Analyse numérique I.

Ceci est un des *pdf* de préparation à la L3. Celui-ci traite de l'unité X31M030 – Analyse Numérique. En particulier, il traite d'intégration numérique, de résolution de systèmes linéaires, et de recherches de valeurs et vecteurs propres.

Ce qui distingue cet ensemble de fiches des autres, c'est qu'il n'est basé sur les polycopiés de l'année précédente, mais est reconstruit en suivant le plan donné dans le programme, à partir de multiples sources traitant sur un ou plusieurs des sujets abordés. Une liste de quelques sources est donnée plus bas.

Sommaire.

Sources principales :

- *Mathématiques Tout-en-un pour la Licence* (volumes 2 et 3), sous la direction de Jean-Pierre Ramis et André Warusfel, de X. Buff, E. Halberstadt, F. Moulin, M. Ramis, J. Sauloy, J. Garnier.
- Page Web *Analyse Numérique*, bibmath.net (https://www.bibmath.net/dico/index.php?action=rub&quoi=307).
- Page Web *Cours de Calcul Scientifique*, Marc Buffat, Université de Lyon (https://perso.univ-lyon1.fr/marc.buffat/COURS/CalculScientique_HTML/courshtml1.html)

I. Intégration numérique.

La note.

On identifie les polynômes (objets formels) avec les fonctions réelles correspondantes. Ainsi, l'objet P(T) = 3T – 7 est identifié avec la fonction f:ℝ→ℝ donnée par f(x) = 3x-7.

Les polynômes orthogonaux.

Fixons nous I un intervalle (quelconque), et w(t) un poids dessus (c'est-à-dire une fonction f:I→ℝ⁺ continue et non-identiquement nulle sur I), tels que

$$\mu_n := \int_I t^n w(t)\,\mathrm{d}t$$

Soit fini pour tout n. Alors, on peut fixer un produit scalaire par

$$(p|q)_w := \int_I p(t)q(t)w(t)\mathrm{d}t$$

On peut même l'expliciter à partir des moments : si $P(T) = a_0+a_1T+a_2T^2+\ldots$, $Q(T) = b_0+b_1T+b_2T^2+\ldots$, alors

$$(P|Q)_w = \sum_{n=0}^{\infty} \mu_n \cdot \sum_{m=0}^{n} a_m b_{n-m}.$$

Une suite de polynômes (réels) $(p_n)_{n\in\mathbb{N}}$ avec p_n de degré n pour tout n, et $(p_n|p_m)_w = 0$ pour tout $n\neq m$, est dite une suite de polynômes orthogonaux.

La normalisation.

Pour une suite de polynômes orthogonaux, on peut en obtenir une autre très similaire en multipliant chaque polynôme par un coefficient non-nul (après tout, si $(p|q)_w = 0$, alors $(\alpha p|\beta q)_w = 0$ aussi, peu importe les α et β choisis). Donc vient le problème de choisir la bonne normalisation. On peut par exemple avoir : $(p_n|p_n)_w = 1$ pour tout n ; ou p_n unitaire (coefficient principal = 1) pour tout n ; ou le plus petit coefficient principal tel que tous les coefficients soient entiers (si possible), etc.

Le procédé de Gram-Schmidt.

Utiliser le procédé de Gram-Schmidt dans l'espace des polynômes ℝ[T] muni du produit scalaire mentionné plus haut (à partir de la suite (1, T, T^2, T^3, …)) permet de construire une suite de polynômes orthogonaux.

La méthode de Newton-Cotes.

Pour les méthodes de Newton-Cotes, la méthode de quadrature (de rang n) consiste en : on subdivise le segment étudié en n sous-segments de mêmes tailles, puis on construit les polynômes d'interpolation de Lagrange dessus, ce qui nous permet d'approximer la fonction par un polynôme de degré n. Voici le détail sur un segment [a,b] : définissons $x_i = a+i\Delta$ où $\Delta = (b-a)/n$. Alors on cherche à avoir

$$\int_a^b f(x)dx \approx \sum_{i=0}^{n} w_i f(x_i)$$

Dans ce cas-là, on est obligé de prendre

$$w_i = \int_a^b \left(\prod_{j=0, j\neq i}^{n} \frac{t-x_j}{x_i-x_j} \right) \mathrm{d}t$$

En effet, pour que la méthode soit exacte sur tous les polynômes de degré n, il faut (et il suffit) qu'elle le soit sur les polynômes d'interpolation de Lagrange associés à x_0, … x_n; d'où l'égalité. Cas particuliers : n = 1 donne la méthode des trapèzes, n = 2 est la méthode de Simpson (et les quelques suivantes sont parfois dites "méthode de Simpson 3/8", "méthode de Boole(-Villarceau)").

La méthode de Gauss-Legendre.

Plaçons-nous sur [-1,1], et essayons d'obtenir $\int_{-1}^{1} f(x)\,dx$ comme $\Sigma_{i=1}^{n} w_i f(x_i)$, où on a choisi convenablement les w_i et x_i de façon à obtenir le degré maximal. Il se trouve que ! Pour n entier, cette méthode est de degré 2n-1 si on prend comme x_i les nœuds du polynôme de Legendre P_n (i.e. les racines, les zéros), et comme w_i, $w_i := 2/(1-x_i^2)[P_n'(x_i)]^2$. On l'appelle méthode de Gauss-Legendre. Voici quelques cas particuliers :

- pour n = 2, on prend $x_0 = -1/\sqrt{3}$, $x_1 = 1/\sqrt{3}$, et $w_0 = w_1 = 1$;
- pour n = 3, on a $x_0 = -\sqrt{3/5}$, $x_1 = 0$, $x_2 = \sqrt{3/5}$, et $w_0 = w_2 = 5/9$, $w_1 = 8/9$.

Bien sûr, par changement affine, on peut utiliser cette méthode sur n'importe quel intervalle, et même en faire une méthode composite (en divisant l'intervalle donné en sous-intervalles sur lesquels on applique la méthode générale).

Les exemples.

	Intervalle	Fonction poids	Normalisation	Premiers polynômes
Polynômes de Legendre P_n	[-1 ; 1]	$w(t) = 1$	$P_n(1) = 1$	$P_0(T) = 1$, $P_1(T) = T$, $P_2(T) = (3T^2-1)/2$, $P_3(T) = (5T^3-3T)/2$, …
Polynômes de Laguerre L_n	[0 ; +∞[	$w(t) = e^{-t}$	Premier coefficient de L_n est $(-1)^n$	$L_0(T) = 1$, $L_1(T) = -T+1$, $L_2(T) = T^2-4T+2$, $L_3(T) = -T^3+9T^2-18T+6$, …
Polynômes de Hermite H_n	]–∞ ; +∞[	$w(t) = e^{-t^2}$	Premier coefficient de H_n est 2^n	$H_0(T) = 1$, $H_1(T) = 2T$, $H_2(T) = 4T^2-2$, $H_3(T) = 8T^3-12T$, …
Polynômes de Tchebychev (du premier type) T_n	]-1 ; 1[	$w(t) = 1/\sqrt{(1-t^2)}$	Premier coefficient de T_n est 2^{n-1}	$T_0(T) = 1$, $T_1(T) = T$, $T_2(T) = 2T^2-1$, $T_3(T) = 4T^3-8T$, …

Quelques autres exemples : les polynômes de Tchebychev du second type U_n, où on utilise $w(t) = \sqrt{(1-t^2)}$ cette fois ; les polynômes de Jacobi $P^{(\alpha,\beta)}_n$, sur I = [-1 ; 1], avec la fonction poids $w(t) = (1-t)^\alpha(1+t)^\beta$. On en obtient plusieurs autres comme cas particulier ; α=β=0 donne les polynômes de Legendre, α=β=-½ donne les polynômes de Tchebychev (de premier type), etc.

I. Intégration numérique.

■ Les méthodes de Gauss.

Plus généralement, avec I fixé et w(t) un poids dessus satisfaisant les mêmes questions que plus haut, une méthode de quadrature de Gauss donne une approximation de $\int_I f(t)w(t)\,dt$ sous la forme d'une somme

$$\sum_{k=1}^{n} \lambda_{k,n} f(t_{k,n})$$

On remarque que la méthode de Gauss-Legendre est le cas particulier I = [-1;1], et w(t) = 1 sur I. Il se trouve qu'on doit utiliser la suite des polynômes orthogonaux (P_n) associés au poids w. En particulier, les $t_{k,n}$ sont les zéros du n-ième polynôme orthogonal P_n. Et les $\lambda_{k,n}$ sont obtenus en s'efforçant à ce que la méthode de quadrature soit correcte pour tous les polynômes d'interpolation de Lagrange $(L_{k,n})$ associés aux points $(t_{1,n}, \dots t_{n,n})$. Cela nous donne donc cette formule pour les déterminer :

$$\lambda_{k,n} = \int_I L_{k,n}(t)w(t)\mathrm{d}t$$

Ces $\lambda_{k,n}$ sont parfois appelés "nombres de Christoffel". Et la méthode de quadrature obtenue est de degré (2n+1). Par exemple, si on veut approximer l'intégrale $\int_I f(x)/\sqrt{(1-x^2)}\,dx$, il faudrait utiliser $x_i = \cos(\pi\cdot(2i-1)/2n)$ et $w_i = \pi/n$ (quadrature de Gauss-Tchebychev, vu qu'on utilise les polynômes de Tchebychev de premier type).

■ Le noyau de Peano.

Soit J(f) une méthode de quadrature (élémentaire) d'ordre N sur un intervalle [a,b]. En notant I(f) la valeur exacte de cette intégrale, on peut poser E(f) := I(f)-J(f) l'erreur de la quadrature. Pour f une fonction réelle, notons f_+ la fonction définie par $f_+(x) = \max(f(x),0)$. Dans le cas où cette méthode est d'ordre N, on appelle noyau de Peano associé à cette méthode la fonction qui associe à $t\in[a,b]$,

$$K_N(t) := E(x \mapsto (x-t)_+^N)$$

On a alors que, pour toute fonction f de classe C^{n+1} sur [a,b] :

$$E(f) = \int_a^b \frac{1}{N!} K_N(t) f^{(N+1)}(t)dt$$

Et si K_N est de signe constant, cela nous permet de donner l'encadrement :

$$|E(f)| \leq \|f^{(N+1)}\|_\infty \int_a^b \frac{1}{N!}|K_N(t)|\mathrm{d}t$$

C'est cet encadrement qui permet de donner la meilleure constante C lorsqu'on veut écrire une inégalité d'erreur du style $|E(f)| \leq C\cdot\|f^{(N+1)}\|_\infty$.

■ Les méthodes composées.

Étant donnée une méthode simple (élémentaire) de quadrature sur un intervalle, on peut la rendre plus précise en découpant l'intervalle étudié en n sous-intervalles de même longueur, en appliquant la méthode simple à chaque sous-intervalle, et en faisant la somme de toutes les valeurs. Par exemple, si l'erreur de la méthode simple des trapèzes est majorée par $\|f''\|_\infty\cdot(b-a)^3/12$, l'erreur de la méthode composite avec n intervalles est majorée cette fois par $\|f''\|_\infty\cdot(b-a)^3/12n^2$, qu'on peut rendre aussi petit qu'on veut.

■ L'ordre.

Rappelons brièvement que l'ordre (d'exactitude) d'une méthode de quadrature est le plus grand entier m pour lequel la méthode est exacte pour tout polynôme de degré inférieur ou égal à m.

■ Les suppléments.

On donne ici quelques informations supplémentaires sur les suite de polynômes orthogonaux. Entre autres, sur leurs zéros, sur leur fonctionnement en tant que base de l'espace vectoriel des polynômes, et les formules de récurrences qui peuvent les lier.

■ Les zéros.

Rapidement, quelques propriétés des zéros des polynômes orthogonaux. Si $(P_n)_n$ est une suite de polynômes orthogonaux (associés au poids w sur]α,β[), alors P_n a n zéros, tous simples, réels, et dans l'intervalle ouvert]α,β[. De plus, P_n et P_{n+1} n'ont aucun zéro en commun, et leurs zéros sont alternés dans]α,β[.

■ Les formules de récurrence.

Soit donc $(P_n)_n$ une suite de polynômes orthogonaux associés au poids w sur un intervalle I (de bornes α et β). Alors :

i) la suite $(P_n)_n$ satisfait une relation de récurrence du type :

$$P_n - (A_n t+B_n)P_{n-1} + C_n P_{n-2} = 0$$

En particulier, si γ_n dénote le coefficient principal de P_n et $\|Q\|^2$ est défini comme $(Q|Q)_w$, alors on a :

$$A_n = \gamma_n/\gamma_{n-1},\ B_n = -\gamma_n/\gamma_{n-1}\ (tP_{n-1}|P_{n-1})_w/\|P_{n-1}\|^2$$
$$\text{et } C_n = \gamma_n\gamma_{n-2}/\gamma_{n-1}^2\ \|P_{n-1}\|^2/\|P_{n-2}\|^2$$

ii) si l'intervalle est symétrie par rapport à l'origine et si le polynôme $P_{n-1}^2 w$ est une fonction paire, alors $B_n = 0$.

iii) lorsque les P_n sont unitaires ($\gamma_n = 1$), $A_n = 1$, et les autres coefficients se simplifient aussi.

Le théorème de Favard donne une condition pour savoir si une suite de polynômes définie par une récurrence donne une suite de polynômes orthogonaux.

■ L'approximation.

Soit donc w un poids (positif) sur un segment [α,β], et $(q_n)_n$ la suite de polynômes orthonormés associées (c'est-à-dire, orthogonaux où on a choisi la normalisation $\|q_k\|_w = 1$). Alors, il se trouve que pour toute fonction f de $C([\alpha,\beta])$ avec $\|f\|_w < +\infty$, alors la suite des polynômes

$$\sum_{k=0}^{n} (f|q_k)_w q_k$$

converge vers f au sens de la norme $\|f\|_w$, et on a même l'égalité de Parseval (comme pour les séries de Fourier) :

$$\|f\|_w^2 = \sum_{k=0}^{+\infty} (f|q_k)_w^2$$

II. Systèmes linéaires.

Le but.

Le but est de résoudre (efficacement) le système AX = B, où $A \in GL_n(R)$ et $B \in M_{n,1}(R)$. Juste inverser A de manière classique (par exemple avec la comatrice), ou utiliser les formules de Cramer, demande, en ordre de grandeur, n! opérations. Alors que la méthode du pivot de Gauss n'en demande généralement, que n^3. Voici d'autres algorithmes ; d'abord les classiques (exacts), et puis des algorithmes itératifs.

La décomposition LU.

Rappelons un algorithme classique (mais si !) ; celui de la décomposition LU. Une matrice $A \in M_n(R)$ admet une décomposition LU si il existe L triangulaire inférieure unipotente (que des 1 sur la diagonale), et U triangulaire supérieure avec A = LU. Une telle décomposition est intéressante car le système LUX = B se résout par descente-remontée ; en effet, en posant Y:=UX, cela revient à résoudre LY = B (pour Y), puis UX = Y (pour X), or les matrices sont triangulaires, donc on le fait comme quand on finit un pivot de Gauss . Dans le cas où A est inversible, la décomposition est unique, et n'existe que si les mineurs principaux sont non-nuls. Pour voir l'algorithme, aller en annexe. Dans le cas général, si il n'y a pas de décomposition LU, il y a toujours une décomposition PLU, avec P matrice de permutation ; et sinon, la méthode du pivot de Gauss.

La décomposition QR.

Similairement, une écriture de A sous la forme QR avec Q orthogonal, R triangulaire supérieure, permet de plus facilement résoudre le système QRX = B (autrement dit, $RX = Q^TB$). Et ici, un des algorithmes pour la faire apparaître est le bien connu procédé de Gram-Schmidt ! En effet, (pour A inversible), si on considère A comme une suite de vecteurs colonnes $A = (A_1 \mid \dots \mid A_n)$, après procédé de Gram-Schmidt, on obtient $\{Q_1, \dots Q_n\}$ une nouvelle base de R^n ; et on peut décrire A_1 comme $r_{1,1} \cdot Q_1$ pour un certain réel $r_{1,1}$, A_2 comme $r_{1,2} \cdot Q_1 + r_{2,2} \cdot Q_2$ pour certains réels $r_{1,2}$ et $r_{2,2}$, etc. D'où, A = QR, en posant $Q = (Q_1 \mid \dots \mid Q_n)$ et $R = (r_{i,j})_{i,j}$ avec ces coefficients (et $r_{i,j} = 0$ quand $i>j$). Et pour calculer les $r_{i,j}$ explicitement, on utilise juste le produit scalaire (canonique) : $r_{i,j} = \langle Q_i | A_j \rangle$.

La factorisation de Cholesky.

On rappelle ici rapidement la factorisation de Cholesky, qui consiste à écrire une matrice symétrique définie positive A comme LL^T avec L matrice triangulaire inférieure réelle. (Une telle décomposition existe toujours!) Si on impose que la diagonale soit strictement positive, alors la factorisation est unique !

Le rayon spectral.

On peut définir le rayon spectral par :

$$\rho(A) = \max \{|\mu| : \mu \text{ valeur propre de } A\}$$

Le théorème de Gelfand nous indique que $\rho(A) = \lim_n \|A^n\|^{1/n}$ et pour toute norme matricielle, $\rho(A) \leq \|A\|$.

Les méthodes itératives.

L'idée est : on écrit A = M–N avec M inversible elle-aussi, et alors AX = B devient MX–NX = B, MX = NX+B, X = PX+C où $P = M^{-1}N$, $C = M^{-1}B$. Alors, on reconnaît qu'une méthode du point fixe pourrait fonctionner : on pose X_0 (par exemple $X_0 = 0$), puis $X_{n+1} = PX_n + C$. Il faut bien sûr bien choisir M, N de manière à ce que ça fonctionne (i.e. converge), et rapidement ! En ce sens que M doit être inversible sans trop de calculs. En particulier, notons D, E et F les matrices diagonales, triangulaire inférieure stricte (nulle sur la diagonale) et triangulaire supérieure stricte telles que M = D–E–F.

La convergence.

Pour qu'une telle méthode converge (c'est-à-dire que la suite définie par $X^{k+1} := PX^k + C$ converge quelque soit le C choisi), il faut et suffit qu'une des conditions suivantes (équivalentes) soit vérifiée :

i) $\lim_k P^k = 0$;

ii) le rayon spectral de P satisfasse $\rho(P) < 1$;

iii) il existe une norme matricielle (subordonnée) telle que $\|P\|<1$.

Remarquons qu'on a $X = (I_n - P)^{-1}C = C+PC+P^2C+P^3C+ \dots$ quand le membre à droite converge ; ce qui explique pourquoi on souhaiterait, entre autres, avoir $P^k \to 0$.

La méthode de Jacobi.

Dans celle-ci, on choisit M:=D, facilement inversible, et donc N:=E+F. Autrement dit, on se retrouve à résoudre $DX_{k+1} = (E+F)X_k+B$, ce qui se fait relativement facilement. Mais problème ! Comme chaque coordonnée de X_{k+1} dépend de (presque) toutes celles de X_k, on ne peut pas utiliser une unique variable X qu'on réécrit coordonnée par coordonnée; on doit (dans une implémentation simple en tout cas) utiliser une variable temporaire Y, puis faire chaque étape en deux temps :

$$Y := D^{-1}((E+F)X+B) \text{ ligne par ligne,}$$
$$\text{puis } X := Y.$$

La méthode de Gauss-Seidel.

On choisit cette fois M:=D-E, relativement facile à inverser (par remontée), et ainsi N:=F. Le truc intéressant, c'est qu'on n'a plus besoin de variable temporaire Y ! En effet, cela revient à résoudre $(D-E)X_{k+1} = FX_k+B$, c'est-à-dire $DX_{k+1} = EX_{k+1}+FX_k+B$; et comme E est triangulaire, la coordonnée i peut être calculée avec les nouvelles coordonnées avant i, et les anciennes après. On remarquera que ça simplifie un peu l'algorithme (voir annexe).

II. Systèmes linéaires.

La relaxation.

Dans le cas du processus de Gauss-Seidel, on peut écrire chaque étape de correction de la i-ème coordonnée par $x_i := x_i - r_i$, où r_i est un terme de correction défini par :

$$r_i := a_{ii}^{-1}(a_{i,1}x_1+\ldots+a_{i,n}x_n - b_i)$$

Les méthodes de relaxation remplacent l'instruction xi := xi-ri par $x_i := x_i - \omega r_i$, où ω est un réel bien choisi. On montre qu'il est nécessaire que $\omega\in]0,2[$ pour que la méthode de relaxation converge. Mais il est difficile, en général, de donner une condition pour que ce soit suffisant; par exemple, si A est symétrique définie positive.

Les matrices tridiagonales.

Une matrice est dite tridiagonale si n'importe quel coefficient qui n'est pas sur la diagonale principale, ni sur celle juste au-dessus ou juste en-dessous est nul. Autrement dit, si $a_{i,j} = 0$ pour tous (i,j) avec $|i-j|\geq 2$. De telles matrices apparaissent souvent lorsqu'on ramène sous forme discrète des équations différentielles.

Les matrices d'itérations.

Posons, pour la suite, $J = D^{-1}(E+F)$ la matrice d'itération du processus de Jacobi, et $\mathcal{L}_\omega$ la matrice d'itération du processus de relaxation, donnée par :

$$\mathcal{L}_\omega = \left(\frac{1}{\omega}D - E\right)^{-1}\left(\frac{1-\omega}{\omega}D + F\right)$$

En particulier, $\mathcal{L}_1$ donne le processus de Gauss-Seidel.

Les résultats.

Par la suite, nous supposons A tridiagonale ; ce n'est pas une condition nécessaire, mais elle est suffisante. Le premier résultat est que $\rho(\mathcal{L}_1) = \rho(J)^2$, donc que si un processus converge, l'autre aussi ; et que le processus de Gauss-Seidel converge deux fois plus rapidement que celui de Jacobi. Le détail est que :

i) μ est valeur propre de $\mathcal{L}_\omega$ ssi $(\mu+\omega-1)^2/\omega^2\mu$ est le carré d'une valeur propre de J (c.à.d., une valeur propre de J^2)

ii) $\lambda \leftrightarrow \lambda^2$ est une bijection entre les paires de valeurs propres non-nulles de J, et les valeurs propres non-nulles de $\mathcal{L}_1$.

Et si toutes les valeurs propres de J sont réelles (ce qui est le cas par exemple si A est symétrique), et $\rho(J) < 1$, alors les processus de Jacobi et de Gauss-Seidel convergent, et $\rho(\mathcal{L}_\omega)$ est minimal si on choisit $\omega = 2/[1 + \sqrt{(1-\rho(J)^2)}]$.

Les normes matricielles.

Étant donné que $M_n(R)$ forme un espace vectoriel, on peut bien sûr définir dessus des normes. En particulier, si on a $\|\cdot\|$ une norme sur R^n, on peut définir une norme sur les matrices par $\|M\| = \sup\{\|MX\|, \|X\| = 1\}$ (où, autrement dit, $\sup_{X\neq 0} \|MX\|/\|X\|$). On appelle ça une norme subordonnée.

Le conditionnement.

Pour $\|\cdot\|$ une norme subordonnée, on peut poser cond(A) le conditionnement d'une matrice A comme $\|A\|\times\|A^{-1}\|$. Bien sûr, ce nombre dépend de la norme choisie ; on note cond_2 et cond_∞ les conditionnements obtenus à partir des normes subordonnées à la norme euclidienne et la norme infinie de R^n.

La majoration de l'erreur.

Revenons au système AX = B. Un des problèmes d'un ordinateur est qu'il doit arrondir les nombres réels (il n'y a pas de mémoire infinie) ; et donc, l'ordinateur travaillera plutôt sur la version arrondie (A+δA)X = (B+δB), où δA et δB sont des erreurs d'arrondis. On étudie les effets séparément, en supposant qu'ils se cumulent. D'abord l'erreur δB. Soit δX l'erreur de X obtenue en résolvant le système ; on a $\delta X = A^{-1}\delta B$. Ainsi, on obtient l'inégalité

$$\|\delta X\|/\|X\| \leq \text{cond}(A)\cdot\|\delta B\|/\|B\|$$

Et similairement, pour δA, on a

$$\|\delta X\|/\|X+\delta X\| \leq \text{cond}(A)\cdot\|\delta A\|/\|A\|$$

On voit bien que le conditionnement d'une matrice joue pour beaucoup ; une matrice mal conditionnée (cond(A) très grand) pourra faire apparaître beaucoup d'erreur. On notera que si $0 < \mu_1 \leq \ldots \leq \mu_n$ sont les valeurs propres de $B = {}^TAA$, alors $\text{cond}_2(A) = \sqrt{(\mu_n/\mu_1)}$.

III. Valeurs propres.

■ Le théorème de Gerschgorin.

Pour $A = (a_{ij}) \in M_n(\mathbb{C})$, on définit pour tout i de 1 à n, le disque

$$D_i = \left\{ z \in \mathbb{C} \middle| |z - a_{ii}| \leq \sum_{1 \leq j \leq n, j \neq i} |a_{ij}| \right\}$$

dit i-ème disque de Gerschgorin (ou encore Geršgorin ou Guerchgorine). Le théorème dit que toute valeur propre de A est contenue dans un de ces disques.

■ Les méthodes directes.

On rappelle que pour calculer une valeur propre, on peut essayer de trouver les racines le polynôme caractéristique. Mais, en pratique, cela demande deux algorithmes : un pour calculer le polynôme (l'algorithme de Faddeev-Leverrier est fait pour ça ; mais on pourrait aussi calculer le déterminant par un pivot de Gauss) ; puis un pour trouver les racines (on en a vu plusieurs en [L2]).

■ La méthode des puissances.

Soit λ_1 la valeur propre la plus grande valeur absolue. À partir de w_0 bien-choisi°, on construit une suite de vecteurs par $w_{n+1} = Aw_n/\|Aw_n\|$. On montre alors que : $\|Aw_n\|$ tend vers $|\lambda_1|$, que $(\lambda_1/|\lambda_1|)^n w_n$ tend vers v un vecteur unitaire associé à la valeur propre λ_1 de A, et en notant $[u]_j$ la j-ième coordonnée du vecteur u, que $[(Aw_n)]_j/[w_n]_j \to \lambda_1$, si $[v]_j \neq 0$.

° La condition exacte est : en écrivant l'espace $\mathbb{R}^n$ comme $E \oplus F$, avec E s.e-v. caractéristique de A associé à la valeur propre λ_1, et F le s.e-v. caractéristique de A associé aux autres valeurs propres, il faut qu'on ait $w_0 \notin F$.

■ La méthode de la puissance inverse.

Cela permet de trouver la valeur propre la plus proche d'un réel μ donné. Et en fait, on prend le même algorithme ; à ceci près qu'on remplace A, par $[A - \mu I]^{-1}$. Autrement dit, on pose

$$w_{n+1} = (A - \mu I)^{-1} w_n / \|(A - \mu I)^{-1} w_n\|$$

On doit donc résoudre le système $(A - \mu I)w'_{n+1} = w_n$, et diviser w' par sa norme !

■ Le quotient de Rayleigh.

Pour A matrice symétrique à coefficients réels et x un vecteur, le quotient de Rayleigh est le scalaire $x^T Ax/x^T x$. Si x est vecteur propre, le quotient de Rayleigh donne la valeur propre associée. On peut alors construire un nouvel algorithme, l'itération du quotient de Rayleigh, en remplaçant le μ de la méthode de la puissance inverse par le quotient de Rayleigh $w_n^T Aw_n/w_n^T w_n$; ce qui donne une approximation de plus en plus correcte de la valeur propre, et le fait converger plus rapidement ! On note qu'on ne peut l'utiliser que pour A symétrique. Note : dans les complexes, on peut utiliser A hermitienne; c'est-à-dire $A = \overline{A^T}$.

■ La diagonale dominante.

Indiquons rapidement qu'une matrice $A = (a_{i,j})_{i,j}$ à diagonale dominante (c'est-à-dire $|a_{ii}| > \Sigma_{j \neq i} |a_{j,i}|$ pour tout i) est inversible (on le déduit du théorème de Gerschgorin), et que les méthodes de Jacobi et Gauss-Seidel convergent pour cette matrice.

■ La méthode de déflation.

On souhaiterait obtenir la deuxième plus grande valeur propre (en valeur absolue) d'une matrice A. Il y a, pour ce problème, plusieurs méthodes de « déflation ». Pour une matrice symétrique, si on a λ_1 la plus grande valeur propre (en valeur absolue), et v_1 un vecteur propre associé, on peut poser $\tilde{A} = A - (\lambda_1/\|v_1\|^2) v_1 v_1^T$, qui a les mêmes valeurs propres que A, excepté λ_1 (la nouvelle valeur propre correspondante est 0). Pour une matrice générale, on cherche un vecteur w_1 propre de A^T correspondant à λ_1 ; puis on pose cette fois

$$\tilde{A} = A - \lambda_1 (v_1 w_1^T)/(w_1^T v_1)$$

■ Les autres algorithmes.

Notons par exemple l'algorithme QR, qui consiste à construire une suite de matrices A_k avec $A_0 = A$, puis à décomposer chaque A_k comme $Q_k R_k$ (par une décomposition QR), et poser $A_{k+1} = R_k Q_k$; alors sous certaines conditions, un des A_k est triangulaire ! Et quand c'est le cas, il suffit de lire les valeurs propres sur la diagonale principale (car $A_{k+1} = Q_k^T A_k Q_k$). Il y a aussi, par exemple, l'algorithme de Lanczos et l'algorithme d'Arnoldi permettent de trouver plusieurs des valeurs propres (en valeur absolue) avec m donné.

IV. Annexe - Algorithmes.

◆ 1. Exemple de méthode de quadrature composite (ici, Newton-Cotes avec n=3).

Données : a,b réels avec a<b ; f fonction de [a,b] dans $\mathbb{R}$; n entier non-nul.
Variables : S, d, x0, x1, x2, x3 réels ; i entier.

```
S:=0, d:=(b-a)/n
pour i de 1 à n faire
• x0 := a + i*d
• x1 := a + (i+1/3)*d
• x2 := a + (i+2/3)*d
• x3 := a + (i+1)*d
• S := S + d*(f(x0)+3*f(x1)+3*f(x2)+f(x3))/8
retourner S.
```

◆ 2. Décomposition LU.

Données : n entier (n≥2) ; A matrice n×n.
Variables : L, U matrices n×n ; s réel ; i,j,k entiers.

```
L:=0, U:=A
pour j de 1 à n-1 faire
• si U[j,j] = 0 alors retourner ÉCHEC.
• sinon, pour i de j+1 à n faire
  • s := - U[i,j]/U[j,j]
  • pour k de 1 à n faire U[i,k] := U[i,k]+s*U[j,k]
  • L[i,j] := -s
L:=I+L
retourner (L,U).
```

◆ 3. Remontée-descente.

Données : n entier ; L, U matrices n×n ; B vecteur colonne n×1.
Variables : X, Y vecteurs colonne n×1.

```
pour i de 1 à n faire
• Y[i] := B[i]
• pour j de 2 à i faire
  • Y[i] := Y[i] - L[i,j]*Y[j]
pour i de n à 1 faire
• X[i] := Y[i]
• pour j de 2 à i faire X[i] := X[i] – U[i,j]*X[j]
• X[i] := X[i]/U[i,i]
retourner X.
```

◆ 4. Méthode de Jacobi.

Données : n entier ; A matrice n×n ; B vecteur colonne n×1 ; ε réel.
Variables : X, R vecteurs colonne n×1 ; Reste réel ; i,j entiers.

```
X := B
répéter
• pour i de 1 à n faire
  • R[i] := B[i]
  • pour j de 1 à n faire R[i] := R[i]+A[i,j]*X[j]
• pour i de 1 à n faire X[i] := X[i]+R[i]/A[i,i]
• Reste := 0
• pour i de 1 à n faire Reste := Reste+R[i]*R[i]
jusqu'à sqrt(Reste) < ε
retourner X.
```

◆ 5. Méthode de Gauss-Seidel.

Données : n entier ; A matrice n×n ; B vecteur colonne n×1 ; ε réel.
Variables : X, R vecteurs colonne n×1 ; Reste réel ; i,j entiers.

```
X := B
répéter
• pour i de 1 à n faire
  • R[i] := B[i]
  • pour j de 1 à n faire R[i] := R[i]+A[i,j]*X[j]
  • X[i] := X[i]+R[i]/A[i,i]
• Reste := 0
• pour i de 1 à n faire Reste := Reste+R[i]*R[i]
jusqu'à sqrt(Reste) < ε
retourner X.
```

◆ 6. Méthode de la puissance itérée.

Données : A matrice n×n ; Q_0 vecteur colonne n×1 ; ε réel.
Variables : X, Q vecteurs colonne n×1 ; γ, γ_{prec} réels.

```
X := AQ₀
γ := ||X||
Q := (1/γ)X
répéter
• γ_prec := γ
• X := AQ
• γ := ||X||
• Q := (1/γ)X
jusqu'à |γ - γ_prec| < ε
retourner γ.
```

Les quelques algorithmes qui suivent sont donnés ou bien en tant que rappels, ou en tant que suppléments.

◆ 7. Procédé d'orthonormalisation de Gram-Schmidt.

Données : n entier, B liste de n vecteurs colonne n×1.
Variables : R liste de vecteurs colonne n×1 ; V vecteur colonne n×1 ; i entier.

```
R[1] := B[1]/||B[1]||
pour i de 2 à n faire
• V := B[i]
• pour j de 1 à i-1 faire V := V – ⟨V|R[j]⟩R[j]
• R[i] := V/||V||
retourner R.
```

◆ 8. Décomposition QR.

Données : n entier ; A matrice n×n.
Variables : A_l, Q_l listes de n vecteurs colonne n×1 ; Q, R matrices n×n.

```
Q := 0, R := 0
pour i de 1 à n faire
• pour j de 1 à n faire A_l[i][j] := A[j,i]
Q_l := GramSchmidt(A_l)
pour i de 1 à n faire
• pour j de 1 à n faire Q[i,j] := Q_l[j][i]
pour i de 1 à n faire
• pour j de 1 à i faire R[j,i] := ⟨Q_l[j]|A_l[i]⟩
retourner (Q,R).
```

IV. Annexe - Algorithmes.

On donne une écriture alternative des méthodes de Jacobi et de Gauss-Seidel afin de montrer que la première demande d'utiliser une variable temporaire, mais qu'on peut s'en passer pour la seconde. On n'explicite pas les conditions d'arrêt ; on pourrait par exemple utiliser la condition que $\|AX - B\| < \varepsilon$ pour un ε donné.

◆ 9. Méthode de Jacobi (version 2).

Données : n entier ; A matrice n×n ; B vecteur colonne n×1 ; condition d'arrêt.
Variables : X, Y vecteurs colonne n×1 ; i, j entiers.

```
pour i de 1 à n faire (X[i] := 0 ; Y[i] := 0)
répète
• pour i de 1 à n faire
  • Y[i] := -B[i]
  • pour j de 1 à i-1 faire Y[i] := Y[i]+A[i,j]*X[i]
  • pour j de i+1 à n faire Y[i] := Y[i]+A[i,j]*X[i]
  • Y[i] := Y[i]/A[i,i]
• pour i de 1 à n faire X[i] := Y[i]
jusqu'à <précision obtenue souhaitée>.
```

◆ 10. Méthode de Gauss-Seidel (ver. 2).

Données : n entier ; A matrice n×n ; B vecteur colonne n×1 ; condition d'arrêt.
Variables : X vecteur colonne n×1 ; i, j entiers.

```
pour i de 1 à n faire X[i] := 0
répète
• pour i de 1 à n faire
  • X[i] := -B[i]
  • pour j de 1 à i-1 faire X[i] := X[i]+A[i,j]*X[i]
  • pour j de i+1 à n faire X[i] := X[i]+A[i,j]*X[i]
  • X[i] := X[i]/A[i,i]
jusqu'à <précision obtenue souhaitée>.
```