TECHNIQUES
DE L'INGÉNIEUR

Réf. : **AF1221 V1**

Date de publication :
10 avril 2006

Méthodes numériques de base - Algèbre numérique

Cet article est issu de : **Sciences fondamentales | Mathématiques**

par **Claude BREZINSKI**

Résumé Cet article est consacré à l'algèbre numérique linéaire et non linéaire. Sont exposées dans un premier temps les méthodes de calcul des racines d'une équation non linéaire à une inconnue, puis celles d'un polynôme, pour conduire à la résolution d'équations non linéaires. Sont abordées ensuite les méthodes numériques pour résoudre les équations linéaires, les directes comme les itératives. Pour terminer, est traité le calcul des valeurs et des vecteurs propres d'une matrice par des méthodes itératives.

Abstract

Méthodes numériques de base

Algèbre numérique

par **Claude BREZINSKI**
Docteur ès sciences mathématiques
Professeur à l'université des Sciences et Technologies de Lille

Ce second dossier sur les méthodes numériques de base concerne l'algèbre numérique linéaire et non linéaire.

Le premier paragraphe est consacré aux méthodes itératives pour calculer les racines d'une équation non linéaire à une inconnue (ou, ce qui revient au même, les points fixes d'une fonction). On traite ensuite le cas particulier de la recherche des racines d'un polynôme. Le paragraphe se termine par les méthodes de résolution des systèmes d'équations non linéaires.

On étudie ensuite les méthodes numériques pour résoudre les systèmes d'équations linéaires. Ces méthodes se divisent en deux classes : les méthodes directes qui fournissent la solution exacte en un nombre fini d'opérations arithmétiques (en supposant nulles les erreurs dues à l'arithmétique de l'ordinateur) et les méthodes itératives qui génèrent une suite de vecteurs convergeant (sous certaines conditions) vers la solution exacte. Pour les systèmes de très grandes dimensions, il est impératif d'utiliser une méthode itérative.

On passe enfin, dans le dernier paragraphe, aux méthodes numériques pour calculer les valeurs propres et les vecteurs propres d'une matrice. Ces méthodes sont toutes des méthodes itératives.

Pour tout renseignement complémentaire, le lecteur se reportera au dossier précédent [AF 1 220].

1. Résolution des équations et des systèmes non linéaires

Soit f une application continue de $\mathbb{R}$ dans lui-même. Le problème auquel nous allons nous intéresser dans ce paragraphe est celui de la recherche de x tel que $f(x) = 0$. On dit alors que x est **racine** de f. Une autre façon, complètement équivalente, de poser le même problème est de rechercher x tel que $x = F(x)$. On dit alors que x est **point fixe** de F. Dans la suite, quand nous utiliserons la lettre f (dans un théorème ou un algorithme), cela signifiera implicitement que le problème à résoudre est mis sous la forme $f(x) = 0$. Quand nous utiliserons la lettre F, cela signifiera que notre problème est écrit sous la forme $x = F(x)$. Ces deux formulations sont équivalentes car, s'il est sous la forme $f(x) = 0$, on a également $x = x + af(x) = F(x)$ avec $a \neq 0$ quelconque. Inversement, si l'on a $x = F(x)$, alors on pourra écrire $f(x) = x - F(x) = 0$.

1.1 Méthode des approximations successives

Pour résoudre numériquement ce type de problème, on utilise une méthode itérative dans laquelle on fabrique une suite (x_n) qui doit converger vers x. On se donne une valeur initiale x_0 puis on fabrique (x_n) par la **méthode des approximations successives** :

$$x_{n+1} = F(x_n) \text{ pour } n = 0, 1, \ldots$$

Étudions d'abord des conditions pour que (x_n) converge vers x point fixe de F et commençons par la définition **1**.

Définition 1 – Soit D une partie de $\mathbb{R}$ et F une application de D dans lui-même. S'il existe une constante positive K, strictement inférieure à 1, telle que pour tout u et tout v appartenant à D on ait :

$$|F(u) - F(v)| \leqslant K|u - v|$$

on dit que F est une **contraction** sur D.

K est appelé **coefficient de contraction** de F.

Le premier résultat est donné par le théorème **1**.

Théorème 1 – Soit $I = [x_0 - a, x_0 + a]$ où $a > 0$.

Supposons que F soit une contraction sur I (de coefficient de contraction K) et que $|F(x_0) - x_0| \leqslant (1 - K)\, a$. Alors la suite (xn), fabriquée par la méthode des approximations successives avec x_0 comme valeur initiale, converge. Soit x la limite de (xn).

x est l'unique point fixe de F dans I. Pour tout n, $x_n \in I$ et l'on a :

$$|x_n - x| \leqslant \frac{K^n}{1 - K} |x_1 - x_0|$$

On voit que, si les hypothèses de ce théorème sont très fortes et difficiles à vérifier en pratique, les conclusions sont également très importantes car on démontre, grâce à la méthode des approximations successives, l'existence et l'unicité d'un point fixe dans I pour l'application F. On démontre également la convergence de la méthode des approximations successives et l'on donne une majoration de l'erreur qui montre que la vitesse de convergence dépend de la proximité de K par rapport à 1.

Donnons maintenant le théorème **2** dont les hypothèses sont plus faibles mais dont les conclusions sont également moins fortes.

Théorème 2 – Soit x un point fixe de F. Si F est dérivable au voisinage de x et si $|F'(x)| < 1$, alors il existe, $V \subset \mathbb{R}$ tel que, pour tout $x_0 \in V$, les itérations $x_{n+1} = F(x_n)$, $n = 0, 1, \ldots$ convergent vers x.

1.2 Ordre d'une suite

Il nous faut maintenant disposer d'un outil mathématique pour mesurer la vitesse de convergence d'une suite. C'est la notion d'ordre d'une suite donnée par la définition **2**.

Définition 2 – Soit (x_n) une suite qui converge vers x. On dit que (x_n) est d'**ordre** r, où r est un nombre réel supérieur ou égal à 1, s'il existe une constante C finie et différente de zéro telle que :

$$C = \lim_{n \to \infty} |x_{n+1} - x| \,/\, |x_n - x|^r$$

C s'appelle **constante asymptotique d'erreur**.

Ces deux notions sont d'une grande importance pratique car elles nous renseignent sur l'évolution du nombre de chiffres décimaux exacts obtenus au fur et à mesure des itérations. En effet, posons :

$$d_n = -\lg |x_n - x|$$

À une constante additive près, indépendante de n, d_n est égal au nombre de chiffres décimaux exacts de x_n. Si nous posons :

$$R = -\lg C$$

alors on voit, d'après la définition **2**, que, lorsque n est suffisamment grand (c'est-à-dire lorsque x_n est suffisamment voisin de x), on a :

$$d_{n+1} \approx r d_n + R$$

Ainsi, en passant de l'itération n à l'itération $n + 1$, on multiplie environ par r le nombre de chiffres décimaux exacts et l'on en ajoute environ R. Cela montre l'avantage des méthodes d'ordre supérieur à 1.

Quand la suite (x_n) est obtenue par la méthode des approximations successives, l'ordre est un nombre entier lorsque F est plusieurs fois dérivable en x. On montre que c'est l'entier r tel que :

$$F'(x) = \ldots = F^{(r-1)}(x) = 0 \qquad \text{et} \qquad F^{(r)}(x) \neq 0$$

On a alors :

$$C = |F^{(r)}(x)|/r!$$

Si l'on sait que $F'(x) = \ldots = F^{(r-1)}(x) = 0$, alors l'ordre est au moins égal à r.

Remarque : tout ce qui a été vu depuis le début du paragraphe se généralise au cas d'un système d'équations non linéaires (ou même au cas d'un espace de Banach général). Il suffit, dans ce qui précède, de remplacer la valeur absolue par la norme.

1.3 Accélération de la convergence

Lorsque la suite (x_n), obtenue par la méthode des approximations successives, converge lentement, on peut chercher à accélérer sa convergence à l'aide du **procédé Δ^2 d'Aitken**. Pour cela, on construit une seconde suite, (y_n), à l'aide de la formule suivante :

$$y_n = x_n - \frac{(x_{n+1} - x_n)^2}{x_{n+2} - 2x_{n+1} + x_n} \quad \text{pour} \quad n = 0, 1, \ldots$$

On voit que cette suite se construit au fur et à mesure de la construction de la suite (x_n) ; il suffit, pour obtenir y_n, de conserver les trois derniers termes de celle-ci.

Remarque : si l'on réduit au même dénominateur la formule précédente, alors on a :

$$y_n = (x_n x_{n+2} - x_{n+1}^2)/(x_{n+2} - 2x_{n+1} + x_n)$$

Cette formule est à proscrire car elle est numériquement instable alors que la première relation donnée était numériquement plus stable.

Pour le procédé Δ^2 d'Aitken, on démontre le théorème **3**.

Théorème 3 – Si l'on applique le procédé Δ^2 d'Aitken à une suite (x_n) qui converge vers x et si, pour tout n :

$$x_{n+1} - x = (a + e_n)(x_n - x)$$

avec $a \neq 1$,

$$\lim_{n \to \infty} e_n = 0,$$

alors la suite (y_n) ainsi obtenue converge vers x plus vite que (x_n), c'est-à-dire que :

$$\lim_{n \to \infty} (y_n - x)/(x_n - x) = 0$$

Lorsque la suite (x_n) est fabriquée par la méthode des approximations successives et que $|F'(x)| < 1$, elle vérifie les hypothèses du théorème **3** et, par conséquent, la convergence est accélérée.

Il existe de nombreuses autres méthodes d'accélération de la convergence. On trouvera leur description, des applications numériques et des sous-programmes FORTRAN dans la référence [7].

1.4 Méthodes particulières

Voyons maintenant un certain nombre de méthodes particulières.

■ La **méthode de Newton** pour résoudre $f(x) = 0$ consiste, partant d'un x_0 arbitraire, à effectuer les itérations :

$$x_{n+1} = x_n - f(x_n)/f'(x_n) \text{ pour } n = 0, 1, \ldots$$

■ Dans la pratique, $f'(x_n)$ pouvant être difficile à évaluer, on le remplace souvent par une valeur approchée. C'est ainsi que si l'on approxime $f'(x_n)$ par :

$$(f(x_n) - f(x_{n-1}))/(x_n - x_{n-1})$$

on obtient une méthode connue sous le nom de **méthode de la sécante** :

$$x_0 \text{ et } x_1 \text{ arbitraires}$$

$$x_{n+1} = x_n - \frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})} f(x_n) \quad \text{pour } n = 1, 2, \ldots$$

■ Si l'on approxime $f'(x_n)$ par $[f(x_n) - f(x_n - f(x_n))]/f(x_n)$, on obtient, en posant $F(x) = x - f(x)$, la **méthode de Steffensen** :

$$x_0 \text{ arbitraire}$$

$$x_{n+1} = x_n - \frac{(F(x_n) - x_n)^2}{F(F(x_n)) - 2F(x_n) + x_n} \quad \text{pour } n = 0, 1, \ldots$$

Pour ces trois méthodes, on a le théorème **4**.

Théorème 4 – Si $f'(x) \neq 0$ et si f'' est continue en x, alors les méthodes de Newton et de Steffensen sont d'ordre deux au moins et la méthode de la sécante est d'ordre $(1 + \sqrt{5})/2$ au moins.

Bien que son ordre soit plus faible ($\approx 1{,}618$), la méthode de la sécante doit être préférée aux deux autres car elle ne nécessite qu'une seule évaluation de fonction par itération au lieu de deux. Une itération de cette méthode dure donc deux fois moins longtemps qu'une itération avec l'une des deux autres.

1.5 Tests d'arrêt

Un problème important posé par les méthodes itératives est celui des tests d'arrêt. Dans la pratique, on ne fait bien évidemment pas une infinité d'itérations. Si l'on désire s'arrêter à une certaine itération, il faut pouvoir contrôler la précision atteinte. On se base pour cela sur l'inégalité du théorème **1**. Comme on ne connaît pas la valeur exacte de K, on la remplace par une valeur approchée :

$$K_n = (x_{n+1} - x_n)/(x_n - x_{n-1})$$

et on arrête les itérations lorsque $-1 < K_n < 1$ et que $|x_{n+1} - x_n|/(1 - |K_n|)$ est inférieur à la précision absolue que l'on désire atteindre.

1.6 Méthode de Bairstow

Un cas particulièrement important de résolution d'équations est celui du calcul des racines d'un polynôme. Il existe de nombreuses méthodes, dont aucune n'est fiable dans toutes les situations (comme c'est d'ailleurs le cas avec toutes les méthodes d'analyse numérique), pour résoudre ce problème. Nous allons en décrire une et renvoyer le lecteur intéressé à la référence [21] qui, bien qu'ancienne, est une source précieuse de renseignements.

Soit P_n le polynôme, de degré n, dont on veut calculer les racines. Soit :

$$Q(x) = x^2 - sx + p$$

avec s et p nombres réels arbitraires.

Effectuons la division euclidienne de P_n par Q. On obtient un quotient P_{n-2} de degré $n-2$ et un reste R du premier degré. Il est évident que les coefficients de P_{n-2} et ceux de R dépendent des valeurs choisies pour s et p. Nous allons donc rechercher s et p tels

que les deux coefficients de R soient nuls (c'est-à-dire que nous avons un système de deux équations non linéaires à deux inconnues à résoudre). S'il en est ainsi, cela signifie que s et p sont respectivement la somme et le produit de deux racines de P_n puisqu'alors Q divise P_n. Nous obtenons donc immédiatement ces deux racines même dans le cas où elles sont complexes.

Pour obtenir deux autres racines, on recommence la même procédure à partir du polynôme P_{n-2} et ainsi de suite jusqu'au moment où l'on obtient un polynôme du second ou du premier degré dont les racines sont calculées directement. Pour résoudre le système de deux équations à deux inconnues constitué par les coefficients de R, on utilise la méthode de Newton qui se généralise facilement au cas d'un système (§ 1.7). L'ensemble de cette procédure s'appelle **méthode de Bairstow**. D'après ce que nous savons de la méthode de Newton, elle sera d'ordre deux au moins si toutes les racines de P_n sont simples. Décrivons maintenant cette méthode. Nous posons :

$$P_n(x) = a_0 x^n + a_1 x^{n-1} + \ldots + a_n$$

$$P_{n-2}(x) = b_0 x^{n-2} + b_1 x^{n-3} + \ldots + b_{n-2}$$

$$R(x) = b_{n-1}(x - s) + b_n$$

Connaissant s et p, arbitraires, les b_i s'obtiennent par :

$$b_0 = a_0$$

$$b_1 = a_1 + s b_0$$

$$b_i = a_i + s b_{i-1} - p b_{i-2} \quad \text{pour} \quad i = 2, \ldots, n$$

Avant la première itération de la méthode de Bairstow, on choisit des valeurs arbitraires s_0 et p_0. Une méthode itérative est complètement définie par le passage de l'itéré k à l'itéré $k+1$. Au début de l'itération $k+1$, on connaît s_k et p_k. Voyons comment obtenir s_{k+1} et p_{k+1} :

— dans les relations précédentes, on prend $s = s_k$ et $p = p_k$ et l'on calcule $b_0, b_1, \ldots, b_n$;

— pour $s = s_k$ et $p = p_k$ on calcule $r_0, r_1, \ldots, r_n$ par :

$$r_0 = 0$$

$$r_1 = b_0$$

$$r_i = b_{i-1} + s r_{i-1} - p r_{i-2} \quad \text{pour} \quad i = 2, \ldots, n$$

— puis l'on pose :

$$s_{k+1} = s_k - (b_n r_{n-2} - b_{n-1} r_{n-1})/d$$

$$p_{k+1} = p_k - (b_n r_{n-1} - b_{n-1} r_n)/d$$

avec $$d = r_n r_{n-2} - r_{n-1}^2$$

On arrête les itérations lorsque $|s_{k+1} - s_k| + |p_{k+1} - p_k|$ est inférieur à la précision absolue désirée. On calcule les deux racines correspondantes et l'on recommence la procédure sur le polynôme P_{n-2} dont les coefficients sont les derniers $b_0, b_1, \ldots, b_{n-2}$ obtenus.

1.7 Systèmes d'équations non linéaires

Dans ce qui précède (sauf dans le cas de la méthode de Bairstow), nous n'avons considéré que le cas d'une seule équation non linéaire à une seule inconnue. Dans le cas d'un système de p équations à p inconnues, il est possible de généraliser les méthodes de Newton, de Steffensen et de la sécante. Les x_n sont maintenant des vecteurs à p composantes ainsi que les $f(x_n)$ [ou, ce qui revient au même, les $F(x_n)$].

Pour ce qui est de la méthode de Newton, les itérations deviennent :

$$x_{n+1} = x_n - [f'(x_n)]^{-1} f(x_n) \quad \text{pour} \quad n = 0, 1, \ldots$$

avec $f'(x_n)$ matrice jacobienne de f en x_n (c'est-à-dire la matrice dont les éléments sont les dérivées partielles des fonctions f par rapport aux différentes variables).

On démontre que (x_n) est d'ordre deux au moins si $f'(x)$ est inversible.

On obtiendra des généralisations de la méthode de la sécante et de la méthode Steffensen en remplaçant, comme dans le cas d'une seule équation, la matrice $f'(x_n)$ par une approximation. Cependant, la diversité des algorithmes obtenus est beaucoup plus grande parce que les possibilités d'approximation sont beaucoup plus nombreuses.

On parle alors de **méthodes quasi-Newton**. Elles se présentent sous l'une des deux formes :

$$x_{n+1} = x_n - H_n^{-1} f(x_n)$$

$$x_{n+1} = x_n - C_n f(x_n)$$

c'est-à-dire :

$$x_{n+1} = x_n + s_n$$

avec s_n défini par $H_n s_n = -f_n$ ou $s_n = -C_n f_n$, $f_n = f(x_n)$.

Pour avoir la convergence la plus rapide possible, les matrices H_n doivent être de bonnes approximations des matrices $f'(x_n)$ ou les matrices C_n de leurs inverses. De telles approximations sont soit difficiles à obtenir soit coûteuses (en termes de nombre d'opérations arithmétiques et d'encombrement mémoire).

L'idée des méthodes quasi-Newton consiste à construire les suites (H_n) ou (C_n) par :

$$H_{n+1} = H_n + D_n \quad \text{ou} \quad C_{n+1} = C_n + E_n$$

avec D_n et E_n matrices de rang 1 ou 2 choisies de sorte que :

$$H_{n+1} s_n = \Delta f_n \quad \text{ou} \quad C_{n+1} y_n = s_n$$

et $y_n = \Delta f_n$, c'est-à-dire $E_n s_n = f_{n+1}$ et $E_n y_n = C_n f_{n+1}$ respectivement.

Les modifications de rang 1 suivantes conduisent à une convergence superlinéaire de la méthode :

$$H_{n+1} = H_n + \frac{(y_n - H_n s_n) v_n^T}{(s_n, v_n)} = H_n + \frac{f_{n+1} v_n^T}{(s_n, v_n)}$$

$$C_{n+1} = C_n + \frac{(s_n - C_n y_n) u_n^T}{(y_n, u_n)} = C_n - \frac{C_n f_{n+1} u_n^T}{(y_n, u_n)}$$

avec u_n et v_n vecteurs devant vérifier certaines conditions.

Le choix $v_n = s_n$ correspond à la *bonne* **méthode de Broyden** tandis que le choix $u_n = y_n$ (ou le choix $v_n = H_n^T y_n$) correspond à sa *mauvaise* méthode. Si $C_n = H_n^{-1}$ et si $v_n = H_n^T u_n$, alors $C_{n+1} = H_{n+1}^{-1}$. Il existe beaucoup d'autres procédures de mise à jour des matrices H_n ou C_n.

Les **méthodes de Barzilai-Borwein** sont plus simples à mettre en œuvre. Elles sont des versions non linéaires de la méthode de la plus profonde descente (cf. § 2.2.2) et consistent en des itérations de la forme $x_{n+1} = x_n - \lambda_n f_n$:

avec $$\lambda_n = \frac{(u_n, \Delta x_{n-1})}{(u_n, \Delta f_{n-1})}$$

$$u_n = \Delta f_{n-1} \quad \text{ou} \quad u_n = \Delta x_{n-1}$$

D'autres choix de u_n peuvent être envisagés.

Sur les méthodes de résolution des systèmes non linéaires, on pourra consulter [5] [20] [35].

2. Résolution des systèmes d'équations linéaires

Soit A une matrice carrée à n lignes et n colonnes dont les éléments a_{ij} sont connus. Soit b un vecteur dont les n composantes b_i sont connues. On recherche le vecteur x, de composantes x_1, x_2, ..., x_n, qui vérifie le système d'équations linéaires :

$$Ax = b$$

La solution de ce problème est connue, on la trouve dans tous les cours d'algèbre linéaire : x_i est égal à un rapport de déterminants, au dénominateur le déterminant de la matrice A et au numérateur le même déterminant dans lequel on a remplacé la $i^{\text{ième}}$ colonne par le vecteur second membre b. Les règles pour calculer un déterminant sont également classiques. Cependant, on oublie souvent de dire qu'un tel calcul demande $n \cdot n!$ multiplications, c'est-à-dire de l'ordre de $n^2 \cdot n!$ multiplications pour résoudre le système. Pour $n = 10$, cela fait $3{,}6 \times 10^8$ multiplications et pour $n = 30$ cela en fait $2{,}3 \times 10^{35}$ (c'est-à-dire 8×10^{12} milliards d'années de travail pour un ordinateur effectuant un million de multiplications par seconde, ce qui représente une durée plus grande que l'âge de l'univers). C'est-à-dire que cette méthode est totalement inutilisable dans la pratique, surtout en sachant qu'il est courant de résoudre actuellement des systèmes d'équations avec plusieurs millions d'inconnues. C'est pour cette raison que l'on fait appel à des méthodes d'analyse numérique.

Celles-ci se divisent en deux classes :

— les **méthodes directes** qui fournissent la solution exacte après un nombre fini (et beaucoup plus petit que $n^2 \cdot n!$) d'opérations arithmétiques ;

— les **méthodes itératives** qui s'apparentent à la méthode des approximations successives et qui donnent la solution comme limite d'une suite de vecteurs.

On peut donc se demander pourquoi utiliser des méthodes itératives alors que l'on dispose de méthodes directes ; la raison principale est que les méthodes itératives ne posent pas de problèmes de stockage dans la mémoire de l'ordinateur. Par ailleurs, chaque itération ne demande que de l'ordre de $2n^2$ opérations arithmétiques.

Sur l'ensemble des questions traitées dans ce paragraphe (et dans le suivant), il faut recommander la lecture des ouvrages [3] [5] [10] [14] [30] [33] [39] [41] [42].

2.1 Méthodes directes

L'idée de ces méthodes est de transformer le système en un système ayant la même solution mais plus facile à résoudre. Le plus simple est donc évidemment d'avoir une matrice diagonale, mais cette stratégie (appelée méthode de Gauss-Jordan) est beaucoup plus onéreuse que la méthode de Gauss que nous allons exposer.

2.1.1 Méthode de Gauss

Cette méthode consiste à transformer le système en un système équivalent ayant une matrice dont tous les éléments au-dessous de la diagonale principale sont nuls (c'est ce que l'on appelle une matrice **triangulaire supérieure**). Pour aboutir à une telle matrice, on procède en $n - 1$ étapes de la manière suivante :

— on pose $A^{(1)} = A$ et $b^{(1)} = b$;

— on construit les systèmes équivalents $A^{(k)} x = b^{(k)}$ pour $k = 2, ..., n$ où les composantes de $b^{(k)}$ sont notées $b_i^{(k)}$, où les éléments de $A^{(k)}$ sont notés $a_{ij}^{(k)}$ et où $A^{(k)}$ est de la forme :

$$A^{(k)} = \begin{pmatrix} a_{11}^{(k)} & \cdots & a_{1k}^{(k)} & \cdots & a_{1n}^{(k)} \\ & \ddots & & & \\ 0 & & a_{kk}^{(k)} & \cdots & a_{kn}^{(k)} \\ & & \vdots & & \vdots \\ & & a_{nk}^{(k)} & \cdots & a_{nn}^{(k)} \end{pmatrix}$$

On voit que $A^{(k)}$ a la forme voulue jusqu'à la $k^{\text{ième}}$ colonne. On passe du $k^{\text{ième}}$ système au $(k+1)^{\text{ième}}$ à l'aide des relations suivantes :

$$a_{ij}^{(k+1)} = a_{ij}^{(k)} \qquad \text{pour } i = 1, ..., k \text{ et } j = 1, ..., n$$

$$b_i^{(k+1)} = b_i^{(k)} \qquad \text{pour } i = 1, ..., k$$

$$a_{ij}^{(k+1)} = 0 \qquad \text{pour } i = k+1, ..., n \text{ et } j = 1, ..., k$$

$$a_{ij}^{(k+1)} = a_{ij}^{(k)} - \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}} a_{kj}^{(k)} \qquad \text{pour } i = k+1, ..., n \text{ et } j = k+1, ..., n$$

$$b_i^{(k+1)} = b_i^{(k)} - \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}} b_k^{(k)} \qquad \text{pour } i = k+1, ..., n$$

Ces règles ne sont bien sûr utilisables que si aucun des nombres $a_{kk}^{(k)}$ (les **pivots**) n'est nul. Nous verrons plus loin ce qu'il y a lieu de faire si cela se produit.

Le système $A^{(n)} x = b^{(n)}$ ainsi obtenu est triangulaire supérieur. Sa résolution est simple ; la dernière équation nous fournit directement x_n, connaissant x_n l'avant-dernière équation nous donne x_{n-1} et ainsi de suite jusqu'à la première équation qui permet de calculer x_1. Plus précisément, nous avons (en supprimant les indices supérieurs n) :

$$x_n = b_n / a_{nn}$$

$$x_i = \left(b_i - \sum_{j=i+1}^{n} a_{ij} x_j \right) / a_{ij} \quad \text{pour} \quad i = n-1, \; n-2, ..., \; 1$$

La résolution d'un système linéaire par la méthode de Gauss demande de l'ordre de $2n^3/3$ opérations arithmétiques.

Si la matrice A est singulière, on rencontrera nécessairement un pivot nul dans la phase de triangularisation (qui peut d'ailleurs être le dernier $a_{nn}^{(n)}$, ce qui n'empêche pas de triangulariser A mais ce qui, naturellement, empêche de résoudre le système triangulaire). Mais l'inverse n'est pas vrai : ce n'est pas parce que l'on rencontre un pivot nul que, pour autant, la matrice A est singulière. Il faut alors pouvoir poursuivre le calcul. Il suffit pour cela d'intervertir deux équations du système, ce qui ne change évidemment pas sa solution. Pour ne pas détruire le travail de triangularisation déjà commencé, on intervertira l'équation concernée avec l'une des suivantes.

Supposons maintenant que le pivot ne soit pas rigoureusement nul, mais seulement très petit. Nous avons appris, dans le

dossier [AF 1 220] § 1.2, à nous méfier des nombres petits ; ils peuvent, en effet, provenir de la différence de deux nombres voisins et être entachés d'une erreur de cancellation importante. C'est d'autant plus dangereux ici que le pivot est au dénominateur. Pour éviter une propagation importante des erreurs numériques, il y a donc lieu d'éviter les pivots petits ; il vaut mieux que le pivot soit le plus grand possible en valeur absolue. C'est pour cela que, à chaque étape k de la phase de triangularisation, nous allons rechercher, dans les $n-k+1$ dernières équations, le pivot le plus grand en valeur absolue et intervertir l'équation correspondante avec la $k^{\text{ième}}$. Plus précisément, à l'étape k on recherche l'indice p tel que :

$$\left| a_{pk}^{(k)} \right| = \max_{k \leqslant i \leqslant n} \left| a_{ik}^{(k)} \right|$$

et l'on permute l'équation p et l'équation k. C'est la méthode de Gauss avec **pivotage partiel**. Si $a_{pk}^{(k)} = 0$, cela signifie que $a_{ik}^{(k)} = 0$ pour $i = k, \ldots, n$ et donc que la matrice A est singulière.

Une autre stratégie, appelée méthode de Gauss avec **pivotage total**, consiste à rechercher les indices p et q tels que :

$$\left| a_{pq}^{(k)} \right| = \max_{k \leqslant i,\, j \leqslant n} \left| a_{ij}^{(k)} \right|$$

et à permuter ensuite les équations p et k et les colonnes q et k. Il faut bien faire attention qu'une permutation de colonnes permute la numérotation des inconnues correspondantes. Il y aura donc lieu, après avoir résolu le système triangulaire, de remettre les inconnues dans le bon ordre ; cela peut se faire aisément grâce à un vecteur d'indices.

Donnons maintenant l'interprétation matricielle de la méthode de Gauss car elle nous sera utile dans le paragraphe 3.4. Cette méthode consiste en fait à décomposer la matrice A en un produit $A = LU$ ou U n'est autre que la matrice $A^{(n)}$ obtenue à la fin de la phase de triangularisation et où la matrice L est la matrice suivante (que l'on appelle **triangulaire inférieure à diagonale unité**) :

$$L = \begin{pmatrix} 1 & & & \\ \ell_{21} & 1 & & 0 \\ \ell_{31} & \ell_{32} & 1 & \\ \vdots & \vdots & \ddots & \ddots \\ \ell_{n1} & \ell_{n2} & \cdots & \ell_{n,n-1} \quad 1 \end{pmatrix}$$

avec $\ell_{ij} = a_{ij}^{(j)} / a_{jj}^{(i)}$.

On a donc à résoudre $LUx = b$. Posons $Ux = y$; le système s'écrit alors $Ly = b$. On résout donc ce système triangulaire inférieur (ce qui est facile) pour obtenir y, puis le système triangulaire supérieur $Ux = y$ pour obtenir x.

Le vecteur y n'est autre que le vecteur $b^{(n)}$ obtenu à la fin de la phase de triangularisation. Celle-ci consiste donc en fait à décomposer A sous la forme d'un produit LU et à calculer le vecteur y. La seconde partie de la méthode de Gauss consiste, nous l'avons vu, à résoudre le système triangulaire supérieur.

D'après cette interprétation matricielle, on a :

$$\det A = \det L \, \det U$$

L et U étant triangulaires, leurs déterminants sont égaux au produit de leurs termes diagonaux. On a donc $\det L = 1$ et :

$$\det A = \prod_{k=1}^{n} a_{kk}^{(k)}$$

Avec la méthode de Gauss, il est possible de résoudre simultanément plusieurs systèmes avec la même matrice A mais avec différents seconds membres. Il suffit, dans la phase de triangularisation, de modifier simultanément tous ces seconds membres. On résout ensuite séparément chacun des systèmes triangulaires supérieurs ainsi obtenus. En prenant comme second membre e_i, vecteur dont toutes les composantes sont nulles sauf la $i^{\text{ième}}$ qui est égale à un, on obtient la $i^{\text{ième}}$ colonne de la matrice A^{-1}. On obtiendra toutes les colonnes de A^{-1} en résolvant simultanément le système avec les seconds membres $e_1, e_2, \ldots, e_n$. Cela nécessite de l'ordre de $8n^3/3$ opérations arithmétiques.

2.1.2 Étude des erreurs

Il nous faut maintenant parler des erreurs introduites par l'arithmétique de l'ordinateur. Normalement, si celle-ci était exacte, la méthode de Gauss devrait fournir la solution exacte du système linéaire, mais il n'en est rien. En premier lieu, avant de commencer tout calcul (que ce soit d'ailleurs par la méthode de Gauss ou par l'une des méthodes que nous verrons ultérieurement), il faut introduire les données dans l'ordinateur. Comme nous l'avons vu dans le dossier [AF 1 220], celles-ci seront donc entachées d'une ***erreur d'affectation***. Nous ne résoudrons donc pas le système que nous voulions résoudre, mais un système perturbé par l'erreur d'affectation. Il se peut très bien que la solution exacte de ce problème perturbé soit très éloignée de la solution exacte du système non perturbé : c'est la notion de conditionnement d'un problème que nous avons déjà évoquée [AF 1 220]. Il est possible de quantifier ce phénomène en introduisant la notion de **norme d'un vecteur** et de **norme d'une matrice**.

Pour étudier les erreurs, il va être nécessaire de disposer d'un outil pour mesurer l'écart entre deux vecteurs ou entre deux matrices. Pour les nombres réels, cet outil est la valeur absolue. La notion que nous allons maintenant définir en est une généralisation. Soient u, $v \in \mathbb{C}^n$ des vecteurs. Toute application qui à u fait correspondre le nombre noté $\|u\|$ s'appelle une **norme de vecteur** si et seulement si elle vérifie les trois conditions caractéristiques :

$$\|u\| \geqslant 0 \text{ et } \|u\| = 0 \text{ si et seulement si } u = 0$$
$$\|\lambda u\| = |\lambda| \cdot \|u\|, \quad \forall \lambda \in \mathbb{C}$$
$$\|u + v\| \leqslant \|u\| + \|v\|$$

Pour un vecteur, on utilise, en général, les normes suivantes appelées **normes de Hölder** ; on les différencie par un indice :

$$\|u\|_1 = \sum_{i=1}^{n} |u_i|$$

$$\|u\|_2 = \left(\sum_{i=1}^{n} |u_i|^2 \right)^{1/2}$$

$$\|u\|_\infty = \max_{1 \leqslant i \leqslant n} |u_i|$$

La norme $\|\cdot\|_2$ s'appelle **la norme euclidienne**.

Soit maintenant $G = (g_{ij})$ une matrice. Toute application qui à G fait correspondre le nombre noté $\|G\|$ s'appelle une **norme de matrice** si et seulement si elle vérifie des conditions caractéristiques identiques à celles vérifiées par les normes de vecteurs.

Une norme de matrice peut être définie à partir d'une norme quelconque de vecteur par $\|G\| = \sup_{u \neq 0} \|Gu\| / \|u\|$. On dit que cette norme de matrice est **subordonnée** à la norme de vecteur correspondante et l'on a alors les inégalités supplémentaires $\|GH\| \leqslant \|G\| \cdot \|H\|$, où H est une matrice et $\|Gu\| \leqslant \|G\| \cdot \|u\|$.

Si, pour définir une norme de matrice subordonnée, on utilise une norme de Hölder de vecteur on obtient les normes suivantes :

$$\| G \|_1 = \max_{1 \leqslant j \leqslant n} \sum_{i=1}^{n} | g_{ij} |$$

$$\| G \|_2 = \sqrt{\rho (G^T G)}$$

$$\| G \|_\infty = \max_{1 \leqslant i \leqslant n} \sum_{j=1}^{n} | g_{ij} |$$

avec $\rho (G^T G)$ la plus grande des valeurs propres de la matrice $G^T G$ (qui sont des nombres réels positifs ou nuls puisque cette matrice est symétrique semi-définie positive).

De plus, pour ces normes, on a $\rho (G) \leqslant \| G \|$ où le **rayon spectral** $\rho (G)$ est le plus grand des modules des valeurs propres de G.

On utilise aussi souvent la **norme de Frobenius** définie par $\| G \|_F^2 = \sum_{i,j=1}^{n} g_{ij}^2$. Ce n'est pas une norme subordonnée, mais on a cependant $\| GH \|_F \leqslant \| G \|_F \cdot \| H \|_F$.

Puisque l'on est dans un espace vectoriel de dimension finie, toutes ces normes sont équivalentes et l'on peut donc, pour étudier la convergence d'une suite de vecteurs ou de matrices, utiliser n'importe laquelle d'entre elles (cf. § 2.2).

On appelle **conditionnement** de la matrice G le nombre :

$$\text{cond } G = \| G \| \cdot \| G^{-1} \|$$

C'est un nombre plus grand ou égal à 1. Comme nous allons le voir, la notion de conditionnement est fondamentale dans l'étude des systèmes linéaires.

Nous voulions résoudre le système $Ax = b$. La matrice A a été perturbée par l'erreur d'affectation : elle est devenue $A + E$. Le second membre b a été également perturbé par l'erreur d'affectation : il est devenu $b + e$. La solution exacte de ce système perturbé n'est plus x mais un vecteur $x + \varepsilon$ où ε est un vecteur. On a donc :

$$(A + E)(x + \varepsilon) = b + e$$

On démontre le théorème **5**.

Théorème 5 – Si $x \neq 0$ et si $\| A^{-1} \| \, \| E \| < 1$, alors :

$$\frac{\| \varepsilon \|}{\| x \|} \leqslant \frac{\text{cond } A}{1 - \text{cond } A \, \| E \| / \| A \|} \left(\| e \| / \| b \| + \| E \| / \| A \| \right)$$

La condition $\| A^{-1} \| \, \| E \| < 1$ assure l'inversibilité de la matrice $A + E$. On voit que cond A est le facteur multiplicatif de l'erreur relative sur A et b. Si ce nombre est grand par rapport à 1, on voit que l'erreur relative sur x peut être grande ; on dit alors que la matrice A est **mal conditionnée** et une petite variation dans les données peut entraîner une grande variation dans le résultat. Si cond A est voisin de 1 (on démontre qu'il ne peut pas être inférieur à 1), alors une petite variation dans les données ne pourra induire qu'une petite variation dans le résultat ; on dit alors que la matrice A est **bien conditionnée**. Naturellement, des expressions telles que *grand par rapport à* 1 et *voisin de* 1 dépendent du nombre de digits t avec lequel travaille l'ordinateur. Si cond $A = 10^4$ et si $t = 6$, on peut dire que A est mal conditionnée, mais si $t = 16$, on peut la considérer comme bien conditionnée.

Cette notion, fondamentale, de conditionnement dépend uniquement du problème mathématique à résoudre et absolument pas de l'algorithme qui sera utilisé pour sa résolution. Viennent donc s'ajouter ensuite les erreurs dues à l'arithmétique de l'ordinateur qui se sont produites pendant l'exécution de l'algorithme : c'est la notion de stabilité numérique d'un algorithme dont nous avons parlé dans le dossier [AF 1 220] § 1.2.

Par conséquent, à cause de l'erreur d'affectation sur les données et de la stabilité numérique de l'algorithme, on n'obtient pas x exactement sur ordinateur mais un vecteur que nous appelons x'. On peut donner une majoration de l'erreur. On pose $r = Ax' - b$. On calcule ensuite, comme nous l'avons vu, la matrice A^{-1}. À cause des erreurs, nous n'obtiendrons pas A^{-1} exactement sur ordinateur, mais une matrice que nous noterons C. On calcule $R = AC - I$. On démontre le théorème **6**.

Théorème 6 – Si $\| R \| < 1$, alors :

$$\| x - x' \| \leqslant \frac{\| r \| \, \| C \|}{1 - \| R \|}$$

Ce théorème est plus intéressant que le théorème **5** car toutes les quantités, qui apparaissent dans la majoration, sont effectivement calculables, ce qui n'est pas le cas dans le théorème **5** (en effet, pour calculer cond A il faut connaître exactement A^{-1}).

L'erreur $e = x - x'$ vérifie :

$$\frac{\| r \|}{\| A \|} \leqslant \| e \| \leqslant \| A^{-1} \| \cdot \| r \|$$

Ces bornes nécessitent la connaissance de la norme de A et de son inverse et, de plus, elles peuvent être beaucoup trop larges. On peut estimer la norme de l'erreur par les quantités :

$$\sigma = \frac{\| r \|^2}{\| Ar \|} \quad \text{ou} \quad \sigma' = \frac{\| r \|^2}{\| A^T r \|}$$

On démontre que les rapports $\| e \| / \sigma$ et $\| e \| / \sigma'$ sont compris entre 1/cond A et cond A. Par conséquent, σ et σ' sont de bonnes estimations de $\| e \|$ si la matrice A est bien conditionnée. Cependant, dans la pratique, ces estimations sont souvent bonnes même si cond A n'est pas voisin de 1. Elles peuvent également être utilisées pour estimer l'erreur dans les méthodes itératives, e et r étant alors respectivement l'erreur et le résidu à la $k^{\text{ième}}$ itération (cf. § 2.2).

Soit e la solution de $Ae = r$. Alors $x' - e$ est une meilleure approximation de x que x'.

Les majorations données dans les théorèmes **5** et **6** sont en général beaucoup trop fortes. Pour les obtenir, on s'est, en effet, placé dans le pire des cas, celui où les erreurs ne se compensent jamais mais s'accumulent toujours (c'est cependant un cas qui peut se produire). Une estimation beaucoup plus réaliste de la précision obtenue est fournie par la méthode de permutation-perturbation décrite dans [2] [43] où les logiciels correspondants sont donnés.

Si le système est mal conditionné, on le remplace par $MAx = Mb$ où la matrice M est choisie de sorte que MA soit mieux conditionnée que A. Puisque le conditionnement de la matrice identité est égal à 1, cette matrice M devra être une approximation de A^{-1}. On dit que le système a été **préconditionné** et une telle matrice M s'appelle un **préconditionneur**. En pratique, on ne calculera jamais directement M^{-1} mais on aura à résoudre des systèmes de la forme :

$$My = c$$

avec c un vecteur quelconque.

La matrice M devra donc être choisie de sorte que de tels systèmes soient faciles à résoudre. Au lieu d'un préconditionneur M, on peut construire directement son inverse C et considérer le système $C^{-1} Ax = C^{-1} b$. Dans ce cas-là, aucun système auxiliaire ne devra être résolu. Il est également possible de considérer les préconditionnements :

$$AM'y = b \text{ et } MAM'y = Mb$$

avec

$$x = M'y$$

Il n'existe pas de choix canonique de M et chaque cas est particulier. Cependant, on peut construire des préconditionneurs par une **décomposition *LU* incomplète**. Comme dans la méthode de Gauss, on décompose A sous forme $A = LU$, mais on ne conserve que quelques-unes des diagonales de L et U les plus voisines de leurs diagonales principales. Soient L' et U' les matrices ainsi obtenues. On aura donc $A = L'U' + R$ et l'on prendra $M = L'U'$. Cette idée peut être adaptée à la décomposition de Cholesky (§ 2.1.3) et à celle de Householder (§ 2.1.4).

Le préconditionnement est une technique extrêmement importante. Non seulement elle peut permettre de réduire les problèmes numériques, mais la vitesse de convergence de certaines méthodes itératives de projection en dépend (cf. § 2.2.2).

Pour résoudre un système mal conditionné, il est également possible de faire appel à la **régularisation**. Cette technique consiste à rechercher le vecteur x_λ qui minimise :

$$\|Ax - b\|^2 + \lambda \|Hx\|^2$$

avec λ paramètre positif,
H matrice,

tous deux laissés au choix de l'utilisateur.

La norme utilisée pour les vecteurs est ici la norme euclidienne définie par $\|u\|^2 = (u, u)$. On montre que le vecteur x_λ, solution de ce problème de minimisation, est également solution du système $(A^TA + \lambda H^TH)\, x_\lambda = A^Tb$. Si $\lambda = 0$, alors x_λ sera égal à x mais il sera mal calculé à cause du mauvais conditionnement de A. Par contre, si λ n'est pas voisin de 0, le vecteur x_λ sera bien calculé mais pourra être très différent de x. Il faut donc faire un compromis dans le choix de ce paramètre. Des techniques basées sur la décomposition en valeurs singulières de A et sur l'extrapolation existent pour traiter ce problème. La question du choix de la matrice H se pose également. La régularisation est souvent utilisée dans la résolution des systèmes linéaires qui proviennent d'équations intégrales comme l'inversion de la transformée de Laplace, par exemple.

Voyons maintenant d'autres méthodes que celle de Gauss. Naturellement, ce que nous venons de dire sur le conditionnement et les erreurs dues à l'arithmétique de l'ordinateur reste valable.

2.1.3 Méthode de Cholesky

Supposons que la matrice A soit symétrique définie positive. On peut, bien entendu, utiliser la méthode de Gauss. Cependant, celle-ci ne prend pas en compte la symétrie de A et elle fait donc trop d'opérations arithmétiques. Dans ce cas, il faut lui préférer la **méthode de Cholesky** qui consiste à décomposer A sous la forme $A = LL^T$ où L est une matrice triangulaire inférieure. On pose ensuite $L^T x = y$ et le système s'écrit alors $Ly = b$. On peut calculer directement les éléments de la matrice L grâce aux relations suivantes :

$$\ell_{11} = \sqrt{a_{11}} \quad \ell_{i1} = a_{i1}/\ell_{11} \quad \text{pour } i = 2, ..., n$$

Puis pour $j = 2, ..., n$:

$$\ell_{jj} = \left(a_{jj} - \sum_{k=1}^{j-1} \ell_{jk}^2 \right)^{1/2}$$

$$\ell_{ij} = \left(a_{ij} - \sum_{k=1}^{j-1} \ell_{ik}\, \ell_{jk} \right) \Big/ \ell_{jj} \quad \text{pour } i = j+1, ..., n$$

On résout ensuite $Ly = b$, ce qui fournit y, puis $L^T x = y$ pour obtenir x.

Si la matrice A est symétrique mais si elle n'est pas définie positive, alors il existera un indice j tel que $\ell_{jj}^2 < 0$. On ne pourra donc pas en prendre la racine carrée et il faudra arrêter l'algorithme.

La méthode de Cholesky nécessite le calcul de n racines carrées et de l'ordre de $n^3/3$ opérations arithmétiques.

2.1.4 Méthode de Householder

Elle consiste à décomposer la matrice A en un produit :

$$A = QU$$

avec U matrice triangulaire supérieure,
Q matrice orthogonale, c'est-à-dire telle que $Q^{-1} = Q^T$.

Comme dans la méthode de Gauss, la triangularisation s'effectue peu à peu en $n - 1$ étapes. À l'étape k, la matrice $A^{(k)}$ est de la même forme que la matrice correspondante de la méthode de Gauss. On passe d'une étape à la suivante par les règles :

$$a_{kk}^{(k+1)} = -\left(\text{signe de } a_{kk}^{(k)}\right) \left(\sum_{i=k}^{n} \left| a_{ik}^{(k)} \right|^2 \right)^{1/2}$$

$$v = a_{kk}^{(k+1)} \left(a_{kk}^{(k+1)} - a_{kk}^{(k)} \right)$$

$$v_k = a_{kk}^{(k)} - a_{kk}^{(k+1)}$$

$$v_i = a_{ik}^{(k)} \quad \text{pour } i = k+1, ..., n$$

$$\beta_j = \sum_{i=k}^{n} v_i\, a_{ij}^{(k)} \quad \text{pour } j = k+1, ..., n$$

$$\gamma_j = \beta_j / v \quad \text{pour } j = k+1, ..., n$$

$$a_{ij}^{(k+1)} = a_{ij}^{(k)} - \gamma_j v_i \quad \text{pour } i = k, ..., n \text{ et } j = k+1, ..., n$$

$$\beta_{n+1} = \sum_{i=k}^{n} v_i\, b_i$$

$$\gamma_{n+1} = \beta_{n+1} / v$$

$$b_i^{(k+1)} = b_i^{(k)} - \gamma_{n+1} v_i \quad \text{pour } i = k, ..., n$$

Naturellement v, les v_i, les β_j et les γ_j dépendent de k. Le système $A^{(n)} x = b^{(n)}$ ainsi obtenu est triangulaire supérieur et on le résout comme dans la méthode de Gauss (§ 2.1.1).

La résolution d'un système linéaire par cette méthode nécessite de l'ordre de $4n^3/3$ opérations arithmétiques, c'est-à-dire environ deux fois plus que la méthode de Gauss ; elle est cependant numériquement plus stable. On démontre que la matrice Q est donnée par $Q = H^{(1)} H^{(2)} \ldots H^{(n-1)}$ avec :

$$H^{(k)} = \begin{pmatrix} I & 0 \\ 0 & \tilde{H}^{(k)} \end{pmatrix}$$

où la matrice $\tilde{H}^{(k)}$, de dimensions $(n - k + 1) \times (n - k + 1)$, est :

$$\tilde{H}^{(k)} = I - v \cdot v^T / v$$

avec v vecteur de composantes $v_{k+1}, ..., v_n$ de la $k^{\text{ième}}$ étape,
v valeur correspondante de cette étape.

2.2 Méthodes itératives

2.2.1 Méthodes de relaxation

Pour résoudre le système $Ax = b$, on pose :

$$A = M - N$$

Cette décomposition additive peut naturellement s'effectuer d'une infinité de façons. Nous supposerons que la matrice M est inversible et que son inverse est facile à calculer. Le système à résoudre s'écrit :

$$Mx = Nx + b$$

soit encore :

$$x = M^{-1} Nx + M^{-1} b$$

Comme dans la méthode des approximations successives décrite au paragraphe 1.1, nous allons choisir arbitrairement un vecteur de départ $x^{(0)}$ et générer une suite de vecteurs $(x^{(k)})$ par une méthode dite de **relaxation** :

$$x^{(k+1)} = M^{-1} Nx^{(k)} + M^{-1} b \text{ pour } k = 0, 1, ...$$

On a le résultat fondamental du théorème **7**.

Théorème 7 – Soit $\lambda_1, \lambda_2, ..., \lambda_n$ les valeurs propres de $M^{-1} N$. Une condition nécessaire et suffisante pour que la suite $(x^{(k)})$ converge vers x est que :

$$\max_{1 \leqslant i \leqslant n} |\lambda_i| < 1$$

Si $\| M^{-1} N \| < 1$, alors $(x^{(k)})$ converge vers x.

Étudions maintenant les choix les plus courants pour les matrices M et N. Appelons D la matrice diagonale formée par la diagonale de A, $-E$ la partie strictement triangulaire inférieure de A et $-F$ la partie strictement triangulaire supérieure de A, on a :

$$A = D - E - F$$

■ **La méthode de Jacobi** consiste à prendre $M = D$ et $N = E + F$. Ses itérations s'écrivent :

$$x_i^{(k+1)} = \left(b_i - \sum_{\substack{j=1 \\ j \neq i}}^{n} a_{ij} x_j^{(k)} \right) / a_{ii} \quad \text{pour } i = 1, ..., n$$

■ **La méthode de Gauss-Seidel** consiste à prendre $M = D - E$ et $N = F$. Ses itérations s'écrivent :

$$x_i^{(k+1)} = \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)} \right) / a_{ii} \quad \text{pour } i = 1, ..., n$$

■ **La méthode de surrelaxation** consiste à prendre $M = (D - \omega E)/\omega$ et $N = [(1 - \omega D) + \omega F]/\omega$ où ω est un paramètre non nul. Ses itérations s'écrivent :

$$\tilde{x}_i^{(k+1)} = \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)} \right) / a_{ii} \quad \text{pour } i = 1, ..., n$$

$$x_i^{(k+1)} = (1 - \omega) x_i^{(k)} + \omega \tilde{x}_i^{(k+1)} \quad \text{pour } i = 1, ..., n$$

On voit que le vecteur intermédiaire $\tilde{x}^{(k+1)}$ est celui obtenu par la méthode de Gauss-Seidel et que le vecteur $x^{(k+1)}$ est une combinaison linéaire de $x^{(k)}$ et de $\tilde{x}^{(k+1)}$. Naturellement, pour $\omega = 1$, on retrouve exactement la méthode de Gauss-Seidel.

Ces trois méthodes ne vérifient pas la condition du théorème **7** quelle que soit la matrice A. Il faut imposer des conditions que l'on trouve dans [14], dans [41] qui est un grand classique ou dans [33] qui est plus récent. Nous allons donner les résultats les plus importants de cette théorie.

Commençons par un résultat général.

Théorème 8 – Soit $A = M - N$ une décomposition additive quelconque de la matrice A. Si A est symétrique définie positive et si $M^T + N$ est symétrique définie positive, alors la condition du théorème **6** est satisfaite.

Le cas où A est symétrique définie positive est important pour les applications. Nous avons le théorème **9**.

Théorème 9 – Soit A une matrice symétrique définie positive. Une condition nécessaire et suffisante pour que la méthode de surrelaxation converge est que $0 < \omega < 2$.

La méthode de Gauss-Seidel converge donc dans ce cas.

Pour certaines classes de matrices (en particulier certaines matrices tridiagonales par blocs), il existe une valeur optimale de ω (c'est-à-dire telle que la convergence soit la plus rapide possible) qu'il est possible de caractériser.

Pour la méthode de Jacobi, nous avons le théorème **10**.

Théorème 10 – Si A est symétrique et si $a_{ii} > 0$ pour $i = 1, ..., n$, alors la méthode de Jacobi est convergente si et seulement si $N (= E + F)$ et $2D - N$ sont définies positives.

Enfin, nous donnons le théorème **11**.

Théorème 11 – Si $a_{ii} > \sum_{j=1,\ j \neq i}^{n} |a_{ij}|$ pour $i = 1, ..., n$, alors les méthodes de Jacobi et de Gauss-Seidel sont convergentes.

On pourra consulter [33] qui contient de nombreux renseignements sur des matrices particulières.

2.2.2 Méthodes de projection

Soit l'itéré $x^{(k)}$; on choisit un vecteur $z^{(k)}$ et l'on considère le plan d'équation :

$$(z^{(k)}, Ay - b) = 0$$

qui passe par la solution x. On projette ensuite $x^{(k)}$ orthogonalement sur ce plan (c'est-à-dire parallèlement à $A^T z^{(k)}$) pour obtenir $x^{(k+1)}$. On a donc :

$$x^{(k+1)} = x^{(k)} - \frac{(z^{(k)}, Ax^{(k)} - b)}{(A^T z^{(k)}, A^T z^{(k)})} A^T z^{(k)}$$

où $(.,.)$ désigne le produit scalaire de deux vecteurs.

On démontre [ce qui n'implique pas la convergence de $(x^{(k)})$ vers x] que :

$$\| x^{(k+1)} - x \| \leqslant \| x^{(k)} - x \|$$

Les différentes méthodes se différencient par le choix de $z^{(k)}$ à chaque itération et les conditions de convergence. Nous n'en parlerons pas ici dans le cas général, mais nous allons maintenant étudier plus en détail le cas où la matrice A est symétrique définie positive.

■ On considère :

$$F : \mathbb{R}^n \to \mathbb{R}$$

définie par :

$$F(y) = (y, Ay)/2 - (y, b)$$

avec y un vecteur quelconque.

L'équation $F(y) = a$ où a est un nombre arbitraire supérieur ou égal à $F(x)$ est l'équation d'un ellipsoïde dans $\mathbb{R}^n$ dont x est le centre de symétrie. Soit l'itéré $x^{(k)}$. On choisit un vecteur $u^{(k)}$ arbitraire passant par $x^{(k)}$ et l'on prend pour $x^{(k+1)}$ le point de $u^{(k)}$ où F atteint son minimum. On a donc :

$$x^{(k+1)} = x^{(k)} - \frac{(u^{(k)}, Ax^{(k)} - b)}{(u^{(k)}, Au^{(k)})} u^{(k)}$$

On démontre que les vecteurs $u^{(k)}$ et $Ax^{(k+1)} - b$ sont orthogonaux. $Ax^{(k+1)} - b$ est, par conséquent, dirigé suivant la normale à l'éllipsoïde passant par $x^{(k+1)}$ et tangent à $u^{(k)}$. La décroissance de F est donc la plus rapide dans la direction de $Ax^{(k)} - b$. On peut alors choisir $u^{(k)} = Ax^{(k)} - b$. C'est ce que l'on appelle la **méthode de la plus profonde descente**.

Puisque la matrice A est symétrique définie positive, ses valeurs propres sont réelles et strictement positives. Soit λ_m la plus petite d'entre elles et λ_M la plus grande. On a le théorème **12**.

Théorème 12 – La méthode de la plus profonde descente converge, quel que soit le vecteur initial $x^{(0)}$, si $\lambda_M/\lambda_m < 2$.

■ Nous allons maintenant choisir $u^{(k)}$ dans le plan des vecteurs $u^{(k+1)}$ et $Ax^{(k)} - b$ et de sorte que :

$$(u^{(k)}, Au^{(k-1)}) = 0$$

On montre que :

$$u^{(k)} = -r^{(k)} + \frac{(r^{(k)}, Au^{(k-1)})}{(u^{(k-1)}, Au^{(k-1)})} u^{(k-1)}$$

avec $r^{(k)} = Ax^{(k)} - b$ ou, ce qui revient au même :

$$u^{(k)} = -r^{(k)} + \frac{(r^{(k)}, r^{(k)})}{(r^{(k-1)}, r^{(k-1)})} u^{(k-1)}$$

Cette méthode s'appelle **méthode du gradient conjugué**. On a le résultat fondamental du théorème **13**.

Théorème 13 – La méthode du gradient conjugué converge en n itérations au plus.

Cela signifie qu'il existe un indice k, inférieur ou égal à n, tel que $x^{(k)} = x$. La méthode du gradient conjugué est donc une méthode directe de résolution des systèmes linéaires.

■ À cause des erreurs dues à l'arithmétique de l'ordinateur, il se peut qu'il faille plus de n itérations pour obtenir x. Il existe des techniques particulières [27] pour remédier à ce problème. La méthode du gradient conjugué est largement utilisée pour la minimisation de fonctions avec ou sans contraintes [20]. Elle s'étend aux matrices non symétriques et l'on obtient la **méthode du gradient biconjugué**. Préconditionné par la matrice M, il consiste à choisir $x^{(0)}$ et z, à poser $r^{(0)} = b - Ax^{(0)}$ et $r'^{(0)} = z$, puis à résoudre $M\tilde{r}^{(0)} = r^{(0)}$ et $M^T \tilde{z}'^{(0)} = z$.

On pose $\tilde{z}^{(0)} = \tilde{r}^{(0)}$ et $\rho_0 = (r'^{(0)}, \tilde{r}^{(0)})$. Puis, pour $k = 0, 1, \ldots$

$$r^{(k+1)} = r^{(k)} - \lambda_{k+1} A\tilde{z}^{(k)}$$

$$x^{(k+1)} = x^{(k)} + \lambda_{k+1} \tilde{z}^{(k)}$$

$$r'^{(k+1)} = r'^{(k)} - \lambda_{k+1} A^T \tilde{z}'^{(k)}$$

$$\tilde{r}^{(k+1)} = M^{-1} r^{(k+1)}$$

$$\tilde{r}'^{(k+1)} = M^{-T} r'^{(k+1)}$$

$$\tilde{z}^{(k+1)} = \tilde{r}^{(k+1)} + \alpha_{k+1} \tilde{z}^{(k)}$$

$$\tilde{z}'^{(k+1)} = \tilde{r}'^{(k+1)} + \alpha_{k+1} \tilde{z}'^{(k)}$$

avec

$$\lambda_{k+1} = \rho_k / (\tilde{z}'^{(k)}, A\tilde{z}^{(k)})$$

$$\rho_{k+1} = (r'^{(k+1)}, \tilde{r}^{(k+1)})$$

$$\alpha_{k+1} = \rho_{k+1}/\rho_k$$

L'intérêt de ces méthodes est que, en général, $x^{(k)}$ est une bonne approximation de la solution x bien avant la $n^{\text{ième}}$ itération. Il est cependant possible de trouver des exemples où cela n'est pas le cas.

D'autre part, dans la méthode du gradient biconjugué, une division par zéro peut se produire. On saute alors directement de l'itération k à l'itération $k+2$. Si une nouvelle division par zéro se produit, on saute à l'itération $k+3$ et ainsi de suite jusqu'à pouvoir poursuivre l'algorithme. De même, on effectuera des sauts lors de la division par un nombre voisin de zéro afin d'éviter une propagation trop importante des erreurs dues à l'arithmétique de l'ordinateur. Ces techniques sont exposées dans [8].

■ De nombreuses méthodes de projection consistent à construire une suite d'itérés définie par les deux conditions :

$$x^{(k)} - x^{(0)} \in \mathcal{K}_k \text{ et } r^{(k)} = b - Ax^{(k)} \perp \mathcal{L}_k$$

avec $\mathcal{K}_k$ et $\mathcal{L}_k$ deux sous-espaces de dimension k.

Dans la **méthode de Lanczos** :

$$\mathcal{K}_k = K_k(A, r^{(0)}) \text{ et } \mathcal{L}_k = K_k(A^T, z)$$

avec $z \neq 0$ vecteur donné,

$K_k(M, u)$ **sous-espace de Krylov** engendré par les vecteurs $u, Mu, \ldots, M^{k-1} u$.

On démontre alors que :

$$r^{(k)} = P_k(A) r^{(0)}$$

avec P_k polynôme qui vérifie les conditions d'orthogonalité $(z, A^i P_k(A) r^{(0)}) = 0$ pour $i = 0, \ldots, k-1$.

La méthode du gradient biconjugué est l'un des algorithmes pour mettre en œuvre la **méthode de Lanczos**. Tous ces algorithmes sont basés sur les **polynômes orthogonaux formels**, c'est-à-dire sur des polynômes qui vérifient les conditions d'orthogonalité précédentes [4].

■ Pour les grands systèmes creux, la matrice est stockée par adressage indirect et le produit d'un vecteur par A^T pose certains problèmes. Des variantes, ne nécessitant pas de tels produits, ont été proposées. Ainsi, dans la méthode **CGS** (*Conjugate Gradient Squared*), les résidus sont définis par :

$$r^{(k)} = P_k^2(A) r^{(0)}$$

alors que, dans la méthode **BiCGSTAB** (*Biconjugated Gradient Stabilized*), on a :

$$r^{(k)} = W_k(A)\, P_k(A)\, r^{(0)}$$

avec W_k polynôme défini par $W_0(\xi) = 1$ et $W_{k+1}(\xi) = (1 - a_k\xi)\, W_k(\xi)$.

Le paramètre a_k est choisi de sorte que $(r^{(k+1)}, r^{(k+1)})$ soit minimal, ce qui a un effet lissant sur la convergence.

Les algorithmes pour mettre en œuvre ces méthodes sont dérivés des relations de récurrence des polynômes orthogonaux formels qui sont à la base de la méthode du gradient biconjugué et des autres algorithmes pour implémenter la méthode de Lanczos.

Les divisions par zéro ou par un nombre voisin de zéro peuvent être traitées de manière similaire à ce qui est fait dans la méthode du gradient biconjugué.

■ Une autre méthode particulièrement intéressante est **GMRES** (*General Minimal Residual*). Les itérés sont tels que :

$$\|r^{(k)}\| = \min_{p \in \mathcal{P}_k} \frac{\|p(A)r^{(0)}\|}{|p(0)|} \quad \text{d'où} \quad \frac{\|r^{(k)}\|}{\|r^{(0)}\|} \leqslant \min_{p \in \mathcal{P}_k} \frac{\|p(A)\|}{|p(0)|}$$

avec $\mathcal{P}_k$ espace vectoriel des polynômes de degré au plus égal à k.

● La mise en œuvre de cette méthode est basée sur la **méthode d'Arnoldi** qui permet de construire une base orthonormale du sous-espace de Krylov :

$$K_k(A, v_1)$$

avec :

$$v_1 = r^{(0)} / \|r^{(0)}\|_2$$

Elle entre dans le cadre des méthodes de projection avec $\mathcal{K}_k = K_k(A, v_1)$ et $\mathcal{L}_k = A \cdot K_k(A, v_1)$. Dans cette méthode, le calcul d'un itéré utilise tous les itérés précédents.

● En pratique, on se contente de n'en conserver que les m derniers et de réinitialiser la méthode toutes les m itérations à partir de $x^{(m)}$. C'est ce que l'on appelle la méthode **GMRES redémarrée**.

Sur ces méthodes, se reporter aux références [3] [5] [10] [33] [39] [42].

3. Calcul des valeurs propres

Les méthodes de calcul des valeurs propres d'une matrice se scindent en deux classes suivant que l'on désire les calculer toutes ou calculer seulement celles de plus grand module.

On trouvera des développements sur les méthodes de calcul des valeurs propres dans les références [1] [21] [30] [38]. Mais il faudra surtout consulter [13] qui est un ouvrage de référence sur le sujet.

3.1 Méthode de la puissance

Elle entre dans la seconde classe.

Soit A la matrice dont on veut calculer les valeurs propres $\lambda_1, ..., \lambda_n$. Nous supposerons celles-ci numérotées suivant l'ordre décroissant de leurs modules :

$$|\lambda_1| \geqslant |\lambda_2| \geqslant ... \geqslant |\lambda_n|$$

Soit $x_1, x_2, ..., x_n$ les vecteurs propres correspondants. La méthode de la puissance consiste à choisir un vecteur u_0 puis à construire la suite de vecteurs (u_k) par :

$$u_{k+1} = Au_k \text{ pour } k = 0, 1, ...$$

et à calculer la suite de nombres :

$$S_k = (y, u_{k+1})/(y, u_k),\ k = 0, 1, ...$$

avec y vecteur arbitraire,

$(.,.)$ produit scalaire de deux vecteurs.

On a le théorème **14**.

Théorème 14 – Si $x_1, x_2, ..., x_n$ forment une base, si $\lambda_1 = ... = \lambda_r$, si $|\lambda_r| > |\lambda_{r+1}|$, si $(u_0, x_i) \neq 0$ et si $(y, x_i) \neq 0$ pour $i = 1, ..., r$, alors la suite (S_k) converge vers λ_1 et l'on a :

$$S_k = \lambda_1 + O[(\lambda_{r+1}/\lambda_1)^k] \quad (k \to \infty)$$

Lorsque les hypothèses de ce théorème ne sont pas satisfaites, il existe, dans certains cas, des variantes de la méthode de la puissance. Si $|\lambda_1|$ est très voisin de $|\lambda_{r+1}|$, la suite (S_k) converge lentement vers λ_1. On peut alors accélérer sa convergence par le procédé Δ^2 d'Aitken (§ 1.3) ; la suite ainsi obtenue converge vers λ_1 avec une vitesse régie par :

$$(\lambda_{r+2}/\lambda_1)^k$$

Une fois la valeur propre λ_1 obtenue, on peut modifier soit la matrice, soit la méthode pour calculer λ_2. On voit qu'une telle procédure nécessite, pour commencer les itérations qui vont converger vers λ_2, d'attendre que la méthode de la puissance ait convergé vers λ_1. Elle est donc peu utilisée et il vaut mieux lui préférer d'autres méthodes (cf § 3.5).

3.2 Calcul du polynôme caractéristique

On appelle **polynôme caractéristique** d'une matrice le polynôme dont les racines sont ses valeurs propres. Nous allons étudier des méthodes permettant d'obtenir ce polynôme caractéristique. On calculera ensuite ses racines (les valeurs propres de A) en utilisant, par exemple, la méthode de Bairstow (§ 1.6). L'idée de ces méthodes est de transformer la matrice A en une matrice tridiagonale semblable, c'est-à-dire en une matrice ayant les mêmes valeurs propres et de la forme :

$$B_n = \begin{pmatrix} a_1 & b_1 & & & 0 \\ c_1 & a_2 & b_2 & & \\ 0 & c_2 & a_3 & b_3 & \\ & & \ddots & \ddots & \ddots \\ & 0 & & c_{n-1} & a_n \end{pmatrix} \quad b_{n-1}$$

Appelons B_k, la matrice formée par les k premières lignes et les k premières colonnes de B_n et P_k le polynôme caractéristique de B_k. On montre que l'on peut calculer récursivement le polynôme caractéristique P_n de B_n (donc de A) par :

$$P_0(t) = 1,\ P_1(t) = a_1 - t$$

$$P_k(t) = (a_k - t)\, P_{k-1}(t) - b_{k-1}\, c_{k-1}\, P_{k-2}(t)$$

pour $k = 2, ..., n$

On voit que l'on obtient une relation de récurrence à trois termes comme dans le cas des polynômes orthogonaux (cf. dossier [AF 1 220] § 3.4). Il n'est donc pas surprenant qu'il y ait des

connexions entre ces deux sujets. C'est ainsi que l'on a le théorème **15**.

Théorème 15 – Si $b_i\, c_i \neq 0$ pour $i = 1, ..., n-1$, alors P_i et P_{i-1} n'ont pas de racine commune pour $i = 1, ..., n$. Si $b_i\, c_i > 0$ pour $i = 1, ..., n-1$, alors les racines de P_i sont réelles, distinctes et séparées par celles de P_{i+1} pour tout i.

On pourra comparer ce résultat à celui du théorème **18** du dossier [AF 1 220].

Pour transformer la matrice A en une matrice tridiagonale semblable, on utilise la **méthode de Lanczos**. On choisit deux vecteurs arbitraires x et y non nuls, on pose :

$$x_0 = 0 \qquad y_0 = 0$$

$$x_1 = x \qquad y_1 = y$$

puis l'on calcule :

$$x_{k+1} = Ax_k - a_k\, x_k - b_{k-1}\, x_{k-1}$$

$$y_{k+1} = A^T y_k - a_k\, y_k - b_{k-1}\, y_{k-1}$$

pour $k = 1, ..., n-1$ avec $b_0 = 0$

et $$a_k = (Ax_k, y_k)/(x_k, y_k)$$

$$b_{k-1} = (Ax_k, y_{k-1})/(x_{k-1}, y_{k-1})$$

La matrice :

$$\begin{pmatrix} a_1 & b_1 & & & 0 \\ 1 & a_2 & b_2 & & \\ 0 & 1 & a_3 & b_3 & \ddots \\ & & \ddots & \ddots & b_{n-1} \\ 0 & & & 1 & a_n \end{pmatrix}$$

est semblable à A. On démontre de plus, que $\forall\ i \neq j$, $(x_i, y_j) = 0$; pour cette raison, on parle souvent de la **méthode de biorthogonalisation de Lanczos**. Lorsque la matrice A est symétrique, on réduit le volume des calculs en prenant $x = y$; on a alors $x_k = y_k$ pour tout k.

Dans cette méthode, il faut faire attention à la propagation des erreurs dues à l'arithmétique de l'ordinateur ; on utilise des techniques particulières pour obvier à cet inconvénient.

La méthode de Lanczos est à la base de la méthode du gradient conjugué décrite dans le paragraphe 2.2.2.

3.3 Forme de Hessenberg

Nous allons maintenant voir comment transformer une matrice quelconque A en une matrice de (la forme de) Hessenberg supérieure. C'est une technique très utile car elle réduit considérablement le volume des calculs dans les méthodes d'obtention des valeurs propres que nous verrons au paragraphe 3.4 suivant.

On dit qu'une matrice H est une **matrice de Hessenberg supérieure** si elle est de la forme :

$$H = \begin{pmatrix} h_{11} & h_{12} & \cdots & h_{1n} \\ h_{21} & h_{22} & \cdots & h_{2n} \\ 0 & h_{32} & \ddots & \vdots \\ & \ddots & \ddots & h_{n,n-1} \\ 0 & & 0 & h_{nn} \end{pmatrix}$$

c'est-à-dire que $h_{ij} = 0$ pour $i > j + 1$. On démontre le théorème **16**.

Théorème 16 – Soit A une matrice quelconque. Il existe une matrice orthogonale P (c'est-à-dire telle que $P^{-1} = P^T$) telle que $H = P^T AP$ soit une matrice de Hessenberg supérieure.

Les matrices A et H sont donc semblables. Pour obtenir H, la technique utilisée est similaire à celle de la méthode de Householder pour la résolution d'un système linéaire (§ 2.1.4). On pose $A_0 = A$ puis on calcule les matrices $A_1, A_2, ..., A_{n-2}$ par :

$$A_k = P_k^T A_{k-1} P_k$$

avec P_k matrice orthogonale,

A_{k-1} de la forme de Hessenberg supérieure jusqu'à la colonne $k-1$, c'est-à-dire de la forme :

$$A_{k-1} = \left(\begin{array}{c|c} H_{k-1} & C_{k-1} \\ \hline 0 \;\; b_{k-1} & B_{k-1} \end{array}\right)$$

H_{k-1} matrice de Hessenberg supérieure de dimension $k \times k$,

b_{k-1} vecteur de dimension $n-k$.

P_k est de la forme :

$$P_k = \left(\begin{array}{c|c} I & 0 \\ \hline 0 & Q_k \end{array}\right)$$

avec I matrice identité $k \times k$,

Q_k matrice $(n-k) \times (n-k)$.

Soit $a_{ij}^{(k-1)}$ les éléments de $A^{(k-1)}$. On a :

$$P_k = I - uu^T/v$$

avec

$$u_i = 0 \text{ pour } i = 1, ..., k$$

$$u_{k+1} = a_{k+1,k}^{(k-1)} \mp q$$

$$u_i = a_{ik}^{(k-1)} \qquad \text{pour } i = k+2, ..., n$$

$$q = \left(\sum_{i=k+1}^{n} (a_{ik}^{(k-1)})^2 \right)^{1/2}$$

$$v = q^2 \mp a_{k+1,k}^{(k-1)} q$$

Dans ces expressions, le signe à placer devant q est celui de $a_{k+1,k}^{(k-1)}$. Après multiplication, on obtient :

$$A_k = \left(\begin{array}{c|c} H_{k-1} & C_{k-1} Q_k \\ \hline 0 \;\; Q_k b_{k-1} & Q_k B_{k-1} Q_k \end{array}\right)$$

On voit que, si A est symétrique, il en est de même des matrices A_k. Donc A_{n-2} est une matrice de Hessenberg supérieure symétrique : elle est tridiagonale. On peut alors utiliser cette procédure à la place de celle de Lanczos pour les matrices symétriques.

Le calcul de A_{n-2} demande de l'ordre de $5n^3/3$ multiplications.

3.4 Méthodes de décomposition

De nombreuses méthodes de calcul des valeurs propres d'une matrice A relèvent de la même idée de base : la décomposition de

A en un produit de deux matrices $A = BC$ où B est inversible. On a donc :

$$CB = B^{-1} AB$$

c'est-à-dire que les matrices BC et CB sont semblables. On va alors recommencer la même décomposition sur la matrice CB et ainsi de suite ; d'où l'algorithme :

$$A_0 = A = B_0 C_0$$

$$A_1 = C_0 B_0 = B_1 C_1$$

.....

$$A_k = C_{k-1} B_{k-1} = B_k C_k$$

$$A_{k+1} = C_k B_k = B_{k+1} C_{k+1}$$

.....

On a $A_{k+1} = B_k^{-1} A_k B_k$ et toutes les matrices A_k ainsi obtenues sont semblables à A. On va donc choisir la décomposition de sorte que la suite de matrices (A_k) converge vers une matrice que nous appellerons A_∞ et dont les valeurs propres sont faciles à calculer.

Posons :

$$P_k = B_0 B_1 \ldots B_k$$

On a :

$$A_{k+1} = P_k^{-1} A P_k$$

Pour que (A_k) converge, il suffit donc que (P_k) converge. Si nous appelons P_∞ sa limite, on a :

$$\lim_{k \to \infty} B_k = P_\infty^{-1} P_\infty = I$$

puisque $B_k = P_{k-1}^{-1} P_k$.

D'où finalement :

$$A_\infty = \lim_{k \to \infty} B_k C_k = \lim_{k \to \infty} C_k$$

La décomposition devra par conséquent être choisie de sorte que les valeurs propres des matrices C_k soient faciles à calculer ; c'est le cas lorsque les matrices C_k sont triangulaires supérieures car les valeurs propres sont alors les termes de la diagonale principale. Or nous connaissons justement deux décompositions où les matrices C sont triangulaires supérieures : la méthode de Gauss et la méthode de Householder pour résoudre les systèmes linéaires (§ 2.1). Elles conduisent donc à deux algorithmes de calcul des valeurs propres de A : l'algorithme LR si l'on utilise la décomposition de Gauss et l'algorithme QR si l'on utilise celle de Householder. Le problème principal, dans les deux cas, est celui de la convergence de l'algorithme car cette convergence n'a lieu que sous certaines conditions.

3.4.1 Algorithme *LR*

Pour cet algorithme, nous avons les théorèmes **17** et **18**.

Théorème 17 – Si A est diagonalisable, si $|\lambda_1| > |\lambda_2| > \ldots > |\lambda_n|$ et si tous les déterminants principaux de la matrice des vecteurs propres de A et de son inverse sont différents de zéro, alors l'algorithme LR converge et :

$$A_\infty = \begin{pmatrix} \lambda_1 & & 0 \\ & \lambda_2 & \\ & & \ddots \\ 0 & & \lambda_n \end{pmatrix}$$

Théorème 18 – Si A est symétrique définie positive, l'algorithme LR converge.

Si A est symétrique définie positive, on peut utiliser la décomposition de Cholesky (§ 2.1.3) au lieu de celle de Gauss mais nous verrons plus loin une autre méthode (§ 3.4.3).

En général, la convergence de l'algorithme LR est lente : elle dépend du rapport $|\lambda_2/\lambda_1|$. Pour aller plus vite, on peut diminuer le volume des calculs à effectuer à chaque itération. Pour cela, on commence par transformer la matrice A en une matrice de Hessenberg supérieure, puis on applique l'algorithme LR. On a le théorème **19**.

Théorème 19 – Si A est une matrice de Hessenberg supérieure, il en est de même de toutes les matrices A_k construites par l'algorithme LR.

De plus, si A est une matrice de Hessenberg supérieure, il est possible d'accélérer la convergence de la méthode LR en effectuant des déplacements d'origine (**shift**). À l'itération k supposons que nous ayons obtenu une bonne approximation p_k de λ_n, on effectuera la décomposition de $A_k - p_k I$ au lieu de celle de A_k, c'est-à-dire que :

$$A_k - p_k I = L_k U_k$$

$$A_{k+1} = U_k L_k + p_k I$$

avec L_k triangulaire inférieure à diagonale unité,

U_k triangulaire supérieure.

Si les valeurs propres de A sont toutes de modules distincts (c'est la condition du théorème **17**, alors l'élément $a_{nn}^{(k)}$ de A_k tend vers λ_n lorsque k tend vers l'infini. On prend donc $p_k = a_{nn}^{(k)}$ et l'on commence ces déplacements d'origine dès que $\left|a_{n,n-1}^{(k)}\right|$ est petit. Lorsque $a_{n,n-1}^{(k)}$ est devenu nul (à la précision désirée), alors $a_{nn}^{(k)}$ est égal à peu près à λ_n et l'on peut réduire la taille du problème en travaillant sur la sous-matrice de dimension $n-1$ qui est, elle aussi, de la forme de Hessenberg supérieure. Dès que $\left|a_{n-1,\,n-2}^{(k)}\right|$ est petit, on recommence les déplacements d'origine avec $p_k = a_{n-1,\,n-1}^{(k)}$ qui sera alors une approximation de λ_{n-1} et ainsi de suite.

3.4.2 Algorithme *QR*

Pour cet algorithme, nous avons le théorème **20**.

Théorème 20 – Si A est non singulière et si ses valeurs propres sont toutes de modules différents, alors l'algorithme QR converge et la matrice A_∞ est triangulaire supérieure.

On voit que les conditions du théorème **20** sont bien moins fortes que celles du théorème **17**, mais c'est surtout le résultat du théorème **21** qui établit la supériorité de l'algorithme QR.

Théorème 21 – Si l'on applique l'algorithme QR à une matrice de Hessenberg supérieure, toutes les matrices A_k sont des matrices de Hessenberg supérieures. De plus, si $a_{i+1,i} \neq 0$ pour $i = 1, \ldots, n-1$, alors l'algorithme QR converge et la matrice A_∞ est triangulaire par blocs. Les valeurs propres de chaque bloc diagonal (qui sont des valeurs propres de A) ont toutes le même module.

On peut également effectuer des déplacements d'origine dans l'algorithme QR.

3.4.3 Méthode de Jacobi

Lorsque la matrice A est symétrique, il existe une méthode intéressante : celle de Jacobi. Nous avons vu, dans l'introduction du paragraphe 3.4, que :

$$A_{k+1} = B_k^{-1} A_k B_k \qquad \text{pour} \quad k = 0, 1, \ldots$$

avec $A_0 = A$.

La méthode de Jacobi consiste à prendre :

$$B_k = \begin{pmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & \cos\theta & & -\sin\theta & & \\ & & & 1 & & & \\ & & & & \ddots & & \\ & & \sin\theta & & \cos\theta & & \\ & & & & & \ddots & \\ & & & & & & 1 \end{pmatrix} \begin{matrix} \\ \\ \leftarrow p \\ \\ \leftarrow q \\ \\ \\ \end{matrix}$$

$$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \uparrow \qquad\qquad \uparrow$$
$$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad p \qquad\qquad q$$

avec p, q, θ dépendant de k et où tous les autres éléments sont nuls.

On peut vérifier facilement que $B_k^{-1} = B_k^T$ et que toutes les matrices A_k sont symétriques. Lorsque l'on passe de A_k à A_{k+1} seuls sont modifiés les éléments des lignes et des colonnes p et q. En notant $a_{ij}^{(k)}$ les éléments de A_k, on a :

$$a_{ij}^{(k+1)} = a_{ij}^{(k)} \qquad \text{pour} \quad i, j \neq p, q$$
$$a_{ip}^{(k+1)} = a_{ip}^{(k)} \cos\theta + a_{iq}^{(k)} \sin\theta \qquad \text{pour} \quad i \neq p, q$$
$$a_{iq}^{(k+1)} = -a_{ip}^{(k)} \sin\theta + a_{iq}^{(k)} \cos\theta \qquad \text{pour} \quad i \neq p, q$$
$$a_{pp}^{(k+1)} = a_{pp}^{(k)} \cos^2\theta + 2a_{pq}^{(k)} \sin\theta\cos\theta + a_{qq}^{(k)} \sin^2\theta$$
$$a_{qq}^{(k+1)} = a_{pp}^{(k)} \sin^2\theta - 2a_{pq}^{(k)} \sin\theta\cos\theta + a_{qq}^{(k)} \cos^2\theta$$
$$a_{pj}^{(k+1)} = a_{pj}^{(k)} \cos\theta + a_{qj}^{(k)} \sin\theta \qquad \text{pour} \quad j \neq p, q$$
$$a_{qj}^{(k+1)} = -a_{pj}^{(k)} \sin\theta + a_{qj}^{(k)} \cos\theta \qquad \text{pour} \quad j \neq p, q$$
$$a_{pq}^{(k+1)} = a_{qp}^{(k+1)} = (a_{qq}^{(k)} - a_{pp}^{(k)}) \sin\theta\cos\theta + a_{pq}^{(k)} (\cos^2\theta - \sin^2\theta)$$

On va choisir θ afin que :

$$a_{pq}^{(k+1)} = a_{qp}^{(k+1)} = 0$$

c'est-à-dire que :

$$\cos\theta = \frac{1}{\sqrt{2}} \left(1 + \frac{1}{\sqrt{1+t^2}} \right)^{1/2}$$

avec

$$t = \tan 2\theta = 2a_{pq}^{(k)} / (a_{pp}^{(k)} - a_{qq}^{(k)})$$

Si $a_{pp}^{(k)} = a_{qq}^{(k)}$, on prendra $\theta = \pi/4$.

p et q varient à chaque itération. Donc $a_{pq}^{(k+1)}$ et $a_{qp}^{(k+1)}$ qui avaient été annulés lors de la $k^{\text{ième}}$ itération ne seront plus obligatoirement nuls après la $(k+1)^{\text{ième}}$. Cependant, en itérant le procédé pour tous les couples (p, q) extra-diagonaux $(p \neq q)$, on arrive peu à peu à annuler tous les termes en dehors de la diagonale principale et donc à converger vers une matrice diagonale dont les termes sont les valeurs propres de A. Posons :

$$N_k = \sum_{\substack{i,j=1 \\ i \neq j}}^{n} (a_{ij}^{(k)})^2$$

On a le théorème **22**.

Théorème 22 – Pour tout k :

$$0 \leqslant N_{k+1} \leqslant N_k$$

Cela ne veut pas dire que (N_k) admet zéro pour limite et donc que la méthode converge. Cela va dépendre du choix de p et q à chaque itération. On a le théorème **23**.

Théorème 23 – Si, à chaque itération, on choisit p et q tels que :

$$\left| a_{pq}^{(k)} \right| = \max_{i \neq j} \left| a_{ij}^{(k)} \right|$$

alors la méthode de Jacobi converge.

En pratique, on arrête les itérations dès que N_k est petit et les éléments de la diagonale de A_k sont des valeurs approchées des valeurs propres de A. Cependant, comme l'a montré F. Chatelin dans sa thèse, ce test d'arrêt est loin d'être optimal : en effet, on peut très bien avoir sur la diagonale de A_k de très bonnes approximations des valeurs propres de A sans pour autant que N_k soit très petit. On peut réduire considérablement le nombre des itérations en effectuant un test d'arrêt basé sur le théorème **24**.

Théorème 24 – Posons, dans la méthode de Jacobi, $A_k = D_k + E_k$ où D_k est la matrice diagonale formée par la diagonale de A_k. Supposons que toutes les valeurs propres de A soient simples. Posons :

$$d_i^{(k)} = \min_{j \neq i} \left| a_{ii}^{(k)} - a_{jj}^{(k)} \right|$$
$$s_i^{(k)} = 2 \| E_k \| / d_i^{(k)}$$
$$t_i^{(k)} = 2 \| E_k e_i \| / d_i^{(k)} \qquad \text{pour} \quad i = 1, \ldots, n$$

où le vecteur e_i a toutes ses composantes nulles sauf la $i^{\text{ième}}$ qui vaut 1. Alors pour chaque i tel que $s_i^{(k)} < 1$, on a :

$$\left\| \lambda_i - a_{ii}^{(k)} \right\| \leqslant f(r_i^{(k)}) \| E_k e_i \|$$

avec $\quad f(r) = \left[\sqrt{1 + r^2} - 1 \right] / r \quad$ et $\quad r_i^{(k)} = \dfrac{2t_i^{(k)}}{2 - s_i^{(k)} + (t_i^{(k)})^2}$

3.5 Méthode de Rayleigh-Ritz

La dimension de certains problèmes de valeurs propres peut être extrêmement élevée. Il est alors hors de question d'utiliser les algorithmes que nous venons de présenter, d'autant plus que, bien souvent, on ne désire calculer que certains éléments propres.

Nous allons maintenant étudier une méthode qui permet d'approcher simultanément plusieurs valeurs propres ainsi que les vecteurs propres correspondants.

Soient $\mathcal{H}_k$ et $\mathcal{L}_k$ deux sous-espaces de dimension k de $\mathbb{C}^n$. On va chercher $(\tilde{\lambda}, \tilde{u}) \in \mathbb{C} \times \mathcal{H}_k$, avec $\tilde{u} \neq 0$, tels que :

$$A\tilde{u} - \tilde{\lambda}\tilde{u} \perp \mathcal{L}_k$$

Soit $u_1, ..., u_k$ une base de $\mathcal{H}_k$ et U_k la matrice dont les colonnes sont $u_1, ..., u_k$. Soit $v_1, ..., v_k$ une base de $\mathcal{L}_k$ et V_k la matrice dont les colonnes sont $v_1, ..., v_k$. La condition précédente s'écrit, en supposant inversible la matrice $V_k^* U_k$:

$$(V_k^* U_k)^{-1} V_k^* A U_k y = \tilde{\lambda} y$$

Par conséquent, $\tilde{\lambda}$ est valeur propre de $H_k = (V_k^* U_k)^{-1} V_k^* A U_k$ et y est le vecteur propre correspondant.

La **méthode de Rayleigh-Ritz** pour approcher k éléments propres de A consiste donc à calculer les éléments propres de H_k. On voit que l'on a intérêt à prendre des bases $u_1, ..., u_k$ et $v_1, ..., v_k$ biorthogonales, c'est-à-dire telles que $V_k^* U_k = I$. Dans ce cas, les étapes de la méthode sont les suivantes :

— **1.** choisir $k \leqslant m \leqslant n$;

— **2.** construire une base orthonormale $u_1, ..., u_m$ de $\mathcal{H}_m$ et poser $U_m = [u_1, ..., u_m]$;

— **3.** calculer $H_m = U_m^* A U_m$;

— **4.** calculer les valeurs propres de H_m et en sélectionner k (par exemple, celles de plus grands modules), $\tilde{\lambda}_1, ..., \tilde{\lambda}_k$;

— **5.** calculer les vecteurs propres y_i de H_m correspondants aux valeurs propres $\tilde{\lambda}_i$ pour $i = 1, ..., k$ et les approximations correspondantes $\tilde{u}_i = U_m y_i$ des vecteurs propres de A.

En pratique, la base orthonormale de $\mathcal{H}_m$ est construite à partir de suites de Krylov engendrées par un vecteur u, c'est-à-dire que $u_i = A^{i-1}u$, $i = 1, 2, ...$ On peut laisser m constant tout au long des itérations, ce qui conduit à la méthode des sous-espaces itérés, ou l'augmenter d'une unité à chaque itération. Prenons, dans ce second cas, $m = k$. Le sous-espace $\mathcal{H}_k$ est engendré par les vecteurs u, Au, ..., $A^k u$ et tout vecteur $v \in \mathcal{H}_k$ s'écrit $v = p_{k-1}(A)u$ où p_{k-1} est un polynôme de degré $k-1$ au plus. De même, $u_i = p_{i-1}(A)u$. Ces polynômes sont des polynômes orthogonaux formels. On a $\mathcal{H}_k \subset \mathcal{H}_{k+1}$. Supposons avoir déjà construit des vecteurs $u_1, u_2, ..., u_k$ orthonormaux. On construit un nouveau vecteur u_{k+1} par :

$$w_k = Au_k - U_k h_k$$

$$u_{k+1} = w_k / \|w_k\|_2$$

avec h_k calculé de sorte que $U_k^* h_k = 0$.

On a donc $U_k^* w_k = 0 = U_k^* A u_k - U_k^* U_k h_k = U_k^* A u_k - h_k$.

D'après cela, on voit que (les normes sont euclidiennes) :

$$\|h_k\| = \min_h \|Au_k - U_k h\| = \min_p \|p(A)u_1\|$$

où la seconde minimisation s'effectue dans l'ensemble des polynômes de degré k dont le coefficient du terme de plus haut degré est le même que le coefficient dominant de p_{k-1}. Le seul cas où cette procédure est en défaut se produit lorsque $h_k = 0$, ce qui implique $A \cdot \mathcal{H}_k = \mathcal{H}_k$ (le sous-espace est invariant).

La construction que nous venons de donner peut s'écrire :

$$AU_k = U_k H_k + w_k e_k^*$$

avec e_k $k^{\text{ième}}$ vecteur de la base canonique de $\mathbb{C}^n$,

$H_k = [h_1, ..., h_k]$ matrice de Hessenberg supérieure.

Nous retrouvons donc la méthode d'Arnoldi dont il a été question au paragraphe 2.2.2. Lorsque A est symétrique, la matrice H_k est tridiagonale, ce qui conduit à la méthode de biorthogonalisation de Lanczos du paragraphe 3.2.

Il est possible d'exprimer différemment les idées précédentes. Posons $P_k = U_k (V_k^* U_k)^{-1} V_k^*$. Il est facile de voir que $P_k^2 = P_k$, ce qui montre que P_k représente une projection. C'est la projection oblique sur $\mathcal{H}_k$ parallèlement à $\mathcal{L}_k^\perp$, c'est-à-dire orthogonalement à $\mathcal{L}_k$. Puisque $\tilde{u} \in \mathcal{H}_k$, on a $P_k \tilde{u} = \tilde{u}$ et l'on obtient $P_k(A\tilde{u} - \tilde{\lambda}\tilde{u}) = 0$, soit encore :

$$P_k A P_k \tilde{u} = \tilde{\lambda}\tilde{u}$$

ce qui montre que $(\tilde{\lambda}, \tilde{u})$ est un élément propre de la matrice $A_k = P_k A P_k \in \mathbb{R}^{k \times k}$ et que c'est une approximation d'un élément propre de la matrice A. Si $U_k \equiv V_k$, alors $P_k = P_k^T$ et la projection est orthogonale. Cette procédure est la méthode des moments de Vorobyev [5].

Sur ces questions, on pourra consulter les références [1] [13] [38].

Méthodes numériques de base

par **Claude BREZINSKI**

Docteur ès sciences mathématiques

Professeur à l'université des Sciences et Technologies de Lille

Bibliographies

Références

[1] BAI (Z.), DEMMEL (J.), DONGARRA (J.), RUHE (A.) et VAN DER VORST (H.). – *Templates for the solution of algebraic eigenvalue problems : a practical guide.* SIAM, Philadelphia (2000).

[2] BARRAUD (A.) éd. – *Outils d'analyse numérique pour l'automatique.* Hermès, Paris (2002).

[3] BARRETT (R.), BERRY (M.), CHAN (T.), DEMMEL (J.), DONATO (J.), EIJKHOUT (V.), POZO (R.), ROMINE (C.) et VAN DER VORST (H.). – *Templates for the solution of linear systems : building blocks for iterative methods.* SIAM, Philadelphia (1994).

[4] BREZINSKI (C.). – *Padé-type approximation and general orthogonal polynomials.* Basel, Birkhäuser (1980).

[5] BREZINSKI (C.). – *Projection methods for systems of equations.* North-Holland, Amsterdam (1997).

[6] BREZINSKI (C.) et GAUTSCHI (W.). – *A compendium of numerical methods.* Birkhäuser, Basel (2007).

[7] BREZINSKI (C.) et REDIVO-ZAGLIA (M.). – *Extrapolation methods. Theory and practice.* North-Holland, Amsterdam (1991).

[8] BREZINSKI (C.), REDIVO-ZAGLIA (M.) et SADOK (H.). – *A review of formal orthogonality in Lanczos-based methods.* J. Comput. Appl. Math., *140*, p. 81-98 (2002).

[9] BREZINSKI (C.) et VAN ISEGHEM (J.). – *Padé approximations,* dans *Handbook of Numerical Analysis.* vol. III, P.G. Ciarlet and J.L. Lions éds., p. 47-222, North-Holland, Amsterdam (1994).

[10] BROYDEN (C.G.) et VESPUCCI (M.T.). – *Krylov solvers for linear algebraic systems.* Elsevier, Amsterdam (2004).

[11] BUHMANN (M.D.). – *Radial basis functions : theory and implementations.* Cambridge University Press, Cambridge (2003).

[12] BUTCHER (J.C.). – *The numerical analysis of ordinary differential equations.* Wiley, Chichester (1987).

[13] CHATELIN (F.). – *Valeurs propres de matrices.* Masson, Paris (1988).

[14] CIARLET (P.G.). – *Introduction à l'analyse numérique matricielle et à l'optimisation.* Masson, Paris (1982).

[15] COHEN (A.). – *Numerical analysis of wavelet methods.* North-Holland, Amsterdam (2003).

[16] CROUZEIX (M.) et MIGNOT (A.L.). – *Analyse numérique des équations différentielles.* 2e éd., Masson, Paris (1989).

[17] DAUBECHIES (I.). – *Ten lectures on wavelets.* SIAM, Philadelphia (1992).

[18] DAUMAS (M.) et MULLER (J.M.) éds. – *Qualité des calculs sur ordinateur.* Masson, Paris (1997).

[19] DAVIS (P.J.). – *Interpolation and approximation.* Dover, New York (1975).

[20] DENNIS (J.E.), JR et SCHNABEL (R.B.). – *Numerical methods for unconstrained optimization and nonlinear equations.* SIAM, Philadelphia (1996).

[21] DURAND (É). – *Solutions numériques des équations algébriques.* 2 vols., Masson, Paris (1972).

[22] ENGELN-MÜLLGES (G.) et UHLIG (F.). – *Numerical algorithms with FORTRAN.* Springer, Berlin (1996).

[23] FARIN (G.). – *Courbes et surfaces pour la GCAO.* Masson, Paris (1992).

[24] FIOROT (J.-C.) et JEANNIN (P.). – *Courbes splines rationnelles. Applications à la CAO.* Masson, Paris (1992).

[25] GAUTSCHI (W.). – *Numerical analysis. An introduction.* Birkhäuser, Boston (1997).

[26] GAUTSCHI (W.). – *Orthogonal polynomials. Computation and approximation.* Oxford University Press, Oxford (2004).

[27] GOLUB (G.H.) et VAN LOAN (C.F.). – *Matrix computations.* 2e éd., The Johns Hopkins Univ. Press, Baltimore (1989).

[28] HAIRER (E.), NØRSETT (S.P.) et WANNR (G.). – *Solving ordinary differential equations. I. Nonstiff problems.* Springer-Verlag, Berlin (1987).

[29] HAIRER (E.) et WANNER (G.). – *Solving ordinary differential equations. II. Stiff and differential-algebraic problems.* Springer-Verlag, Berlin (1991).

[30] LASCAUX (P.) et THÉODOR (R.). – *Analyse numérique matricielle appliquée à l'art de l'ingénieur.* 2 vols., Masson, Paris (1986).

[31] LAURENT (P.J.). – *Approximation et optimisation.* Hermann, Paris (1972).

[32] LORENTZEN (L.) et WAADELAND (H.). – *Continued fractions with applications.* North-Holland, Amsterdam (1992).

[33] MEURANT (G.). – *Computer solution of large linear systems.* North-Holland, Amsterdam (1999).

[34] MÜHLBACH (G.). – *The general Neville-Aitken algorithm and some applications.* Numer. Math., *31*, p. 97-110 (1978).

[35] ORTEGA (J.M.) et RHEINBOLDT (W.C.). – *Iterative solution of nonlinear equations in several variables.* Academic Press, New York (1970).

[36] PICHAT (M.). – *Correction d'une somme en arithmétique à virgule flottante.* Numer. Math., *19*, p. 400-406 (1972).

[37] QUARTERONI (A.), SACCO (R.) et SALERI (F.). – *Méthodes numériques pour le calcul scientifique.* Springer, Paris (2000).

[38] SAAD (Y.). – *Numerical methods for large eigenvalue problems.* Halstead Press, New York (1992).

[39] SAAD (Y.). – *Iterative methods for sparse linear systems.* PWS, Boston (1996).

[40] UEBERHUBER (C.W.). – *Numerical computation. Methods, software, and analysis.* 2 vols., Springer, Berlin (1997).

[41] VARGA (R.S.). – *Matrix iterative analysis.* 2e éd., Springer, Berlin (2000).

[42] VAN DER VORST (H.A.). – *Iterative Krylov methods for large linear systems.* Cambridge University Press, Cambridge (2003).

[43] VIGNES (J.). – *Discrete stochastic arithmetic for validating results of numerical software.* Numer. Algorithms, *37*, p. 377-390 (2004).

[44] WENDLAND (H.). – *Scattered data approximation.* Cambridge University Press, Cambridge (2005).

Dans les Techniques de l'Ingénieur

Traité Sciences fondamentales

MARTINEZ (J.), GAJAN (P.) et STRZLECKI (A.). – *Analyse temps-fréquence. Ondelettes-théorie.* AF 4 510 (2002).

MARTINEZ (J.), GAJAN (P.) et STRZLECKI (A.). – *Analyse temps-fréquence. Ondelettes. Applications.* AF 4 511 (2002).

COHEN (A.). – *Les bases d'ondelettes.* AF 210 (2002).

QUEFFÉLEC (H.). – *Séries de Fourier.* AF 141 (1999).

Publications concernant l'analyse numérique

(liste par ordre alphabétique et non limitative)

Advances in Computational Mathematics
Applied Numerical Mathematics
BIT Numerical Mathematics
Computer Aided Geometric Design
Constructive Approximation
Journal of Approximation Theory
Journal of Computational and Applied Mathematics
Mathematics of Computation
Numerische Mathematik
Numerical Algorithms
SIAM Journal on Matrix Analysis and Applications
SIAM Journal on Numerical Analysis
SIAM Journal on Scientific Computing

Il existe également des journaux plus spécialisés dans certains domaines de l'analyse numérique.

Logiciels de calcul

(liste non exhaustive)

Le plus connu est bien évidemment **MATLAB**
http://www.mathworks.fr/

Un guide concernant les logiciels de calcul disponibles peut être consulté sur :
http://gams.nist.gov/

On pourra également se procurer des logiciels à l'adresse :
http://www.netlib.org