la base des vecteurs propres (à droite).

Observons en figure V.1 (à gauche) le champ de vecteurs d'une équation différentielle $y' = Ay$. Deux directions sautent aux yeux : ce sont les directions où le vecteur Av prend la même direction que le vecteur v, i.e., où

$$Av = \lambda v \qquad \text{ou} \qquad (A - \lambda I)v = 0\,. \tag{0.1}$$

Si cette équation est vérifiée, $\lambda \in \mathbb{C}$ s'appelle *valeur propre* de la matrice A et $v \in \mathbb{C}^n$ ($v \neq 0$) est le *vecteur propre* correspondant. L'équation (0.1) possède une solution v non nulle si et seulement si

$$\chi_A(\lambda) = \det(A - \lambda I) = 0. \tag{0.2}$$

Le polynôme $\chi_A(\lambda)$ est le *polynôme caractéristique* de la matrice A. Les valeurs propres de A sont alors les zéros du polynôme caractéristique.

Bibliographie sur ce chapitre

Tous les livres cités dans le chapitre IV, et en plus . . .

B.N. Parlett (1980): *The Symmetric Eigenvalue Problem*. Prentice-Hall, Englewood Cliffs, NJ.

B.T. Smith, J.M. Boyle, Y. Ikebe, V.C. Klema & C.B. Moler (1970): *Matrix Eigensystem Routines:* EISPACK *Guide*. 2nd ed., Springer-Verlag, New York.

J.H. Wilkinson (1965): *The Algebraic Eigenvalue Problem*. Clarendon Press. [MA 65/72]

J.H. Wilkinson & C. Reinsch (1971): *Handbook for Automatic Computation, Volume II, Linear Algebra*. Springer-Verlag, New York.

V.1 La condition du calcul des valeurs propres

A cause des erreurs d'arrondi, les éléments d'une matrice A, pour laquelle on cherche les valeurs propres, ne sont pas exacts. Ils sont plutôt égaux à $\widehat{a}_{ij} = a_{ij}(1+\epsilon_{ij})$ avec $|\epsilon_{ij}| \leq$ *eps* (*eps*, la précision de l'ordinateur, est supposée être très petite). Il est alors très important d'étudier l'influence de ces perturbations sur les valeurs propres et sur les vecteurs propres de la matrice. Pour montrer ceci, considérons la famille de matrices

$$A(\epsilon) = A + \epsilon C \qquad \text{où} \qquad |\epsilon| \leq \textit{eps} \quad \text{et} \quad |c_{ij}| \leq |a_{ij}| \tag{1.1}$$

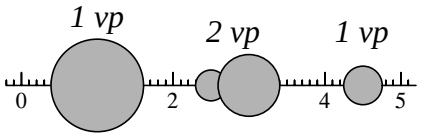
(souvent, la dernière hypothèse va être remplacée par $\|C\| \leq \|A\|$).

Théorème 1.1 (Gershgorine) *Soit A une matrice $n \times n$ (avec des éléments dans $\mathbb{R}$ ou dans $\mathbb{C}$).*
a) Si λ est une valeur propre de A, alors il existe un indice i tel que

$$|\lambda - a_{ii}| \leq \sum_{\substack{j=1 \\ j\neq i}}^{n} |a_{ij}|, \tag{1.2}$$

c.-à-d. que toutes les valeurs propres de A se trouvent dans l'union des disques

$$D_i = \{\lambda\,;\ |\lambda - a_{ii}| \leq \sum_{j\neq i} |a_{ij}|\,\}.$$



b) Si une composante connexe de $\bigcup_{i=1}^{n} D_i$ consiste de k disques, elle contient exactement k valeurs propres de A.

Démonstration. Soit $v \neq 0$ un vecteur propre et choisissons l'indice i tel que $|v_i| \geq |v_j|$ pour tout j. La ligne i de l'équation $Av = \lambda v$ donne

$$\sum_{j\neq i} a_{ij} v_j = (\lambda - a_{ii}) v_i.$$

En divisant par v_i et en utilisant l'inégalité de triangle, on obtient

$$|\lambda - a_{ii}| = \Big|\sum_{j\neq i} a_{ij} \cdot \frac{v_j}{v_i}\Big| \leq \sum_{j\neq i} |a_{ij}|.$$

L'affirmation (b) est vraie si A est une matrice diagonale. Le cas général est obtenu par un argument de continuité en faisant tendre les éléments en dehors de la diagonale vers zéro (voir le cours "Analyse II" concernant la dépendance continue de racinces d'un polynôme en fonction d'un paramètre). □

Théorème 1.2 *Soit A une matrice diagonalisable, i.e., il existe T avec $T^{-1}AT = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$, et soit $A(\epsilon) = A + \epsilon C$. Alors, pour chaque valeur propre $\lambda(\epsilon)$ de $A(\epsilon)$, il existe un λ_i avec*

$$|\lambda(\epsilon) - \lambda_i| \leq |\epsilon| \cdot \kappa_\infty(T) \cdot \|C\|_\infty.$$

Démonstration. Nous transformons la matrice $A(\epsilon) = A + \epsilon C$ par la même matrice, qui transforme A sous forme diagonale :

$$T^{-1}A(\epsilon)T = \mathrm{diag}(\lambda_1, \ldots, \lambda_n) + \epsilon T^{-1}CT.$$

Si l'on dénote par e_{ij} les éléments de $T^{-1}CT$, le théorème de Gershgorine implique l'existence d'un indice i tel que $|\lambda(\epsilon) - (\lambda_i + \epsilon e_{ii})| \leq |\epsilon| \sum_{j \neq i} |e_{ij}|$. L'inégalité de triangle donne alors

$$|\lambda(\epsilon) - \lambda_i| \leq |\epsilon| \cdot \max_i \Big(\sum_j |e_{ij}| \Big) \leq |\epsilon| \cdot \|T^{-1}CT\|_\infty \leq |\epsilon| \cdot \|T^{-1}\|_\infty \|C\|_\infty \|T\|_\infty,$$

ce qui démontre l'affirmation du théorème, car $\kappa_\infty(T) = \|T\|_\infty \|T^{-1}\|_\infty$ (condition de T). □

La condition du calcul des valeurs propres depend de la condition de la matrice de transformation T. Si la matrice A est symétrique (T est orthogonale), le problème est bien conditionné. Toutefois, observons qu'on obtient seulement une estimation pour l'*erreur absolue* et non pour l'erreur relative.

Théorème 1.3 (différentiabilité des valeurs propres) *Soit λ_1 une racine simple de $\chi_A(\lambda) = 0$. Alors, pour $|\epsilon|$ suffisamment petit, la matrice $A(\epsilon) = A + \epsilon C$ possède une valeur propre unique $\lambda_1(\epsilon)$ proche de λ_1. La fonction $\lambda_1(\epsilon)$ est différentiable (même analytique) et on a*

$$\lambda_1(\epsilon) = \lambda_1 + \epsilon \cdot \frac{u_1^* C v_1}{u_1^* v_1} + \mathcal{O}(\epsilon^2) \tag{1.3}$$

où v_1 est le vecteur propre à droite ($Av_1 = \lambda_1 v_1$) et u_1 est le vecteur propre à gauche ($u_1^* A = \lambda_1 u_1^*$). On peut supposer que $\|v_1\| = \|u_1\| = 1$.

Démonstration. Soit $p(\lambda, \epsilon) := \chi_{A+\epsilon C}(\lambda) = \det(A + \epsilon C - \lambda I)$. Comme

$$p(\lambda_1, 0) = 0 \qquad \text{et} \qquad \frac{\partial p}{\partial \lambda}(\lambda_1, 0) \neq 0,$$

le théorème des fonctions implicites garantie l'existence d'une fonction différentiable $\lambda_1(\epsilon)$ (même analytique), tel que $\lambda_1(0) = \lambda_1$ et $p(\lambda_1(\epsilon), \epsilon) = 0$. Il existe donc un vecteur $v_1(\epsilon) \neq 0$ tel que

$$\Big(A(\epsilon) - \lambda_1(\epsilon) I\Big) v_1(\epsilon) = 0. \tag{1.4}$$

La matrice dans (1.4) étant de rang $n - 1$, on peut fixer une composante à 1 et appliquer la règle de Cramer. Ceci montre que les autres composantes sont des fonctions rationnelles des éléments de la matrice $A + \epsilon C - \lambda_1(\epsilon) I$ et donc différentiables. Après la normalisation à $v_1(\epsilon)^T v_1(\epsilon) = 1$, la fonction $v_1(\epsilon)$ reste différentiable.

Pour calculer $\lambda_1'(0)$, nous pouvons dériver l'équation (1.4) par rapport à ϵ et poser ensuite $\epsilon = 0$. Ceci donne

$$(A - \lambda_1 I) v_1'(0) + (C - \lambda_1'(0) I) v_1 = 0. \tag{1.5}$$

En multipliant cette relation par u_1^*, on obtient $u_1^*(C - \lambda_1'(0) I) v_1 = 0$, ce qui permet de calculer $\lambda_1'(0)$ et démontre la formule (1.3). □

Conséquences. La formule (1.3) du thórème précédent montre que le plus le vecteur propre de droite est parallèle au vecteur propre de gauche, le mieux la valeur propre correspondante est *bien conditionnée* (par exemple, pour les matrices symétriques les deux vecteurs sont identiques) ; le plus ils se rapprochent de l'orthogonalité, le plus la valeur propre est *mal conditionnée*.

Si la matrice n'est pas symétrique (ou normale), le calcul de λ_1 (valeur propre simple) peut être mal conditionné. Considérons par exemple la matrice

$$A = \begin{pmatrix} 1 & \alpha \\ 0 & 2 \end{pmatrix} \qquad \text{où} \qquad v_1 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \qquad u_1 = \frac{1}{\sqrt{1+\alpha^2}} \begin{pmatrix} 1 \\ -\alpha \end{pmatrix}.$$

Dans cette situation, la formule (1.3) nous donne $\lambda_1(\epsilon) - \lambda_1 = \epsilon \cdot (c_{11} - \alpha c_{21}) + \mathcal{O}(\epsilon^2)$ et le calcul de $\lambda_1 = 1$ est mal conditionné si α est grand.

Exemple 1.4 Considérons la matrice (boîte de Jordan)

$$A = \left.\begin{pmatrix} \lambda_1 & 1 & & \\ & \lambda_1 & \ddots & \\ & & \ddots & 1 \\ & & & \lambda_1 \end{pmatrix}\right\} n \tag{1.6}$$

Le polynôme caractéristique de $A + \epsilon C$ satisfait

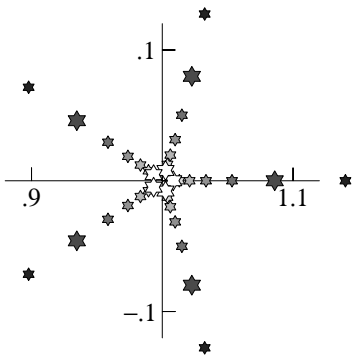
$$\det(A + \epsilon C - \lambda I) = (\lambda_1 - \lambda)^n - (-1)^n \cdot \epsilon \cdot c_{n1} + \mathcal{O}(\epsilon^2) + \mathcal{O}(\epsilon \cdot |\lambda_1 - \lambda|).$$

Si $c_{n1} \neq 0$, les termes $\mathcal{O}(\epsilon^2)$ et $\mathcal{O}(\epsilon \cdot |\lambda_1 - \lambda|)$ sont négligeables par rapport à $\epsilon \cdot c_{n1}$. Les valeurs propres de $A + \epsilon C$ sont alors approximativement données par les racines de

$$(\lambda_1 - \lambda)^n - (-1)^n \cdot \epsilon \cdot c_{n1} = 0, \qquad \text{c.-à-d.} \qquad \lambda \approx \lambda_1 + (\epsilon \cdot c_{n1})^{1/n} \tag{1.7}$$

(observer que $(\epsilon \cdot c_{n1})^{1/n}$ donne n valeurs complexes distinctes – multiples des racines de l'unité).

Expérience numérique. Prenons la matrice (1.6) avec $\lambda_1 = 1$ et $n = 5$. Les éléments de la matrice C sont des nombres aléatoires dans l'intervalle $[-1, 1]$. Le dessin ci-contre montre les 5 valeurs propres de $A + \epsilon C$ pour $\epsilon = 10^{-4}, 10^{-5}, \ldots, 10^{-10}$. L'erreur est $\approx 10^{-1}$ pour $\epsilon = 10^{-5}$ et $\approx 10^{-2}$ pour $\epsilon = 10^{-10}$, ce qui correspond à la formule (1.7) pour $n = 5$.



Conséquence. Si la dimension n d'une boîte de Jordan est plus grande que 1, le calcul de la valeur propre de cette matrice est *très mal conditionné*.

Condition du calcul des vecteurs propres

Considérons la situation où toutes les valeurs propres de A sont distinctes. La démonstration du théorème sur la différentiabilité des valeurs propres montre (voir formule (1.4)) que les vecteurs propres normalisés $v_i(\epsilon)$ de $A+\epsilon C$ sont des fonctions différentiables de ϵ. Pour étudier la condition du calcul des vecteurs propres, nous exprimons $v_1'(0)$ dans la base des vecteurs propres (de droite)

$$v_1'(0) = \sum_{i=1}^{n} \alpha_i v_i. \tag{1.8}$$

La formule (1.5) donne alors

$$\sum_{j=2}^{n}(\lambda_j - \lambda_1)\alpha_j v_j + (C - \lambda_1'(0)I)v_1 = 0. \tag{1.9}$$

En multipliant (1.9) par le vecteur propre de gauche u_i^* (observer que $u_i^* v_j = 0$ pour $i \neq j$), on obtient α_i (pour $i \geq 2$) de la relation $(\lambda_i - \lambda_1)\alpha_i u_i^* v_i + u_i^* C v_1 = 0$. La normalisation $\|v_1(\epsilon)\|_2^2 = 1$ donne (en la dérivant) $v_1^* v_1'(0) = 0$ et on en déduit que $\alpha_1 = -\sum_{i=2}^{n} \alpha_i v_1^* v_i$. Si l'on insère les formules pour α_i dans (1.8), on obtient pour $v_1(\epsilon) = v_1 + \epsilon v_1'(0) + \mathcal{O}(\epsilon^2)$ la relation

$$v_1(\epsilon) = v_1 + \epsilon \sum_{i=2}^{n} \frac{u_i^* C v_1}{(\lambda_1 - \lambda_i) u_i^* v_i}(v_i - v_1 v_1^* v_i) + \mathcal{O}(\epsilon^2). \tag{1.10}$$

De cette formule, on voit que la condition du calcul du vecteur propre v_1 dépend de la grandeur $u_i^* v_i$ (comme c'est le cas pour la valeur propre; voir la formule (1.3)) et aussi de la distance entre λ_1 et les autres valeurs propres de A.

Un algorithme dangereux

La première méthode (déjà utilisée par Lagrange) pour calculer les valeurs propres d'une matrice A est la suivante: *calculer d'abord les coefficients du polynôme caractéristique $\chi_A(\lambda)$ et déterminer ensuite les zéros de ce polynôme*. Si la dimension de A est très petite (disons $n \leq 3$) ou si l'on fait le calcul en arithmétique exacte, cet algorithme peut être très utile. Par contre, si l'on fait le calcul en virgule flottante, cet algorithme peut donner des mauvaises surprises.

Considérons, par exemple, le problème de calculer les valeurs propres de la matrice diagonale

$$A = \mathrm{diag}(1, 2, 3, \ldots, n) \tag{1.11}$$

dont le polynôme caractéristique est

$$\begin{aligned} \chi_A(\lambda) &= (1-\lambda)(2-\lambda)(3-\lambda)\cdot\ldots\cdot(n-\lambda) \\ &= (-1)^n\lambda^n + a_{n-1}\lambda^{n-1} + a_{n-2}\lambda^{n-2} + \ldots + a_1\lambda + a_0. \end{aligned} \tag{1.12}$$

Les coefficients calculés satisfont $\widehat{a}_i = a_i(1+\epsilon_i)$ avec $|\epsilon_i| \leq$ *eps*. Cette perturbation dans les coefficients provoque une grande erreur dans les zéros de (1.12). Les résultats numériques pour $n = 9, 11, 13, 15$ (avec *eps* $\approx 6 \cdot 10^{-8}$, simple précision) sont dessinés dans la figure V.2.

Conclusion. Eviter le calcul des coefficients du polynôme caractéristique. Un tel algorithme est numériquement instable.

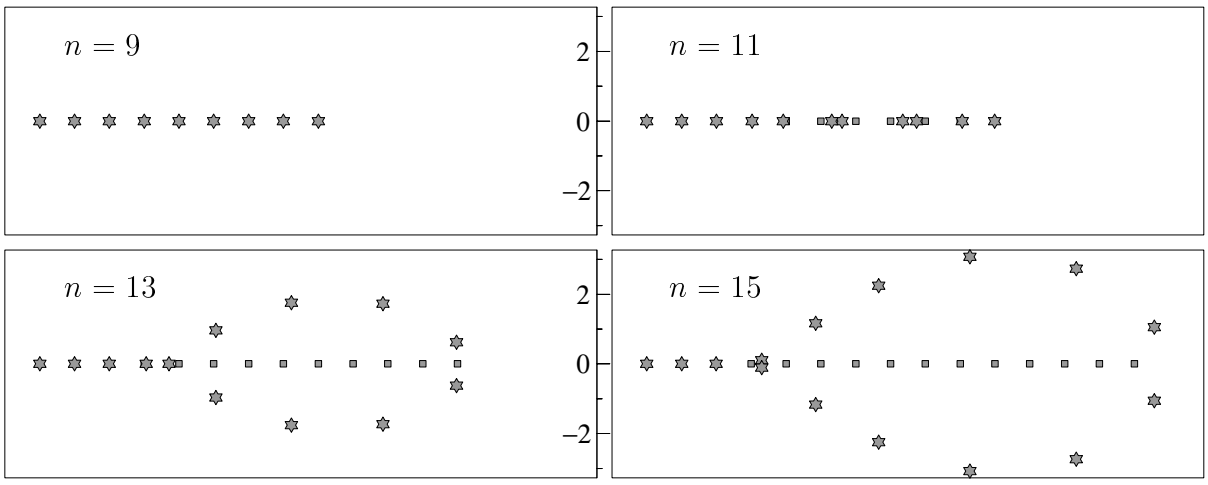


FIG. V.2: Zéros de (1.12) avec des coefficients perturbés

V.2 La méthode de la puissance

Un algorithme simple pour calculer les valeurs propres d'une matrice A est basé sur l'itération

$$y_{k+1} = Ay_k \tag{2.1}$$

où y_0 est un vecteur arbitraire. Dans le théorème suivant, on démontre que $y_k = A^k y_0$ (méthode de la *puissance*) tend vers un vecteur propre de A et que le *quotient de Rayleigh* $y_k^* A y_k / y_k^* y_k$ est une approximation d'une valeur propre de A.

Théorème 2.1 *Soit A une matrice diagonalisable de valeurs propres $\lambda_1, \dots, \lambda_n$ et de vecteurs propres $v_1, \dots, v_n$ (normalisés par $\|v_i\|_2 = 1$).*

Si $|\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq \dots \geq |\lambda_n|$, les vecteurs y_k de l'itération (2.1) vérifient

$$y_k = \lambda_1^k (a_1 v_1 + \mathcal{O}(|\lambda_2/\lambda_1|^k)) \tag{2.2}$$

(le nombre a_1 est défini par $y_0 = \sum_i a_i v_i$). Le quotient de Rayleigh satisfait (si $a_1 \neq 0$)

$$\frac{y_k^* A y_k}{y_k^* y_k} = \lambda_1 + \mathcal{O}\Big(\Big|\frac{\lambda_2}{\lambda_1}\Big|^k\Big). \tag{2.3}$$

Si A est une matrice normale (c.-à-d. que les vecteurs propres sont orthogonaux), l'erreur dans (2.3) est $\mathcal{O}(|\lambda_2/\lambda_1|^{2k})$.

Démonstration. Exprimons le vecteur de départ y_0 dans la base des vecteur propres, c.-à-d., $y_0 = \sum_{i=1}^n a_i v_i$. Par récurrence, on voit que

$$y_k = A^k y_0 = \sum_{i=1}^n a_i \lambda_i^k v_i = \lambda_1^k \Big(a_1 v_1 + \sum_{i=2}^n a_i \Big(\frac{\lambda_i}{\lambda_1}\Big)^k v_i \Big), \tag{2.4}$$

ce qui démontre la formule (2.2). De cette relation, on déduit que

$$y_k^* A y_k = y_k^* y_{k+1} = \sum_{i=1}^n |a_i|^2 |\lambda_i|^{2k} \lambda_i + \sum_{i \neq j} \overline{a}_i a_j \overline{\lambda}_i^k \lambda_j^{k+1} v_i^* v_j \tag{2.5}$$

$$y_k^* y_k = \sum_{i=1}^n |a_i|^2 |\lambda_i|^{2k} + \sum_{i \neq j} \overline{a}_i a_j \overline{\lambda}_i^k \lambda_j^k v_i^* v_j. \tag{2.6}$$

Si $a_1 \neq 0$, la formule (2.3) est une conséquence de

$$\frac{y_k^* A y_k}{y_k^* y_k} = \frac{|a_1|^2 \cdot |\lambda_1|^{2k} \cdot \lambda_1 \cdot (1 + \mathcal{O}(|\lambda_2/\lambda_1|^k))}{|a_1|^2 \cdot |\lambda_1|^{2k} \cdot (1 + \mathcal{O}(|\lambda_2/\lambda_1|^k))}. \tag{2.7}$$

Pour une matrice normale, le deuxième terme dans les formules (2.5) et (2.6) est absent et l'expression $\mathcal{O}(|\lambda_2/\lambda_1|^k)$ peut être remplacée par $\mathcal{O}(|\lambda_2/\lambda_1|^{2k})$ dans (2.7) et dans (2.3). □

Exemple 2.2 Considérons la matrice $A = \begin{pmatrix} 2 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 2 \end{pmatrix}$ dont la valeur propre la plus grande est $\lambda_1 = 2(1 + \cos(\pi/4)) \approx 3.414213562373095$. Quelques itérations de la méthode de la puissance nous donnent

$$y_0 = (1,1,1)^T, \qquad y_1 = (3,4,3)^T, \qquad y_2 = (10,14,10)^T$$

et une première approximation de λ_1 est obtenue par

$$\frac{y_1^* A y_1}{y_1^* y_1} = \frac{y_1^* y_2}{y_1^* y_1} = \frac{116}{34} \approx 3.41176.$$

Remarques. Les éléments du vecteur y_k croissent exponentiellement avec k. Il est alors recommandé de normaliser y_k après chaque itération, c.-à-d. de remplacer y_k par $y_k/\|y_k\|$. Sinon, on risque un "overflow". Si $|\lambda_2/\lambda_1|$ est proche de 1, la convergence est très lente. Pour accélérer la convergence, on utilise la modification suivante.

Méthode de la puissance inverse de Wielandt

Supposons qu'on connaisse une approximation μ de la valeur propre cherchée λ_1 (il n'est pas nécessaire de supposer que λ_1 soit la plus grande valeur propre de A). L'idée est d'appliquer l'itération (2.1) à la matrice $(A-\mu I)^{-1}$. Les valeurs propres de cette matrice sont $(\lambda_i-\mu)^{-1}$. Si μ est proche de λ_1, on a

$$\frac{1}{|\lambda_1-\mu|} \gg \frac{1}{|\lambda_i-\mu|} \qquad \text{pour} \qquad i \geq 2$$

et la convergence va être très rapide. L'itération devient alors $y_{k+1}=(A-\mu I)^{-1}y_k$ ou

$$(A-\mu I)y_{k+1} = y_k. \tag{2.8}$$

Après avoir calculé la décomposition LR de la matrice $A-\mu I$, une itération de (2.8) ne coûte pas plus cher qu'une de (2.1).

Exemple 2.3 Pour la matrice A de l'exemple précédent, choisissons $\mu = 3.41$ et $y_0 = (1, 1.4, 1)^T$. Deux itérations de (2.8) nous donnent

$$y_1 = \begin{pmatrix} 236.134453781513 \\ 333.949579831933 \\ 236.134453781513 \end{pmatrix}, \qquad y_2 = \begin{pmatrix} 56041.9461902408 \\ 79255.2785820210 \\ 56041.9461902408 \end{pmatrix}$$

et on obtient

$$\frac{1}{\lambda_1-3.41} \approx \frac{y_1^*(A-\mu I)^{-1}y_1}{y_1^*y_1} = \frac{y_1^*y_2}{y_1^*y_1} \approx 237.328870774159.$$

De cette relation, on calcule λ_1 et on obtient l'approximation 3.41421356237333. Les 13 premiers chiffres sont corrects.

La méthode de la puissance (et celle de Wielandt) est importante pour la compréhension d'autres algorithmes. Si l'on veut calculer toutes les valeurs propres d'une matrice, on utilise des méthodes encore plus sophistiquées. En pratique, on procède de la manière suivante:

- on distingue les cas: A symétrique ou A quelconque.
- on cherche T telle que $T^{-1}AT = H$ devienne une matrice de Hessenberg (ou une matrice tridiagonale, si A est symétrique); voir V.3.
- on applique l'algorithme QR à la matrice H (voir V.6).
- si H est une matrice tridiagonale et symétrique, on peut également appliquer la méthode de bissection (voir V.4).

V.3 Transformation sous forme de Hessenberg (ou tridiagonale)

Avec la transformation $v = Tu$ (où T est une matrice inversible) le problème

$$Av = \lambda v \qquad \text{devient} \qquad T^{-1}ATu = \lambda u.$$

Donc, les valeurs propres de A et de $T^{-1}AT$ sont les mêmes et les vecteurs propres v_i de A se transforment par $v_i = Tu_i$. Le but de ce paragraphe est de trouver une matrice T telle que $T^{-1}AT$ devienne "plus simple". La situation idéale serait trouvée si $T^{-1}AT$ devenait diagonale ou triangulaire – mais une telle transformation nécessiterait déjà la connaissance des valeurs propres.

Alors, on cherche T tel que $T^{-1}AT$ soit sous *forme de Hessenberg*

$$T^{-1}AT = H = \begin{pmatrix} * & * & \dots & \dots & * \\ * & * & \ddots & & \vdots \\ & * & \ddots & \ddots & * \\ & & \ddots & \ddots & * \\ & & & * & * \end{pmatrix}, \tag{3.1}$$

c.-à-d., $h_{ij} = 0$ pour $i > j + 1$. Pour arriver à ce but, nous considérons deux algorithmes.

a) A l'aide des transformations élémentaires

Comme pour l'élimination de Gauss, nous utilisons les transformations L_i pour faire apparaître les zéros – colonne par colonne – dans (3.1).

Dans un premier pas, nous choisissons $k \geq 2$ tel que $|a_{k1}| \geq |a_{j1}|$ pour $j \geq 2$ et nous permutons les lignes 2 et k, c.-à-d., nous formons PA où P est une matrice de permutation convenable. Pour ne pas changer les valeurs propres, il faut également permuter les colonnes 2 et k (ceci correspond au calcul de $A' = PAP^{-1}$ car $P^2 = I$ et donc $P = P^{-1}$). Si $a'_{21} = 0$, on a aussi $a'_{i1} = 0$ pour $i \geq 3$ et le premier pas est terminé. Sinon, nous déterminons

$$L_2 = \begin{pmatrix} 1 & & & & \\ 0 & 1 & & & \\ 0 & -\ell_{32} & 1 & & \\ \vdots & \vdots & \ddots & \ddots & \\ 0 & -\ell_{n2} & \dots & 0 & 1 \end{pmatrix} \qquad \text{telle que} \qquad L_2 A' = \begin{pmatrix} a'_{11} & a'_{12} & \dots & a'_{1n} \\ a'_{21} & a'_{22} & \dots & a'_{2n} \\ 0 & * & \dots & * \\ \vdots & \vdots & & \vdots \\ 0 & * & \dots & * \end{pmatrix}.$$

Pour ceci, on définit $\ell_{i2} = a'_{i1}/a'_{21}$. Une multiplication à droite avec

$$L_2^{-1} = \begin{pmatrix} 1 & & & & \\ 0 & 1 & & & \\ 0 & \ell_{32} & 1 & & \\ \vdots & \vdots & \ddots & \ddots & \\ 0 & \ell_{n2} & \dots & 0 & 1 \end{pmatrix}$$

ne change pas la première colonne de $L_2 A'$.

On répète la même procédure avec la sous-matrice de $L_2 A' L_2^{-1}$ de dimension $n - 1$, et ainsi de suite. A cause des multiplications à droite avec L_i^{-1}, cet algorithme coûte deux fois plus cher que l'élimination de Gauss (donc $\approx 2n^3/3$ opérations).

Exemple. Pour la matrice

$$A = \begin{pmatrix} 3 & 2 & 1 \\ 2 & 1 & 3 \\ 1 & 3 & 1 \end{pmatrix} \qquad \text{on prend} \qquad L_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -1/2 & 1 \end{pmatrix}$$

et on obtient

$$L_2 A = \begin{pmatrix} 3 & 2 & 1 \\ 2 & 1 & 3 \\ 0 & 5/2 & -1/2 \end{pmatrix}, \qquad \text{puis} \qquad L_2 A L_2^{-1} = \begin{pmatrix} 3 & 5/2 & 1 \\ 2 & 5/2 & 3 \\ 0 & 9/4 & -1/2 \end{pmatrix} = H.$$

Cet exemple montre un désavantage de cet algorithme : si l'on part avec une matrice symétrique A, la matrice de Hessenberg H, obtenue par cet algorithme, n'est plus symétrique en général.

b) A l'aide des transformations orthogonales

Il est souvent préférable de travailler avec des réflexions de Householder (voir le paragraphe IV.7). Commençons par une réflexion pour les coordonnées $2, \ldots, n$ laissant fixe la première coordonnée : $\bar{Q}_2 = I - 2\bar{u}_2\bar{u}_2^T$ ($\|\bar{u}_2\|_2 = 1$) tel que $\bar{Q}_2\bar{A}_1 = \alpha_2 e_1$ où $\bar{A}_1 = (a_{21}, \ldots, a_{n1})^T$. En posant $u_2 = (0, \bar{u}_2)^T$ et $Q_2 = I - 2u_2u_2^T$, la matrice Q_2A contient des zéros dans la première colonne à partir du troisième élément. La multiplication à droite avec $Q_2^{-1} = Q_2^T = Q_2$ ne change pas cette colonne :

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \quad \overset{Q_2 \cdot A}{\longmapsto} \quad \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ \boxed{\alpha_2} & \boxed{*} & \boxed{*} \\ \boxed{0} & \boxed{*} & \boxed{*} \end{pmatrix} \quad \overset{Q_2 \cdot A \cdot Q_2}{\longmapsto} \quad \begin{pmatrix} a_{11} & \boxed{*} & \boxed{*} \\ \alpha_2 & \boxed{*} & \boxed{*} \\ 0 & \boxed{*} & \boxed{*} \end{pmatrix}.$$

Dans le pas suivant, on applique la même procédure à la sous-matrice de dimension $n-1$, etc. Finalement, on arrive à la forme de Hessenberg (3.1) avec la tranformation $T^{-1} = Q_{n-1} \cdot \ldots \cdot Q_2$, qui est une matrice orthogonale (c.-à-d., $T^{-1} = T^T$).

Nous avons un double avantage avec cet algorithme :

- il ne faut pas faire une recherche de pivot ;
- si A est symétrique, alors $T^{-1}AT$ est aussi symétrique, et donc *tridiagonale*.

V.4 Méthode de bissection pour des matrices tridiagonales

Considérons une matrice symétrique tridiagonale

$$A = \begin{pmatrix} d_1 & e_2 & & & \\ e_2 & d_2 & e_3 & & \\ & e_3 & \ddots & \ddots & \\ & & \ddots & \ddots & e_n \\ & & & e_n & d_n \end{pmatrix}. \tag{4.1}$$

On observe tout d'abord que si un élément e_i est nul, la matrice A est déjà décomposée en deux sous-matrices du même type, qui ensemble fournissent les valeurs propres de A.

On peut donc supposer, sans restreindre la généralité, que

$$e_i \neq 0 \qquad \text{pour} \qquad i = 2, \ldots, n. \tag{4.2}$$

Pour cette matrice, il est possible de calculer la valeur $\chi_A(\lambda)$ du polynôme caractéristique sans connaître ses coefficients. En effet, si l'on pose

$$A_1 = (d_1), \qquad A_2 = \begin{pmatrix} d_1 & e_2 \\ e_2 & d_2 \end{pmatrix}, \qquad A_3 = \begin{pmatrix} d_1 & e_2 & \\ e_2 & d_2 & e_3 \\ & e_3 & d_3 \end{pmatrix}, \qquad \ldots$$

et si l'on définit $p_i(\lambda) := \det(A_i - \lambda I)$, on obtient

$$\begin{aligned} p_0(\lambda) &= 1 \\ p_1(\lambda) &= d_1 - \lambda \\ p_i(\lambda) &= (d_i - \lambda)p_{i-1}(\lambda) - e_i^2 p_{i-2}(\lambda), \qquad i = 2, \ldots, n. \end{aligned} \tag{4.3}$$

La formule de récurrence dans (4.3) est obtenue en développant le déterminant de la matrice $A_i - \lambda I$ par rapport à la dernière ligne (ou colonne).

En principe, on peut maintenant calculer les valeurs propres de A (c.-à-d. les zéros de $p_n(\lambda)$) de la manière suivante : chercher un intervalle où $p_n(\lambda)$ change de signe et localiser une racine de $p_n(\lambda) = 0$ par bissection. Les évaluations de $p_n(\lambda)$ sont faites à l'aide de la formule (4.3). Mais il existe une astuce interessante qui permet d'améliorer cet algorithme.

Théorème 4.1 *Si (4.2) est vérifié, les polynômes $p_i(\lambda)$ définis par (4.3) satisfont*

a) $p_n'(\widehat{\lambda})\, p_{n-1}(\widehat{\lambda}) < 0$ *si* $p_n(\widehat{\lambda}) = 0$ *($\widehat{\lambda} \in \mathbb{R}$);*

b) $p_{i-1}(\widehat{\lambda})\, p_{i+1}(\widehat{\lambda}) < 0$ *si* $p_i(\widehat{\lambda}) = 0$ *pour un* $i \in \{1, \ldots, n-1\}$*;*

c) $p_0(\lambda)$ *ne change pas de signe sur* $\mathbb{R}$*.*

Démonstration. L'affirmation (c) est triviale.

Si $p_i(\widehat{\lambda}) = 0$ pour un $i \in \{1, \ldots, n-1\}$, la formule de récurrence (4.3) donne l'inégalité $p_{i+1}(\widehat{\lambda}) p_{i-1}(\widehat{\lambda}) \leq 0$. Pour démontrer (b), il suffit d'exclure le cas $p_{i+1}(\widehat{\lambda}) p_{i-1}(\widehat{\lambda}) = 0$. Si deux valeurs consécutives de la suite $\{p_i(\widehat{\lambda})\}$ sont nulles, la formule de récurrence montre que $p_i(\widehat{\lambda}) = 0$ pour tout i, ce qui contredit $p_0(\widehat{\lambda}) = 1$.

Nous démontrons par récurrence que

$$\text{toutes les racines de } p_i(\lambda) \text{ sont réelles, simples et séparées par celles de } p_{i-1}(\lambda). \tag{4.4}$$

Il n'y a rien à démontrer pour $i = 1$. Supposons la propriété vraie pour i et montrons qu'elle est encore vraie pour $i+1$. Comme les zéros $\lambda_1 < \ldots < \lambda_i$ de $p_i(\lambda)$ sont séparés par ceux de $p_{i-1}(\lambda)$ et comme $p_{i-1}(-\infty) = +\infty$, nous avons $\operatorname{sign} p_{i-1}(\lambda_j) = (-1)^{j+1}$. Alors, on déduit de (b) que $\operatorname{sign} p_{i+1}(\lambda_j) = (-1)^j$. Ceci et le fait que $p_{i+1}(\lambda) = (-1)^{i+1}\lambda^{i+1} + \ldots$ montrent que $p_{i+1}(\lambda)$ possède un zéro réel dans chacun des intervalles ouverts $(-\infty, \lambda_1)$, (λ_1, λ_2), $\ldots$, (λ_i, ∞).

L'affirmation (a) est maintenant une conséquence de (b) et de (4.4); voir la figure V.3. □

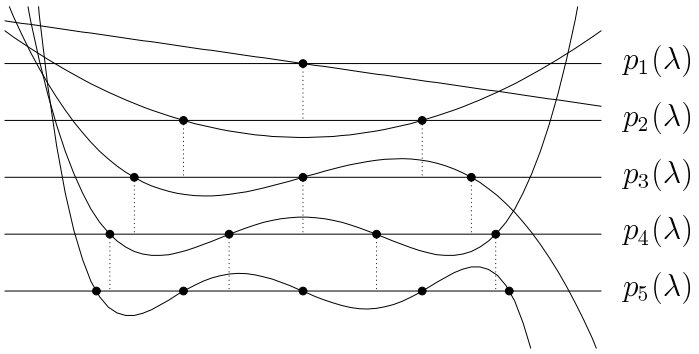


FIG. V.3: Suite de Sturm

Définition 4.2 (suite de Sturm) *Une suite $(p_0, p_1, \ldots, p_n)$ de polynômes à coefficients réelles s'appelle une suite de Sturm*[1]*, si elle vérifie les conditions (a), (b), (c) du Théorème 4.1.*

Théorème 4.3 *Considérons une suite de Sturm $(p_0, p_1, \ldots, p_n)$. Si l'on définit*

$$\omega(\lambda) = \text{ nombre de changements de signes de } \{p_0(\lambda), p_1(\lambda), \ldots, p_n(\lambda)\}, \tag{4.5}$$

alors le polynôme $p_n(\lambda)$ possède exactement

$$\omega(b) - \omega(a) \tag{4.6}$$

zéros dans l'intervalle $[a, b)$ (si $p_i(\lambda) = 0$, on définit $\operatorname{sign} p_i(\lambda) = \operatorname{sign} p_{i-1}(\lambda)$*).*

Démonstration. Par continuité, l'entier $\omega(\lambda)$ peut changer sa valeur seulement si une valeur des fonctions $p_i(\lambda)$ devient nulle. La fonction $p_0(\lambda)$ ne change pas de signe. Supposons alors que $p_i(\widehat{\lambda}) = 0$ pour un $i \in \{1, \ldots, n-1\}$. La condition (b) et la continuité de $p_j(\lambda)$ montrent que seulement les deux situations suivantes sont possibles (ϵ petit):

[1] Jacques Charles François Sturm (1803 – 1855), né le 29 septembre 1803 à Genève.

	$\widehat{\lambda}-\epsilon$	$\widehat{\lambda}$	$\widehat{\lambda}+\epsilon$
$p_{i-1}(\lambda)$	$+$	$+$	$+$
$p_i(\lambda)$	$\pm$	0	$\mp$
$p_{i+1}(\lambda)$	$-$	$-$	$-$

	$\widehat{\lambda}-\epsilon$	$\widehat{\lambda}$	$\widehat{\lambda}+\epsilon$
$p_{i-1}(\lambda)$	$-$	$-$	$-$
$p_i(\lambda)$	$\pm$	0	$\mp$
$p_{i+1}(\lambda)$	$+$	$+$	$+$

Chaque fois, on a $\omega(\widehat{\lambda}+\epsilon)=\omega(\widehat{\lambda})=\omega(\widehat{\lambda}-\epsilon)$ et la valeur de $\omega(\lambda)$ ne change pas si λ traverse un zéro de $p_i(\lambda)$ pour $i\in\{1,\ldots,n-1\}$.

Il reste à étudier la fonction $\omega(\lambda)$ dans un voisinage d'un zéro $\widehat{\lambda}$ de $p_n(\lambda)$. La propriété (a) implique que pour les signes de $p_j(\lambda)$ on a seulement les deux possibilités suivantes:

	$\widehat{\lambda}-\epsilon$	$\widehat{\lambda}$	$\widehat{\lambda}+\epsilon$
$p_{n-1}(\lambda)$	$+$	$+$	$+$
$p_n(\lambda)$	$+$	0	$-$

	$\widehat{\lambda}-\epsilon$	$\widehat{\lambda}$	$\widehat{\lambda}+\epsilon$
$p_{n-1}(\lambda)$	$-$	$-$	$-$
$p_n(\lambda)$	$-$	0	$+$

c.-à-d., $\omega(\widehat{\lambda}+\epsilon)=\omega(\widehat{\lambda}-\epsilon)+1$. Ceci démontre que la fonction $\omega(\lambda)$ est constante par morceaux et augmente de 1 sa valeur si λ traverse un zéro de $p_n(\lambda)$. □

Méthode de bissection. Si l'on applique ce théorème à la suite (4.3), la différence $\omega(b)-\omega(a)$ est égale au nombre de valeurs propres de (4.1) dans l'intervalle $[a,b)$. On obtient toutes les valeurs propres de A de la manière suivante:

- on cherche un intervalle $[a,b]$ qui contienne toutes les valeurs propres de A (p.ex., en appliquant le théorème de Gershgorin). On a donc que $\omega(a)=0$ et $\omega(b)=n$.
- on pose $c=(a+b)/2$ et on calcule $\omega(c)$. Les différences $\omega(c)-\omega(a)$ et $\omega(b)-\omega(c)$ indiquent combien de valeurs propres de A sont dans $[a,c)$ et combien sont dans $[c,b)$.
- on continue à diviser les intervalles qui contiennent au moins une valeur propre de A.

On peut facilement modifier cet algorithme pour calculer la valeur propre la plus petite ou la 3ème plus grande valeur propre, etc.

Pour éviter un "overflow" dans le calcul de $p_n(\lambda)$ (si n et λ sont grands), il vaut mieux travailler avec

$$f_i(\lambda):=p_i(\lambda)/p_{i-1}(\lambda)\qquad i=1,\ldots,n \tag{4.7}$$

et utiliser le fait que

$$\omega(\lambda)=\quad\text{nombre d'éléments négatifs parmi }\ \{f_1(\lambda),\ldots,f_n(\lambda)\} \tag{4.8}$$

(attention: si $p_{i-1}(\lambda)$ est zéro, on pose $f_i(\lambda)=-\infty$; cette valeur compte pour un élément négatif). Pour une programmation de l'algorithme, on utilise la récurrence

$$\begin{aligned} f_1(\lambda)&=d_1-\lambda\\ f_i(\lambda)&=d_i-\lambda-\begin{cases} e_i^2/f_{i-1}(\lambda) & \text{si } f_{i-1}(\lambda)\neq 0\\ |e_i|/\textit{eps} & \text{si } f_{i-1}(\lambda)=0\,. \end{cases} \end{aligned} \tag{4.9}$$

La formule pour le cas $f_{i-1}(\lambda)\neq 0$ est une conséquence de (4.3). Si $f_{i-1}(\lambda)=0$ (c.-à-d., $p_{i-1}(\lambda)=0$), on remplace cette valeur par $|e_i|\cdot\textit{eps}$. Ceci correspond à ajouter la perturbation $|e_i|\cdot\textit{eps}$ à d_{i-1}.

V.5 L'itération orthogonale

Dans ce paragraphe, nous allons généraliser la méthode de la puissance (voir le paragraphe V.2) afin de pouvoir calculer les deux (trois, ...) valeurs propres dominantes en même temps. Cette généralisation motivera l'itération QR qui constitue l'algorithme le plus important pour le calcul des valeurs propres d'une matrice.

Généralisation de la méthode de la puissance (pour calculer les deux valeurs propres dominantes). Considérons une matrice A dont les valeurs propres satisfont

$$|\lambda_1| > |\lambda_2| > \ldots > |\lambda_n|. \tag{5.1}$$

La méthode de la puissance est basée sur l'itération $y_{k+1} = Ay_k$ (voir (2.1)) et nous permet d'obtenir une approximation de λ_1 à l'aide du quotient de Rayleigh. Pour calculer (en même temps) la deuxième valeur propre λ_2, nous prenons deux vecteurs y_0 et z_0 satisfaisant $y_0^* z_0 = 0$ et nous considérons l'itération

$$\begin{aligned} y_{k+1} &= Ay_k \\ z_{k+1} &= Az_k - \beta_{k+1} y_{k+1} \end{aligned} \tag{5.2}$$

où β_{k+1} est déterminé par la condition $y_{k+1}^* z_{k+1} = 0$. Par induction, on voit que

$$\begin{aligned} y_k &= A^k y_0 \\ z_k &= A^k z_0 - \gamma_k y_k \end{aligned} \tag{5.3}$$

où γ_k est tel que

$$y_k^* z_k = 0. \tag{5.4}$$

Ceci signifie que le calcul de $\{z_k\}$ correspond à la méthode de la puissance appliquée à z_0, combinée avec une orthogonalisation (projection de $A^k z_0$ sur le complément orthogonal de y_k).

En exprimant les vecteurs initiaux dans la base de vecteurs propres $v_1, \ldots, v_n$ de la matrice A (on suppose $\|v_i\|_2 = 1$),

$$y_0 = \sum_{i=1}^{n} a_i v_i, \qquad\qquad z_0 = \sum_{i=1}^{n} b_i v_i,$$

les vecteurs y_k, z_k deviennent

$$y_k = \sum_{i=1}^{n} a_i \lambda_i^k v_i, \qquad\qquad z_k = \sum_{i=1}^{n} (b_i - \gamma_k a_i) \lambda_i^k v_i. \tag{5.5}$$

Comme nous l'avons constaté dans le paragraphe V.2, pour $k \to \infty$, le terme $a_1 \lambda_1^k v_1$ est dominant dans y_k (si $a_1 \neq 0$) et on obtient une approximation du premier vecteur propre v_1. Que peut-on dire pour la suite z_k?

La condition (5.4) d'orthogonalité implique que

$$\sum_{i=1}^{n} \sum_{j=1}^{n} \bar{a}_i (b_j - \gamma_k a_j) \bar{\lambda}_i^k \lambda_j^k v_i^* v_j = 0. \tag{5.6}$$

Cette relation définit γ_k. Comme le terme avec $i = j = 1$ est dominant, on voit que $\gamma_k \approx b_1/a_1$. Par la suite, nous allons supposer que $a_1 \neq 0$ et $a_1 b_2 - a_2 b_1 \neq 0$. En divisant (5.6) par $\bar{\lambda}_1^k$, on obtient

$$\bar{a}_1 (b_1 - \gamma_k a_1) \lambda_1^k \Big(1 + \mathcal{O}(|\lambda_2/\lambda_1|^k)\Big) = -\bar{a}_1 (b_2 - \gamma_k a_2) \lambda_2^k \Big(v_1^* v_2 + \mathcal{O}(|\lambda_2/\lambda_1|^k) + \mathcal{O}(|\lambda_3/\lambda_2|^k)\Big).$$

Maintenant, on peut insérer cette formule dans (5.5) et on en déduit

$$z_k = \lambda_2^k(b_2 - \gamma_k a_2)\Big(v_2 - v_1^* v_2 \cdot v_1 + \mathcal{O}(|\lambda_2/\lambda_1|^k) + \mathcal{O}(|\lambda_3/\lambda_2|^k)\Big). \tag{5.7}$$

Visiblement, le vecteur z_k s'approche (pour $k \to \infty$) d'un multiple de $v_2 - v_1^* v_2 \cdot v_1$, qui est la projection orthogonale de v_2 à l'hyperplan $v_1^\perp$. Concernant les valeurs propres, on a le résultat suivant.

Théorème 5.1 *Considérons les vecteurs y_k, z_k donnés par (5.2) et notons*

$$U_k = (y_k/\|y_k\|_2,\ z_k/\|z_k\|_2) \tag{5.8}$$

(observer que $U_k^* U_k = I$). Si (5.1) est vérifié, on a que

$$U_k^* A U_k \ \to\ \begin{pmatrix} \lambda_1 & * \\ 0 & \lambda_2 \end{pmatrix} \qquad \textit{pour} \qquad k \to \infty. \tag{5.9}$$

Démonstration. L'élément (1,1) de la matrice $U_k^* A U_k$ est le quotient de Rayleigh (2.3) qui converge vers λ_1. En utilisant (5.7), on voit que l'élément (2,2) satisfait

$$\frac{z_k^* A z_k}{z_k^* z_k} \ \to\ \frac{(v_2 - v_1^* v_2 \cdot v_1)^*(\lambda_2 v_2 - \lambda_1 v_1^* v_2 \cdot v_1)}{(v_2 - v_1^* v_2 \cdot v_1)^*(v_2 - v_1^* v_2 \cdot v_1)} = \frac{\lambda_2(1 - |v_1^* v_2|^2)}{1 - |v_1^* v_2|^2} = \lambda_2.$$

De façon similaire, on obtient pour l'élément (2,1)

$$\frac{z_k^* A y_k}{\|z_k\|_2 \cdot \|y_k\|_2} \ \to\ \frac{(v_2 - v_1^* v_2 \cdot v_1)^* \lambda_1 v_1}{\|v_2 - v_1^* v_2 \cdot v_1\|_2 \cdot \|v_1\|_2} = 0.$$

Finalement, l'élément (1,2) de $U_k^* A U_k$ satisfait

$$\frac{y_k^* A z_k}{\|y_k\|_2 \cdot \|z_k\|_2} \ \to\ \frac{v_1^*(\lambda_2 v_2 - \lambda_1 v_1^* v_2 \cdot v_1)}{\|v_1\|_2 \cdot \|v_2 - v_1^* v_2 \cdot v_1\|_2} = \frac{(\lambda_2 - \lambda_1) v_1^* v_2}{\sqrt{1 - |v_1^* v_2|^2}}.$$

Cette expression est en général non nulle. □

Remarque. Avec la notation (5.8), l'itération (5.2) peut être écrite sous la forme

$$A U_k = U_{k+1} R_{k+1} \tag{5.10}$$

où R_{k+1} est une matrice 2×2 qui est triangulaire supérieure.

Méthode de la puissance (pour le calcul de toutes les valeurs propres) ou simplement **itération orthogonale.** La généralisation de l'algorithme précédent au cas où l'on veut calculer toutes les valeurs propres d'une matrice est évidente: on choisit une matrice orthogonale U_0, c.-à-d., on choisit n vecteurs orthogonaux (les colonnes de U_0) qui jouent le rôle de y_0, z_0, etc. Puis, on effectue l'itération

for $k = 1, 2, \ldots$
$\quad Z_k = A U_{k-1}$
$\quad U_k R_k = Z_k$ (décomposition QR)
end

Si (5.1) est vérifié et si la matrice U_0 est bien choisie ($a_1 \neq 0$, $a_1 b_2 - a_2 b_1 \neq 0$, etc), une généralisation du théorème précédent donne la convergence de

$$T_k := U_k^* A U_k \tag{5.11}$$

vers une matrice triangulaire dont les éléments de la diagonale sont les valeurs propres de A. On a donc transformé A en forme triangulaire à l'aide d'une matrice orthogonale (*décomposition de Schur*).

Il y a une possibilité intéressante pour calculer T_k de (5.11) directement à partir de T_{k-1}. D'une part, on déduit de (5.10) que

$$T_{k-1} = U_{k-1}^* A U_{k-1} = (U_{k-1}^* U_k) R_k. \tag{5.12a}$$

D'autre part, on a

$$T_k = U_k^* A U_k = U_k^* A U_{k-1} U_{k-1}^* U_k = R_k (U_{k-1}^* U_k). \tag{5.12b}$$

On calcule la décomposition QR de la matrice T_{k-1} et on échange les deux matrices de cette décomposition pour obtenir T_k.

V.6 L' algorithme QR

> La méthode QR, due à J.C.F. Francis et à V.N. Kublanovskaya, est la méthode la plus couramment utilisée pour le calcul de l'ensemble des valeurs propres . . . (P.G. Ciarlet 1982)
> . . . the QR iteration, and it forms the backbone of the most effective algorithm for computing the Schur decomposition. (G.H. Golub & C.F. van Loan 1989)

La version simple du célèbre algorithme QR n'est rien d'autre que la méthode du paragraphe précédent. En effet, si l'on pose $Q_k = U_{k-1}^* U_k$ et si l'on commence l'itération avec $U_0 = I$, les formules (5.11) et (5.12) nous permettent d'écrire l'algorithme précédent comme suit:

$T_0 = A$
for $k = 1, 2, \ldots$
  $Q_k R_k = T_{k-1}$ (décomposition QR)
  $T_k = R_k Q_k$
end

Les T_k qui sont les mêmes que dans le paragraphe V.5, convergent (en général) vers une matrice triangulaire. Ceci nous permet d'obtenir toutes les valeurs propres de la matrice A car les T_k ont les mêmes valeurs propres que A (voir (5.11)).

Cet algorithme important a été développé indépendamment par J.G.F. Francis (1961) et par V.N. Kublanovskaya (1961). Un algorithme similaire, qui utilise la décomposition LR à la place de la décomposition QR, a été introduit par H. Rutishauser (1958).

Exemple numérique. Appliquons la méthode QR à la matrice

$$A = \begin{pmatrix} 10 & 2 & 3 & 5 \\ 3 & 6 & 8 & 4 \\ 0 & 5 & 4 & 3 \\ 0 & 0 & 4 & 3 \end{pmatrix}. \tag{6.1}$$

On peut montrer (voir exercice 14) que, pour une matrice de Hessenberg A, toutes les matrices T_k sont aussi sous forme de Hessenberg. Pour étudier la convergence vers une matrice triangulaire, il suffit alors de considérer les éléments $t_{i+1,i}^{(k)}$ ($i = 1, \ldots, n-1$) de la sous-diagonale. On constate que

$$\frac{t_{i+1,i}^{(k+1)}}{t_{i+1,i}^{(k)}} \approx \frac{\lambda_{i+1}}{\lambda_i} \tag{6.2}$$

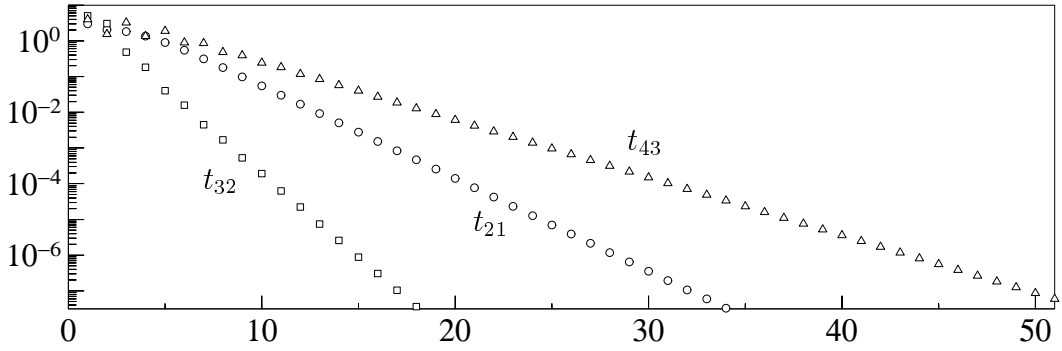


FIG. V.4: Convergence de la méthode QR (sans shift)

($\lambda_1 \approx 14.3$, $\lambda_2 \approx 7.86$, $\lambda_3 \approx 2.70$, $\lambda_4 \approx -1.86$). Comme $|\lambda_{i+1}/\lambda_i| < 1$, les éléments $t^{(k)}_{i+1,i}$ convergent, pour $k \to \infty$, linéairement vers 0 (voir la figure V.4, où les valeurs $|t^{(k)}_{i+1,i}|$ sont dessinées en fonction du nombre k de l'itération).

Remarques. (a) Comme le calcul de la décomposition QR d'une matrice pleine est très coûteux ($\mathcal{O}(n^3)$ opérations), on applique l'algorithme QR uniquement aux matrices de Hessenberg. Dans cette situation une itération nécessite seulement $\mathcal{O}(n^2)$ opérations.

(b) La convergence est très lente en général (seulement *linéaire*). Pour rendre efficace cet algorithme, il faut absolument trouver un moyen pour accélérer la convergence.

(c) Considérons la situation où A est une matrice réelle qui possède des valeurs propres complexes (l'hypothèse (5.1) est violée). L'algorithme QR produit une suite de matrices T_k qui sont toutes réelles. Dans cette situation, les T_k ne convergent pas vers une matrice triangulaire, mais deviennent *triangulaires par blocs* (sans démonstration). Comme la dimension des blocs dans la diagonale vaut en général 1 ou 2, on obtient également des approximations des valeurs propres.

Accélération de la convergence

D'après l'observation (6.2), nous savons que

$$t^{(k)}_{n,n-1} = \mathcal{O}(|\lambda_n/\lambda_{n-1}|^k). \tag{6.3}$$

La convergence vers zéro de cet élément ne va être rapide que si $|\lambda_n| \ll |\lambda_{n-1}|$. Une idée géniale est d'appliquer l'algorithme QR à la matrice $A - pI$ où $p \approx \lambda_n$. Comme les valeurs propres de $A - pI$ sont $\lambda_i - p$, on a la propriété $|\lambda_n - p| \ll |\lambda_i - p|$ pour $i = 1, \ldots, n-1$ et l'élément $t^{(k)}_{n,n-1}$ va converger rapidement vers zéro. Rien ne nous empêche d'améliorer l'approximation p après chaque itération. L'algorithme QR avec "shift" devient alors:

$T_0 = A$
for $k = 1, 2, \ldots$
  déterminer le paramètre p_{k-1}
  $Q_k R_k = T_{k-1} - p_{k-1} I$ (décomposition QR)
  $T_k = R_k Q_k + p_{k-1} I$
end

Les matrices T_k de cette itération satisfont

$$Q_k^* T_{k-1} Q_k = Q_k^* (Q_k R_k + p_{k-1} I) Q_k = R_k Q_k + p_{k-1} I = T_k. \tag{6.4}$$

Ceci implique que, indépendamment de la suite p_k, les matrices T_k ont toutes les mêmes valeurs propres que $T_0 = A$.

Pour décrire complètement l'algorithme QR avec shift, il faut encore discuter le choix du paramètre p_k et il faut donner un critère pour arrêter l'itération.

Choix du "shift"-paramètre. On a plusieurs possibilités:
- $p_k = t_{nn}^{(k)}$: ce choix marche très bien si les valeurs propres de la matrice sont réelles.
- on considère la matrice

$$\begin{pmatrix} t_{n-1,n-1}^{(k)} & t_{n-1,n}^{(k)} \\ t_{n,n-1}^{(k)} & t_{n,n}^{(k)} \end{pmatrix}. \tag{6.5}$$

Si les valeurs propres de (6.5) sont réelles, on choisit pour p_k celle qui est la plus proche de $t_{n,n}^{(k)}$. Si elles sont de la forme $\alpha \pm i\beta$ avec $\beta \neq 0$ (donc complexes), on prend d'abord $p_k = \alpha + i\beta$ et pour l'itération suivante $p_{k+1} = \alpha - i\beta$.

Critère pour arrêter l'itération. L'idée est d'itérer jusqu'à ce que $t_{n,n-1}^{(k)}$ ou $t_{n-1,n-2}^{(k)}$ soit suffisamment petit. Plus précisément, on arrête l'itération quand

$$|t_{\ell,\ell-1}^{(k)}| \leq \mathit{eps} \cdot (|t_{\ell-1,\ell-1}^{(k)}| + |t_{\ell,\ell}^{(k)}|) \qquad \text{pour} \quad \ell = n \quad \text{ou} \quad \ell = n-1. \tag{6.6}$$

- Si (6.6) est vérifié pour $\ell = n$, on accepte $t_{n,n}^{(k)}$ comme approximation de λ_n et on continue l'itération avec la matrice $(t_{ij}^{(k)})_{i,j \leq n-1}$.
- Si (6.6) est vérifié pour $\ell = n-1$, on accepte les deux valeurs propres de (6.5) comme approximations de λ_n et λ_{n-1} et on continue l'itération avec la matrice $(t_{ij}^{(k)})_{i,j \leq n-2}$.

Exemple numérique. Nous avons appliqué l'algorithme QR à la matrice (6.1) avec le shift $p_k = t_{nn}^{(k)}$. La convergence de $t_{i+1,i}^{(k)}$ vers zéro est illustrée dans la figure V.5. Une comparaison avec la figure V.4 nous montre que la convergence est beaucoup plus rapide (convergence quadratique). Après 5 itérations, on a $|t_{43}^{(k)}| \leq 10^{-15}$. Encore 4 itérations pour la matrice de dimension 3 donnent $|t_{32}^{(k)}| \leq 10^{-15}$. Il ne reste plus que 3 itérations à faire pour la matrice de dimension 2 pour avoir $|t_{21}^{(k)}| \leq 10^{-15}$. En tout, 12 itérations ont donné toutes les valeurs propres avec une précision de 15 chiffres.

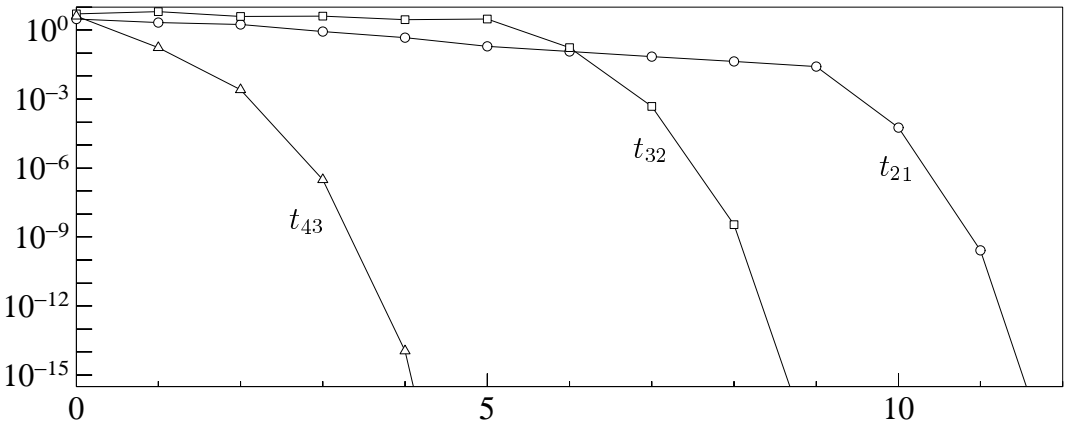


FIG. V.5: Convergence de la méthode QR (avec shift)

Le "double shift" de Francis

Dans la situation où A est une matrice réelle ayant des valeurs propres complexes, il est recommandé de choisir un shift-paramètre p_k qui soit complexe. Une application directe de l'algorithme précédent nécessite un calcul avec des matrices complexes. L'observation suivante permet d'éviter ceci.

Lemme 6.1 *Soit T_k une matrice réelle, $p_k = \alpha + i\beta$ et $p_{k+1} = \alpha - i\beta$. Alors, on peut choisir les décompositions dans l'algorithme QR de manière à ce que T_{k+2} soit réelle.*

Remarque. La décomposition QR d'une matrice est unique sauf qu'on peut remplacer QR par $(QD)(D^{-1}R)$ où $D = \mathrm{diag}(d_1, \ldots, d_n)$ avec $|d_i| = 1$.

Démonstration. La formule (6.4) montre que

$$T_{k+2} = (Q_{k+1}Q_{k+2})^* T_k (Q_{k+1}Q_{k+2}). \tag{6.7}$$

Il suffit alors de démontrer que le produit $Q_{k+1}Q_{k+2}$ est réel. Une manipulation à l'aide de formules pour T_k donne

$$\begin{aligned} Q_{k+1}Q_{k+2}R_{k+2}R_{k+1} &= Q_{k+1}(T_{k+1} - p_{k+1}I)R_{k+1} = Q_{k+1}(R_{k+1}Q_{k+1} + p_k I - p_{k+1}I)R_{k+1} \\ &= (Q_{k+1}R_{k+1})^2 + (p_k - p_{k+1})Q_{k+1}R_{k+1} = (T_k - p_k I)^2 + (p_k - p_{k+1})(T_k - p_k I) \\ &= T_k^2 - (p_k + p_{k+1})T_k + p_k p_{k+1} I =: M. \end{aligned} \tag{6.8}$$

On a donc trouvé une décomposition QR de la matrice M qui, en conséquence des hypothèses du lemme, est une matrice réelle. Si, dans l'algorithme QR, la décomposition est choisie de manière à ce que les éléments diagonaux de R_{k+1} et R_{k+2} soient réels, alors, à cause de l'unicité de la décomposition QR, les matrices $Q_{k+1}Q_{k+2}$ et $R_{k+2}R_{k+1}$ sont réelles. □

Une possibilité de calculer T_{k+2} à partir de T_k est de calculer M de (6.8), de faire une décomposition QR (réelle) de M et de calculer T_{k+2} à l'aide de (6.7). Cet algorithme n'est pas pratique car le calcul de T_k^2 nécessite $\mathcal{O}(n^3)$ opérations, même si T_k est sous forme de Hessenberg.

Il y a une astuce intéressante pour obtenir T_{k+2} à partir de T_k en $\mathcal{O}(n^2)$ opérations. Elle est basée sur la propriété suivante.

Théorème 6.2 *Soit T une matrice donnée et supposons que*

$$Q^* T Q = S \tag{6.9}$$

où Q est orthogonale et S est sous forme de Hessenberg satisfaisant $s_{i,i-1} \neq 0$ pour $i = 2, \ldots, n$. Alors, Q et S sont déterminées de manière "unique" par la première colonne de Q.

Remarque. On a "unicité" dans le sens suivant: si $\widehat{Q}^* T \widehat{Q}$ est de type Hessenberg avec une matrice orthogonale $\widehat{Q}$ satisfaisant $\widehat{Q}e_1 = Qe_1$, alors $\widehat{Q} = QD$ où $D = \mathrm{diag}(d_1, \ldots, d_n)$ avec $|d_i| = 1$.

Démonstration. Notons les colonnes de Q par q_i. Alors, la relation (6.9) implique

$$T q_i = \sum_{j=1}^{i+1} s_{ji} q_j, \qquad\qquad q_j^* T q_i = s_{ji}. \tag{6.10}$$

Si q_1 est fixé, la valeur s_{11} est donnée par la deuxième formule de (6.10). Avec cette valeur, on obtient de la première formule de (6.10) que q_2 est un multiple de $Tq_1 - s_{11}q_1$. Ceci détermine q_2 à une unité d_2 près. Maintenant, les valeurs s_{21}, s_{12}, s_{22} sont déterminées et q_3 est un multiple de $Tq_2 - s_{21}q_1 - s_{22}q_2$, etc. □

Si les hypothèses du lemme précédent sont vérifiées, on peut calculer la matrice réelle T_{k+2} en $\mathcal{O}(n^2)$ opérations de la manière suivante:

- calculer Me_1, la première colonne de M (formule (6.8));
- déterminer une matrice de Householder H_1 telle que $H_1(Me_1) = \alpha e_1$;
- transformer $H_1^T T_k H_1$ sous forme de Hessenberg à l'aide de matrices de Householder $H_2, \ldots$, H_{n-1} (voir le paragraphe V.3); c.-à-d., calculer $H^T T_k H$ où $H = H_1 H_2 \cdot \ldots \cdot H_{n-1}$.

Comme $H_i e_1 = e_1$ pour $i = 2, \ldots, n-1$, la première colonne de H est un multiple de celle de M (observer que $H_1^T = H_1$). Par la formule (6.8), la première colonne de $Q_{k+1}Q_{k+2}$ est aussi un multiple de Me_1. Par conséquent, pour un bon choix des décompositions $Q_{k+1}R_{k+1}$ et $Q_{k+2}R_{k+2}$, on a $H = Q_{k+1}Q_{k+2}$ et la matrice obtenue par cet algorithme est égale à T_{k+2} (voir (6.7)).

Etude de la convergence

Supposons d'être déjà proche de la limite et considérons, par exemple, la matrice

$$T_0 = A = \begin{pmatrix} 2 & a \\ \epsilon & 1 \end{pmatrix}$$

où ϵ est un nombre petit. Avec le choix $p_0 = 1$ pour le shift-paramètre, on obtient

$$T_0 - p_0 I = \begin{pmatrix} 1 & a \\ \epsilon & 0 \end{pmatrix} = \begin{pmatrix} \frac{1}{\sqrt{1+\epsilon^2}} & -\frac{\epsilon}{\sqrt{1+\epsilon^2}} \\ \frac{\epsilon}{\sqrt{1+\epsilon^2}} & \frac{1}{\sqrt{1+\epsilon^2}} \end{pmatrix} \begin{pmatrix} \sqrt{1+\epsilon^2} & \frac{a}{\sqrt{1+\epsilon^2}} \\ 0 & -\frac{a\epsilon}{\sqrt{1+\epsilon^2}} \end{pmatrix} = Q_1 R_1$$

et

$$T_1 - p_0 I = R_1 Q_1 = \begin{pmatrix} * & * \\ -\frac{a\epsilon^2}{1+\epsilon^2} & * \end{pmatrix}.$$

- si A est symétrique (c.-à-d., $a = \epsilon$) on a $t^{(1)}_{n,n-1} = \mathcal{O}(\epsilon^3)$, donc convergence *cubique*.
- si A n'est pas symétrique (p.ex. $a = 1$) on a $t^{(1)}_{n,n-1} = \mathcal{O}(\epsilon^2)$, donc convergence *quadratique*.

Ces propriétés restent vraies pour des matrices générales (sans démonstration).

V.7 Exercices

1. Calculer les valeurs propres de la matrice tridiagonale (dimension n, $b \cdot c > 0$)

$$A = \begin{pmatrix} a & c & & & \\ b & a & c & & \\ & b & a & c & \\ & & b & \ddots & \ddots \\ & & & \ddots & \end{pmatrix}.$$

Indication. Les composants du vecteur propre $(v_1, v_2, \ldots, v_n)^T$ satisfont une équation aux différences finies avec $v_0 = v_{n+1} = 0$. Vérifier que $v_j = Const \cdot (\alpha_1^j - \alpha_2^j)$ où

$$\alpha_1 + \alpha_2 = \frac{\lambda - a}{c}, \qquad \alpha_1 \cdot \alpha_2 = \frac{b}{c}, \qquad \Big(\frac{\alpha_1}{\alpha_2}\Big)^{n+1} = 1.$$

Résultat. $\lambda_j = a - 2\sqrt{bc} \cdot \cos\Big(\frac{j\pi}{n+1}\Big), \qquad j = 1, 2, \ldots, n.$

2. Considérer la matrice

$$A(\varepsilon) = \begin{pmatrix} 1 & \varepsilon & 0 \\ -1 & 0 & 1 \\ 1 & -1+\varepsilon & -\varepsilon \end{pmatrix}$$

D'après le théorème 1.3 cette matrice possède une valeur propre de la forme

$$\lambda(\varepsilon) = i + \varepsilon \cdot d + \mathcal{O}(\varepsilon^2).$$

Calculer d et dessiner la tangente à la courbe $\lambda(\varepsilon)$ au point $\lambda(0)$.

3. (a) Calculer par la méthode de la puissance, la plus grande valeur propre de la matrice

$$A = \begin{pmatrix} 99 & 1 & 0 \\ 1 & 100 & 1 \\ 0 & 1 & 98 \end{pmatrix}.$$

   (b) Pour accélerer considérablement la vitesse de convergence, appliquer la méthode de la puissance à la matrice $A - pI$ avec un choix intelligent de p.

   (c) Avec quel choix de p obtient-on la valeur propre la plus petite?

4. Considérons la matrice tridiagonale

$$A = \begin{pmatrix} b_1 & c_1 & & \\ a_1 & b_2 & c_2 & \\ & a_2 & & \\ & & \ddots & \end{pmatrix}.$$

   Montrer que, si $a_i \cdot c_i > 0$ pour $i = 1, \ldots, n-1$, toutes les valeurs propres de A sont réelles.

   Indication. Trouver $D = \text{diag}(d_1, \ldots, d_n)$ telle que DAD^{-1} soit symétrique.

5. Soit A une matrice symétrique et B quelconque. Montrer que pour chaque valeur propre λ_B de B il existe une valeur propre λ_A de A telle que

$$|\lambda_A - \lambda_B| \leq \|A - B\|_2.$$

   Indication. Montrer l'existence d'un vecteur v tel que $v = (A - \lambda_B)^{-1}(A - B)v$. En déduire que $1 \leq \|(A - \lambda_B)^{-1}(A - B)\| \leq \|(A - \lambda_B)^{-1}\|\|(A - B)\|$.

6. (Schur, 1909). Soit A une matrice symétrique. Montrer que pour chaque indice i il existe une valeur propre λ de A telle que

$$|\lambda - a_{ii}| \leq \sqrt{\textstyle\sum_{j \neq i} |a_{ij}|^2}.$$

   Indication. Appliquer l'exercice 5 avec une B convenable.

7. Soit A une matrice réelle avec pour valeur propre $\alpha + i\beta$. Montrer que l'itération

$$\begin{pmatrix} \bar{\alpha}I - A & -\bar{\beta}I \\ \bar{\beta}I & \bar{\alpha}I - A \end{pmatrix} \begin{pmatrix} u_{k+1} \\ v_{k+1} \end{pmatrix} = \begin{pmatrix} u_k \\ v_k \end{pmatrix}$$

   (où $\bar{\alpha} \approx \alpha$ et $\bar{\beta} \approx \beta$) permet de calculer la valeur propre $\alpha + i\beta$ et le vecteur propre correspondant.

   Indication. Considérer les parties réelles et complexes de l'itération de Wielandt. On obtient alors

$$\frac{u_k^T A u_k + v_k^T A v_k}{u_k^T u_k + v_k^T v_k} \to \alpha, \qquad \frac{u_k^T A v_k - v_k^T A u_k}{u_k^T u_k + v_k^T v_k} \to \beta\,.$$

8. Considérons la matrice de Hilbert,

$$A = \begin{pmatrix} 1/2 & 1/3 & 1/4 \\ 1/3 & 1/4 & 1/5 \\ 1/4 & 1/5 & 1/6 \end{pmatrix}$$

   (a) Transformer A en une matrice tridiagonale ayant les mêmes valeurs propres.

   (b) En utilisant une suite de Sturm, montrer que toutes les valeurs propres sont positives et qu'une valeur propre est plus petite que 0.001.

   (c) Calculer approximativement la condition de A pour la norme Euclidienne.

9. La formule de récurrence

$$(k+1)P_{k+1}(x) = (2k+1)xP_k(x) - kP_{k-1}(x)$$

pour les *polynômes de Legendre* (voir aussi (I.4.5)) ressemble à

$$p_i(\lambda) = (d_i - \lambda)p_{i-1}(\lambda) - e_i^2 p_{i-2}(\lambda), \qquad i = 2, \ldots, n,$$

pour les polynômes $\det(A_i - \lambda I)$. Trouver une matrice tridiagonale A de dimension n telle que les valeurs propres de A sont les racines de $P_n(x)$.

10. Soit $p(x)$ un polynôme de degré n et supposons que toutes les racines soient simples. Démontrer que la suite définie par l'algorithme d'Euclid,

$$\begin{aligned} &p_n(x) = p(x), \quad p_{n-1}(x) = -p'(x) \\ &p_i(x) = q_i(x)p_{i-1}(x) - \gamma_i^2 p_{i-2}(x), \quad i = n, \ldots, 2, \end{aligned}$$

est une suite de Sturm.

Pour le polynôme $p(x) = x^5 - 5x^4 + 3x^3 + 3x^2 + 2x + 8$.

(a) déterminer le nombre de racines réelles.

(b) Combien de racines sont complexes?

(c) Combien de racines sont réelles et positives?

11. Pour un φ donné notons $c = \cos\varphi$ et $s = \sin\varphi$. La matrice $\Omega_{k\ell}$, définie par

$$(\Omega_{k\ell})_{ij} = \begin{cases} 1 & \text{si } i = j,\ j \neq k,\ j \neq \ell \\ c & \text{si } i = j = k \text{ où } i = j = \ell \\ s & \text{si } i = k \text{ et } j = \ell \\ -s & \text{si } i = \ell \text{ et } j = k \\ 0 & \text{sinon,} \end{cases}$$

s'appelle *rotation de Givens.*

a) Montrer qu'elle est orthogonale.

b) Soit A une matrice symétrique. Déterminer φ tel que le (k,ℓ)-ième élément de $A' = \Omega_{k\ell} A \Omega_{k\ell}^T$ s'annule.

Resultat. ctg $2\varphi = (a_{kk} - a_{\ell\ell})/(2a_{k\ell})$.

12. La *méthode de Jacobi* (1846) pour le calcul des valeurs propres d'une matrice symétrique:

i) on choisit $a_{k\ell}$ $(k > \ell)$ tel que $|a_{k\ell}| = \max_{i>j} |a_{ij}|$;

ii) on détermine A' comme dans l'exercice 11.

Montrer que, si on répète cette procédure, on a convergence vers une matrice diagonale, dont les éléments sont les valeurs propres de A.

Indication. Montrer que $\sum_{i>j} |a'_{ij}|^2 = \sum_{i>j} |a_{ij}|^2 - |a_{k\ell}|^2$.

13. On considère la matrice

$$A = \begin{pmatrix} 7 & 0.5 \\ 0.0001 & 8 \end{pmatrix}$$

dont on cherche à calculer les valeurs propres.

(a) Faire une itération de l'algorithme QR sans shift.

(b) Faire une itération de l'algorithme QR avec shift.

(c) Estimer la position des valeurs propres de A à l'aide du Théorème de Gershgorine.

(d) Calculer les valeurs propres de A à l'aide du polynôme caractéristique.

14. Montrer que si la matrice $T_0 = A$ est une matrice de Hessenberg (ou tridiagonale), alors les matrices T_k, $k \geq 1$, construites par l'algorithme QR sont également des matrices de Hessenberg (tridiagonales).

15. Donner une estimation grossière du nombre d'opérations qui sont nécessaires pour effectuer la décomposition QR d'une matrice de Hessenberg et pour calculer ensuite le produit RQ.

16. Soit T_0 une matrice de Hessenberg dont tous les éléments de la sous-diagonale sont non-nuls. Montrer que, si p_0 est une valeur propre de T_0, une itération de l'algorithme QR avec shift p_0 donne $t^{(1)}_{n,n-1} = 0$.

17. Expliquer, comment le calcul de T_k à partir de T_{k-1}

$$Q_k R_k = T_{k-1} - p_{k-1} I, \qquad T_k = R_k Q_k + p_{k-1} I$$

peut être effectué sans soustraire (et additionner) explicitement la matrice $p_{k-1} I$.

Indication. Laissez-vous inspirer par le "double shift" algorithme de Francis.