TECHNIQUES
DE L'INGÉNIEUR

Réf. : **AF1220 V1**

Date de publication :
10 avril 2006

Méthodes numériques de base - Analyse numérique

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

par **Claude BREZINSKI**

Résumé L'analyse numérique étudie les méthodes, appelées constructives, de résolution numérique des problèmes. Cet article débute par la présentation de la problématique posée par la programmation sur ordinateur des méthodes d'analyse numérique. Sont ensuite abordées successivement l'erreur d'interpolation, l'approche de la quadrature numérique, l'intégration des équations différentielles puis la théorie de l'approximation, qui constitue à elle seule une partie fondamentale de l'analyse numérique.

Abstract

Pour toute question :
Service Relation clientèle
Techniques de l'Ingénieur
Immeuble Pleyad 1
39, boulevard Ornano
93288 Saint-Denis Cedex

Par mail :
infos.clients@teching.com
Par téléphone :
00 33 (0)1 53 35 20 20

Méthodes numériques de base

Analyse numérique

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

Il est bien connu que les méthodes utilisées en mathématiques classiques sont incapables de résoudre tous les problèmes. On ne sait pas, par exemple, donner une formule pour calculer exactement le nombre x unique qui vérifie x = exp(– x) ; on ne sait pas non plus trouver la solution analytique de certaines équations différentielles ni calculer certaines intégrales définies. On remplace alors la résolution mathématique exacte du problème par sa résolution numérique qui est, en général, approchée. ***L'analyse numérique est la branche des mathématiques qui étudie les méthodes de résolution numérique des problèmes, méthodes que l'on appelle constructives****. Par méthode constructive, on entend un ensemble de règles (on dit :* ***algorithme****) qui permet d'obtenir la solution numérique d'un problème avec une précision désirée après un nombre fini d'opérations arithmétiques.*

L'analyse numérique est une branche assez ancienne des mathématiques. Autrefois, en effet, les mathématiciens développaient les outils dont ils avaient besoin pour résoudre les problèmes posés par les sciences de la nature. C'est ainsi que Newton était avant tout un physicien, Gauss un astronome... Ils s'aperçurent rapidement que les problèmes pratiques qui se posaient étaient trop compliqués pour leurs outils et c'est ainsi que, peu à peu, s'élaborèrent les techniques de l'analyse numérique. Ces méthodes ne connurent cependant leur essor actuel qu'avec l'avénement des ordinateurs à partir des années 1945-1947.

Ce qui suit n'est pas un cours théorique d'analyse numérique. Il existe d'excellents livres pour cela. Ce n'est pas non plus un catalogue de méthodes et de recettes. Pour être utilisées correctement et pour que leurs résultats soient interprétés correctement, les méthodes d'analyse numérique nécessitent une connaissance des principes de base qui ont guidé les mathématiciens ; il est très difficile, voire impossible, d'utiliser un algorithme d'analyse numérique comme une boîte noire. Pour ces raisons, une voie médiane a été choisie et les algorithmes sont toujours replacés dans leur contexte théorique ; le lecteur soucieux des démonstrations pourra se référer à la littérature correspondante.

Les méthodes d'analyse numérique sont destinées à être programmées sur ordinateur. L'arithmétique de l'ordinateur n'a qu'une précision limitée (par la technologie), ce qui pose souvent des problèmes extrêmement importants qu'il faut pouvoir analyser et éviter. C'est pour cela que le premier paragraphe est consacré à cette question.

Il existe, naturellement, de très nombreux ouvrages d'analyse numérique. Comme références, on pourra consulter [2] [6] [22] [25] [30] [37] [40].

1. Arithmétique de l'ordinateur

1.1 Virgule flottante normalisée

Soit a un nombre réel. On peut toujours l'écrire sous la forme :

$$a = \pm\, 0,\, a_1\, a_2\, a_3 \ldots 10^q$$

avec q nombre entier relatif,

$a_1, a_2 \ldots a_i$ chiffres décimaux de a avec $a_1 \neq 0$.

On dit alors que a est écrit en virgule flottante normalisée. En général, la **mantisse** $a_1\, a_2\, a_3 \ldots$ de a possède une infinité de chiffres (on dit : **digits** ou **bits**).

Dans un ordinateur, chaque nombre est placé dans un mot. Un mot est un ensemble (fini) de petites cases qui peuvent contenir un 0 ou un 1 car les ordinateurs travaillent, pour des raisons technologiques, dans un système de numération dérivé du système binaire. Le problème qui se pose maintenant à nous est simple : comment placer un nombre ayant une infinité de digits dans un mot qui n'en comporte qu'un nombre fini ?

Il y a deux façons de procéder : la **troncature** ou l'**arrondi**. Supposons qu'un mot de l'ordinateur ne puisse contenir que t digits de la mantisse (pour simplifier le raisonnement, nous supposerons que notre ordinateur travaille lui aussi en base 10, ce qui ne changera pratiquement rien à nos conclusions). On peut tout simplement couper la mantisse de a après son $t^{\text{ième}}$ digit : c'est la troncature. On peut aussi, suivant la valeur du digit a_{t+1}, arrondir le digit a_t : si $a_{t+1} \geqslant 5$, on remplacera a_t par $a_t + 1$ et l'on tronquera, sinon on tronquera directement. La plupart des ordinateurs travaillent en arrondi. Le nombre réel a est donc représenté dans l'ordinateur par une valeur approchée, que nous noterons $fl(a)$, obtenue par troncature ou par arrondi selon la technologie de l'ordinateur. L'erreur commise en remplaçant a par $fl(a)$ s'appelle **erreur d'affectation**. Elle est donnée par le théorème **1**.

Théorème 1.

$$|a - fl(a)| \leqslant K |a|\ 10^{-t}$$

avec t nombre de digits décimaux de la mantisse des mots de l'ordinateur,

K = 10 si l'ordinateur travaille par troncature ou $K = 5$ s'il travaille par arrondi.

1.2 Opérations arithmétiques et conséquences

Les quatre opérations arithmétiques élémentaires (+, –, × et /) ne s'effectuent pas directement dans la mémoire centrale de l'ordinateur, mais dans une unité arithmétique dont les mémoires comportent plus de t digits. Une fois le calcul effectué dans cette unité arithmétique, le résultat est renvoyé dans la mémoire de l'ordinateur ; celui-ci doit donc le tronquer ou l'arrondir puisqu'il possède plus de t digits. Par conséquent, l'erreur commise sur une opération arithmétique élémentaire est régie par le théorème précédent, d'où l'on déduit le théorème **2**.

Théorème 2.

$$|a \circ b - fl(a \circ b)| \leqslant K |a \circ b| 10^{-t}$$

où $\circ$ désigne l'une des opérations +, –, × ou /.

Voyons maintenant les conséquences pratiques fondamentales qui se déduisent de ce résultat.

Soit à calculer $1 + \varepsilon$. On voit que, si l'ordinateur travaille par arrondi et si $|\varepsilon| < 5 \times 10^{-t}$ (ou si $|\varepsilon| < 10^{1-t}$ dans le cas de la troncature), alors on aura :

$$fl(1 + \varepsilon) = 1$$

La même conclusion restera valable dans le calcul de $a \pm b$ si les ordres de grandeur de a et de b sont très différents puisque :

$$a \pm b = a(1 \pm b/a)$$

On peut penser que l'erreur commise est minime, mais il n'en est rien. En effet, soit à calculer :

$$u = \frac{(y + x) - x}{y} \quad \text{et} \quad v = \frac{y + (x - x)}{y}$$

où $y \neq 0$ et où les parenthèses indiquent celle des opérations à effectuer en premier. On a $u = v = 1$. Sur l'ordinateur si l'on prend $x = 1$ et $y = \varepsilon$ tel que $fl(1 + \varepsilon) = 1$, alors on obtient $fl(v) = 1$ et $fl(u) = 0$. Par conséquent, sur ordinateur, l'addition n'est pas associative et n'est pas commutative. L'exemple précédent montre également que les erreurs peuvent être importantes et qu'une formule mathématiquement exacte peut conduire, sur ordinateur, à des résultats complètement faux.

Calculons maintenant sur ordinateur la différence $a = b - c$ lorsque b et c sont très voisins.

Exemple : si $b = 0{,}183\,256$ et $c = 0{,}183\,255$ et si $t = 6$, on obtient $a = 0{,}000\,001$, c'est-à-dire $0{,}100\,000 \times 10^{-5}$ en virgule flottante normalisée, résultat parfaitement exact. Il faut cependant bien voir que les cinq 0 qui suivent le 1 dans le résultat n'ont aucune signification et qu'ils sont complètement arbitraires puisque l'on ne connaissait que les 6 premiers chiffres significatifs de b et de c. Si l'on utilise maintenant la valeur de a dans des calculs ultérieurs, tout se passera donc comme si l'on ne disposait plus que d'un seul chiffre significatif exact, comme si l'ordinateur ne travaillait plus qu'avec $t = 1$.

On voit donc le risque énorme que l'on prend en continuant les calculs. C'est l'**erreur de cancellation** qui se produit dans la différence de deux nombres voisins ; elle est la principale source d'erreur sur ordinateur.

Exemple d'erreur de cancellation : soit à calculer les deux racines de :

$$ax^2 + bx + c = 0$$

à l'aide des formules classiques.

Pour $a = 10^{-4}$, $b = 0{,}8$ et $c = -10^{-4}$, les racines sont -8×10^3 et $-1{,}25 \times 10^{-4}$. Un ordinateur travaillant par arrondi avec $t = 6$ trouve bien la première racine mais donne $5{,}96 \times 10^{-4}$ pour la seconde. L'erreur provient de la cancellation dans le calcul de $-b + \sqrt{b^2 - 4ac}$: on dit que l'algorithme utilisé est **numériquement instable**.

Si l'on veut obtenir un algorithme qui ne présente pas cet inconvénient, un algorithme numériquement stable, il faut éliminer la différence de nombres voisins qui engendre une erreur de cancellation. Cela est possible. En effet, l'une des deux racines est toujours bien calculée : celle pour laquelle le signe devant la racine carrée est le même que celui de $-b$. Posons donc :

$$x_1 = (-b + \varepsilon\sqrt{b^2 - 4ac})/2a$$

avec $\varepsilon = +1$ si $b < 0$ et $\varepsilon = -1$ si $b \geqslant 0$.

x_1 sera toujours bien calculé. Il faut alors se souvenir que le produit des racines est égal à c/a. On calculera donc la seconde racine par :

$$x_2 = c/ax_1$$

x_2 sera toujours bien calculé : l'algorithme est **numériquement stable**, nous en avons éliminé les causes possibles d'erreurs de cancellation.

Cette notion de stabilité numérique est liée à un algorithme.

1.3 Conditionnement d'un problème

À la notion de stabilité numérique d'un algorithme vient s'adjoindre une notion liée au problème mathématique lui-même : le conditionnement. Avant de résoudre un problème, il faut introduire les données dans l'ordinateur. Celles-ci sont entachées d'une erreur d'affectation et le problème que l'on va résoudre diffère donc un peu de celui que l'on aurait dû résoudre. Il se peut que la solution exacte du problème ainsi perturbé soit très différente de la solution exacte du problème initial non perturbé : c'est la notion de **conditionnement d'un problème**.

On dit qu'un problème est **bien conditionné** si une petite variation des données n'entraîne qu'une petite variation des résultats. Inversement, un problème est **mal conditionné** si une petite variation des données peut entraîner une grande variation des résultats.

Naturellement, les notions de *petite* et *grande* variations dépendent de t, le nombre de digits de la mantisse des mots de l'ordinateur. On voit que la notion de conditionnement est liée au problème mathématique lui-même et qu'elle est indépendante de la stabilité numérique de l'algorithme qui sera ensuite utilisé pour le résoudre. Ces deux notions sont à prendre en compte simultanément dans l'analyse des résultats numériques fournis par l'ordinateur, de même qu'il faudra également tenir compte de la précision de la méthode de résolution utilisée, puisque nous avons dit, dans l'introduction, que la majorité des méthodes d'analyse numérique étaient des méthodes approchées.

1.4 Correction de l'arithmétique

En face des erreurs dues à l'arithmétique de l'ordinateur, on peut avoir plusieurs attitudes. On peut d'abord chercher à estimer ces erreurs en se basant sur les majorations des théorèmes **1** et **2**. On se place alors dans le pire des cas, celui où les erreurs ne se compensent jamais, et les bornes obtenues ainsi ne sont pas réalistes. À de telles majorations, il vaut mieux préférer une estimation statistique des erreurs dues à l'arithmétique de l'ordinateur : c'est la **méthode de permutation-perturbation** due à La Porte et Vignes [18] [43]. On trouvera le logiciel correspondant à cette méthode sur le site http://www-anp.lip6.fr/cadna/. Une autre attitude consiste à corriger l'arithmétique de l'ordinateur. Comme c'est dans une somme de termes que les erreurs peuvent le plus s'accumuler, nous allons montrer comment essayer de corriger un tel calcul par une méthode due à Pichat [36]. Soit à calculer :

$$S = \sum_{i=1}^{n} x_i$$

et soit $fl(S)$ la valeur obtenue sur ordinateur après calcul. Pour obtenir $fl(S)$, on effectue une boucle. On pose :

$$S_1 = x_1$$

puis on calcule :

$$S_i = S_{i-1} + x_i \quad \text{pour } i = 2, \ldots, n$$

On obtient :

$$S_n = fl(S)$$

Soit e_i l'erreur faite sur la $i^{\text{ième}}$ somme. Naturellement, on aura :

$$S = fl(S) + \sum_{i=1}^{n-1} e_i$$

Les e_i se calculent à l'aide des formules :

$$e_i = \begin{cases} -S_i + S_{i-1} + x_i & \text{si } |S_{i-1}| \geqslant |x_i| \\ -S_i + x_i + S_{i-1} & \text{si } |S_{i-1}| < |x_i| \end{cases}$$

Tous les chiffres décimaux de $T = S_n + \sum_{i=1}^{n-1} e_i$ sont exacts.

Exemple : calculer :

$$S = 1 + \sum_{i=1}^{1\,000} 10^{-6} = 1{,}001$$

Sur un ordinateur travaillant en arrondi avec $t = 6$, on obtient $fl(S) = 1{,}000\,95$ et $T = 1{,}001\,00$.

2. Interpolation

2.1 Polynôme d'interpolation et son calcul

Soit f une fonction réelle d'une variable réelle (ou, ce qui ne change rien, une fonction complexe d'une variable complexe). On suppose que l'on connaît les valeurs de $f(x_0)$, $f(x_1)$, ..., $f(x_n)$ et l'on cherche un polynôme P tel que :

$$P(x_i) = f(x_i) \text{ pour } i = 0, ..., n$$

On dit que P est le **polynôme d'interpolation** de f (ou qu'il **interpole** f) en $x_0, x_1, ..., x_n$. On a le résultat fondamental du théorème **3** en supposant qu'au moins l'une des quantités $f(x_i)$ est différente de zéro.

Théorème 3. Une condition nécessaire et suffisante pour qu'il existe un unique polynôme P de degré au plus égal à n qui interpole f en $x_0, x_1, ..., x_n$ est que les abscisses d'interpolation $x_0, x_1, ..., x_n$ soient toutes distinctes les unes des autres.

Pour obtenir ce polynôme P, il y a deux possibilités principales. La première est d'utiliser la **formule d'interpolation de Lagrange** qui dit que P est donné par :

$$P(x) = \sum_{i=0}^{n} L_i(x)\, f(x_i)$$

avec

$$L_i(x) = \prod_{\substack{j=0 \\ j \neq i}}^{n} (x - x_j)/(x_i - x_j)$$

Il est facile de voir que $L_i(x_i) = 1$ et que $L_i(x_k) = 0$ pour $k \neq i$, donc d'après l'unicité du polynôme d'interpolation, cette formule nous fournit bien P puisque $P(x_k) = f(x_k)$ pour $k = 0, ..., n$. Naturellement, les L_i dépendent de n et donc, si l'on veut ajouter de nouveaux points d'interpolation et augmenter n, tous les calculs seront à recommencer.

Pour cette raison, on utilise souvent le **schéma de Neville-Aitken** qui est particulièrement bien adapté à l'adjonction de nouveaux points d'interpolation. Appelons $T_k^{(i)}$ le polynôme de degré au plus égal à k qui interpole f en $x_i, ..., x_{i+k}$, c'est-à-dire que :

$$T_k^{(i)}(x_j) = f(x_j) \quad \text{pour} \quad j = i, ..., i + k$$

D'après cette définition, on a donc :

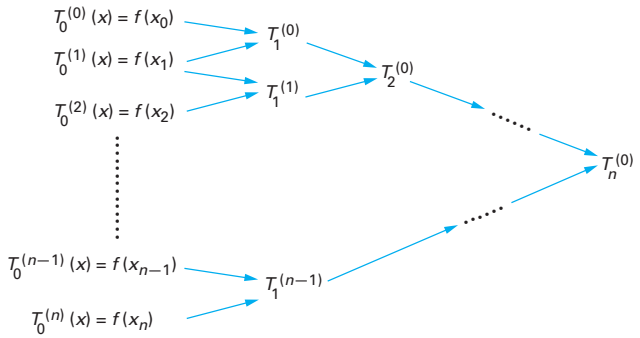
$$T_0^{(i)}(x) = f(x_i) \quad \text{pour} \quad i = 0, ..., n$$

On montre que les autres polynômes $T_k^{(i)}$ peuvent se calculer récursivement à l'aide du schéma de Neville-Aitken :

$$T_{k+1}^{(i)}(x) = \frac{(x_{i+k+1} - x)\, T_k^{(i)}(x) - (x_i - x)\, T_k^{(i+1)}(x)}{x_{i+k+1} - x_i}$$

pour $k = 0, ..., n-1$ et $i = 0, ..., n-k-1$.

Le polynôme $T_n^{(0)}$ ainsi obtenu est le polynôme d'interpolation de f en $x_0, ..., x_n$. On place habituellement ces polynômes dans un tableau à double entrée :



On voit que l'on se déplace dans ce tableau à partir de la colonne de gauche qui est connue, en allant vers la droite et de haut en bas. Les flèches indiquent comment obtenir, à l'aide de la formule précédente, un polynôme $T_{k+1}^{(i)}$ de la colonne $k+1$ à partir de deux polynômes $T_k^{(i)}$ et $T_k^{(i+1)}$ de la colonne k. Si l'on garde en mémoire la dernière diagonale montante $T_0^{(n)}, T_1^{(n-1)}, ..., T_n^{(0)}$, il est alors facile d'ajouter un nouveau point d'interpolation.

Le polynôme d'interpolation peut également s'exprimer à l'aide des **différences divisées**. Celles-ci sont définies récursivement de la manière suivante :

$$[x_i]_f = f(x_i)$$

$$[x_i, ..., x_{i+k}]_f = \frac{[x_{i+1}, ..., x_{i+k}]_f - [x_i, ..., x_{i+k-1}]_f}{x_{i+k} - x_i}$$

Le polynôme d'interpolation P est alors donné par la formule :

$$P(x) = [x_0]_f + (x - x_0)\,[x_0, x_1]_f + (x - x_0)(x - x_1)\,[x_0, x_1, x_2]_f + ... + (x - x_0) ... (x - x_{n-1})\,[x_0, ..., x_n]_f$$

On peut ainsi adjoindre de nouveaux points d'interpolation un par un. Pour passer du polynôme d'interpolation de degré n au polynôme de degré $n+1$ sur les mêmes points et un point supplémentaire, il suffit de rajouter un terme dans la formule précédente.

L'erreur s'exprime par :

$$f(x) - P(x) = (x - x_0) ... (x - x_n)\,[x_0, ..., x_n, x]_f.$$

Définissons l'opérateur Δ et ses puissances par $\Delta^0 f(x_i) = f(x_i)$ et $\Delta^{k+1} f(x_i) = \Delta^k f(x_{i+1}) - \Delta^k f(x_i)$ pour $k \geqslant 0$. Lorsque les points d'interpolation sont équidistants, c'est-à-dire $x_i = x_0 + ih$ pour $i = 0, 1, ...$, on a $k!\, h^k\, [x_i, ..., x_{i+k}]_f = \Delta^k f(x_i)$ et le polynôme d'interpolation P s'exprime à l'aide de la **formule de Newton** :

$$P(x) = f(x_0) + (x - x_0)\frac{\Delta f(x_0)}{1!\,h} + (x - x_0)(x - x_1)\frac{\Delta^2 f(x_0)}{2!\,h^2} + ... + (x - x_0) ... (x - x_{n-1})\frac{\Delta^n f(x_0)}{n!\,h^n}$$

2.2 Erreur d'interpolation

Dans la pratique, l'interpolation polynomiale sert à remplacer une fonction f, qui est soit inconnue, soit trop compliquée, par une fonction plus simple, en l'occurrence un polynôme. On dit que l'on approxime f par le polynôme d'interpolation P. Quand on utilise une approximation, comme c'est le cas dans de nombreuses méthodes d'analyse numérique, il est fondamental d'étudier l'erreur d'approximation. Naturellement, sauf cas particulier, l'expression de l'erreur ne permet pas de calculer cette erreur exactement (car, s'il en était ainsi, il n'y aurait plus d'erreur) ; elle peut cependant être très utile pour en calculer une borne supérieure. C'est ainsi que, pour l'interpolation polynomiale, on démontre le théorème **4**.

Théorème 4. Soit I un intervalle contenant $x_0, ..., x_n$ et x. Si f est $(n+1)$ fois continûment dérivable sur I, alors il existe $\xi \in I$ et dépendant de x tel que :

$$f(x) - P(x) = \frac{v(x)}{(n+1)!} f^{(n+1)}(\xi)$$

avec $$v(x) = (x - x_0)(x - x_1) ... (x - x_n)$$

Cette expression ne permet pas de calculer la valeur exacte de l'erreur parce que, en général, ξ est inconnu. Elle peut permettre d'en calculer une majoration ou de choisir les points d'interpolation $x_0, ..., x_n$ de façon optimale lorsque ceux-ci ne sont pas imposés.

2.3 Choix des points d'interpolation

Supposons que $x_0, ..., x_n$ et tous les points x possibles appartiennent à l'intervalle $[-1, +1]$ (auquel on pourra toujours se ramener par changement de variable). On a alors :

$$\max_{x \in [-1, +1]} |f(x) - P(x)| \leqslant \frac{1}{(n+1)!} \max_{x \in [-1, +1]} |v(x)| \max_{x \in [-1, +1]} \left|f^{(n+1)}(x)\right|$$

La borne supérieure de l'erreur ainsi obtenue contient deux termes : un qui dépend de $f^{(n+1)}$ et sur lequel on ne peut rien et un qui dépend uniquement des points d'interpolation, c'est :

$$\max_{x \in [-1, +1]} |v(x)|$$

On peut alors se poser la question de savoir comment choisir les points d'interpolation $x_0, ..., x_n$ de façon à rendre ce terme le plus petit possible. On aura ainsi minimisé une borne supérieure de l'erreur (et non pas l'erreur elle-même, ce qui est différent). Ce problème, très célèbre en mathématiques, a été posé et résolu par Tchebychev et les polynômes qui répondent à cette question ont reçu son nom. Les **polynômes de Tchebychev** vérifient la relation de récurrence :

$$T_0(x) = 1 \quad T_1(x) = x$$

$$T_{n+1}(x) = 2x\, T_n(x) - T_{n-1}(x) \text{ pour } n = 1, 2, ...$$

T_n est de degré n et, sur $[-1, +1]$, on a :

$$T_n(x) = \cos(n \arccos x) \text{ pour } n = 0, 1, ...$$

On montre que, parmi les polynômes v de degré $n+1$, ayant un coefficient du terme de plus haut degré égal à 1 et leurs racines toutes réelles, distinctes et dans $[-1, +1]$, celui qui minimise $\max_{x \in [-1, +1]} |v(x)|$ est $T_{n+1}(x)/2^n$.

Le choix optimal des points d'interpolation consiste donc à prendre les racines $x_0, ..., x_n$ de T_{n+1} qui sont données par :

$$x_i = \cos \frac{2i+1}{2n+2} \pi \quad \text{pour} \quad i = 0, ..., n$$

2.4 Convergence

Puisque l'on cherche à approximer une fonction f par un polynôme d'interpolation, il est une seconde question qu'il est naturel de se poser : celle de la convergence (en un sens à préciser) de ces polynômes d'interpolation lorsque n augmente indéfiniment.

On se donne n et des abscisses d'interpolation distinctes $x_0^{(n)}, x_1^{(n)}, ..., x_n^{(n)}$. Soit P_n le polynôme tel que :

$$P_n(x_i^{(n)}) = f(x_i^{(n)}) \quad \text{pour} \quad i = 0, ..., n$$

Soit $C_\infty[-1, +1]$ l'espace des fonctions continues sur $[-1, +1]$ muni de la norme :

$$\|f\| = \max_{x \in [-1, +1]} |f(x)|$$

On démontre le résultat négatif du théorème **5**.

Théorème 5. Quelles que soient les abscisses $x_i^{(n)}$ pour $i = 0, ..., n$ et pour $n = 0, 1, ...$, il existe au moins une fonction $f \in C_\infty[-1, +1]$ telle que la suite des polynômes d'interpolation (P_n) ne converge pas vers f dans $C_\infty[-1, +1]$, c'est-à-dire telle que :

$$\max_{x \in [-1, +1]} |f(x) - P_n(x)|$$

ne tende pas vers zéro lorsque n tend vers l'infini.

On voit donc qu'il faut faire attention : le résultat ne sera pas toujours meilleur en augmentant n. Nous avons obtenu un résultat négatif parce que nous demandions beaucoup : nous avons seulement imposé à f d'être continue et nous n'avons imposé aucune contrainte sur les points d'interpolation $x_i^{(n)}$. Dans la pratique, il n'y a pas lieu d'être aussi pessimiste car, dès que l'on demande moins, en imposant soit des conditions sur f, soit des conditions sur les $x_i^{(n)}$, on obtient des résultats positifs. C'est ainsi que l'on a le théorème **6**.

Théorème 6. Quelle que soit $f \in C_\infty[-1, +1]$, il existe des abscisses $x_i^{(n)}$ ($i = 0, ..., n$ et $n = 0, 1, ...$) telles que :

$$\lim_{n \to \infty} \max_{x \in [-1, +1]} |f(x) - P_n(x)| = 0$$

Cependant, il n'existe pas de famille d'abscisses $x_i^{(n)}$ qui conviennent pour toutes les fonctions continues et il est plus intéressant d'ajouter des conditions sur f comme le montre le théorème **7**.

Théorème 7. Si $f \in C_\infty[-1, +1]$ a une dérivée $k^{\text{ième}}$ continue (pour un certain $k \geqslant 1$), alors :

$$\lim_{n \to \infty} \max_{x \in [-1, +1]} |f(x) - P_n(x)| = 0$$

lorsque les $x_i^{(n)}$ sont les racines de T_{n+1}.

De plus, on a :

$$\max_{x \in [-1, +1]} |f(x) - P_n(x)| = o(\lg n / n^k)$$

On trouvera les démonstrations des résultats précédents ainsi que de nombreux autres résultats théoriques dans [19] [31].

2.5 Polynôme d'interpolation d'Hermite

Jusqu'à présent, nous avons imposé à notre polynôme d'interpolation P de satisfaire à :

$$P(x_i) = f(x_i), \text{ pour } i = 0, ..., n$$

Nous allons maintenant lui imposer de satisfaire en plus à :

$$P'(x_i) = f'(x_i), \text{ pour } i = 0, ..., n$$

en supposant naturellement connues les valeurs de $f'(x_0)$, ..., $f'(x_n)$. On dit alors que P est le **polynôme d'interpolation d'Hermite** de f en $x_0, ..., x_n$. Nous avons le théorème **8**.

Théorème 8. Une condition nécessaire et suffisante pour qu'il existe un unique polynôme d'interpolation d'Hermite de f en $x_0, ..., x_n$ de degré au plus égal à $2n+1$ est que les abscisses $x_0, ..., x_n$ soient toutes distinctes les unes des autres.

On montre que ce polynôme est donné par la formule :

$$P(x) = \sum_{i=0}^{n} H_i(x)\, f(x_i) + \sum_{i=0}^{n} V_i(x)\, f'(x_i)$$

avec

$$H_i(x) = \left[1 - 2(x - x_i) L_i'(x_i)\right] L_i^2(x)$$

$$V_i(x) = (x - x_i) L_i^2(x)$$

et

$$L_i(x) = \prod_{\substack{j=0 \\ j \neq i}}^{n} (x - x_j)/(x_i - x_j)$$

Pour l'erreur, on a le théorème **9**.

Théorème 9. Soit I un intervalle contenant $x_0, ..., x_n$ et x. Si f est $(2n+2)$ fois continûment dérivable sur I, alors il existe $\xi \in I$ et dépendant de x tel que :

$$f(x) - P(x) = \frac{v^2(x)}{(2n+2)!} f^{(2n+2)}(\xi)$$

avec $v(x) = (x - x_0)(x - x_1) ... (x - x_n)$

2.6 Exemples d'interpolation non polynomiale

Jusqu'à présent, nous avons toujours interpolé f par un polynôme parce que c'est un cas simple et qui couvre de nombreuses applications.

Cependant, les polynômes auront du mal à approcher correctement la fonction tan x qui présente des pôles ou la fonction exp $(-x)$ qui tend vers zéro lorsque x tend vers l'infini. Il est donc utile d'étudier l'interpolation par des familles autres que polynomiales, mais c'est un problème qui devient rapidement difficile même dès les conditions d'existence. Nous en donnerons cependant deux exemples.

■ Le **premier exemple** concerne l'interpolation par des fractions rationnelles. On commence par définir les différences réciproques de f par :

$$\rho_{-1}^{(i)} = 0 \quad \text{et} \quad \rho_0^{(i)} = f(x_i) \qquad \text{pour } i = 0, 1, ...$$

$$\rho_{k+1}^{(i)} = \rho_{k-1}^{(i+1)} + \frac{x_{i+k+1} - x_i}{\rho_k^{(i+1)} - \rho_k^{(i)}} \qquad \text{pour } k, i = 0, 1, ...$$

Puis on calcule les polynômes A_k et B_k par :

$$A_{-1}(x) = 1 \qquad A_0(x) = \rho_0^{(0)}$$
$$B_{-1}(x) = 0 \qquad B_0(x) = 1$$

$$A_{k+1}(x) = (\rho_{k+1}^{(0)} - \rho_{k-1}^{(0)}) A_k(x) + (x - x_k) A_{k-1}(x)$$

$$B_{k+1}(x) = (\rho_{k+1}^{(0)} - \rho_{k-1}^{(0)}) B_k(x) + (x - x_k) B_{k-1}(x), \text{ pour } k = 0, 1, ...$$

A_{2k}, A_{2k+1} et B_{2k+1} sont des polynômes de degré k au plus et B_{2k} est un polynôme de degré $k-1$ au plus sous certaines conditions d'existence que nous ne détaillerons pas ici. On démontre que la fraction rationnelle :

$$R_n(x) = A_n(x)/B_n(x)$$

interpole f en $x_0, ..., x_{2n}$, c'est-à-dire que :

$$R_n(x_i) = f(x_i), \text{ pour } i = 0, ..., 2n$$

■ Voyons maintenant un **second exemple**. Soit $g_0, ..., g_n$ des fonctions données. Nous allons rechercher P de la forme :

$$P(x) = a_0\, g_0(x) + ... + a_n\, g_n(x)$$

qui interpole f en $x_0, ..., x_n$, c'est-à-dire tel que :

$$P(x_i) = f(x_i), \text{ pour } i = 0, ..., n$$

On voit que ce problème généralise l'interpolation polynomiale que l'on retrouve pour $g_i(x) = x^i$. Il existe, pour calculer P, un algorithme qui généralise le schéma de Neville-Aitken et qui est dû à Mühlbach [34]. Soit $P_k^{(i)}$ tel que :

$$P_k^{(i)}(x) = a_0 g_0(x) + ... + a_k g_k(x)$$

et

$$P_k^{(i)}(x_j) = f(x_j) \qquad \text{pour } j = i, ..., i+k$$

Naturellement, comme dans le schéma de Neville-Aitken pour l'interpolation polynomiale, les coefficients $a_0, ..., a_k$ dépendent de k et de i. On montre que ces $P_k^{(i)}$ peuvent être calculés récursivement à l'aide du schéma suivant :

$$P_0^{(i)}(x) = f(x_i) g_0(x)/g_0(x_i) \qquad \text{pour } i = 0, 1, ...$$

$$g_{0,j}^{(i)}(x) = g_j(x_i) g_0(x)/g_0(x_i) - g_j(x) \qquad \text{pour } i = 0, 1, ... ; j = 1, 2, ...$$

$$P_{k+1}^{(i)}(x) = \frac{g_{k,k+1}^{(i+1)}(x)\, P_k^{(i)}(x) - g_{k,k+1}^{(i)}(x)\, P_k^{(i+1)}(x)}{g_{k,k+1}^{(i+1)}(x) - g_{k,k+1}^{(i)}(x)}$$

pour $i, k = 0, 1, ...$

$$g_{k+1,j}^{(i)}(x) = \frac{g_{k,k+1}^{(i+1)}(x)\, g_{k,j}^{(i)}(x) - g_{k,k+1}^{(i)}(x)\, g_{k,j}^{(i+1)}(x)}{g_{k,k+1}^{(i+1)}(x) - g_{k,k+1}^{(i)}(x)}$$

pour $i, k = 0, 1, ... ; j = k+2, ...$

On aura $P(x) = P_n^{(0)}(x)$ si aucune division par zéro ne se produit dans l'algorithme. On trouvera dans [34] une étude de ces conditions ainsi que des résultats théoriques concernant cet algorithme.

2.7 Fonctions splines

Représenter une fonction par un polynôme sur un intervalle nécessite souvent un degré élevé pour obtenir une bonne précision. On peut alors diviser l'intervalle en sous-intervalles et représenter la fonction par un polynôme de faible degré sur chacun d'eux. On obtient ainsi une approximation polynomiale par morceaux. Afin

que cette fonction par morceaux soit la plus lisse possible, on demandera à ces polynômes de se raccorder aux points de la subdivision ainsi que leurs dérivées jusqu'à un certain ordre. Une telle fonction s'appelle une **fonction spline**.

Sur les fonctions splines, voir [23] et [24].

Nous commencerons par étudier les **polynômes de Bernstein** qui forment une base de l'espace vectoriel des polynômes ainsi que la **représentation de Bézier** d'un polynôme.

2.7.1 Polynômes de Bernstein et représentation de Bézier

Les **polynômes de Bernstein** sont définis par :

$$B_k^n(x) = C_n^k(1-x)^{n-k}x^k, \qquad 0 \leqslant k \leqslant n, \qquad C_n^k = \frac{n!}{k!(n-k)!}$$

Pour tout k, B_k^n est un polynôme de degré k et l'on a :

$$B_0^{n+1}(x) = (1-x)B_0^n(x)$$
$$B_k^{n+1}(x) = (1-x)B_k^n(x) + xB_{k-1}^n(x)$$
$$B_{n+1}^{n+1}(x) = xB_n^n(x)$$

De plus

$$\sum_{k=0}^{n} B_k^n(x) = 1$$

$$\sum_{k=0}^{n} \frac{k}{n} B_k^n(x) = x$$

$$\sum_{k=0}^{n} \frac{k(k-1)}{n(n-1)} B_k^n(x) = x^2$$

Pour $n \geqslant 1$, le polynôme B_k^n admet les points 0 et 1 comme racines avec les multiplicités respectives k et $n-k$ et il atteint son maximum en k/n.

Les polynômes de Bernstein $B_0^n, ..., B_n^n$ forment une base de l'espace vectoriel des polynômes de degré au plus égal à n. Tout polynôme $P(x) = a_0 + a_1x + ... + a_nx^n$ peut donc être écrit dans cette base $P(x) = b_0B_0^n(x) + ... + b_nB_n^n(x)$. Cette forme s'appelle la **représentation de Bézier** de P et $b_0, ..., b_n$ sont ses **coefficients de Bézier**. Ces coefficients b_i sont reliés aux a_i par $b_0 = a_0$ et $b_i = -\Delta^i a_{0i}$ (Δ agit sur le premier indice inférieur), où $a_{ki} = b_k$ pour $k = 0, ..., i-1$ et $a_{ii} = -a_i / C_n^i$. Réciproquement $a_k = C_n^k \Delta^k b_0$ pour $k = 0, ..., n$.

Dans $\mathbb{R}^2$, les points $(i/n, b_i)$, pour $i = 0, ..., n$, sont les **points de Bézier** (ou **points de contrôle**) et la ligne polygonale qui les relie s'appelle le **polygone de Bézier** de P. Il contient le graphe de P et a une forme similaire. De plus, les points $(0, b_0)$ et $(1, b_n)$ appartiennent au graphe de P. Pour modifier P afin d'obtenir une forme désirée, il suffit de modifier la position des points de Bézier. C'est cette propriété qui fait tout l'intérêt de la représentation de Bézier en conception géométrique assistée par ordinateur [CAGD (*Computer Aided Geometric Design*) ou CAO (*Conception assistée par ordinateur*)].

Il est possible de calculer récursivement les valeurs de P et de ses dérivées en tout point par **l'algorithme de De Casteljau** à partir des coefficients de Bézier.

Soit $0 \leqslant r \leqslant s \leqslant n$.

On pose :

$$b_{rs}(x) = \sum_{i=r}^{s} b_i B_{i-r}^{s-r}(x)$$

On a :

$$b_{rr}(x) = b_r$$

et $$b_{rs}(x) = (1-x)b_{r,s-1}(x) + xb_{r+1,s}(x), \quad r < s$$

On a $b_{0n}(x) = P(x)$

et $$P^{(k)}(x) = \frac{n!}{(n-k)!}\Delta^k b_{0n}(x)$$

où l'opérateur Δ agit sur le premier indice inférieur.

La représentation de Bézier et l'algorithme de De Casteljau s'étendent aux courbes paramétrées et aux surfaces. Ce sont des ingrédients majeurs dans la représentation de courbes et de surfaces par des fonctions splines.

2.7.2 Courbe de Bézier

Soit P_i, pour $i = 0, ..., n$, des points de $\mathbb{R}^2$. On définit par récurrence la suite d'applications de $[0, 1]$ dans $\mathbb{R}^2$ de la manière suivante :

$$B_0(P_i)(t) = P_i, \quad \forall i$$

$$B_k(P_i, ..., P_{i+k})(t) = (1-t)B_{k-1}(P_0, ..., P_{i+k-1})(t) + tB_{k-1}(P_{i+1}, ..., P_{i+k})(t), \qquad k \geqslant 1$$

L'arc paramétré $B_n(P_0, ..., P_n) : t \mapsto B_n(P_0, ..., P_n)(t)$ s'appelle **courbe de Bézier** associée aux points de Bézier $P_0, ..., P_n$. Pour simplifier, on notera B_{rs} ($r \leqslant s \in \mathbb{N}$) l'application qui à t fait correspondre $B_{s-r}(P_r, ..., P_s)(t)$. On démontre que :

$$B_{rs}(t) = \sum_{i=r}^{s} B_{i-r}^{s-r}(t) P_i$$

En particulier :

$$B_{0n}(t) = \sum_{i=0}^{n} B_i^n(t) P_i$$

et le support de cet arc paramétré est contenu dans l'enveloppe convexe de $P_0, ..., P_n$.

Soit f une fonction définie sur un certain intervalle. Par un changement de variable, on peut ramener cet intervalle à $[0, m]$, $m \in \mathbb{N}$. Sur chaque sous-intervalle $[k, k+1]$, pour $k = 0, ..., m-1$, f sera représentée par un polynôme P_k de degré n avec la condition de raccordement $P_k(k+1) = P_{k+1}(k+1)$. Pour utiliser la représentation de Bézier, on effectue le changement de variable $t = x - k$ dans $[k, k+1]$.

Soit $b_{nk}, ..., b_{nk+n}$ les coefficients de Bézier pour l'intervalle $[k, k+1]$, c'est-à-dire $b_0, ..., b_n$ pour le premier sous-intervalle, $b_n, ..., b_{2n}$ pour le second et ainsi de suite jusqu'à $b_{nm-n}, ..., b_{nm}$ pour le dernier. On obtient ainsi une approximation polynomiale par morceaux $\tilde{f}$ de f qui, pour le choix $P_k(k+1) = P_{k+1}(k+1) = f(k+1)$, interpole f aux points $1, ..., m-1$. La fonction $\tilde{f}$ est définie par les $nm+1$ coefficients $b_0, ..., b_{nm}$ et elle peut être calculée en tout point $x \in [0, m]$ par l'algorithme de De Casteljau. Pour cela, il faut commencer par déterminer $k \in \{0, ..., nm-1\}$ tel que $x \in [k, k+1]$, puis appliquer l'algorithme de De Casteljau aux coefficients de Bézier $b_{nk}, ..., b_{nk+n}$. Le graphe de $\tilde{f}$ est une courbe de Bézier et elle est contenue dans l'enveloppe convexe des points de Bézier.

On peut imposer des conditions de raccordement supplémentaires à $\tilde{f}$. Ainsi, cette fonction sera de classe C^1 si et seulement si $n(b_{nk} - b_{nk-1}) = n(b_{nk+1} - b_{nk})$, c'est-à-dire :

$$2b_{nk} = b_{nk-1} + b_{nk+1} \quad \text{pour} \quad k = 1, \ldots, m-1$$

$\tilde{f}$ sera de classe C^2 si, de plus :

$$2b_{nk-1} - b_{nk-2} = 2b_{nk+1} - b_{nk+2} \quad \text{pour} \quad k = 1, \ldots, m-1$$

Et ainsi de suite pour des conditions de raccordement sur les dérivées d'ordres supérieurs.

La représentation de Bézier et l'algorithme de De Casteljau s'étendent à $\mathbb{R}^m$.

2.7.3 Spline et spline naturelle

Soit $D = \{x_0, \ldots, x_{n+1}\}$ une subdivision de $[a, b]$ avec $a = x_0 < x_1 < \ldots < x_n < x_{n+1} = b$. Une fonction $S : [a, b] \mapsto \mathbb{R}$ est une **fonction spline** de degré k par rapport à D si et seulement si S est un polynôme de degré k au plus sur chaque sous-intervalle $[x_i, x_{i+1}]$ pour $i = 0, \ldots, n$ et est de classe C^{k-1} sur $[a, b]$, c'est-à-dire, pour $i = 1, \ldots, n$:

$$S^{(j)}(x_i^-) = S^{(j)}(x_i^+), \qquad j = 0, \ldots, k-1$$

L'espace vectoriel des splines de degré k par rapport à D est dénoté $S_{k,D}$.

Posons :

$$(x - x_i)_+^k = \begin{cases} (x - x_i)^k & \text{si } x \geqslant x_i \\ 0 & \text{si } x < x_i \end{cases}$$

La fonction S s'appelle une **spline naturelle** de degré $2k-1$ par rapport à D si c'est un polynôme de degré $2k-1$ au plus sur chaque sous-intervalle $]x_i, x_{i+1}[$ pour $i = 1, \ldots, n-1$, si c'est un polynôme de degré $k-1$ au plus sur $[a, x_1[$ et $]x_n, b]$, et si elle est de classe C^{2k-2} sur $[a, b]$. L'espace vectoriel des splines naturelles par rapport à D est dénoté $S_{k,D}^N$. Une fonction $S : [a, b] \mapsto \mathbb{R}$ est une spline naturelle de degré $2k-1$ par rapport à D si et seulement si elle est de la forme :

$$S(x) = p_0(x) + \sum_{i=1}^{n} a_i (x - x_i)_+^{2k-1}$$

avec p_0 polynôme de degré $k-1$ au plus et si les coefficients a_i, $i = 1, \ldots, n$, satisfont :

$$\sum_{i=1}^{n} a_i x_i^j = 0, \qquad j = 0, \ldots, k-1$$

On a $a_i = (S^{(2k-1)}(x_i^+) - S^{(2k-1)}(x_i^-))/(2k-1)!$. Quels que soient $x_1, \ldots, x_n \in [a, b]$, $y_1, \ldots, y_n$, et $k \in \{1, \ldots, n\}$, il existe une unique spline naturelle d'interpolation $S \in S_{k,D}^N$ telle que $S(x_i) = y_i$ pour $i = 1, \ldots, n$.

Si, sur chaque sous-intervalle, les polynômes sont écrits dans leur représentation de Bézier, alors les conditions de raccordement des polynômes et de leurs dérivées jusqu'à un certain ordre aux points de la subdivision se traduisent par des conditions sur les coefficients de Bézier correspondants à chaque sous-intervalle.

2.7.4 Splines cubiques d'interpolation

Les **splines cubiques** sont parmi les plus utilisées. Elles correspondent à $k = 3$, ou à $2k - 1 = 3$ dans le cas des splines naturelles. Posons $h_i = x_{i+1} - x_i$ pour $i = 1, \ldots, n-1$. Dans chaque sous-intervalle, S est un polynôme de degré 3 au plus. En écrivant les conditions d'interpolation $S(x_i) = y_i$ pour $i = 1, \ldots, n$ et la continuité des dérivées premières en $x_2, \ldots, x_{n-1}$, on obtient le système tridiagonal :

$$\begin{pmatrix} H_1 & h_2 & & & \\ h_2 & H_2 & h_3 & & \\ & \ddots & \ddots & \ddots & \\ & & h_{n-3} & H_{n-3} & h_{n-2} \\ & & & h_{n-2} & H_{n-2} \end{pmatrix} \begin{pmatrix} z_2 \\ z_3 \\ \vdots \\ z_{n-2} \\ z_{n-1} \end{pmatrix} = \begin{pmatrix} v_2 \\ v_3 \\ \vdots \\ v_{n-2} \\ v_{n-1} \end{pmatrix}$$

avec $z_i = S''(x_i)$ pour $i = 1, \ldots, n$,

$z_1 = z_n = 0$ puisque S est affine sur $[a, x_1]$ et $[x_n, b]$,

$H_i = 2(h_i + h_{i+1})$, $i = 2, \ldots, n-2$,

$v_i = [(y_{i+1} - y_i)/h_i - (y_i - y_{i-1})/h_{i-1}]/6$, $i = 2, \ldots, n-1$

En résolvant ce système, on obtient :

$$p_i(x) = \frac{z_{i+1}}{6h_i}(x - x_i)^3 + \frac{z_i}{6h_i}(x_{i+1} - x)^3 + \left(\frac{y_{i+1}}{h_i} - \frac{h_i}{6} z_{i+1}\right)(x - x_i) + \left(\frac{y_i}{h_i} - \frac{h_i}{6} z_i\right)(x_{i+1} - x)$$

pour $i = 1, \ldots, n-1$.

$S = \alpha x + \beta$ est déterminé sur $[a, x_1]$ par y_1 et $p_1'(x_1)$, et par y_n et $p_{n-1}'(x_n)$ sur $[x_n, b]$.

2.7.5 *B*-splines

La base de $S_{k,D}$ telle qu'elle vient d'être construite est non locale. D'autre part, elle est mal conditionnée si deux points x_i sont voisins. On peut construire une base de $S_{k,D}$ consistant en des fonctions positives à support compact, appelées ***B*-splines**.

Les B-splines de degré k sont définies par :

$$N_i^k(x) = (-1)^{k+1}(x_{i+k+1} - x_i)[x_i, \ldots, x_{i+k+1}]_g$$

avec $g(t) = (x - t)_+^k$.

Elles satisfont la relation de récurrence :

$$N_i^k(x) = \frac{x - x_i}{x_{i+k} - x_i} N_i^{k-1}(x) + \frac{x_{i+k+1} - x}{x_{i+k+1} - x_{i+1}} N_{i+1}^{k-1}(x), \quad k \geqslant 1$$

Soit $S(x) = \sum_{i=-k}^{n} d_i N_i^k(x)$ une spline de degré k par rapport à D écrite dans la base des B-splines. Pour tout $x \in [a, b[$, elle peut être calculée par l'**algorithme de de Boor** :

soit $0 \leqslant j \leqslant n$ tel que $x \in [x_j, x_{j+1}[$. On pose $d_i^0 = d_i$ et, pour $m \geqslant 1$, on calcule :

$$d_i^m = \alpha_i^{k-m+1} d_i^{m-1} + (1 - \alpha_i^{k-m+1}) d_{i-1}^{m-1} \quad \text{avec} \quad \alpha_i^m = \frac{x - x_i}{x_{i+m} - x_i}$$

Pour $m \geqslant 1$, les d_i^m sont des fonctions de x et l'on a, pour $1 \leqslant m \leqslant k$:

$$S(x) = \sum_{i=j-k+m}^{j} d_i^m N_i^{k-m}(x)$$

Donc, pour $m = k$:

$$S(x) = d_j^k(x)$$

Si $k \geqslant 1$, on peut utiliser cet algorithme pour $x = b$ avec $j = m$.

2.7.6 Cas multivarié

Un avantage important de la représentation de Bézier est qu'elle s'étend facilement au cas de plusieurs variables. En effet, les polynômes de Bernstein $B_r^n(x)$ et $B_s^m(y)$ forment une base de l'espace vectoriel des polynômes de degré au plus n et m par rapport aux variables x et y et l'on a donc :

$$P(x, y) = \sum_{i=0}^{n} \sum_{k=0}^{m} b_{ik} B_i^n(x) B_k^m(y)$$

Les coefficients de Bézier constituent maintenant une matrice. Si y est fixé, le polynôme en x, $P(x, y)$, a comme coefficients de Bézier les nombres $b_i(y) = \sum_{k=1}^{m} b_{ik} B_k^m(y)$. Ainsi, pour y fixé, l'algorithme de De Casteljau appliqué à chaque ligne de la matrice de Bézier génère les coefficients de Bézier de $P(x, y)$ pour y fixé [23].

Les fonctions de plusieurs variables peuvent aussi être représentées par des **fonctions à base radiale**. On suppose que la fonction f est connue sur un ensemble discret de points $D \in \mathbb{R}^n$. Soit $\phi : \mathbb{R}^+ \mapsto \mathbb{R}$ une fonction donnée. Un approximation à base radiale de f a la forme générale :

$$S(x) = \sum_{r \in D} \lambda_r \phi(\|x - r\|), \quad x \in \mathbb{R}^n$$

Exemple : le plus simple est $\phi(r) = r$.

D'autres choix intéressants sont $\phi(r) = r^2 \ln r$ (*splines plaque mince*), $\phi(r) = (r^2 + c^2)^{1/2}$ où c est un paramètre positif (*multiquadriques*) et $\phi(r) = \exp(-\alpha r^2)$ avec α un paramètre positif (*cas gaussien*) [11] [44].

3. Quadrature numérique

3.1 Quadrature de type interpolation

Nous allons maintenant nous intéresser à l'obtention d'une valeur numérique approchée de l'intégrale définie :

$$I = \int_a^b f(x)\, \omega(x)\, \mathrm{d}x$$

avec $\omega(x) > 0, \forall x \in]a, b[$,

$$\int_a^b \omega(x)\, \mathrm{d}x < +\infty .$$

L'idée de base des méthodes numériques pour résoudre ce problème (que l'on appelle : méthodes de quadrature) est de remplacer la fonction f que l'on ne sait pas intégrer par son polynôme d'interpolation. On a une **méthode de quadrature de type interpolation**.

Soit donc $a \leqslant x_0 < x_1 < \ldots < x_{n-1} < x_n \leqslant b$ et soit P_n le polynôme d'interpolation de f (et non pas de $f\omega$) en ces points. Nous avons, d'après le théorème **9**, $f(x) = P_n(x) + E_n(x)$ où $E_n(x)$ est l'erreur d'interpolation en x. Par conséquent :

$$\int_a^b f(x)\, \omega(x)\, \mathrm{d}x = \int_a^b P_n(x)\, \omega(x)\, \mathrm{d}x + \int_a^b E_n(x)\, \omega(x)\, \mathrm{d}x$$

En remplaçant P_n par son expression donnée par la formule de Lagrange (§ 2.1), on obtient :

$$I = I_n + R_n$$

avec

$$I_n = \sum_{i=0}^{n} A_i^{(n)} f(x_i)$$

$$A_i^{(n)} = \int_a^b L_i(x)\, \omega(x)\, \mathrm{d}x$$

$$R_n = \int_a^b E_n(x)\, \omega(x)\, \mathrm{d}x$$

I_n est une valeur approchée de I et R_n est l'erreur de la formule de quadrature.

Par construction même de cette formule et d'après le théorème **9**, on a le théorème **10**.

Théorème 10. Si f est un polynôme de degré n au plus, alors $I_n = I$.

On dit que I_n est exact sur $\mathcal{P}_n$, l'espace vectoriel des polynômes de degré inférieur ou égal à n. Ce résultat est valable quel que soit le choix des abscisses d'interpolation. Nous allons donc, dans un premier temps, nous borner à un choix particulièrement simple des x_i. On pose :

$$h = (b - a)/n$$

et l'on prend :

$$x_i = a + ih \text{ pour } i = 0, \ldots, n$$

Le premier travail est de calculer les coefficients $A_i^{(n)}$ de la formule de quadrature. Lorsque $\omega(x) = 1$ (ce qui est le cas le plus courant et nous nous placerons dans ce cas jusqu'à nouvel avis), il existe des tables qui donnent les valeurs numériques des $A_i^{(n)}$. C'est ainsi que l'on a :

$$A_0^{(1)} = A_1^{(1)} = (b - a)/2$$

$$A_0^{(2)} = A_2^{(2)} = (b - a)/6 \quad \text{et} \quad A_1^{(2)} = 4(b - a)/6$$

Les formules correspondantes s'appellent : **formules de quadrature de Newton-Cotes**.

3.2 Convergence et stabilité

La première question importante qu'il nous faut résoudre est celle de la **convergence** : la suite (I_n) converge-t-elle vers I lorsque n tend vers l'infini, $\forall f \in C_\infty[a, b]$?

La seconde question à laquelle il nous faut répondre est celle de la **stabilité numérique** de la formule de quadrature. En effet, dans la pratique, les $f(x_i)$ ne sont pas connus exactement parce qu'ils proviennent de mesures ou parce qu'ils sont entachés d'une erreur

de calcul due à l'arithmétique de l'ordinateur. Donc, au lieu de calculer I_n, on calcule :

$$\sum_{i=0}^{n} A_i^{(n)} (f(x_i) + \varepsilon_i)$$

La différence entre ce que l'on calcule réellement et ce que l'on voulait calculer est donc :

$$\sum_{i=0}^{n} A_i^{(n)} \varepsilon_i$$

On dira qu'une formule de quadrature est **stable** s'il existe une constante M telle que pour tout n et quels que soient $\varepsilon_0, ..., \varepsilon_n$ on ait :

$$\left| \sum_{i=0}^{n} A_i^{(n)} \varepsilon_i \right| \leqslant M \max_{0 \leqslant i \leqslant n} |\varepsilon_i|$$

On démontre les théorèmes **11** et **12**.

Théorème 11. Une condition nécessaire et suffisante pour qu'une méthode de quadrature soit convergente sur $C_\infty [a, b]$ est que :

a) elle soit convergente lorsque f est un polynôme ;

b) il existe une constante M telle que, pour tout n :

$$\sum_{i=0}^{n} \left| A_i^{(n)} \right| \leqslant M$$

Théorème 12. Une condition nécessaire et suffisante pour qu'une méthode de quadrature soit stable est que la condition *b* du théorème **11** soit satisfaite.

On démontre que, pour la méthode de Newton-Cotes, cette condition n'est pas vérifiée et, par conséquent, on a le théorème **13**.

Théorème 13. La méthode de Newton-Cotes n'est ni stable ni convergente sur $C_\infty [a, b]$.

3.3 Méthodes des trapèzes et de Romberg

En raison du théorème **13**, on ne peut pas utiliser directement la méthode de Newton-Cotes, mais on est obligé de passer par le biais d'une **méthode de quadrature composite**. On écrit que :

$$\int_a^b f(x)\,dx = \int_{x_0}^{x_1} f(x)\,dx + \int_{x_1}^{x_2} f(x)\,dx + ... + \int_{x_{n-1}}^{x_n} f(x)\,dx$$

et l'on va calculer séparément une valeur numérique approchée de chacune de ces n intégrales en utilisant la plus simple de toutes les formules de Newton-Cotes, celle à deux points qui est de la forme (cf. fin § 3.1) :

$$\int_{x_i}^{x_{i+1}} f(x)\,dx \approx \frac{x_{i+1} - x_i}{2} [f(x_i) + f(x_{i+1})]$$

En prenant les abscisses x_i équidistantes, comme nous l'avons fait dans les méthodes de Newton-Cotes, on obtient une valeur approchée de I que nous noterons T_n ou $T(h)$ selon qu'il vaut mieux faire ressortir sa dépendance en n ou celle en h [on rappelle que $h = (b - a)/n$] :

$$T_n = \frac{h}{2} \left[f(a) + 2 \sum_{i=1}^{n-1} f(a + ih) + f(b) \right]$$

Cette formule s'appelle **méthode des trapèzes** et l'on démontre que :

$$T_n - I = \frac{(b-a)^3}{12 n^2} f''(\xi) \quad \text{avec} \quad \xi \in [a, b]$$

La méthode des trapèzes est de la forme :

$$T_n = \sum_{i=0}^{n} A_i^{(n)} f(x_i)$$

avec $x_i = a + ih$

$$A_0^{(n)} = A_n^{(n)} = h/2$$

$$A_i^{(n)} = h \quad \text{pour} \quad i = 1, ..., n-1$$

On a donc :

$$\sum_{i=0}^{n} \left| A_i^{(n)} \right| = \sum_{i=0}^{n} A_i^{(n)} = b - a$$

qui est une constante indépendante de n. Par ailleurs, d'après la formule de l'erreur, on voit que la méthode des trapèzes est convergente lorsque f est un polynôme ; par conséquent, nous avons démontré le théorème **14**.

Théorème 14. La méthode des trapèzes est stable et convergente sur $C_\infty [a, b]$.

Par contre, les résultats fournis par cette méthode ne sont souvent pas très précis et la suite (T_n) ne converge pas très vite vers I. Nous allons donc voir comment améliorer la précision de la formule des trapèzes ou, ce qui revient ici au même, comment accélérer la convergence de (T_n). L'idée est d'abord d'utiliser la méthode des trapèzes pour différentes valeurs du pas h. On obtient ainsi différentes valeurs approchées de I :

$$T(h_0),\ T(h_1),\ T(h_2),\ ...$$

telles que $\lim_{n \to \infty} T(h_n) = I$ si $\lim_{n \to \infty} h_n = 0$. Puis, on fait passer un polynôme d'interpolation par ces valeurs de $T(h_i)$ et enfin on calcule la valeur en 0 de ce polynôme d'interpolation, ce qui nous fournit, en général, une valeur approchée de I bien meilleure que les $T(h_i)$. Pour des raisons de stabilité numérique et de minimisation du nombre d'évaluations de f à effectuer, on prend $h_{i+1} = h_i/2$. Pour des raisons basées sur la justification théorique de cette procédure, on effectue le changement de variable $x = h^2$ dans le polynôme qui interpole les $T(h_i)$. Enfin, dans la pratique, le schéma de Neville-Aitken est particulièrement bien adapté au calcul des valeurs en 0 de ces polynômes d'interpolation. Appelons $T_k^{(n)}$ la valeur en 0 du polynôme d'interpolation de $T(h)$ en $x_n = h_n^2, ..., x_{n+k} = h_{n+k}^2 = h_n^2 / 2^{2k}$.

Nous obtenons l'algorithme suivant :

$$T_0^{(n)} = T(h_n) = T(h_0 / 2^n) \quad \text{pour} \quad n = 0, 1, ...$$

$$T_{k+1}^{(n)} = \frac{4^{k+1} T_k^{(n+1)} - T_k^{(n)}}{4^{k+1} - 1} \quad \text{pour } k, n = 0, 1, ...$$

Si l'on suppose f suffisamment dérivable, alors on démontre que :

— $\lim_{n \to \infty} T_k^{(n)} = \lim_{k \to \infty} T_k^{(n)} = I$;

— $T_k^{(n)} - I = O(h_n^{2k+2})$ lorsque n tend vers l'infini ;

— la suite $(T_{k+1}^{(n)})$ pour k fixé converge vers I plus vite que $(T_k^{(n)})$, c'est-à-dire que, pour tout k :

$$\lim_{n \to \infty} (T_{k+1}^{(n)} - I)/(T_k^{(n)} - I) = 0$$

Les $T_k^{(n)}$ ainsi obtenus sont, en général, bien plus précis que les $T_0^{(n)}$ fournis par la méthode des trapèzes. Cette méthode s'appelle **méthode de Romberg**.

3.4 Méthode de Gauss et polynômes orthogonaux

Revenons maintenant au cas général avec une fonction ω vérifiant :

$$\forall x \in]a, b[, \ \omega(x) > 0 \quad \text{et} \quad \int_a^b \omega(x) \mathrm{d}x < +\infty$$

Nous avons vu que, quel que soit le choix de $x_0, x_1, ..., x_n$, I_n est exact sur $\mathcal{P}_n$ (théorème **10**). Peut-on avoir mieux ou, en d'autres termes, est-il possible de choisir $x_0, ..., x_n$ de sorte que I_n soit exact pour des polynômes de degré le plus élevé possible ? Nous allons maintenant étudier un tel choix. On démontre le théorème **15**.

Théorème 15. Une condition nécessaire et suffisante pour que I_n soit exact sur $\mathcal{P}_{2n+1}$ est que :

$$\int_a^b x^i v_{n+1}(x) \omega(x) \mathrm{d}x = 0, \text{ pour } i = 0, ..., n$$

avec $$v_{n+1}(x) = (x - x_0)(x - x_1)...(x - x_n)$$

Ce théorème nous permet donc de répondre à la question que nous nous étions posée. Il faut d'abord rechercher un polynôme v_{n+1}, de degré $n + 1$, qui satisfait aux relations du théorème **15** (les **relations d'orthogonalité**). On démontre que, grâce aux conditions imposées sur ω, un tel polynôme existe toujours, que ses racines $x_0, x_1, ..., x_n$ sont réelles, distinctes et appartiennent à $[a, b]$. On construit donc I_n en prenant pour abscisses d'interpolation $x_0, ..., x_n$ les racines de v_{n+1} qui satisfont bien aux conditions précisées au début de ce paragraphe ($x_0, ..., x_n$ dépendent bien évidemment de n). On calcule les coefficients $A_i^{(n)}$ par la formule donnée au paragraphe 3.1. La méthode de quadrature ainsi obtenue s'appelle **méthode de Gauss**. On démontre qu'elle est optimale car on a le théorème **16**.

Théorème 16. Dans la méthode de quadrature de Gauss, I_n n'est pas exact sur $\mathcal{P}_{2n+2}$ (c'est-à-dire qu'il existe au moins un $f \in \mathcal{P}_{2n+2}$ tel que $I_n \neq I$).

On dit que la famille de polynômes $\{v_0, v_1, v_2, ...\}$, qui satisfait aux conditions d'orthogonalité du théorème **15**, forme la **famille de polynômes orthogonaux** sur l'intervalle [a, b] par rapport à la **fonction poids** ω. Les familles de polynômes orthogonaux ont des propriétés caractéristiques importantes. D'abord ils vérifient une relation de récurrence à trois termes qui permet de les calculer récursivement. Écrivons v_k sous la forme :

$$v_k(x) = t_k x^k + s_k x^{k-1} + ...$$

On démontre le théorème **17**.

Théorème 17. Toute famille de polynômes orthogonaux $\{v_k\}$ satisfait à une relation de récurrence de la forme :

$$v_{k+1}(x) = (A_{k+1} x + B_{k+1}) v_k(x) - C_{k+1} v_{k-1}(x), \text{ pour } k = 0, 1, ...$$

avec $$v_{-1}(x) = 0, \quad v_0(x) = t_0$$

$$A_{k+1} = t_{k+1}/t_k$$

$$B_{k+1} = A_{k+1}\left(\frac{s_{k+1}}{t_{k+1}} - \frac{s_k}{t_k}\right)$$

$$C_{k+1} = \frac{t_{k-1} t_{k+1}}{t_k^2} \frac{h_k}{h_{k-1}}$$

et $$h_k = \int_a^b v_k^2 x \omega(x) \mathrm{d}x$$

Comme les polynômes v_k sont déterminés à une constante multiplicative près (car alors les racines restent inchangées), on peut toujours prendre, dans les relations précédentes, $t_k = 1$ pour tout k. Connaissant v_{k-1} et v_k on peut alors calculer v_{k+1}.

Comme nous venons de le voir, les racines de v_k possèdent également certaines propriétés intéressantes. On a ainsi le théorème **18**.

Théorème 18. Pour tout k, les racines de v_k sont réelles, distinctes et appartiennent à $[a, b]$; v_k et v_{k+1} n'ont pas de racine commune. Entre deux racines consécutives de v_k il y a une et une seule racine de v_{k+1} et inversement.

Après avoir calculé récursivement le polynôme v_{n+1} grâce à la relation de récurrence du théorème **17**, il faut calculer ses racines. Nous verrons dans le dossier [AF 1 221], § 1 des méthodes numériques qui permettent de résoudre ce problème (par exemple, la méthode de Bairstow). Il nous faut ensuite calculer les coefficients $A_i^{(n)}$ de la formule de quadrature de Gauss. On démontre que ces coefficients sont tous strictement positifs et que l'on a :

$$A_i^{(n)} = \frac{A_{n+1} h_n}{v'_{n+1}(x_i) v_n(x_i)} \qquad \text{pour } i = 0, ..., n$$

avec $x_0, ..., x_n$ racines de v_{n+1}.

Non seulement les méthodes de Gauss sont optimales par rapport à un choix arbitraire des abscisses d'interpolation puisque I_n est exact sur $\mathcal{P}_{2n+1}$ au lieu de $\mathcal{P}_n$ mais de plus on a le théorème **19**.

Théorème 19. Les formules de quadrature de Gauss sont stables et convergentes sur $C_\infty[a, b]$.

On démontre de plus que :

$$R_n = I - I_n = \int_a^b v_{n+1}^2(x) \frac{f^{(2n+2)}(\xi)}{(2n+2)!} \omega(x) \mathrm{d}x$$

avec ξ dépendant de x.

Nous allons maintenant passer en revue les familles de polynômes orthogonaux les plus couramment utilisées. Ces polynômes ont des applications dans de nombreux autres problèmes d'analyse numérique. Chaque famille a reçu un nom particulier et est désignée par une lettre souvent normalisée.

Polynômes de Legendre P_n

Ils sont définis sur [– 1, + 1] avec $\omega(x) = 1$. Leur relation de récurrence est :

$$(n+1)\,P_{n+1}(x) = (2n+1)\,x\,P_n(x) - n\,P_{n-1}(x) \quad \text{pour } n = 1, 2, \ldots$$

avec $P_0(x) = 1$ et $P_1(x) = x$.

On a :

$$A_i^{(n)} = \frac{2(1-x_i^2)}{(n+1)^2\,P_n^2(x_i)} \qquad \text{pour } i = 0, \ldots, n$$

et $$R_n = \frac{2^{2n+3}[(n+1)!]^4}{(2n+3)[(2n+2)!]^3}\,f^{(2n+2)}(\xi) \quad \text{avec } \xi \in [-1, +1]$$

Polynômes de Laguerre L_n

Ils sont définis sur [0, + ∞) avec $\omega(x) = \exp(-x)$. Leur relation de récurrence est :

$$(n+1)\,L_{n+1}(x) = (2n+1-x)\,L_n(x) - n\,L_{n-1}(x) \quad \text{pour } n = 1, 2, \ldots$$

avec $L_0(x) = 1$ et $L_1(x) = 1 - x$.

On a :

$$A_i^{(n)} = \frac{[(n+1)!]^2}{x_i[L'_{n+1}(x_i)]^2} \qquad \text{pour } i = 0, \ldots, n$$

et $$R_n = \frac{[(n+1)!]^2}{(2n+2)!}\,f^{(2n+2)}(\xi) \quad \text{avec } \xi \in [0, +\infty]$$

Polynômes d'Hermite H_n

Ils sont définis sur (– ∞, + ∞) avec $\omega(x) = \exp(-x^2)$. Leur relation de récurrence est :

$$H_{n+1}(x) = 2x\,H_n(x) - 2n\,H_{n-1}(x) \quad \text{pour } n = 1, 2, \ldots$$

avec $H_0(x) = 1$ et $H_1(x) = 2x$.

On a :

$$A_i^{(n)} = \frac{2^{n+2}(n+1)!\,\sqrt{\pi}}{[H'_{n+1}(x_i)]^2} \qquad \text{pour } i = 0, \ldots, n$$

et $$R_n = \sqrt{\pi}\,\frac{(n+1)!}{2^{n+1}(2n+2)!}\,f^{(2n+2)}(\xi) \quad \text{avec } \xi \in (-\infty, +\infty)$$

Polynômes de Tchebychev de première espèce T_n

Ils sont définis sur [– 1, + 1] avec $\omega(x) = 1/\sqrt{1-x^2}$. Leur relation de récurrence est :

$$T_{n+1}(x) = 2x\,T_n(x) - T_{n-1}(x) \quad \text{pour } n = 1, 2, \ldots$$

avec $T_0(x) = 1$ et $T_1(x) = x$.

Ce sont les polynômes que nous avons déjà rencontrés dans le paragraphe 2.3. On a :

$$x_i = \cos\frac{2i+1}{2n+2}\pi \qquad \text{pour } i = 0, \ldots, n$$

$$A_i^{(n)} = \pi/(n+1) \qquad \text{pour } i = 0, \ldots, n$$

et $$R_n = \frac{2\pi}{2^{2n+2}(2n+2)!}\,f^{(2n+2)}(\xi) \quad \text{avec } \xi \in [-1, +1]$$

Il existe de nombreuses autres familles importantes de polynômes orthogonaux. Nous renvoyons le lecteur intéressé aux ouvrages sur cette question, par exemple [26].

Estimation de l'erreur

Un principe général en analyse numérique est que, si l'on dispose de deux valeurs approchées d'une même quantité, leur différence est, en général, une bonne estimation de l'erreur sur la moins bonne des deux approximations. Selon ce principe, on peut donc estimer l'erreur d'une formule de quadrature de Gauss à $n+1$ points (exacte sur $\mathcal{P}_{2n+1}$) par sa différence avec le résultat fourni par une formule de Gauss à $n+2$ points. Cependant, l'estimation de l'erreur ainsi obtenue ne sera pas très bonne car la seconde formule, qui est censée fournir une solution de référence, n'est exacte que sur $\mathcal{P}_{2n+3}$.

L'idée de la **procédure de Kronrod** est de bâtir une seconde formule de quadrature meilleure. Elle est construite en conservant les $n+1$ points de la première formule de quadrature (les racines du polynôme orthogonal v_{n+1}) et en leur ajoutant $n+2$ nouveaux points choisis au mieux, c'est-à-dire afin que cette seconde formule soit exacte sur $\mathcal{P}_{3n+4}$. On montre que ces nouveaux points doivent être les racines du **polynôme de Stieltjes** s_{n+2} tel que :

$$\int_a^b x^i\,s_{n+2}(x)\,v_{n+1}(x)\,\omega(x)\,dx = 0, \quad \text{pour} \quad i = 0, \ldots, n+1$$

Un tel polynôme s_{n+2} existe toujours mais, pour conduire à une formule de quadrature, ses racines doivent être réelles, dans $[a, b]$, distinctes entre elles et distinctes de celles de v_{n+1}. Ces restrictions ne sont vérifiées que pour certaines fonctions poids ω, comme celles qui correspondent aux polynômes de Tchebychev, de Gegenbauer et de Jacobi. La procédure de Kronrod nécessite le calcul de f en $2n+3$ points comme cela aurait été le cas en faisant appel à deux formules de Gauss, mais l'estimation de l'erreur ainsi obtenue est de bien meilleure qualité. Cette procédure s'étend aux approximants de Padé qui peuvent être considérés comme des quadratures de Gauss formelles (cf. dossier [AF 1 221]).

Sur la méthode de quadrature de Gauss et la procédure de Kronrod, on pourra consulter la référence [26].

4. Intégration des équations différentielles

4.1 Définition du problème

Soit $[a, b]$ un intervalle fermé de $\mathbb{R}$, soit f une application de $\mathbb{R} \times \mathbb{R}^p$ dans $\mathbb{R}^p$ et soit y une application différentiable de $\mathbb{R}$ dans $\mathbb{R}^p$. On appelle système différentiel du premier ordre la relation :

$$y'(x) = f(x, y(x))$$

On dit que y est solution de ce système sur $[a, b]$ si y vérifie cette relation pour tout x de $[a, b]$. On sait que la solution y d'un tel système dépend de constantes arbitraires. Ces constantes peuvent être déterminées par la connaissance de la solution en un point. On appelle **problème de Cauchy** le système différentiel précédent auquel on adjoint la **condition initiale** :

$$y(a) = y_0$$

avec y_0 vecteur donné de $\mathbb{R}^p$.

Le théorème **20** nous donne des conditions qui assurent l'existence et l'unicité de la solution de ce problème de Cauchy.

Théorème 20. Si f est définie et continue sur $[a, b] \times \mathbb{R}^p$ et s'il existe une constante L strictement positive telle que, pour tout $x \in [a, b]$ et pour tout u et tout $v \in \mathbb{R}^p$, on ait :

$$\|f(x, u) - f(x, v)\| \leqslant L\|u - v\|$$

alors le problème de Cauchy admet une solution unique quel que soit $y_0 \in \mathbb{R}^p$.

On dit alors que f satisfait à une **condition de Lipschitz** et L est sa **constante de Lipschitz**. Nous nous placerons toujours dans les conditions de ce théorème.

Si l'on prend un problème de Cauchy très simple [par exemple $y' = ay$ avec $y(0) = 1$], on sera capable d'obtenir sa solution exacte par les méthodes de l'analyse mathématique [$y(x) = \exp(ax)$ dans notre exemple]. On pourra alors calculer la valeur numérique exacte de cette solution en tout point de l'intervalle $[a, b]$ en remplaçant, dans cette formule, x par sa valeur numérique. Si le problème est plus compliqué, on ne sera en général plus capable d'obtenir la solution à l'aide des outils de l'analyse mathématique classique et il faudra faire appel à des méthodes d'analyse numérique qui nous fourniront seulement une valeur approchée de la solution en certains points $x_0, x_1, ..., x_N$ de l'intervalle d'intégration. Nous appellerons $y(x_n)$ la solution exacte (inconnue en général) en x_n et y_n la valeur approchée de la solution en x_n fournie par la méthode d'analyse numérique considérée. Les méthodes de calcul de $y_0, y_1, ..., y_N$ se divisent en deux classes selon la façon de calculer ces y_n :

— les **méthodes à pas séparés** (ou à un pas) dans lesquelles le calcul de y_{n+1} ne fait intervenir que y_n ;

— les **méthodes à pas liés** (ou à plusieurs pas) dans lesquelles le calcul de y_{n+1} fait intervenir $y_n, y_{n-1}, ..., y_{n-k}$ pour k fixé.

Soit $h = (b - a)/N$, le pas d'intégration. Nous allons prendre les abscisses x_n équidistantes dans $[a, b]$, c'est-à-dire :

$$x_n = a + nh \text{ pour } n = 0, ..., N$$

4.2 Méthodes à pas séparés

Dans une méthode à pas séparés, les y_n sont calculés par une relation de la forme :

$$y_{n+1} = y_n + h\phi(x_n, y_n, h) \text{ pour } n = 0, ..., N - 1$$

avec y_0 condition initiale donnée.

Les différentes méthodes se distinguent les unes des autres par le choix de ϕ.

4.2.1 Notions théoriques

Le problème fondamental est celui de la convergence : la solution approchée y_n converge-t-elle vers la solution exacte $y(x_n)$ en tout point de $[a, b]$ lorsque le pas h tend vers zéro ? Pour pouvoir répondre plus facilement à cette question, on est obligé d'introduire deux notions intermédiaires : la **consistance** et la **stabilité** (à ne pas confondre avec la stabilité numérique). Nous allons donc étudier cette question en nous restreignant au cas d'une seule équation, $p = 1$.

Définition 1. On dit qu'une méthode à pas séparés est consistante avec l'équation différentielle si, pour toute solution y de celle-ci, on a :

$$\lim_{h \to 0} \max_{0 \leqslant n \leqslant N-1} \left|(y(x_{n+1}) - y(x_n))/h - \phi(x_n, y(x_n), h)\right| = 0$$

On a le résultat suivant qui nous permet de savoir si une méthode à pas séparés est consistante.

Théorème 21. Une condition nécessaire et suffisante pour qu'une méthode à pas séparés soit consistante est que pour tout $x \in [a, b]$ et pour tout $u \in \mathbb{R}$ on ait : $\phi(x, u, 0) = f(x, u)$.

On voit que cette condition est particulièrement simple à vérifier dans la pratique.

Pour la stabilité, considérons les z_n donnés par :

$$z_0 \in \mathbb{R} \text{ arbitraire}$$

$$z_{n+1} = z_n + h[\phi(x_n, z_n, h) + \varepsilon_n] \text{ pour } n = 0, ..., N - 1$$

Définition 2. On dit qu'une méthode à pas séparés est stable s'il existe deux constantes positives M_0 et M_1 telles que, quels que soient h et $\varepsilon_0, ..., \varepsilon_{N-1}$, on ait :

$$\max_{0 \leqslant n \leqslant N} |y_n - z_n| \leqslant M_0|y_0 - z_0| + M_1 \max_{0 \leqslant n \leqslant N-1} |\varepsilon_n|$$

Le théorème **22** nous permet de savoir si une méthode à pas séparés est stable.

Théorème 22. S'il existe une constante positive M telle que, pour tout h suffisamment petit, pour tout $x \in [a, b]$ et pour tout u et tout $v \in \mathbb{R}$, on ait :

$$|\phi(x, u, h) - \phi(x, v, h)| \leqslant M|u - v|$$

alors la méthode à pas liés est stable.

Nous pouvons maintenant passer à l'étude de la **convergence** en commençant par la définition **3**.

Définition 3. On dit qu'une méthode à pas séparés est convergente si :

$$\lim_{h \to 0} \max_{0 \leqslant n \leqslant N} |y_n - y(x_n)| = 0$$

Les notions intermédiaires de consistance et de stabilité nous permettent maintenant de répondre à cette question grâce au théorème **23**.

Théorème 23.
Si une méthode à pas séparés est consistante et stable, alors elle est convergente.

La convergence est une notion qualitative. Il est évident que, dans la pratique, il serait très intéressant de savoir comment la quantité $\max_{0 \leqslant n \leqslant N} |y_n - y(x_n)|$ tend vers zéro avec h. Naturellement, puisqu'il était déjà trop difficile de répondre directement à la question de la convergence, il sera encore plus difficile de répondre directement à cette nouvelle question. Pour cela, nous allons retourner à la définition **1** de la consistance où il y a aussi une quantité qui tend vers zéro avec h et poser la définition **4**.

Définition 4. On dit qu'une méthode à pas séparés est d'**ordre** r s'il existe une constante positive K telle que :

$$\max_{0 \leqslant n \leqslant N-1} \left|(y(x_{n+1}) - y(x_n))/h - \phi(x_n, y(x_n), h)\right| \leqslant Kh^r$$

Grâce au théorème **24**, nous allons connaître le comportement de l'erreur.

Théorème 24. Si une méthode à pas séparés est d'ordre r et si elle vérifie la condition de stabilité du théorème **22**, alors :

$$\max_{0 \leqslant n \leqslant N} |y_n - y(x_n)| \leqslant K' h^r$$

avec $K' = K(\exp[(b-a)M] - 1)/M$

Au cours de la démonstration du théorème **24**, on obtient l'inégalité :

$$|e_{n+1}| \leqslant (1 + hM)|e_n| + Kh^{r+1}$$

avec $e_n = y_n - y(x_n)$.

Cela montre que l'erreur globale en x_{n+1}, e_{n+1}, provient de deux sources : l'erreur locale sur le passage de x_n à x_{n+1} (Kh^{r+1}) et les erreurs qui se sont accumulées depuis l'abscisse initiale a. Ainsi, en cumulant des erreurs locales en h^{r+1}, on obtient une erreur globale en h^r, ce qui est normal puisque :

$$Nh^{r+1} = (b-a)\, h^r$$

4.2.2 Méthodes de Runge-Kutta

La plus simple de toutes les méthodes à pas séparés est la **méthode d'Euler** qui consiste à prendre :

$$\phi(x, u, h) = f(x, u)$$

C'est une méthode du premier ordre et les résultats qu'elle fournit ne sont pas très précis.

On appelle **méthode de Runge-Kutta** une méthode où la fonction ϕ est définie par :

$$k_1 = f(x, u)$$
$$k_2 = f(x + \theta_2 h,\ u + a_{21} h k_1)$$
$$\vdots$$
$$k_m = f(x + \theta_m h,\ u + a_{m1} h k_1 + \ldots + a_{m,m-1} h k_{m-1})$$
$$\phi(x, u, h) = c_1 k_1 + c_2 k_2 + \ldots + c_m k_m$$

m s'appelle le rang et les constantes θ_i, a_{ij} et c_i sont en général choisies pour que la méthode soit d'ordre le plus élevé possible.

Il n'y a qu'une seule méthode de rang 1 et d'ordre 1, c'est la méthode d'Euler ; il y a une infinité de méthodes de rang m et d'ordre m pour $m = 2$, 3 et 4 ; il n'y a aucune méthode de rang 5 et d'ordre 5. Pour atteindre l'ordre 5, il faut aller au rang 6.

La plus utilisée des méthodes de Runge-Kutta est la suivante qui est de rang et d'ordre 4 :

$$k_1 = f(x, u)$$
$$k_2 = f(x + h/2,\ u + hk_1/2)$$
$$k_3 = f(x + h/2,\ u + hk_2/2)$$
$$k_4 = f(x + h,\ u + hk_3)$$
$$\phi(x, u, h) = (k_1 + 2k_2 + 2k_3 + k_4)/6$$

4.2.3 *A*-stabilité

On appelle raides (ou **stiff**) les équations différentielles dont la solution présente des variations très rapides. Cela pose de très sérieux problèmes de stabilité numérique à la plupart des méthodes numériques. De telles équations étant très fréquentes dans de nombreux domaines des mathématiques appliquées, il est important d'étudier ce phénomène et de savoir lui apporter une solution. Cette question a donné lieu à une très abondante littérature (cf. par exemple [28] [29]), mais on peut cependant l'étudier sur un problème test qui, bien que très simple, est suffisant pour analyser les difficultés.

Considérons l'équation différentielle :

$$y'(x) = -\lambda\, y(x), \quad y(0) = 1$$

avec λ nombre complexe dont la partie réelle est strictement positive.

La solution de ce problème est :

$$y(x) = \exp(-\lambda x)$$

et par conséquent $\lim_{x \to \infty} y(x) = 0$.

Il est naturellement souhaitable que la solution approchée reproduise ce comportement asymptotique, c'est-à-dire tende vers zéro à l'infini. C'est ce qu'exprime la définition **5**.

Définition 5. On dit qu'une méthode d'intégration numérique est ***A*-stable** si, lorsque l'on intègre le problème de Cauchy $y' = -\lambda y$, $\lim_{n \to \infty} y_n = 0$ quel que soit le nombre complexe $h\lambda$ dont la partie réelle est strictement positive.

Si, par exemple, nous appliquons la méthode d'Euler à ce problème, nous obtenons :

$$y_{n+1} = (1 - h\lambda)\, y_n$$

c'est-à-dire :

$$y_n = (1 - h\lambda)^n$$

Lorsque n tend vers l'infini, y_n tend donc vers zéro si et seulement si le nombre complexe $1 - h\lambda$ est de module strictement inférieur à 1. Cette condition n'est évidemment pas satisfaite pour tous les nombres $h\lambda$ dont la partie réelle est strictement positive : la méthode d'Euler n'est pas *A*-stable.

Par conséquent, si l'on choisit un pas h tel que $|1 - h\lambda| > 1$, y_n tendra vers l'infini avec n au lieu de tendre vers zéro comme la solution exacte. On voit donc que c'est un ennui sérieux auquel il faut absolument remédier. On doit naturellement choisir une valeur du pas h telle que la condition $|1 - h\lambda| < 1$ soit satisfaite.

Exemple : si $\lambda = 10\,000$, il faudra prendre $h < 2 \times 10^{-4}$ et le temps de calcul sera très long alors que, puisque la solution est très rapidement presque nulle, on pouvait espérer prendre un pas relativement grand et avoir un temps de calcul court.

On démontre de même qu'aucune des méthodes de Runge-Kutta vues plus haut n'est *A*-stable. Il faut donc s'orienter vers un autre type de méthodes. Toutes les méthodes précédentes étaient **explicites** car, dès que y_n était connu, la relation :

$$y_{n+1} = y_n + h\phi(x_n, y_n, h)$$

nous permettait de calculer explicitement la valeur de y_{n+1}. Lorsque ce n'est pas le cas, la méthode est dite **implicite** ; par exemple, on peut avoir :

$$y_{n+1} = y_n + h\phi(x_n, y_n, x_{n+1}, y_{n+1}, h)$$

y_{n+1} est alors donné implicitement comme solution de cette équation (ou de ce système d'équations) qui peut être non linéaire. Pour calculer y_{n+1}, il faut utiliser les méthodes itératives étudiées dans le dossier [AF 1 221], § 1. La plus simple de toutes ces méthodes est la méthode d'Euler implicite :

$$y_{n+1} = y_n + hf(x_{n+1}, y_{n+1})$$

Appliquée à notre problème test, elle fournit :

$$y_{n+1} = y_n / (1 + h\lambda)$$

c'est-à-dire :

$$y_n = (1 + h\lambda)^{-n}$$

Par conséquent, y_n tend vers zéro quel que soit $h\lambda$ dont la partie réelle est strictement positive : la méthode d'Euler implicite est *A*-stable.

Il existe des méthodes de Runge-Kutta implicites définies par :

$$k_1 = f(x + \theta_1 h,\ u + a_{11} hk_1 + \ldots + a_{1m} hk_m)$$

$$k_2 = f(x + \theta_2 h,\ u + a_{21} hk_1 + \ldots + a_{2m} hk_m)$$

$$\vdots$$

$$k_m = f(x + \theta_m h,\ u + a_{m1} hk_1 + \ldots + a_{mm} hk_m)$$

$$\phi(x, u, h) = c_1 k_1 + c_2 k_2 + \ldots + c_m k_m$$

Si $a_{ij} = 0$ pour $j > i$, on dit que la méthode est **semi-implicite** ; le système non linéaire à résoudre à chaque pas est alors plus simple.

4.2.4 Mise en œuvre

Lors de la mise en œuvre effective d'une méthode d'intégration numérique, de nombreux autres problèmes pratiques se posent. Le premier d'entre eux concerne le choix du pas h pour obtenir la précision désirée. Ensuite, si la solution varie beaucoup dans l'intervalle d'intégration, on peut être amené à changer la valeur du pas soit pour le diminuer si la précision atteinte n'est pas suffisante, soit pour l'augmenter si elle est beaucoup trop petite. Avec les méthodes à pas séparés, cela ne pose aucune difficulté car il suffit de remplacer h par h_n et de prendre :

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

Mais, pour effectuer ces changements de pas à bon escient, il faut être capable de contrôler l'erreur globale. Cela se fait généralement en programmant simultanément deux méthodes : une d'ordre r et une d'ordre $r+1$. En chaque point x_n la différence entre les deux valeurs approchées ainsi obtenues est une bonne estimation de l'erreur globale sur la méthode d'ordre r. Il faut aussi tenir compte de la propagation des erreurs numériques dues à l'arithmétique de l'ordinateur. Plus le pas h est petit et plus l'erreur de méthode est faible. Mais plus le pas h est petit et plus il faut faire de calculs pour parcourir l'intervalle d'intégration, donc les erreurs dues à l'arithmétique de l'ordinateur augmentent lorsque h diminue. L'erreur totale, qui est la somme de l'erreur de méthode et de l'erreur due à l'arithmétique de l'ordinateur, passe donc par un minimum en fonction de h ; il existe une valeur optimale de h car il faut faire un compromis entre une erreur de méthode petite et une erreur d'arithmétique importante ou l'inverse. Ces erreurs numériques sont plus difficiles à contrôler que l'erreur de méthode.

Pour estimer l'erreur dans les méthodes de Runge-Kutta, on se sert de méthodes de Runge-Kutta **emboîtées**. Cette technique consiste à utiliser simultanément une méthode de Runge-Kutta d'ordre s et une autre d'ordre $s' < s$ avec les mêmes k_i afin d'éviter de nouvelles évaluations de f. Ainsi nous avons :

$$y_{n+1} = y_n + h(c_1 k_1(x_n, y_n) + \ldots + c_r k_r(x_n, y_n)), \text{ ordre } s$$

$$y^*_{n+1} = y_n + h(c^*_1 k_1(x_n, y_n) + \ldots + c^*_r k_r(x_n, y_n)), \text{ ordre } s' < s$$

Un terme supplémentaire $hc^*_{r+1} f(x_{n+1}, y_{n+1})$ peut être ajouté à la seconde formule pour plus de flexibilité. La différence $y^*_{n+1} - y_{n+1}$ est une approximation de l'erreur locale sur y^*_{n+1}. Les coefficients c^*_i s'obtiennent en développant les k_i et $f(x_{n+1}, y_{n+1})$ en série de Taylor et en comparant à la solution exacte. Puisque les c_i et les a_{ij} sont déjà connus, les c^*_i sont solution d'un système d'équations linéaires.

Exemple : considérons la méthode de Runge-Kutta d'ordre 4 :

$$k_1(x, u) = f(x, u)$$

$$k_2(x, u) = f(x + h/3,\ u + hk_1/3)$$

$$k_3(x, u) = f(x + 2h/3,\ u - hk_1/3 + hk_2)$$

$$k_4(x, u) = f(x + h,\ u + hk_1 - hk_2 + hk_3)$$

avec $\Phi(x, u, h) = (k_1 + 3k_2 + 3k_3 + k_4)/8$.

La méthode emboîtée correspondante est d'ordre 3 et elle est donnée par :

$$y^*_{n+1} = y_n + \frac{h}{24}[2k_1 + 12k_2 + 6k_3 + 4f(x_{n+1}, y_{n+1})]$$

L'erreur locale sur y^*_{n+1} est estimée par la formule :

$$y^*_{n+1} - y_{n+1} = \frac{h}{24}[-k_1 + 3k_2 - 3k_3 - 3k_4 + 4f(x_{n+1}, y_{n+1})]$$

Les méthodes emboîtées permettent de choisir le pas à utiliser afin d'obtenir une précision désirée. On a, avec la valeur de h utilisée :

$$y_{n+1} - y^*_{n+1} = [y_{n+1} - \tilde{y}(x_{n+1})] - [\tilde{y}(x_{n+1}) - y^*_{n+1}]$$
$$= O(h^{s+1}) + O(h^{s'+1}) \approx Ch^{s'+1},\ C \in \mathbb{R}^p$$

avec $\tilde{y}(x_{n+1})$ solution exacte en x_{n+1} telle que $\tilde{y}(x_n) = y_n$.

La valeur optimale de h, dénotée $\hat{h}$, est choisie de sorte que $\eta \approx \|C\| \hat{h}^{s'+1}$, où η est la tolérance prescrite pour l'erreur locale. En éliminant $\|C\|$ entre ces deux relations, on trouve :

$$\hat{h} = 0{,}9h(\eta / \| y_{n+1} - y^*_{n+1} \|)^{1/(s'+1)}$$

où le facteur 0,9 a été inclus pour des raisons de sécurité. Dans cette formule, la norme utilisée est, en général, un mélange d'erreur relative et d'erreur absolue :

$$\| y_{n+1} - y^*_{n+1} \| = \sqrt{\frac{1}{p} \sum_{i=1}^{p} \left(\frac{y_{n+1,i} - y^*_{n+1,i}}{d_i} \right)^2}$$

avec $d_i = 1 + \max(|y_{n+1,i}|, |y^*_{n+1,i}|)$,

$y_{n+1,i}$ et $y^*_{n+1,i}$ respectivement les composantes des vecteurs y_{n+1} et y^*_{n+1}.

De plus, pour éviter de grandes variations dans le pas, on ajoute souvent une restriction du type $0{,}2h \leqslant \hat{h} \leqslant 5h$. Le pas n'est modifié que si $\hat{h}$ ne vérifie pas cette condition.

4.3 Méthodes à pas liés

Dans ces méthodes, les y_n sont calculés récursivement par une relation de la forme :

$$\alpha_k y_{n+k} + \ldots + \alpha_0 y_n = h[\beta_k f_{n+k} + \ldots + \beta_0 f_n] \text{ pour } n = 0, \ldots, N - k$$

avec $f_i = f(x_i, y_i)$.

On voit que lorsque $n = 0$ il faut, pour calculer y_k, connaître $y_0, y_1, \ldots, y_{k-1}$. y_0 est notre condition initiale : elle est donc connue. $y_1, \ldots, y_{k-1}$ sont des valeurs approchées de $y(x_1), \ldots, y(x_{k-1})$. Elles

devront donc être calculées par une méthode à pas séparés ; une méthode à pas liés ne démarre pas toute seule. Nous reviendrons sur ce point plus tard (théorème **29**). On voit aussi que, si $\beta_k = 0$, on a une méthode à pas liés explicite tandis que si $\beta_k \neq 0$ elle est implicite. Dans le cas d'une méthode implicite, il faut que la solution y_{n+k} de la relation précédente existe. En utilisant les théorèmes de points fixes démontré dans le dossier [AF 1 221], § 1, on démontre le théorème **25**.

Théorème 25. Si $\beta_k \neq 0$, l'équation implicite précédente a une solution unique pour tout n si :

$$h < \frac{1}{L}\left|\frac{\alpha_k}{\beta_k}\right|$$

avec L constante de Lipschitz de f (cf. § 4.1).

On voit que, si $\beta_k = 0$, la condition précédente n'impose aucune restriction sur h, ce qui est normal puisque la méthode est explicite et que y_{n+k} existe toujours.

La supériorité des méthodes à pas liés sur les méthodes à pas séparés réside dans le fait qu'elles ne nécessitent pas d'évaluations de f en des points intermédiaires (sauf au démarrage). Par rapport à une méthode de Runge-Kutta du même ordre, le temps de calcul est donc réduit dans une proportion importante.

Il existe de nombreux livres entièrement consacrés aux méthodes numériques pour les équations différentielles [12] [16] [28] [29].

4.3.1 Notions théoriques

Comme pour les méthodes à pas séparés, il nous faut maintenant nous occuper de la **consistance** et de la **stabilité**. Nous avons la définition **6**.

Définition 6. On dit qu'une méthode à pas liés est **consistante** avec l'équation différentielle si, pour toute solution y de celle-ci, on a :

$$\lim_{h\to 0} \max_{0\leqslant n\leqslant N-k} \left|\frac{1}{h}\sum_{i=0}^{k}\alpha_i y(x_{n+i}) - \sum_{i=0}^{k}\beta_i f(x_{n+i}, y(x_{n+i}))\right| = 0$$

Posons :

$$\alpha(t) = \alpha_0 + \alpha_1 t + \ldots + \alpha_k t^k$$

$$\beta(t) = \beta_0 + \beta_1 t + \ldots + \beta_k t^k$$

On a le théorème **26**.

Théorème 26. Une condition nécessaire et suffisante pour qu'une méthode à pas liés soit consistante est que $\alpha(1) = 0$ et que $\alpha'(1) = \beta(1)$.

Pour la stabilité, on considère les z_n donnés par :

$$z_0, z_1, \ldots, z_{k-1} \in \mathbb{R} \text{ arbitraires}$$

$$\alpha_k z_{n+k} + \ldots + \alpha_0 z_n = h[\beta_n f(x_{n+k}, z_{n+k}) + \ldots + \beta_0 f(x_n, z_n) + \varepsilon_n]$$

pour $n = 0, \ldots, N-k$.

Définition 7. On dit qu'une méthode à pas liés est **stable** s'il existe deux constantes M_0 et M_1 telles que, quels que soient h et $\varepsilon_0, \ldots, \varepsilon_{N-k}$, on ait :

$$\max_{0\leqslant n\leqslant N}|y_n - z_n| \leqslant M_0 \max_{0\leqslant i\leqslant k-1}|y_i - z_i| + M_1 \max_{0\leqslant n\leqslant N-k}|\varepsilon_n|$$

On voit que la différence avec le cas des méthodes à pas séparés provient du fait que les méthodes à pas liés ne démarrent pas avec la seule connaissance de y_0. On a le théorème **40**.

Théorème 27. Une condition nécessaire et suffisante pour qu'une méthode à pas liés soit stable est que toutes les racines du polynôme α soient de module inférieur ou égal à 1 et que les racines de module 1 soient des racines simples.

On voit que, contrairement au cas des méthodes à pas séparés, on a maintenant une condition nécessaire et suffisante, ce qui nous permet d'obtenir les résultats du théorème **28**. Auparavant donnons la définition **8**.

Définition 8. On dit qu'une méthode à pas liés est **convergente** si :

$$\lim_{h\to 0} \max_{0\leqslant n\leqslant N}|y_n - y(x_n)| = 0$$

lorsque $\lim_{h\to 0} y_i = y_0$ pour $i = 0, 1, \ldots, k-1$.

Il est tout à fait normal d'être obligé d'imposer, dans cette définition, que les k valeurs approchées $y_0, y_1, \ldots, y_{k-1}$ convergent vers la valeur exacte y_0 lorsque h tend vers zéro.

Théorème 28. Une condition nécessaire et suffisante pour qu'une méthode à pas liés soit convergente est qu'elle soit consistante et stable.

Pour l'**ordre** nous repartons, comme dans le cas des méthodes à pas séparés (§ 4.2.1), de la définition **6** de la consistance.

Définition 9. On dit qu'une méthode à pas liés est d'**ordre** r s'il existe une constante positive K telle que :

$$\max_{0\leqslant n\leqslant N-k}\left|\frac{1}{h}\sum_{i=0}^{k}\alpha_i y(x_{n+i}) - \sum_{i=0}^{k}\beta_i f(x_{n+i}, y(x_{n+i}))\right| \leqslant Kh^r$$

Théorème 29. Si une méthode à pas liés est d'ordre r, si elle est stable et si $\max_{0\leqslant i\leqslant k-1}|y_i - y(x_i)| = O(h^r)$, alors :

$$\max_{0\leqslant n\leqslant N}|y_n - y(x_n)| = O(h^r)$$

Nous avons vu au début du paragraphe 4.3 qu'une méthode à pas liés ne démarrait pas toute seule, mais qu'il fallait utiliser une méthode à pas séparés pour calculer $y_0, y_1, \ldots, y_{k-1}$. Le théorème **29** nous dit que ces valeurs doivent aussi être calculées avec une méthode d'ordre r, ce qui est parfaitement logique. Si l'on prenait une méthode à pas séparés d'ordre plus faible, $y_0, \ldots, y_{k-1}$ ne seraient pas assez précis ; d'un autre côté, il ne sert à rien de les obtenir avec une trop grande précision qui serait ensuite perdue.

On a également le théorème **30**.

Théorème 30. Une condition nécessaire et suffisante pour qu'une méthode à pas liés soit consistante est qu'elle soit au moins d'ordre un.

4.3.2 Méthodes d'Adams

Les plus utilisées des méthodes à pas liés sont les **méthodes d'Adams** qui sont de la forme :

$$y_q - y_j = h \sum_{i=0}^{k} \beta_i f_{n+i} \quad \text{pour } j \leqslant q$$

Il en existe plusieurs types selon les valeurs de j et de q. Les coefficients β_i dépendent de k. On a ainsi les méthodes suivantes :

■ **Méthodes d'Adams-Bashforth :**

$$y_{n+2} = y_{n+1} + \frac{h}{2}(3f_{n+1} - f_n) \quad \text{ordre 2}$$

$$y_{n+3} = y_{n+2} + \frac{h}{12}(23f_{n+2} - 16f_{n+1} + 5f_n) \quad \text{ordre 3}$$

$$y_{n+4} = y_{n+3} + \frac{h}{24}(55f_{n+3} - 59f_{n+2} + 37f_{n+1} - 9f_n) \quad \text{ordre 4}$$

■ **Méthodes d'Adams-Moulton :**

$$y_{n+1} = y_n + \frac{h}{2}(f_{n+1} + f_n) \quad \text{ordre 2}$$

$$y_{n+2} = y_{n+1} + \frac{h}{12}(5f_{n+2} + 8f_{n+1} - f_n) \quad \text{ordre 3}$$

$$y_{n+3} = y_{n+2} + \frac{h}{24}(9f_{n+3} + 19f_{n+2} - 5f_{n+1} + f_n) \quad \text{ordre 4}$$

■ **Méthodes de Nyström :**

$$y_{n+2} = y_n + 2hf_{n+1} \quad \text{ordre 2}$$

$$y_{n+3} = y_{n+1} + \frac{h}{3}(7f_{n+2} - 2f_{n+1} + f_n) \quad \text{ordre 3}$$

$$y_{n+4} = y_{n+2} + \frac{h}{3}(8f_{n+3} - 5f_{n+2} + 4f_{n+1} - f_n) \quad \text{ordre 4}$$

■ **Méthodes de Milne-Simpson :**

$$y_{n+2} = y_n + 2hf_{n+2} \quad \text{ordre 2}$$

$$y_{n+3} = y_{n+1} + \frac{h}{3}(f_{n+3} + 4f_{n+2} + f_{n+1}) \quad \text{ordre 3}$$

Il a été démontré qu'aucune méthode à pas liés explicite ne pouvait être A-stable. On est donc, là encore, obligé de s'orienter soit vers les méthodes implicites, soit vers les méthodes explicites non linéaires.

Les questions posées par la mise en œuvre d'une méthode à pas liés sont similaires à celles évoquées pour les méthodes à pas séparés. Lorsque l'on veut changer de pas en cours d'intégration, une difficulté supplémentaire se présente du fait que les méthodes à pas liés demandent impérativement un pas constant. Supposons que, arrivé à l'abscisse x_n, nous désirions changer la valeur du pas. Cela est possible en utilisant de nouveau la méthode à pas séparés du démarrage sur k pas pour calculer $y_{n+1}, ..., y_{n+k-1}$ avec la nouvelle valeur du pas. Après, on continuera avec la méthode à pas liés. Le contrôle de l'erreur s'effectue, comme dans le cas des méthodes à pas séparés, en utilisant simultanément une méthode d'ordre r et une méthode d'ordre $r+1$.

4.3.3 Méthodes de prédiction-correction

Si l'on désire utiliser une méthode à pas liés implicite, il faut normalement à chaque pas résoudre l'équation implicite pour obtenir y_{n+k}. Pour cela, il faut mettre en œuvre la méthode des approximations successives comme cela sera expliqué dans le dossier [AF 1 221] § 1, ce qui risque d'être coûteux en temps de calcul. Cependant, si le point de départ de la méthode des approximations successives est suffisamment bon, on peut se contenter de ne faire qu'une seule itération : c'est ce que l'on appelle une **méthode de prédiction-correction**. On programme simultanément une méthode à pas liés explicite, qui fournit la valeur de départ (c'est le **prédicteur**), et une méthode implicite avec laquelle on ne fait qu'une itération (c'est le **correcteur**). On a ainsi le schéma suivant :

$$\alpha_k^* y_{n+k}^* + \alpha_{k-1}^* y_{n+k-1} + \ ... \ + \alpha_0^* y_n = h[\beta_{k-1}^* f_{n+k-1} + \ ... \ + \beta_0 f_n]$$

$$\alpha_k y_{n+k} + \alpha_{k-1} y_{n+k-1} + \ ... \ + \alpha_0 y_n = h[\beta_k f(x_{n+k}, y_{n+k}^*) + \beta_{k-1} f_{n+k-1} + \ ... \ + \beta_0 f_n]$$

La première relation nous sert à calculer la valeur prédite y_{n+k}^* à partir de $y_n, ..., y_{n+k-1}$ et la seconde relation nous donne y_{n+k} à partir de $y_n, ..., y_{n+k-1}$ et y_{n+k}^*.

On a le théorème **31**.

Théorème 31. Si le prédicteur est d'ordre $r-1$ et si le correcteur est d'ordre r, alors :

$$\max_{0 \leqslant n \leqslant N} |y_n - y(x_n)| = O(h^r)$$

à condition que :

$$\max_{0 \leqslant i \leqslant k-1} |y_i - y(x_i)| = O(h^r)$$

On voit que le correcteur a amélioré le résultat du prédicteur et qu'il est inutile de prendre un prédicteur d'ordre supérieur car la précision serait ensuite perdue.

L'avantage que possèdent les méthodes de prédiction-correction est que le contrôle de l'erreur globale est facile à effectuer. En effet, la différence $|y_{n+k}^* - y_{n+k}|$ entre les valeurs prédite et corrigée est une bonne estimation de l'erreur globale.

4.3.4 Méthodes de différenciation rétrograde

Ces méthodes sont obtenues par une approche complémentaire de celle des méthodes d'Adams. Elles consistent à approximer la dérivée première de y, c'est-à-dire f, par la dérivée du polynôme d'interpolation q qui satisfait :

$$q\ (x_{n-i+1}) = y_{n-i+1} \text{ pour } i = 0, ..., k$$

et

$$q'(x_{n+1}) = f(x_{n+1}, q\ (x_{n+1}))$$

On obtient ainsi les méthodes implicites suivantes d'ordre respectif 2, 3, 4, 5 et 6

$$y_{n+2} = \frac{1}{3}(4y_{n+1} - y_n) + \frac{2h}{3} f_{n+2} \quad \text{ordre 2}$$

$$y_{n+3} = \frac{1}{11}(18y_{n+2} - 9y_{n+1} + 2y_n) + \frac{6h}{11} f_{n+3} \quad \text{ordre 3}$$

$$y_{n+4} = \frac{1}{25}(48y_{n+3} - 36y_{n+2} + 16y_{n+1} - 3y_n) + \frac{12h}{25} f_{n+4} \quad \text{ordre 4}$$

$$y_{n+5} = \frac{1}{137}(300y_{n+4} - 300y_{n+3} + 200y_{n+2} - 75y_{n+1} + 12y_n) + \frac{60h}{137} f_{n+5} \quad \text{ordre 5}$$

$$y_{n+6} = \frac{1}{147}(360y_{n+5} - 450y_{n+4} + 400y_{n+3} - 225y_{n+2} + 72y_{n+1} - 10y_n) + \frac{60h}{147} f_{n+6} \quad \text{ordre 6}$$

L'un des avantages de ces méthodes est que le changement de pas y est facilité. Ainsi, en exprimant le polynôme q à l'aide des différences divisées, elles peuvent s'écrire (h_n a été conservé de part et d'autre pour une raison d'uniformité avec les autres formules) :

$$\sum_{j=1}^{k} h_n \prod_{i=1}^{j-1} (x_{n+1} - x_{n+1-i})[x_{n+1}, ..., x_{n-j+1}]_y = h_n f(x_{n+1}, y_{n+1})$$

avec $[x_{n+1}, ..., x_{n-j+1}]_y$ différence divisée de y aux points entre crochets (cf. § 2.1).

4.4 Problèmes aux limites

Considérons le cas d'un système d'équations différentielles. Jusqu'à présent, nous avons considéré le cas d'un problème de Cauchy, c'est-à-dire le cas où les constantes d'intégration étaient déterminées par la donnée des conditions initiales $y(a) = y_0$. Mais la situation peut également se présenter de façon différente : il se peut que l'on connaisse certaines composantes du vecteur y à l'abscisse initiale a et les autres composantes de y à l'abscisse b. C'est ce que l'on appelle un problème de **conditions aux limites**. On peut transformer ce problème en un problème de Cauchy en recherchant les conditions initiales manquantes telles que les conditions connues en b soient satisfaites. On a donc à résoudre un système d'équations non linéaires dont les inconnues sont les conditions initiales manquantes. De façon plus générale, il faut trouver les conditions initiales à prendre de sorte qu'un certain système de p équations non linéaires soit satisfait :

$$g(y(x_1), y(x_2), ..., y(x_q)) = 0$$

Comme $y(x_1), y(x_2), ..., y(x_q)$ dépendent des conditions initiales inconnues y_0 on a bien à résoudre un système de p équations à p inconnues.

Pour résoudre un tel système, on utilise une méthode d'intégration numérique du système différentiel, couplée à une méthode itérative pour résoudre le système d'équations non linéaires. On part d'une valeur arbitraire des conditions initiales manquantes et l'on itère sur cette valeur jusqu'à ce que les équations non linéaires soient satisfaites à la précision demandée. Naturellement, dans les équations g, les valeurs exactes $y(x_1), ..., y(x_q)$ de la solution seront remplacées par les valeurs approchées $y_1, ..., y_q$ obtenues par intégration numérique. C'est ce que l'on appelle une **méthode de tir**. Du point de vue pratique, il faut choisir une méthode de résolution du système non linéaire qui ne fasse pas intervenir les dérivées des fonctions g par rapport aux composantes de y_0 ; on pourra, par exemple, utiliser les méthodes quasi-Newton (cf. dossier [AF 1 221], § 1.7.

5. Approximation

La théorie de l'approximation constitue une partie fondamentale de l'analyse numérique. De nombreuses questions étudiées dans les paragraphes précédents peuvent se formuler dans le cadre de cette théorie : approximation d'une fonction par un polynôme d'interpolation, d'une intégrale par une somme finie, de la solution d'une équation différentielle, etc.

Nous allons donner les notions de base de la théorie de l'approximation ; elles feront appel à un minimum de connaissances en analyse fonctionnelle, mais nous n'entrerons pas dans les détails qui pourront être étudiés dans [19] [31] ; nous nous intéresserons plus à des exemples pour montrer la richesse de cette théorie et la puissance de l'outil que constitue l'analyse fonctionnelle. Le premier mathématicien à attirer l'attention sur l'intérêt que pouvait présenter l'analyse fonctionnelle dans le développement de l'analyse numérique fut le Russe L.V. Kantorovich en 1948.

Les idées et les méthodes de l'analyse fonctionnelle jouent un rôle important, voire fondamental, en analyse numérique quand les mathématiques du problème dépendent beaucoup de l'analyse fonctionnelle (par exemple, dans les équations aux dérivées partielles), lorsque l'on cherche à traiter d'un seul coup une classe entière de méthodes (par exemple, les méthodes de quadrature de type interpolation) ou encore à démontrer l'existence de méthodes numériques présentant certaines caractéristiques. L'analyse fonctionnelle apporte alors une simplification importante ; elle joue, par contre, un rôle moins fondamental pour étudier un algorithme précis ou pour résoudre un problème spécifique et ne sera d'aucune utilité en ce qui concerne l'implémentation d'une méthode sur ordinateur. Actuellement, l'analyse fonctionnelle est un outil essentiel pour comprendre bon nombre de méthodes d'analyse numérique, mais on peut également trouver de nouvelles méthodes numériques sans son secours.

Réciproquement, certains algorithmes découlent directement des méthodes de l'analyse fonctionnelle. On doit la considérer comme un outil privilégié pour résoudre certains problèmes d'analyse numérique, mais on ne peut pas, inversement, considérer l'analyse numérique comme une branche de l'analyse fonctionnelle ; ce sont deux domaines différents mais complémentaires puisque l'analyse numérique peut suggérer l'étude de nouvelles questions d'analyse fonctionnelle et que, inversement, celle-ci est le pivot de certains sujets d'analyse numérique.

5.1 Meilleure approximation. Théorie

Nous allons maintenant étudier le problème de la recherche de la meilleure approximation qui peut se formuler ainsi : soit H un espace vectoriel normé et C une partie de H ; on dit que $\overline{g} \in C$ est la meilleure approximation de $f \in H$ dans C si, pour tout $g \in C$, $\|f - \overline{g}\| \leqslant \|f - g\|$. On a le résultat d'existence donné par le théorème **32**.

Théorème 32. Si C est soit une partie compacte de H, soit un sous-espace vectoriel de dimension finie de H, alors, quel que soit $f \in H$, il existe au moins une meilleure approximation $\overline{g}$ de f dans C.

Nous allons maintenant supposer que H est muni d'un produit scalaire ; on dira que c'est un espace préhilbertien et l'on posera $\|.\|^2 = (.,.)$. On a le résultat de caractérisation donné par le théorème **33**.

Théorème 33. Si H est un espace préhilbertien et si C est un sous-espace vectoriel de dimension finie de H, alors une condition nécessaire et suffisante pour que $\overline{g} \in C$ soit meilleure approximation de f dans C est que pour tout $g \in C$:

$$(f - \overline{g}, g - \overline{g}) \leqslant 0$$

Finalement, on a le théorème **34**.

Théorème 34. Sous les hypothèses du théorème **33**, la meilleure approximation $\overline{g}$ de f dans C est unique.

Après avoir démontré l'existence et l'unicité de $\overline{g}$ et l'avoir caractérisé, nous allons passer à la question qui intéresse l'analyste numéricien : la construction effective de $\overline{g}$.

Soit $g_1, g_2, ..., g_n$ des éléments linéairement indépendants de H et soit C le sous-espace engendré par leurs combinaisons linéaires finies. On a d'abord le théorème **35**.

Théorème 35. Une condition nécessaire et suffisante pour que $g_1, ..., g_n$ soient linéairement indépendants est que leur **déterminant de Gram** :

$$D(g_1, ..., g_n) = \begin{vmatrix} (g_1, g_1) ..... (g_1, g_n) \\ ..... \\ (g_n, g_1) ..... (g_n, g_n) \end{vmatrix}$$

soit différent de zéro.

Puisque $\overline{g} \in C$ et que C est engendré par $g_1, ..., g_n$, $\overline{g}$ peut s'écrire comme combinaison linéaire des g_i avec des coefficients donnés par le théorème **36**.

Théorème 36. La meilleure approximation $\overline{g}$ de $f(\notin C)$ dans C s'écrit :

$$\overline{g} = a_1 g_1 + ... + a_n g_n$$

où les a_i sont solutions du système :

$$\sum_{i=1}^{n} a_i (g_i, g_j) = (f, g_j) \quad \text{pour } j = 1, ..., n$$

et l'on a :

$$\| f - \overline{g} \|^2 = D(g_1, ..., g_n, f)/D(g_1, ..., g_n)$$

On voit que la solution de ce système est immédiate dans le cas où $(g_i, g_j) = 0$ pour $j \neq i$. Cela montre l'importance des systèmes orthogonaux en approximation, par exemple les polynômes orthogonaux en approximation polynomiale. Si, de plus, le système est normé, c'est-à-dire si $(g_i, g_j) = 1$, alors on a :

$$a_i = (f, g_i) \quad \text{et} \quad \| f - \overline{g} \|^2 = \| f \|^2 - \sum_{i=1}^{n} a_i^2$$

On dit alors que les a_i sont les **coefficients de Fourier** de f relativement au système orthonormé $g_1, ..., g_n$ et $\overline{g}$ est appelé **somme de Fourier**.

Étant donné une base quelconque $g_1, ..., g_n$ de C, il est toujours possible de la transformer en une base orthonormée $h_1, ..., h_n$ à l'aide du procédé d'**orthonormalisation de Gram-Schmidt** :

$$h_1 = g_1 / \| g_1 \|$$

$$u_i = g_i - \sum_{j=1}^{i-1} (g_i, h_j) h_j$$

$$h_i = u_i / \| u_i \| \text{ pour } i = 2, ..., n$$

Pour des raisons de stabilité numérique, on lui préfère le **procédé de Gram-Schmidt modifié**. Il consiste à poser $h_i^{(1)} = g_i$ pour $i = 1, ..., n$, puis, pour $i = 1, ..., n$, à calculer $h_i = h_i^{(i)} / \| h_i^{(i)} \|_2$ et, pour $j = i+1, ..., n$, $h_j^{(i+1)} = h_j^{(i)} - (h_i, h_j^{(i)}) h_i$.

5.2 Meilleure approximation. Exemples

5.2.1 Approximation au sens des moindres carrés

Il faut distinguer le cas continu et le cas discret.

■ Soit $C[a, b]$ l'espace des fonctions continues sur $[a, b]$ et soit ω une fonction poids strictement positive sur $]a, b[$. Nous supposerons que $\int_a^b f(x)\, \omega(x)\, dx$ existe pour toute fonction f de $C[a, b]$ et **nous définirons le produit scalaire** par :

$$(f, g) = \int_a^b f(x)\, g(x)\, \omega(x)\, dx$$

Soit $g_0, ..., g_n$ des éléments linéairement indépendants de H et soit C le sous-espace vectoriel qu'ils engendrent. D'après ce qui précède, on a :

$$\overline{g} = a_0 g_0 + ... + a_n g_n$$

où les a_i sont solutions du système :

$$\sum_{i=0}^{n} a_i \int_a^b g_i(x)\, g_j(x)\, \omega(x)\, dx = \int_a^b f(x)\, g_j(x)\, \omega(x)\, dx$$

pour $j = 0, ..., n$

• Si $g_i(x) = x^i$, on retrouve l'**approximation polynomiale**, ce qui montre l'importance des familles de polynômes orthogonaux étudiées au paragraphe 3.4.

• Si $g_0(x) = 1$, $g_1(x) = \sin x$; $g_2(x) = \cos x$, ..., $g_{2n-1}(x) = \sin nx$; $g_{2n}(x) = \cos nx$; $a = -\pi$ et $b = \pi$

on est dans le cas de l'**approximation trigonométrique**.

$g_0, ..., g_{2n}$ forment un système orthonormé et :

$$\overline{g}(x) = a_0 + \sum_{k=1}^{n} (a_k \cos kx + b_k \sin kx)$$

■ Étudions maintenant l'**approximation discrète au sens des moindres carrés**. Définissons le produit scalaire par :

$$(f, g) = \sum_{i=0}^{N} f(x_i)\, g(x_i)\, \omega(x_i)$$

où les x_i sont distincts les uns des autres et $\omega(x_i) > 0$.

Soit C le sous-espace engendré par des éléments $g_0, ..., g_n$ linéairement indépendants. On aura $\overline{g} = a_0 g_0 + ... + a_n g_n$ avec les a_i solutions du système :

$$\sum_{i=0}^{n} a_i \sum_{k=0}^{N} g_i(x_k)\, g_j(x_k)\, \omega(x_k) = \sum_{k=0}^{N} f(x_k)\, g_j(x_k)\, \omega(x_k)$$

pour $j = 0, ..., n$

Posons :

$$b = (\sqrt{\omega(x_0)}\, f(x_0), ..., \sqrt{\omega(x_N)}\, f(x_N))^T$$

$$a = (a_0, ..., a_n)^T$$

$$A = \begin{pmatrix} \sqrt{\omega(x_0)}\, g_0(x_0) ..... \sqrt{\omega(x_0)}\, g_n(x_0) \\ ..... \\ \sqrt{\omega(x_N)}\, g_0(x_N) ..... \sqrt{\omega(x_N)}\, g_n(x_N) \end{pmatrix}$$

Le système précédent s'écrit $A^T Aa = A^T b$. La matrice $A^T A$ étant symétrique définie positive, la méthode de Cholesky cf. dossier [AF 1 221], § 2.1.3 est particulièrement bien adaptée à la résolution de ce système.

Si $g_i(x) = x^i$ on obtient l'**approximation polynomiale discrète**. Si $N = n$, on retrouve l'interpolation polynomiale (§ 2.1) et l'on a $\|f - \overline{g}\| = 0$.

■ Prenons maintenant pour H l'espace vectoriel $H^q[a, b]$ $(q \geqslant 1)$ des fonctions définies sur $[a, b]$ qui ont une dérivée $(q-1)^{\text{ième}}$ absolument continue et une dérivée $q^{\text{ième}}$ de carré sommable, c'est-à-dire que :

$$\int_a^b (f^{(q)}(x))^2 \, dx < +\infty$$

On considère le produit scalaire :

$$(f, g) = \int_a^b f^{(q)}(x) \, g^{(q)}(x) \, dx + c \sum_{i=1}^{N} f(x_i) \, g(x_i)$$

où les x_i sont distincts les uns des autres et où c est un paramètre non négatif. Si C est un sous-espace vectoriel de $H^q[a, b]$, les résultats théoriques précédents s'appliquent. On dit alors que $\overline{g}$ est la **fonction spline d'ajustement** de f. On peut également définir des **fonctions spline d'interpolation**. Ces sujets sont développés dans [31].

5.2.2 Meilleure approximation uniforme

Soit maintenant $C_\infty[a, b]$ l'espace vectoriel des fonctions continues sur $[a, b]$ muni de la norme :

$$\|f\| = \max_{x \in [a, b]} |f(x)|$$

et soit C un sous-espace vectoriel de $C_\infty[a, b]$. $\overline{g}$ est **meilleure approximation uniforme** de f dans C si pour tout $g \in C$, on a :

$$\max_{x \in [a, b]} |f(x) - \overline{g}(x)| \leqslant \max_{x \in [a, b]} |f(x) - g(x)|$$

Nous avons rencontré un exemple de meilleure approximation uniforme lorsque nous avons étudié les polynômes de Tchebychev au paragraphe 2.3.

$C_\infty[a, b]$ n'est pas un espace préhilbertien et les résultats des théorèmes **33**, **34**, **35**, **36** ne sont plus valables. Il nous faut donc donner une caractérisation de la meilleure approximation uniforme (quelquefois appelée **approximation au sens de Tchebychev**), démontrer des résultats d'unicité et en fournir un procédé de construction.

Nous prendrons toujours pour C le sous-espace engendré par des éléments linéairement indépendants $g_1, ..., g_n$ de $C_\infty[a, b]$.

Nous commencerons par un résultat connu sous le nom de **condition de Haar** (théorème **37**).

Théorème 37. Une condition nécessaire et suffisante pour que la meilleure approximation uniforme $\overline{g}$ de f dans C soit unique pour tout $f \in C_\infty[a, b]$ est que tout $g \in C$ admette au plus $n-1$ racines dans $[a, b]$.

On dit alors que les g_i forment un **système de Tchebychev** sur $[a, b]$.

On a le théorème de caractérisation suivant.

Théorème 38. Si la condition de Haar est vérifiée et s'il existe $g \in C$ tel que :

$$f(x_i) - g(x_i) = c(-1)^i a_i \quad \text{pour } i = 0, ..., n$$

avec $c = \pm 1$, $a_i > 0$ et $a \leqslant x_0 < x_1 < ... < x_n \leqslant b$,

alors la meilleure approximation uniforme $\overline{g}$ de f dans C vérifie :

$$\|f - \overline{g}\| \geqslant \min_{0 \leqslant i \leqslant n} a_i$$

Donnons maintenant le **théorème d'alternance de Tchebychev** (théorème **39**).

Théorème 39. Une condition nécessaire et suffisante pour que $\overline{g}$ soit meilleure approximation uniforme de f dans C est que les conditions :

$$|f(x_i) - \overline{g}(x_i)| = \|f - \overline{g}\|$$

$$f(x_i) - \overline{g}(x_i) = -(f(x_{i+1}) - \overline{g}(x_{i+1}))$$

soient satisfaites en au moins $n+1$ abscisses $[a, b]$.

La construction effective de $\overline{g}$ est réalisée grâce à l'algorithme de Rémès.

Dans le cas $g_i(x) = x^i$, $i = 0, ..., n$, on voit immédiatement que la condition du théorème **36** est vérifiée et donc que la meilleure approximation uniforme par des polynômes est unique. Nous renvoyons le lecteur intéressé aux ouvrages spécialisés [31].

La meilleure approximation uniforme par des polynômes s'obtient à l'aide de l'**algorithme de Rémès**.

D'après un théorème de Weierstrass, toute fonction continue peut être approchée aussi près que l'on veut par des polynômes au sens de la norme uniforme. Une réponse constructive à ce résultat est obtenue à l'aide des polynômes de Bernstein (cf. § 2.7.1). Si f est une fonction continue dans [0, 1], alors la suite de polynômes :

$$B_n(x) = \sum_{k=0}^{n} f(k/n) \, B_k^n(x)$$

converge uniformément vers f dans [0, 1].

5.3 Approximation de Padé

La solution de nombreux problèmes s'obtient sous forme de développement en série formelle au voisinage de zéro. Souvent, seuls les premiers coefficients sont connus. D'autre part, la série peut ne pas converger ou, quand elle le fait, sa convergence peut être très lente.

Soit donc :

$$f(t) = c_0 + c_1 t + c_2 t^2 + ...$$

Nous allons en chercher une approximation par une fraction rationnelle.

Un **approximant de Padé de *f*** est une fraction rationnelle dont le développement en série coïncide avec celui de f aussi loin que possible, c'est-à-dire jusqu'au terme dont le degré est la somme

des degrés du numérateur et du dénominateur inclusivement. Soit :

$$[p/q]_f(t) = \frac{P(t)}{Q(t)} = \frac{\sum_{i=0}^{p} a_i t^i}{\sum_{i=0}^{q} b_i t^i}$$

un tel approximant. On veut que :

$$f(t)Q(t) - P(t) = O(t^{p+q+1}), \quad (t \to 0)$$

En identifiant les coefficients des termes de même degré, on trouve que :

$$\begin{aligned} a_0 &= c_0 b_0 \\ a_1 &= c_1 b_0 + c_0 b_1 \\ &\vdots \\ a_p &= c_p b_0 + c_{p-1} b_1 + \ldots + c_{p-q} b_q \\ 0 &= c_{p+1} b_0 + c_p b_1 + \ldots + c_{p-q+1} b_q \\ &\vdots \\ 0 &= c_{p+q} b_0 + c_{p+q-1} b_1 + \ldots + c_p b_q \end{aligned}$$

avec $c_i = 0$ si $i < 0$.

Puisqu'une fraction rationnelle n'est déterminée qu'à un coefficient multiplicatif près, on prendra $b_0 = 1$ (ce terme ne doit pas être nul pour ne pas réduire l'ordre de l'approximation). Les q dernières équations donnent alors $b_1, \ldots, b_q$ comme solution d'un système linéaire, puis les $p+1$ premières équations fournissent directement $a_0, \ldots, a_p$. Le calcul de $[p/q]_f$ nécessite la connaissance des coefficients $c_0, \ldots, c_{p+q}$.

Il existe des méthodes de calcul récursif des suites d'approximants de Padé adjacents (c'est-à-dire dont les degrés diffèrent d'une unité). Elles sont basées sur les **polynômes orthogonaux formels** [4].

Bien que la convergence d'une suite d'approximants de Padé vers la fonction f n'ait pu être démontrée que pour certaines classes de séries (comme les séries de Stieltjes), ils fournissent souvent d'excellentes approximations et sont très utiles dans de nombreuses branches des mathématiques appliquées, en particulier pour le calcul des fonctions spéciales. Ils sont reliés aux **fractions continues** [32].

Les approximants de Padé peuvent s'interpréter comme des formules de quadrature de Gauss formelles. Il est donc possible d'en estimer l'erreur en leur adaptant la procédure de Kronrod décrite dans le paragraphe 3.4. Les racines des polynômes de Stieltjes n'ont alors même plus à être calculées et aucune restriction ne leur est imposée.

Sur les approximants de Padé, on pourra consulter les références [4] et [9] en [Doc. AF 1 221].

5.4 Ondelettes

Le problème du développement en série de Fourier d'une fonction est qu'elle n'est pas locale (cf. dossier [AF 141] dans ce traité ainsi que le paragraphe 5.2.1).

Pour un x donné, il faut donc sommer un très grand nombre de termes pour obtenir une bonne précision. L'idée des **ondelettes** est d'utiliser des fonctions de base ayant soit un support compact soit une décroissance exponentielle.

Soit $L_2(\mathbb{R})$ l'espace vectoriel des fonctions de carré intégrable avec le produit scalaire :

$$(u, v) = \int_{\mathbb{R}} u(x) v(x) \mathrm{d}x$$

On va construire une base de cet espace de la manière suivante. On considère la fonction w à support compact [0, 1] :

$$w(x) = \begin{cases} 0 & \text{si} \quad x < 0 \\ 1 & \text{si} \quad 0 \leqslant x < 1/2 \\ -1 & \text{si} \quad 1/2 \leqslant x < 1 \\ 0 & \text{si} \quad 1 \leqslant x \end{cases}$$

On a :

$$(w(x), w(2x)) = (w(x), w(2x-1)) = (w(2x), w(2x-1)) = 0$$

De même, la fonction $w(2^j x - k)$ est à support compact dans $[k2^{-j}, (k+1)2^{-j}]$, et $\forall r, s$ avec $r \neq j$ ou $s \neq k$, on a :

$$(w(2^j x - k), w(2^r x - s)) = 0$$

$\forall\, j, k \in \mathbb{Z}$, on pose :

$$w_{jk}(x) = 2^{j/2} w(2^j x - k)$$

On a :

$$(w_{jk}(x), w_{rs}(x)) = \delta_{jr} \delta_{ks}$$

ce qui montre que ces fonctions sont orthonormales. Une fonction w qui conduit à une base de $L_2(\mathbb{R})$ après dilatation et translation s'appelle une **ondelette**. Les ondelettes forment une base de $L_2(\mathbb{R})$ comme on va le voir dans un cas particulier (les autres cas sont similaires).

Considérons la fonction ϕ définie par :

$$\phi(x) = \begin{cases} 0 & \text{si} \quad x < 0 \\ 1 & \text{si} \quad 0 \leqslant x < 1 \\ 0 & \text{si} \quad 1 \leqslant x \end{cases}$$

On a :

$$w(x) = \phi(2x) - \phi(2x-1)$$

De plus :

1. $\{\phi(x-k),\ k \in \mathbb{Z}\}$ est un système orthonormal ;

2. $\phi(x) = \phi(2x) + \phi(2x-1)$.

Une telle fonction ϕ s'appelle une **fonction d'échelle**.

Considérons la famille de sous-espaces V_j engendrés par $\phi(2^j x - k)$ pour $k \in \mathbb{Z}$. On a :

$$\ldots \subset V_{-1} \subset V_0 \subset V_1 \subset \ldots$$

et :

1. V_j a une base orthonormale $\{\phi_{jk}(x) = 2^{j/2} \phi(2^j x - k),\ k \in \mathbb{Z}\}$;

2. $f(x) \in V_j$ si et seulement si $f(2^{k-j} x) \in V_k$;

3. $\cap_{j \in \mathbb{Z}} V_j = \{0\}$ et $\overline{\cup_{j \in \mathbb{Z}} V_j} = L_2(\mathbb{R})$.

Cette famille est un exemple d'**analyse multirésolution**. Cependant, l'ensemble des fonctions ϕ_{jk} n'est pas une base de $L_2(\mathbb{R})$. Si l'on a déjà une base orthonormale de V_j, on obtient une base orthonormale de V_{j+1} en ajoutant le complémentaire orthogonal W_j de V_j dans V_{j+1} :

$$V_{j+1} = V_j \oplus W_j$$

Les fonctions $\{w_{jk},\ k \in \mathbb{Z}\}$ forment une base de W_j et W_j est un **espace d'ondelettes**. De plus, $\forall\ i \leqslant j$:

$$V_{j+1} = V_i \oplus W_i \oplus W_{i+1} \oplus \ldots \oplus W_j$$

Par conséquent, $\{w_{jk}(x), j, k \in \mathbb{Z}\}$ est une base orthonormale de $L_2(\mathbb{R})$. Ces ondelettes s'appellent **ondelettes de Haar**.

Supposons que les fonctions w_{jk}, j, $k \in \mathbb{Z}$ forment un système orthonormal d'ondelettes pour $L_2(\mathbb{R})$. Alors, toute fonction $f \in L_2(\mathbb{R})$ peut être représentée comme :

$$f(x) = \sum_{j,\, k \in \mathbb{Z}} (f(x), w_{jk}(x))\, w_{jk}(x)$$

En pratique, on n'utilise qu'un nombre fini de termes selon la précision désirée. Les coefficients se calculent par une procédure similaire à celle de la transformée de Fourier rapide. On pose :

$$\alpha_{jk} = (f(x),\, w_{jk}(x))$$

$$\beta_{jk} = (f(x),\, \phi_{jk}(x))$$

Si, pour n fixé, les coefficients β_{jn}, $j \in \mathbb{Z}$ sont connus, il existe des formules simples reliant les β_{jn} aux α_{jn} et aux $\beta_{j,\, n-1}$. Ainsi, on utilise β_{jn} pour calculer les α_{jn} et l'on obtient 2^n coefficients en $O(2^n)$ opérations arithmétiques. Pour les ondelettes de Haar, chaque nouveau terme ne demande qu'une addition et une multiplication supplémentaires.

Les ondelettes de Haar sont les seules à support compact qui soient à la fois orthogonales et symétriques. Puisqu'elles ne sont pas continues, d'autres types d'ondelettes ont été étudiées, en particulier les **ondelettes biorthogonales**.

Sur les ondelettes, on pourra consulter les références [15] et [17] en [Doc AF 1 221] ainsi que les dossiers [AF 4 510], [AF 4 511] et [AF 210] dans ce traité des Techniques de l'Ingénieur.

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