M206: Analyse numérique pour informaticien(ne)s

Bernhard Beckermann, Labo Painlevé, UST Lille
bbecker@math.univ-lille1.fr
http://math.univ-lille1.fr/∼bbecker

Villeneuve d'Ascq, 28/1/2008

Analyse Numérique, c'est quoi ?

L'analyse numérique intervient après la phase de modélisation où on a traduit une situation concrète en un modèle mathématique (par exemple on cherche l'inconnue dans une équation (différentielle))...

Les étapes :

1. simplification/discretisation du problème, par exemple on cherche à approcher quelques valeurs au lieu de chercher une fonction entière ;
2. choix/construction des algorithmes, critères possibles : efficacité, adaptation à un ordinateur/langage de programmation, stabilité ;
3. analyse d'erreurs de calcul, erreurs de la discrétisation.

Dans ce cours : arithmétique de l'ordinateur, approcher fonctions, droite de régression, analyse de fréquences, FFT, jpeg, approcher intégrales, et plus si besoin...

Références : C. Brezinski, Introduction à la pratique du calcul scientifique, BU
J. Baranger, Introduction à l'analyse numérique, BU.

Arithmétique de l'ordinateur

L'ordinateur se trompe

Calculons $y = 1 + x - x$ pour différentes valeurs de x ($x = 10^j$, $j = 1, 2, ..., 20$)

Petit code Scilab :

```
for j=1:20
   x=10^j; y=1+x-x;
   disp(["x=10^",string(j),' , 1+x-x=',string(y)]);
end;
```

On trouve $y = 1$ pour $j \leqslant 15$, mais $y = 0$ pour $j \geqslant 16$.

POURQUOI ? En maths, $1 + x - x = 1 + (x - x) = 1$, mais l'ordinateur calcule

$$y = (1 + x) - x$$

et $1 + x = x$? Oui, à cause de la représentation des nombres machine.

CONCLUSION : l'addition (multiplication) n'est plus associative, l'ordre du calcul est important.

CONCLUSION ENCORE PLUS IMPORTANTE : ... il faut se méfier de l'ordinateur ... !
Des petites erreurs peuvent s'accumuler et avoir des impacts majeurs !

Il y a des fameux exemples comme l'échec d'une Missile "Patriot" pendant la première guerre de l'Irak, ou l'explosion d'une Ariane 5 montrant que l'arithmétique de l'ordinateur est primordiale, voir

- http ://www.ima.umn.edu/∼arnold/455.f96/disasters.html
- http ://www.irisa.fr/sage/jocelyne/cours/precision/insa-prec-1101.pdf

.

Les nombres machine (en base 10)

Un nombre machine en virgule flottante normalisée dit nombre flottant vaut 0 ou s'écrit sous la forme

$$x = \pm m b^e, \quad m = a_0.a_1 a_2 ... a_s = \sum_{j=0}^{s} \frac{a_j}{b^j},$$

$a_0, a_1, ..., a_s$ entiers entre 0 et $b-1$, e entier entre $e_{\min}$ et $e_{\max}$, $a_0 \neq 0$,

avec m la mantisse (à s chiffres), b la base, (ici $b = 10$ décimal, sur ordinateur $b = 16$ hexa-décimal), et e l'exposant (qui aussi est limité en taille).

EXEMPLES : $-1/(3 * 10^{17})$ s'écrit $-3.3333333\,10^{-18}$ ou `- 3.333D-18` sous Scilab. $100/7$ s'écrit $1.4285714\,10^{1}$ ou `14.285714` sous Scilab.

D'ailleurs, on voit bien aux exemples que les flottants sont des réels, mais la réciproque est fausse.

Sous Scilab : $b = 2$, $s = 52$, $\epsilon = \%eps = b^{-s} \approx 2.210^{-16}$, $e_{\max} = -e_{\min} = 1024$.

1.1. Définition : Pour associer un flottant $fl(x)$ a un réel x, il faut distinguer trois cas :

- si $b^{e_{\min}} \leqslant |x| \leqslant b^{e_{\max}}$ alors on procède (ici) par arrondi et choisit comme $fl(x)$ le nombre flottant le plus près ;
- si $|x| > b^{e_{\max}}$ on parle d'overflow : $fl(x) \in \{NeN, \pm\infty\}$;
- si $|x| < b^{e_{\min}}$ on parle d'underflow : $fl(x) \in \{NeN, 0\}$.

1.2. Lemme : si il n'y a pas dépassement vers $0, \pm\infty$ (underflow, overflow) alors l'erreur relative est bornée par

$$\frac{|x - fl(x)|}{|x|} \leqslant \frac{\epsilon}{2}, \quad \text{avec précision machine} \quad \epsilon = b^{-s}.$$

Un bon modèle pour les 4 opérations arithmétiques de base $\oplus$, $\ominus$, $\otimes$, $\oslash$ sur ordinateur est

$$x \oplus y = fl(x + y), \quad x \ominus y = fl(x - y), \quad x \otimes y = fl(x \times y), \quad x \oslash y = fl(x/y)$$

c'est-à-dire, le calcul est effectué avec une précision plus grande, et ensuite stocké sous format d'un flottant.

Cancellation

1.3. Définition : On dira que $\overline{x}$ approximation de x a p chiffres significatifs si $fl(x) = fl(\overline{x})$ avec une mantisse à p chiffres ($\longrightarrow$ l'erreur relative vérifie $\frac{|\overline{x}-x|}{|x|} \leqslant 10^{-p}$).

EXEMPLES : 0.333 approximation de $1/3$ admet 2 chiffres significatifs, 3.13 approximation de $\pi = 3.1415927...$ admet 1 chiffre significatif.

1.4. Lemme : Si $\overline{x}$ approximation de x et $\overline{y}$ approximation de y admettent p chiffres significatifs alors $\overline{x} \otimes \overline{y}$ approximation de $x \times y$ admet au moins $p-1$ chiffres significatifs (on s'y trompe pas trop).

Le lemme précédent est faux pour l'addition : si on additionne sur machine deux nombres de même taille et de signe opposé, il se peut que le résultat n'a plus aucun chiffre significatif. On parle du phénomène de cancellation.

Dans notre premier exemple, $1 = fl(1) = 1.0000000\,10^0$, $x = 10^{18} = fl(10^{18}) = 1.0000000\,10^{18}$, mais $1 \oplus x = fl(1+x) = x$ (absorption), et $(1 \oplus x) \ominus x = 0$ (cancellation).

D'ailleurs, toujours d'après le phénomène d'absorption, pour toute suite $(a_n)_n$ convergeant vers 0, la somme sur ordinateur $\bigoplus_{n=1}^{\infty} a_n$ est finie, même $\bigoplus_{n=1}^{\infty} \frac{1}{n}$!

Conditionnement d'un problème

Considérons un problème de calcul de $f(b)$ pour une donnée $b \in \mathbb{R}$ ou $b \in \mathbb{R}^n$, par exemple

- évaluation de la fonction $f(b) = b^7$ pour $b = 17$;
- résolution $x = f(b) = A^{-1}b$ d'un système d'équations linéaires $Ax = b$ avec second membre $b \in \mathbb{R}^n$.

Ce problème est **mal conditionné** (en b) (c'est-à-dire, délicat à résoudre) si une petite erreur relative sur b peut entraîner une grande erreur relative sur la valeur : on mesure le facteur d'amplification (au pire)

$$\kappa_f(b) = \limsup_{\Delta b \to 0} \frac{|f(b+\Delta b) - f(b)|}{|f(b)|} \Big(\frac{|\Delta b|}{|b|} \Big)^{-1}.$$

EXEMPLE 1 : Dans le cas d'une fonction $f : \mathbb{R} \mapsto \mathbb{R}$ nous avons $\kappa_f(b) = |\frac{bf'(b)}{f(b)}|$, faites un développement asymptotique (plus délicat dans $\mathbb{R}^n$).

EXEMPLE 2 : Pour la résolution d'un système $Ax = b$:

$$\kappa_f(b) = \sup_{\Delta b} \frac{||A^{-1}\Delta b||}{||\Delta b||} \frac{||b||}{||A^{-1}b||}, \qquad \sup_b \kappa_f(b) = ||A||\,||A^{-1}|| \geqslant 1.$$

Stabilité numérique d'un algorithme de résolution

Un algorithme de calcul de $x = f(b)$ est dit numériquement stable si on trouve par une suite de calculs une réponse $\overline{x}$ étant "bon", plus précisément :

stabilité forward : l'erreur $|\overline{x} - x|$ est petite (inconvénient : souvent surestimation importante de l'erreur).

stabilité backward : la solution numérique $\overline{x}$ est solution exacte $\overline{x} = f(\overline{b})$ pour des données proches ($|\overline{b} - b|/|b|$ petit).

Comme les données sont souvent entachées d'erreurs (des résultats des expériences, obtenues par passage aux flottants), cette deuxième notion est plus réaliste.

EXEMPLE : algorithme de Gauss avec pivotage partiel est backward stable.

NB : La stabilité numérique n'est pas à confondre avec le conditionnement : l'un décrit la qualité d'un algorithme, l'autre le problème lui-même ! Pour les problèmes bien conditionnés, backward stable implique forward stable.

EXEMPLE : comment trouver les solutions d'une équation de deuxième degré $x^2 + bx + c = 0$?

En maths on a vu la formule

$$x_1 = \frac{-b + \sqrt{b^2 - 4c}}{2}, \quad x_2 = \frac{-b - \sqrt{b^2 - 4c}}{2}.$$

Regardons le cas d'un zéro proche de 0 (c petit devant b). On obtient sous Scilab $b = 3.333333333\,10^{-1}, c = 1.428571429\,10^{-14}, \overline{x_1} = -4.285460875\,10^{-14}, \overline{x_2} = -3.333333333\,10^{-1}$, avec $\overline{x}_2$ approximation de x_2 ayant 9 chiffres significatives, mais $\overline{x}_1$ approximation de x_1 ayant seulement 2 chiffres significatifs.

Pourquoi ? Dans la formule pour x_1 on a une cancellation entre $0 < b$ et $0 < \sqrt{b^2 - 4c} \approx b$.

Comment faire mieux ? On peut écrire x_1 sous une autre forme

$$x_1 = x_1 \frac{-b - \sqrt{b^2 - 4c}}{-b - \sqrt{b^2 - 4c}} = \frac{2c}{-b - \sqrt{b^2 - 4c}}$$

et dans l'expression à droite il n'y a plus cancellation : on obtient sous Scilab la valeur $\widetilde{x}_1 = -4.285714286\,10^{-14}$, et sous ordinateur

$$(x - \overline{x}_1)(x - \overline{x}_2) = x^2 + 3.333333333\,10^{-1}\,x + 1.428486958\,10^{-14},$$
$$(x - \widetilde{x}_1)(x - \overline{x}_2) = x^2 + 3.333333333\,10^{-1}\,x + 1.428571429\,10^{-14}$$

Interpolation polynomiale

Position du problème

Etant donné des abscisses distincts $x_0, x_1, ..., x_n \in \mathbb{R}$ et des valeurs $y_0, y_1, ..., y_n \in \mathbb{R}$, trouver un polynôme p de degré au plus n de sorte que $p(x_j) = y_j$ pour $j = 0, 1, ..., n$. Exemple 1, Exemple 2.

Motivation : Reconstruire/approcher une fonction f avec $f(x_j) = y_j$...

Notons par $\mathbb{P}_n$ l'espace des polynômes à coefficients réels de degré $\leqslant n$.

$\mathbb{P}_n$ espace vectoriel de dimension $n+1$, base $g_0, ..., g_n \in \mathbb{P}_n$, c'est-à-dire,
$\forall p \in P \ \exists_1 a_0, ..., a_n \in \mathbb{R}$ avec $p = a_0 g_0 + a_1 g_1 + ... + a_n g_n$.

Quelle base ? Il suffit que g_j soit un polynôme de degré exactement j, par exemple

- base de monômes $g_j(x) = x^j$;
- base décentrée $g_j(x) = (x - x_0)^j$ pour un $x_0 \in \mathbb{R}$;
- base de Newton pour abscisses $x_0, x_1, ..., x_{n-1} \in \mathbb{R}$ définie par $g_j(x) = (x - x_0)(x - x_1)...(x - x_{j-1})$ (en particulier $g_0(x) = 1$).

Préambule : comment évaluer un polynôme ?

Comment évaluer d'une manière efficace $p(x) = x^3 + 3x^2 - 4x - 12$ en $z = 2$?

Idée : mettre x en facteur $p(x) = (((1 * x + 3) * x - 4) * x) - 12...$

2.1 Algorithme de Horner. Tâche : évaluer en $x = z$ l'expression $p(x) = a_0 + a_1(x-x_0)+a_2(x-x_0)(x-x_1)+...+a_n(x-x_0)(x-x_1)...(x-x_{n-1}) = \sum_{j=0}^{n} a_j \prod_{k=0}^{j-1}(x-x_j)$.

Code Scilab (n multiplications et $2n$ additions) :

```
v=a(n) ;
for j=n-1 :-1 :0, v=( z-x(j) )*v + a(j) ; end ;
```

Ecriture sous forme de tableau :

	a_n	a_{n-1}	....	a_1	a_0
z	0	$(z - x_{n-1})v_n$	....	...	$(z - x_0)v_1$
	$v_n = a_n$	$v_{n-1} = somme$	....	$v_1 = somme$	$v_0 = v$

EXEMPLE : évaluer $p(x) = x^3 + 3x^2 - 4x - 12$ en $z = 2$ (en rouge les données)

	1	3	-4	-12
$\boxed{2}$	0	2	10	12
	1	5	6	$0 = p(2)$

2.2. Lemme : Notons $(v_n, ..., v_0) = Horner(z_0; a_n, ..., a_0; x_{n-1}, ..., x_0)$ (coefficients et abscisses à l'envers !), alors

$$a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + ... + a_n(x - x_0)(x - x_1)...(x - x_{n-1})$$
$$= v_0 + v_1(x - z_0) + v_2(x - z_0)(x - x_0) + ... + v_n(x - z_0)(x - x_0)(x - x_1)...(x - x_{n-2})$$

RETOUR A L'EXEMPLE : dans la dernière ligne on trouve coefficients dans la base de Newton pour abscisses $2, 0, 0$: $p(x) = 1(x - 2)(x - 0)(x - 0) + 5(x - 2)(x - 0) + 6(x - 2) + 0$.

Conséquence : Si on applique n fois le schéma de Horner

pour $j = 0, 1, ..., n - 1$: $\quad (a_{j+1,n}, ..., a_{j+1,j}) = Horner(z_j; a_{j,n}, ..., a_{j,j}; x_{n-j-1}, x_{n-j-2}, ..., x_0)$

et $a_{n+1,n} = a_{n,n}$ alors

$$a_{0,0} + a_{0,1}(x - x_0) + a_{0,2}(x - x_0)(x - x_1) + ... + a_{0,n}(x - x_0)(x - x_1)...(x - x_{n-1})$$
$$= a_{1,0} + a_{2,1}(x - z_0) + a_{3,2}(x - z_0)(x - z_1) + ... + a_{n+1,n}(x - z_0)(x - z_1)...(x - z_{n-1})$$

EXEMPLE 1 : $p(x) = x^3 + 3x^2 - 4x - 12 = b_3(x-1)(x+1)(x-2) + b_2(x-1)(x+1) + b_1(x-1) + b_0$

Ici $x_0 = x_1 = x_2 = 0,\ z_0 = 1,\ z_1 = -1,\ z_2 = 2.$

	1	3	-4	-12	abscisses à l'envers : $0, 0, 0$
$\boxed{1}$	0	1	4	0	
	1	4	0	$-12 = b_0$	abscisses à l'envers : $0, 0(, 1)$
$\boxed{-1}$	0	-1	-3		
	1	3	$-3 = b_1$		abscisses à l'envers : $0(, -1, 1)$
$\boxed{2}$	0	2			
	$1 = b_3$	$5 = b_2$			abscisses à l'envers : $(2, -1, 1)$

et donc

$$p(x) = 1 * (x-1)(x+1)(x-2) + 5 * (x-1) * (x+1) - 3 * (x-1) - 12.$$

EXEMPLE 2 : $p(x) = (x-1)(x+1)(x-2) + 5(x-1)(x+1) - 3(x-1) - 12 = b_3(x-2)x^2 + b_2x^2 + b_1x + b_0$

Ici $x_0 = 1,\ x_1 = -1,\ x_2 = 2,\ z_0 = 0, z_1 = 0, z_2 = 2.$

	1	5	-3	-12	abscisses à l'envers :	$2, -1, 1$
$\boxed{0}$	0	-2	3	0		
	1	3	0	$-12 = b_0$	abscisses à l'envers :	$-1, 1(, 0)$
$\boxed{0}$	0	1	-4			
	1	4	$-4 = b_1$		abscisses à l'envers :	$1(, 0, 0)$
$\boxed{2}$	0	1				
	$1 = b_3$	$5 = b_2$			abscisses à l'envers :	$(2, 0, 0)$

et donc

$$p(x) = 1(x-2)x^2 + 5 * x^2 - 4 * x - 12.$$

Existence, unicité : les polynômes de Lagrange

2.3. Théorème : Etant donné des abscisses distincts $x_0, x_1, ..., x_n \in \mathbb{R}$ et des valeurs $y_0, y_1, ..., y_n \in \mathbb{R}$, il existe un et un seul polynôme p de degré au plus n de sorte que $p(x_j) = y_j$ pour $j = 0, 1, ..., n$ (dit polynôme d'interpolation).
Avec les polynômes de Lagrange $\ell_0, ..., \ell_n$ associés aux abscisses $x_0, x_1, ..., x_n$

$$\ell_k(x) = \frac{(x-x_0)(x-x_1)...(x-x_{k-1})(x-x_{k+1})...(x-x_n)}{(x_k-x_0)(x_k-x_1)...(x_k-x_{k-1})(x_k-x_{k+1})...(x_k-x_n)} = \prod_{j=0, j\neq k}^{n} \frac{x-x_j}{x_k-x_j},$$

ce polynôme peut s'écrire à l'aide de la formule de Lagrange

$$p(x) = y_0\ell_0(x) + y_1\ell_1(x) + ... + y_n\ell_n(x).$$

Preuve 1 : Existence : vérifier propriétés du candidat. Unicité : théorème fondamental d'algèbre.

Preuve 2 : écrire le système sous-jacent d'équations linéaires pour coefficients inconnus a_j d'un candidat $p = a_0g_0 + a_1g_1 + ... + a_ng_n$.

NB : la quantité $|\ell_0(x)| + ... + |\ell_n(x)|$ indique le changement de $p(x)$ (au pire) si on perturbe $y_0, ..., y_n$.

Calcul récursif : différences divisées, Formule de Newton

Etudiant : La formule de Lagrange n'est pas utile si on veut ajouter une condition d'interpolation.

2.4. Définition : Etant donné une fonction $f : \mathbb{R} \mapsto \mathbb{R}$ et $x_0, x_1, ... \in \mathbb{R}$ distincts, on notera par $p_{i,j}$ le polynôme de degré $\leqslant j - i$ de sorte que $p_{i,j}(x_k) = f(x_k)$ pour $k = i, i+1, ..., j$, et par

$$[x_i, x_{i+1}, ..., x_j]f \qquad \text{dit "différence divisée"}$$

le coefficient de $p_{i,j}$ devant x^{j-i} (coefficient "de tête").

2.5. Lemme : Nous avons $p_{i,i}(x) = [x_i]f = f(x_i)$, et pour $j > i$

$$p_{i,j}(x) - (x - x_i)(x - x_{i+1})...(x - x_{j-1})\,[x_i, x_{i+1}, ..., x_j]f = p_{i,j-1}(x).$$

2.6. Corollaire : Nous avons la formule de Newton

$$p_{0,n}(x) = \sum_{j=0}^{n} (x - x_0)(x - x_1)...(x - x_{j-1})\,[x_0, x_1, ..., x_j]f.$$

Prof : Donc il suffit d'ajouter un terme à cette somme !

Etudiant : Mais comment calculer ces différences divisées ? D'où vient leur nom ?

Observation : Ni le polynôme d'interpolation $p_{i,j}$ ni la différence divisée $[x_i, x_{i+1}..., x_j]f$ dépend de l'ordre des abscisses $x_i, x_{i+1}..., x_j$.

En retirant de notre formule de Newton celle obtenue pour les abscisses $x_0, x_1, ..., x_{n-2}, x_n, x_{n-1}$, on trouve

$$(x-x_0)(x-x_1)....(x-x_{n-2})\Big([x_0,...,x_{n-2},x_{n-1}]f + (x-x_{n-1})[x_0,...,x_{n-2},x_{n-1},x_n]f\Big)$$
$$= (x-x_0)(x-x_1)....(x-x_{n-2})\Big([x_0,...,x_{n-2},x_n]f + (x-x_n)[x_0,...,x_{n-2},x_n,x_{n-1}]f\Big)...$$

2.7 Corollaire : Nous pouvons calculer récursivement pour $i < j$

$$[x_i]f = f(x_i), \quad [x_i, x_{i+1}..., x_j]f = \frac{[x_{i+1}, x_{i+2}..., x_j]f - [x_i, x_{i+1}..., x_{j-1}]f}{x_j - x_i}.$$

Petit code Scilab ($n(n+1)/2$ divisions, $n(n+1)$ additions)

```
for j=0:n, x(j)=1+j/5; d(j,j)=log(x(j)); end;
for k=1:n
   for j=k:n
      d(j-k,j)=( d(j-k+1,j) - d(j-k,j-1) )/ (x(j) - x(j-k));
   end;
end;
```

On remarque qu'il suffit de travailler avec un tableau 1D par $dd(j) = d(j-k, j)$.

Schéma de calcul sur papier :

$$\begin{array}{c|c} x_0 & [x_0]f \\ x_1 & [x_1]f \\ x_2 & [x_2]f \\ \vdots & \vdots \end{array} \quad [x_0,x_1]f = \frac{[x_1]f-[x_0]f}{x_1-x_0} \quad [x_1,x_2]f = \frac{[x_2]f-[x_1]f}{x_2-x_1} \quad \vdots \quad [x_0,x_1,x_2]f = \frac{[x_1,x_2]f-[x_0,x_1]f}{x_2-x_0} \quad \vdots$$

EXEMPLE : calculer la valeur en $1/2$ du polynôme d'interpolation $p \in \mathbb{P}_3$ de $f(x) = \sin(\pi x/2)$ aux points $0, 1, 2, 3$.

On commence à calculer les différences divisées et applique ensuite Newton et Horner

$$\begin{array}{c|c} 0 & f(0) = 0 \\ 1 & f(1) = 1 \\ 2 & f(2) = 0 \\ 3 & f(3) = -1 \end{array} \quad \begin{array}{l} [x_0,x_1]f = \frac{1-0}{1-0} = 1 \\ [x_1,x_2]f = \frac{0-1}{2-1} = -1 \\ [x_2,x_3]f = \frac{-1-0}{3-2} = -1 \end{array} \quad \begin{array}{l} [x_0,x_1,x_2]f = \frac{-1-1}{2-0} = -1 \\ [x_1,x_2,x_3]f = \frac{-1-(-1)}{3-1} = 0 \end{array} \quad [x_0,x_1,x_2,x_3]f = \frac{1}{3}$$

donc d'après Newton $p(x) = 0 + 1(x-0) + (-1)(x-0)(x-1) + \frac{1}{3}(x-0)(x-1)(x-2)$. Horner :

	$1/3$	-1	1	0
$\boxed{1/2}$	0	$-1/2$	$3/4$	$7/8$
	$1/3$	$-3/2$	$7/4$	$7/8 = p(1/2) = 0.875$

abscisses à l'envers $2, 1, 0$

$f(1/2) \approx 0.707$

Estimation d'erreur : formule de Cauchy

2.8. Théorème : Pour $x, x_0, x_1, ..., x_n \in [a,b]$ distincts

$$f(x) - p_{0,n}(x) = (x-x_0)(x-x_1)....(x-x_n)\,[x_0, x_1, ..., x_n, x]f$$

et si $f \in \mathcal{C}^{n+1}([a,b])$ alors

$$\exists \xi_x \in [a,b] \quad \text{de sorte que} \quad [x_0, x_1, ..., x_n, x]f = \frac{f^{(n+1)}(\xi_x)}{(n+1)!}.$$

La preuve de la deuxième partie est admise ici, elle utilise le théorème de Rolle. Notons que la dépendance de ξ_x de x est compliquée. La deuxième propriété contient ce que l'on appelle le théorème des accroissements finis :

$$f \in \mathcal{C}^1([a,b]) : \qquad \exists \xi \in [a,b] \quad \text{t.q.} \quad [a,b]f = \frac{f(b)-f(a)}{b-a} = f'(\xi).$$

2.9. Corollaire : Si $f \in \mathcal{C}^{n+1}([a,b])$ alors pour tout $x \in [a,b]$

$$|f(x) - p_{0,n}(x)| \leqslant |(x-x_0)(x-x_1)....(x-x_n)| \max_{\xi \in [a,b]} \frac{|f^{(n+1)}(\xi)|}{(n+1)!}.$$

2.10. Résultat sans preuve : Si $x_j = \frac{b+a}{2} + \frac{b-a}{2}\cos(\pi\frac{2j+1}{2n+2}) \in]a,b[$ alors

$\forall x \in [a,b]$: $\quad |(x-x_0)(x-x_1)....(x-x_n)| \leqslant 2((b-a)/4)^{n+1}$ (borne optimale...)

EXEMPLE : retour au polynôme d'interpolation $p_{0,3}$ interpolant $f(x) = \sin(\pi x/2)$ aux points $x_j = j$, $j = 0,1,2,3$.

Posons $[a,b] = [0,3]$, alors $f \in \mathcal{C}^4([a,b])$, avec

$$f'(x) = \frac{\pi}{2}\cos(\frac{\pi x}{2}), \quad f''(x) = -(\frac{\pi}{2})^2\sin(\frac{\pi x}{2}), \quad f'''(x) = -(\frac{\pi}{2})^3\cos(\frac{\pi x}{2}), \quad f''''(x) = (\frac{\pi}{2})^4\sin(\frac{\pi x}{2}),$$

et donc avec $\omega(x) = (x-x_0)(x-x_1)(x-x_2)(x-x_3)$,

$$M := \max_{\xi\in[a,b]} \frac{|f^{(4)}(\xi)|}{4!} = \frac{(\pi/2)^4}{24} \approx 0.254, \quad |\omega(\frac{1}{2})| = \frac{15}{16} \approx 0.94$$

ce qui implique que $|f(1/2) - p_{0,3}(1/2)| \leqslant M\,|\omega(1/2)| \leqslant 0.238$ (c'est une surestimation, $f(1/2) - p_{0,3}(1/2) \approx -0.16$).

Par un calcul élémentaire, on trouve que ω s'annule en a, b, et que ω' s'annule en $3/2$ et $3/2 \pm \sqrt{5}/2$, donc

$$\max_{x\in[a,b]} |f(x) - p_{0,3}(x)| \leqslant M \max_{x\in[a,b]} |\omega(x)|$$

$$= M\ \max\{|\omega(a)|, |\omega(\frac{3-\sqrt{5}}{2})|, |\omega(\frac{3}{2})|, |\omega(\frac{3+\sqrt{5}}{2})|, |\omega(b)|\} = M \approx 0.254.$$

Notons que l'erreur devient plus petit si on prend plus de points d'interpolation dans $[0,3]$.

Au delà de l'interpolation polynômiale

Dilemme de l'interpolation polynômiale :
- si le degré est petit le polynôme d'interpolation est lisse mais l'erreur n'est pas forcement petite ;
- si le degré est grand alors le polynôme risque d'osciller, à ce moment l'erreur n'est pas petite non plus !

Phénomène de Runge :

$$[a,b]=[-5,5], \quad f(x)=\frac{1}{1+x^2}, \quad (n+1) \text{ abscisses équidistantes } x_{j,n}=-5+10\frac{j}{n}.$$

Solutions :
- approximation au sens des moindres carrés (droite de régression) ;
- interpoler avec d'autres fonctions.

Approximation au sens des moindres carrés

Ici on dispose des couples (x_i, y_i) pour $i = 0, 1, ..., m$ et on cherche les coefficients a_j dans l'expression $g(x) = a_0 g_0(x) + a_1 g_1(x) + ... + a_m g_m(x)$ avec m bien plus petit que n de sorte que la somme des distances au carré

$$F(a_0, a_1, ..., a_m) = \sum_{j=0}^{n} |g(x_j) - y_j|^2$$

soit le plus petit possible.

APPLICATION $m = 1$, $g_0(x) = 1$, $g_1(x) = x$: droite de régression.

EXEMPLE : $n + 1 = 11$ points avec $m = 1$ et $m = 2$.

SOLUTION 1 : en annulant les dérivées de

$$F(a_0, a_1, ..., a_m) = \sum_{j=0}^{n} \Big(y_j - \sum_{k=0}^{m} a_k g_k(x_j) \Big)^2$$

par rapport à $a_0, a_1, ..., a_m$ on obtient un unique point stationnaire par un système d'équations linéaires avec inconnues $a_0, a_1, ..., a_m \Longrightarrow$ solution de ce système donne coefficients optimaux.

SOLUTION 2 : on peut écrire notre problème sous forme matricielle

$$\min_{a \in \mathbb{R}^{m+1}} \|Aa - y\|, \quad A = \begin{bmatrix} g_0(x_0) & \cdots & g_m(x_0) \\ g_0(x_1) & \cdots & g_m(x_1) \\ \vdots & & \vdots \\ g_0(x_n) & \cdots & g_m(x_n) \end{bmatrix}, \quad a = \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_m \end{bmatrix}, \quad y = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{bmatrix}$$

avec $\|\cdot\|$ la norme (distance) euclidienne. Cela donne un problème des moindres carrés, résolu sous Scilab par `a=A \ y`.

Interpolation avec splines cubiques

Ici on interpole avec des fonctions étant par morceaux des polynômes, raccordés de sorte que globalement la fonction soit lisse (l'oeil perçoit seulement régularité C^2).

Etant donnés noeuds $... < t_0 < t_1 < t_2 < ...$ on définit la B-spline B_j étant

- un polynôme de degré $\leqslant 3$ sur chaque sous-intervalle $[t_k, t_{k+1}]$;
- de classe $\mathcal{C}^2(\mathbb{R})$, identiquement zéro en dehors de $[t_j, t_{j+4}]$;
- $\forall x \in \mathbb{R} : B_j(x) \geqslant 0$, $\sum_j B_j(x) = 1$.

L'expression mathématique pour $B_j(x)$ dépendant seulement de $t_j, .., t_{j+4}$ est très compliquée :

$$B_j(x) = \begin{cases} \dfrac{(x-t_j)^3}{(t_{j+3}-t_j)(t_{j+2}-t_j)(t_{j+1}-t_j)} & \text{si } x \in [t_j, t_{j+1}[, \\ \dfrac{x-t_j}{t_{j+3}-t_j}\dfrac{x-t_j}{t_{j+2}-t_j}\dfrac{t_{j+2}-x}{t_{j+2}-t_{j+1}} + \dfrac{x-t_j}{t_{j+3}-t_j}\dfrac{t_{j+3}-x}{t_{j+3}-t_{j+1}}\dfrac{x-t_{j+1}}{t_{j+2}-t_{j+1}} & \\ \quad + \dfrac{t_{j+4}-x}{t_{j+4}-t_{j+1}}\dfrac{(x-t_{j+1})^2}{(t_{j+3}-t_{j+1})(t_{j+2}-t_{j+1})} & \text{si } x \in [t_{j+1}, t_{j+2}[, \\ \dfrac{x-t_j}{t_{j+3}-t_j}\dfrac{(t_{j+3}-x)^2}{(t_{j+3}-t_{j+1})(t_{j+3}-t_{j+2})} + \dfrac{t_{j+4}-x}{t_{j+4}-t_{j+1}}\dfrac{x-t_{j+1}}{t_{j+3}-t_{j+1}}\dfrac{t_{j+3}-x}{t_{j+3}-t_{j+2}} & \\ \quad + \dfrac{t_{j+4}-x}{t_{j+4}-t_{j+1}}\dfrac{t_{j+4}-x}{t_{j+4}-t_{j+2}}\dfrac{x-t_{j+2}}{t_{j+3}-t_{j+2}} & \text{si } x \in [t_{j+2}, t_{j+3}[, \\ \dfrac{(t_{j+4}-x)^3}{(t_{j+4}-t_{j+1})(t_{j+4}-t_{j+2})(t_{j+4}-t_{j+3})} & \text{si } x \in [t_{j+3}, t_{j+4}[, \\ 0 & \text{sinon.} \end{cases}$$

En choisissant les noeuds t_j en fonction des x_j on peut interpoler : par exemple, avec

$$t_{-1} < t_0 < t_1 < t_2 \leqslant x_0 < t_3, ..., t_{n-1} < x_n \leqslant t_n < t_{n+1} < t_{n+2} < t_{n+3}$$

on peut définir $B_{-1}, B_0, ..., B_{n-1}$, et on trouve $p \in \text{span}(B_{-1}, B_0, ..., B_{n-1})$ avec $p(x_j) = y_j$ pour $j = 0, 1, ..., n$ par

$$p(x) = \Big[B_{-1}(x), B_0(x), ..., B_{n-1}(x)\Big] \begin{bmatrix} B_{-1}(x_0) & B_0(x_0) & \cdots & B_{n-1}(x_0) \\ B_{-1}(x_1) & B_0(x_1) & \cdots & B_{n-1}(x_1) \\ \vdots & \vdots & & \vdots \\ B_{-1}(x_n) & B_0(x_n) & \cdots & B_{n-1}(x_n) \end{bmatrix}^{-1} \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{bmatrix}.$$

Attention : la matrice est inversible ssi $x_j \in]t_{j-1}, t_{j+3}[$ pour $j = 0, 1, ..., n$.

On peut aussi utiliser une approche moindres carrés (si les données (x_j, y_j) sont entachées d'erreur) : pour $m < n$, on trouve $p \in \text{span}(B_{-1}, B_0, ..., B_{m-1})$ avec $\sum_{j=0}^{n} \big(p(x_j) - y_j\big)^2$ minimum par

$$p(x) = \Big[B_{-1}(x), B_0(x), ..., B_{m-1}(x)\Big] \begin{bmatrix} B_{-1}(x_0) & B_0(x_0) & \cdots & B_{m-1}(x_0) \\ B_{-1}(x_1) & B_0(x_1) & \cdots & B_{m-1}(x_1) \\ \vdots & \vdots & & \vdots \\ B_{-1}(x_n) & B_0(x_n) & \cdots & B_{m-1}(x_n) \end{bmatrix} \setminus \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{bmatrix}.$$

EXEMPLE avec $n = 10, m \in \{10, 7, 4\}$, sans bruit et avec bruit.

Points de contrôle, courbes et surfaces splines

Courbes $g : \mathbb{R} \mapsto \mathbb{R}^2$

$$g(x) = \sum_{j=0}^{m} P_j B_j(x), \qquad P_j \in \mathbb{R}^2 \quad \text{dits points de contrôle.}$$

Dans la CAO (conception assistée par ordinateur), le choix de ces points de contrôle permet de donner à la courbe une allure souhaitée, tout en gardant une régularité $\mathcal{C}^2$.

– la courbe "suit" des points de contrôle, mais elle n'y passe pas ;

– changer un point de contrôle change seulement une partie de la courbe.

Simulation

Surfaces $g : \mathbb{R}^2 \mapsto \mathbb{R}^3$

$$g(u,v) = \sum_{j=0}^{m} \sum_{k=0}^{n} P_{j,k} B_j(u) B_k(v), \qquad P_{j,k} \in \mathbb{R}^3 \quad \text{dits points de contrôle.}$$

Le choix de ces points de contrôle permet de déformer localement d'une manière lisse la surface. Application : capot d'une voiture. Simulation 1, Simulation 2, Simulation 3.

Analyse de fréquence et jpeg

Une fonction $f : \mathbb{R} \mapsto \mathbb{R}$ est dite périodique de période $T > 0$ si $\forall x \in \mathbb{R} : f(x+T) = f(x)$.

Pour analyser la vibration d'une barre où le son d'une guitare, on décompose un signal donné par une fonction $s : [0, 2\pi[\mapsto \mathbb{R}$ (et prolongé périodiquement sur $\mathbb{R}$ par la relation $s(x) = s(x + 2\pi)$) par une somme dite de Fourier de la forme

$$\sum_{j=0}^{\infty} (a_j \cos(jx) + b_j \sin(jx)).$$

Les a_j, b_j sont alors les amplitudes des fréquences propres, utilisés aussi dans la compression d'images, la téléphonie mobile, etc etc.

Problème : Comment définir/calculer rapidement ces coefficients a_j, b_j ?

Idée 1 : on remplace la somme infinie par une somme partielle, et on demande que cette expression coïncide avec f en un certain nombre de points.

Idée 2 : on se sert de la formule d'Euler $e^{ijx} = \cos(jx) + i \sin(jx)$ pour travailler avec une seule classe de fonctions. Défaut : on passe par les nombres complexes $\mathbb{C}$ (on y reviendra).

La DFT et la IDFT

Conventions : $x_k^{\alpha,n} = 2\pi\frac{k+\alpha}{n}$.
On note les coefficients d'une somme de Fourier d'ordre n par des majuscules

$$F(x) = \sum_{j=0}^{n-1} F_j \exp(ijx)$$

et les valeurs $f_k = F(x_k^{\alpha,n})$ par des miniscules.

DFT (discrete Fourier transform) : passage des valeurs aux coefficients

$$(F_k)_{k=0,\ldots,n-1} = DFT((f_k)_{k=0,\ldots,n-1}) = DFT^{\alpha,n}((f_k)_{k=0,\ldots,n-1}).$$

IDFT (inverse discrete Fourier transform) : passage des coefficients aux valeurs

$$(f_k)_{k=0,\ldots,n-1} = IDFT((F_k)_{k=0,\ldots,n-1}) = IDFT^{\alpha,n}((F_k)_{k=0,\ldots,n-1}),$$
$$f_k = \sum_{j=0}^{n-1} F_j \exp(2\pi i\frac{j(k+\alpha)}{n}).$$

Formule pour la DFT

4.1. Lemme : Pour j, ℓ, n des entiers

$$\sum_{k=0}^{n-1} \exp(-2\pi i \frac{jk}{n}) \exp(2\pi i \frac{\ell(k+\alpha)}{n}) = \begin{cases} n \exp(2\pi i \frac{\alpha \ell}{n}) & \text{si } j-\ell \text{ est divisible par } n, \\ 0 & \text{sinon.} \end{cases}$$

Idée de la preuve : On pose $q = \exp(2\pi i \frac{\ell - j}{n})$, alors $q^n = 1$, et la somme devient

$$\exp(2\pi i \frac{\alpha \ell}{n}) \sum_{k=0}^{n-1} q^k.$$

4.2. Théorème : Pour toute fonction f de périodicité 2π il existe une et une seule somme de Fourier discrète F d'ordre n vérifiant les conditions d'interpolation $f(x_k^{\alpha,n}) = F(x_k^{\alpha,n})$ pour $k = 0, 1, ..., n-1$.
Ses coefficients sont donnés par la DFT explicite (pour $j = 0, 1, ..., n-1$)

$$F_j = \frac{1}{n} \exp(-2\pi i \frac{\alpha j}{n}) \sum_{k=0}^{n-1} \exp(-2\pi i \frac{jk}{n}) f_k.$$

Simulation 1 sans bruit, Simulation 2 avec bruit.

Lien entre moindres carrés et DFT

Comme dans le Théorème 4.2. on montre le résultat suivant :

4.3. Corollaire : Etant donné une fonction f de périodicité 2π et sa somme de Fourier discrète d'ordre n

$$F(x) = \sum_{j=0}^{n-1} F_j \exp(ijx),$$

alors la somme tronquée d'ordre $m \leqslant n$

$$\widetilde{F}(x) = \sum_{j=0}^{m-1} F_j \exp(ijx)$$

réalise une erreur

$$\sum_{k=0}^{n-1} |f(x_k^{\alpha,n}) - \widetilde{F}(x_k^{\alpha,n})|^2$$

minimale parmi toutes les somme de Fourier d'ordre m.

La transformée de Fourier discrète permet alors de combiner à la fois les avantages d'approximation au sens des moindres carrés avec le principe d'interpolation.

: En cas d'erreurs sur les données, il vaut mieux tronquer la somme de Fourier.

La FFT : transformée de Fourier rapide

Le calcul $DFT = DFT^{\alpha,n}$ par les formules

$$f_k = F(x_k^{\alpha,n}) = \sum_{j=0}^{n-1} F_j e^{2\pi i \frac{j(k+\alpha)}{n}}, \qquad F_j = \frac{1}{n} e^{-2\pi i \frac{\alpha j}{n}} \sum_{k=0}^{n-1} e^{-2\pi i \frac{jk}{n}} f_k$$

nécessite $\mathcal{O}(n^2)$ multiplications pour l'ensemble des coefficients. Pour n une puissance de 2, on peut faire mieux par une approche récursive :

Comme $x_k^{\alpha,n} = 2\pi \frac{k+\alpha}{2n}$, on trouve pour le coefficient F_j de $(F_j)_{j=0,\ldots,2n-1} = DFT^{\alpha,2n}$

$$\frac{e^{-2\pi i \frac{\alpha j}{2n}}}{2n} \sum_{k=0}^{n-1} \underbrace{e^{-2\pi i \frac{2jk}{2n}}}_{=e^{-2\pi i \frac{jk}{n}}} \underbrace{f_{2k}}_{=f(x_{2k}^{\alpha,2n})=f(x_k^{\alpha/2,n})} + \frac{e^{-2\pi i \frac{\alpha j}{2n}}}{2n} \sum_{k=0}^{n-1} \underbrace{e^{-2\pi i \frac{j(2k+1)}{2n}}}_{=e^{-2\pi i \frac{j}{2n}} e^{-2\pi i \frac{jk}{n}}} \underbrace{f_{2k+1}}_{=f(x_{2k+1}^{\alpha,2n})=f(x_k^{(\alpha+1)/2,n})}$$

4.4. Théorème : Pour $j = 0, 1, \ldots, n-1$

$$DFT_j^{\alpha,2n} = \frac{1}{2}[DFT_j^{\alpha/2,n} + DFT_j^{(\alpha+1)/2,n}]$$

$$DFT_{j+n}^{\alpha,2n} = \frac{1}{2}\exp(-2\pi i \frac{\alpha}{2})[DFT_j^{\alpha/2,n} - DFT_j^{(\alpha+1)/2,n}]$$

L'algorithme FFT

Notons que pour les vecteurs de valeurs :

$$f = f(1:2n) = (F(x_k^{\alpha,2n}))_{k=0,\ldots,2n-1} \Longrightarrow$$
$$f(1:2:2n-1) = (F(x_k^{\alpha/2,n}))_{k=0,\ldots,n-1},\ f(2:2:2n) = (F(x_k^{(\alpha+1)/2,n}))_{k=0,\ldots,n-1}.$$

Donc sous Scilab on obtient la fonction suivante

```
function [F]=mon_FFT(f,α)
   // f vecteur de valeurs au points 2π(k+α)/n pour k=0,1,..,n-1
   // n puissance de 2, F=DFT^{α,n}
   n=length(f)
   if (n==1) then
      F=f ;
   else
      F1=mon_FFT(f(1 :2 :n/2),α/2) ; F2=mon_FFT(f(2 :2 :n/2),(α+1)/2) ;
      e=exp(-πiα) ; // valeurs tabulées
      F=[ F1 + F2 , e* (F1-F2) ]/2 ;
   end ;
endfunction ;
```

Complexité : $\mathcal{O}(n \log n)$ $(mul_{2n} = 2\,mul_n + 2n,\ mul_1 = 0)$

Et comment se ramener à une arithmétique réelle ?

Ecrivons $IDFT = IDFT^{1/2,2n}$, $DFT = DFT^{1/2,2n}$,

$$f_k = F(\pi\frac{2k+1}{2n}) = \sum_{j=0}^{2n-1} F_j e^{\pi i \frac{j(2k+1)}{2n}}, \quad F_j = \frac{1}{2n} e^{-\pi i \frac{j}{2n}} \sum_{k=0}^{2n-1} e^{-\pi i \frac{jk}{n}} f_k.$$

Si la fonction f de départ est 2π-périodique **et** symétrique $f(\pi + x) = f(\pi - x)$, alors $f_k = f(\pi\frac{2k+1}{2n}) = f(2\pi - \pi\frac{2k+1}{2n}) = f_{2n-1-k}$, et donc

$$F_j = \frac{1}{2n} e^{-\pi i \frac{j}{2n}} \sum_{k=0}^{n-1} [e^{-\pi i \frac{jk}{n}} + e^{-\pi i \frac{j(2n-1-k)}{n}}] f_k = \frac{1}{n} \sum_{k=0}^{n-1} \cos(j\pi\frac{2k+1}{2n}) f_k =: DCT_j^n = -F_{2n-j},$$

$$f_k = F_0 + \sum_{j=1}^{n-1} F_j [e^{\pi i \frac{j(2k+1)}{2n}} - e^{\pi i \frac{(2n-j)(2k+1)}{2n}}] = F_0 + 2\sum_{j=1}^{n-1} F_j \cos(j\pi\frac{2k+1}{2n}) =: IDCT_k^n.$$

4.5. Théorème : Pour tout $f : [0, \pi] \mapsto \mathbb{R}$ il existe un unique polynôme trigonometrique de la forme $C(x) = C_0 + 2\sum_{j=1}^{n-1} C_j \cos(jx)$ vérifiant les conditions d'interpolation $f(\pi\frac{2k+1}{2n}) = C(\pi\frac{2k+1}{2n})$ pour $k = 0, 1, ..., n-1$. Dans ce cas, nous avons des transformées de cosinus discrètes $c = IDCT(C)$, $C = DCT(c)$, avec

$$c_k = C(\pi\frac{2k+1}{2n}) = C_0 + 2\sum_{j=1}^{n-1} C_j \cos(j\pi\frac{2k+1}{2n}) \in \mathbb{R}, \quad C_j = \frac{1}{n} \sum_{k=0}^{n-1} \cos(j\pi\frac{2k+1}{2n}) c_k \in \mathbb{R}.$$

Quelques remarques sur nos transformées de cosinus discrètes

- Il existe plusieurs transformées de cosinus discrètes (de DCT-1 à DCT-8). Dans cette nomenclature, notre IDCT est DCT-3 et notre DCT est DCT-2 (à des facteurs près), voir par exemple Wikipedia (anglais).
- Il existent des méthodes FCT pour le calcul "rapide" comme FFT, mais ces méthodes sont plus compliquées.
- Le résultat du corollaire 4.3 (sommes partielles et approximation au sens des moindres carrés) peut être généralisé aux sommes des cosinus.
- On peut résumer schématiquement le calcul $C = DCT(c)$, $c = IDCT(C)$ comme

$$c = M * C, \quad C = M^{-1} * c,$$

avec

$$C = \begin{bmatrix} C_0 \\ C_1 \\ \vdots \\ C_{n-1} \end{bmatrix}, \; c = \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_{n-1} \end{bmatrix}, \; M = \begin{bmatrix} 1 & 2\cos(\frac{\pi}{2n}) & 2\cos(2\frac{\pi}{2n}) & \dots & 2\cos((n-1)\frac{\pi}{2n}) \\ 1 & 2\cos(\frac{3\pi}{2n}) & 2\cos(2\frac{3\pi}{2n}) & \dots & 2\cos((n-1)\frac{3\pi}{2n}) \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & 2\cos(\frac{(2n-1)\pi}{2n}) & 2\cos(2\frac{(2-1)\pi}{2n}) & \dots & 2\cos((n-1)\frac{(2n-1)\pi}{2n}) \end{bmatrix}$$

et $M^{-1} = \text{diag}(\frac{1}{n}, \frac{1}{2n}, ..., \frac{1}{2n}) * M^T$.

Principe de seuillage

Idée : Si on supprime dans

$$C(x) = C_0 + 2\sum_{j=1}^{n-1} C_j \cos(jx)$$

un terme $C_j \cos(jx)$ avec $|C_j| < \epsilon$ "petit" alors on obtient une erreur absolue $\leqslant 2\epsilon$.

Stratégie de seuillage : on se fixe un seuil $\epsilon > 0$ et on remplace un C_j par 0 si

$$|C_j| < \epsilon \quad \text{(seuillage absolu), ou} \quad |C_j| < \epsilon \sum_{\ell=0}^{n-1} |C_\ell| \quad \text{(seuillage rélatif).}$$

Avantage : compression de données, car en stockant seulement les coefficients C_j non nuls on garde moins d'information, tout en gardant la possibilité de pouvoir reconstruire (assez bien) le signal.

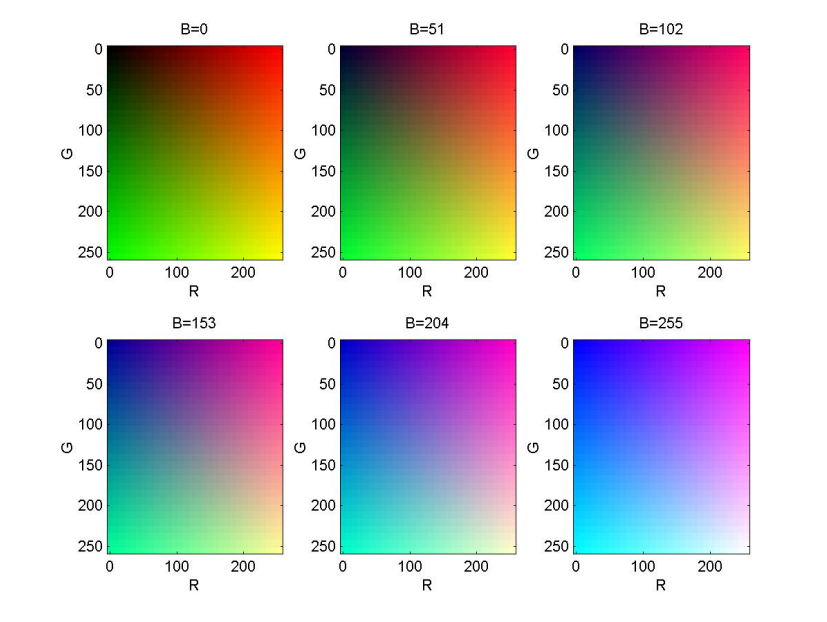
Simulation 1 sans bruit, Simulation 2 avec bruit.

On vient de comprendre le principe de base du mode de compression **jpeg**, même si les images, c'est du 2D.

C'est quoi une image en couleurs ?

Une image 334×460 est représenté par trois matrices AR, AV, AB (ici de taille 334×460) :

- Trois couleurs de base : rouge - vert - bleu
- On introduit une grille de 334 lignes et 460 colonnes : $AR(j,k)$, $AV(j,k)$, $AB(j,k) \in \{0,1,...,255\}$ (deux chiffres en hexadécimal) décrivent l'intensité de chaque couleur (codage RGB) du pixel à la position (j,k).



Stratégie compression par JPEG

Philosophie : Stocker une matrice creuse (avec beaucoup d'éléments $= 0$) prend moins de place.

- Faire pour chacune des trois matrices couleurs :
  - Couper en morceaux y de taille 8×8 (ou 16×16), et faire pour chaque morceau
    - Calculer $Y = DCT(y) = M * y * M^T$, de même taille que y
    - Seuillage : remplacer des éléments "petits" dans Y par 0
    - garder les nouvelles matrices Y_0 en mémoire
  - au moment de l'affichage, calculer $y_0 = IDCT(Y_0) = (M^T)^{-1} * Y_0 * M^{-1}$.

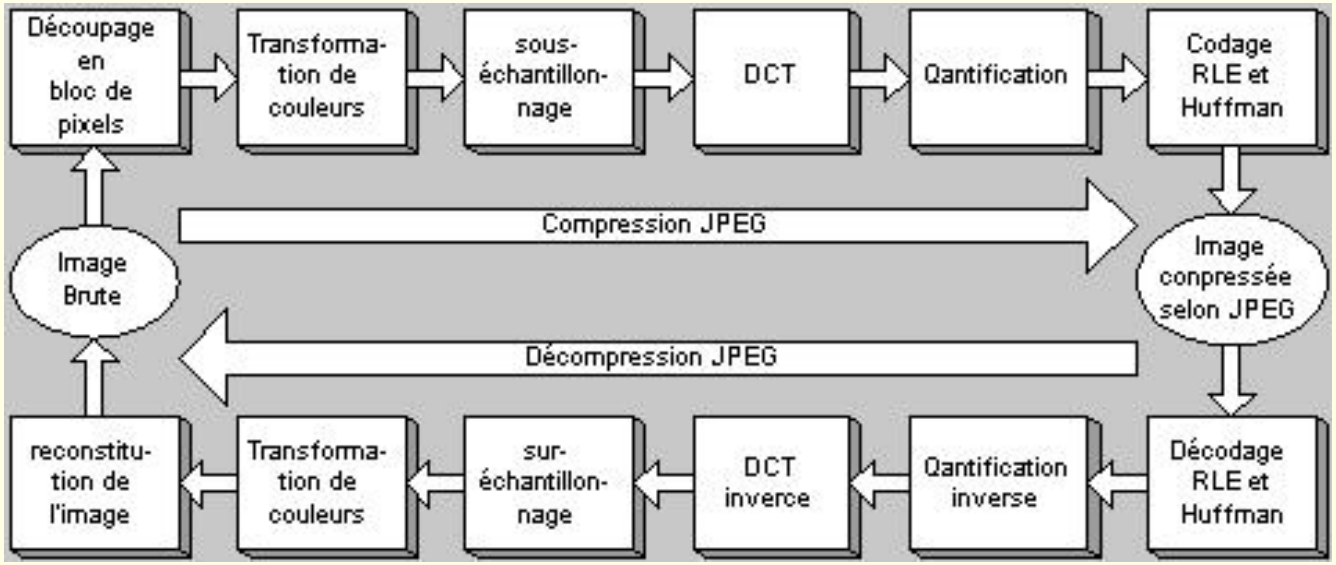
Question : $y_0 \approx y$? ?

Réponse : ca dépend du seuil (d'ailleurs aussi le taux de compression).

Photo 1 Photo 2

Détails JPEG

En réalité il y a un peu plus de détails.



Aussi, JPEG date de 1991, et maintenant d'autres compressions sont plus prometteurs (JPEG2000, etc)

Calcul approché des intégrales

Objectif : calculer par exemple $\int_{-1}^{1} e^{-x^2} dx$.
Problème : pas de primitive élémentaire ! Il faut donc approcher l'intégrale (on parle des méthodes de **quadrature**).

Idée de l'intégrale de Riemann : avec $c = y_0 < y_1 < ... < y_N = d$ on décompose

$$I_{[c,d]}(f) := \int_c^d f(x)\,dx = I_{[y_0,y_1]}(f) + I_{[y_1,y_2]}(f) + ... + I_{[y_{N-1},y_N]}(f)$$

et l'intégrale pour chaque sous-intervalle $[a,b]$ "petit" est approchée par une formule simple (approchant l'aire sous la courbe), par exemple (ex1 , ex2)

formule du rectangle : $Q^{Rect}_{[a,b]}(f) = (b-a)f(a)$

for. du point milieu : $Q^{Mi}_{[a,b]}(f) = (b-a)f(\frac{a+b}{2})$

formule du trapèze : $Q^{Tr}_{[a,b]}(f) = (b-a)\frac{f(a)+f(b)}{2}$

formule de Simpson : $Q^{Si}_{[a,b]}(f) = (b-a)\frac{f(a)+4f(\frac{a+b}{2})+f(b)}{6} = \frac{1}{3}Q^{Tr}_{[a,b]}(f) + \frac{2}{3}Q^{Mi}_{[a,b]}(f).$

Formule versus méthode de quadrature

Pour simplifier, on prend ici toujours points équidistants $y_j = c + (d-c)\frac{j}{N}$, et alors $y_0 = c$, $y_N = d$, $y_{j+1} - y_j = (d-c)/N$.

Chaque formule de quadrature donne une méthode de quadrature à N sous-intervalles, à titre d'exemple la méthode des trapèzes :

$$\begin{aligned} Q_N^{Tr}(f) &= \sum_{j=0}^{N-1} Q^{Tr}_{[y_j,y_{j+1}]}(f) = \frac{d-c}{N} \sum_{j=0}^{N-1} \frac{f(y_j)+f(y_{j+1})}{2} \\ &= \frac{d-c}{2N}(f(c)+f(d)) + \frac{d-c}{N} \sum_{j=1}^{N-1} f(y_j). \end{aligned}$$

Estimation d'erreur ? On majore l'erreur pour la méthode en majorant l'erreur pour une formule

$$|I_{[c,d]}(f) - Q_N(f)| \leqslant \sum_{j=0}^{N-1} |I_{[c,d]}(f) - Q_{[y_j,y_{j+1}]}(f)|,$$

mais comment traiter les formules ? Lien avec interpolation polynomiale !

Formules de quadrature et interpolation par constantes

Regardons d'abord un cas particulier : avec $\xi \in [a,b]$, nous trouvons $p(x) = f(\xi)$ comme polynôme d'interpolation de f de degré $\leqslant n = 0$ au point $x_0 = \xi$. Aussi,

$$I_{[a,b]}(p) = \int_a^b f(\xi)dx = (b-a)f(\xi), \quad \forall x \in [a,b] : \quad |f(x) - p(x)| \leqslant |x-\xi| \max_{y\in[a,b]} |f'(y)|$$

d'après le chapitre sur l'interpolation polynômiale (ou le théorème des accroissements finis), et alors

$$\begin{aligned} |I_{[a,b]}(p) - (b-a)f(\xi)| &= |I_{[a,b]}(f-p)| \leqslant I_{[a,b]}(|f-p|) \\ &\leqslant \max_{y\in[a,b]} |f'(y)| \int_a^b |x-\xi|\, dx = \max_{y\in[a,b]} |f'(y)| \, \frac{|b-\xi|^2 + |\xi-a|^2}{2}. \end{aligned}$$

5.1. Corollaire : Si $f \in \mathcal{C}^1([c,d])$ alors

$$|I_{[c,d]}(f) - Q_N^{Rect}(f)| \leqslant \frac{(d-c)^2}{2N} \max_{y\in[c,d]} |f'(y)|, \quad |I_{[c,d]}(f) - Q_N^{Mi}(f)| \leqslant \frac{(d-c)^2}{4N} \max_{y\in[c,d]} |f'(y)|.$$

Formules de quadrature interpolatoires

5.2. Définition : Une formule $Q_{[a,b]}$ est dite interpolatoire pour des abscisses distincts $x_0, x_1, ..., x_n \in [a,b]$ si $Q_{[a,b]}(f) = I_{[a,b]}(p)$ avec p le polynôme de degré au plus n interpolant f aux abscisses $x_0, ..., x_n$.

EXEMPLE : la formule du rectangle et du point milieu sont interpolatoire pour $x_0 = a$, et $x_0 = \frac{a+b}{2}$, respectivement.

5.3. Lemme : Une formule $Q_{[a,b]}$ est interpolatoire pour des abscisses $x_0, x_1, ..., x_n$ si et seulement si elle prend la forme

$$Q_{[a,b]}(f) = \sum_{j=0}^{n} w_j f(x_j), \quad w_j = I_{[a,b]}(\ell_j),$$

avec $\ell_0, ..., \ell_n$ les polynômes de Lagrange.

EXEMPLE : montrons que la méthode des trapèzes est interpolatoire pour les abscisses $x_0 = a$, $x_1 = b$

$$\ell_0(x) = \frac{x-a}{b-a}, \quad I_{[a,b]}(\ell_0) = \int_a^b \frac{x-a}{b-a} dx = \frac{b-a}{2}, \quad I_{[a,b]}(\ell_1) = \int_a^b \frac{x-b}{a-b} dx = \frac{b-a}{2}.$$

EXERCICE : Montrer que la formule de Simpson est interpolatoire pour les abscisses $x_0 = a, x_1 = \frac{a+b}{2}$, $x_2 = b$.

Estimation d'erreur pour les formules interpolatoires

5.4 Théorème : Si $Q_{[a,b]}$ est interpolatoire pour des abscisses $x_0, x_1, ..., x_n$ et $f \in \mathcal{C}^{n+1}([a,b])$ alors avec $M_{n,[c,d]}(f) = \max_{y\in[c,d]} |f^{(n)}(y)|/n!$

$$|I_{[a,b]}(f) - Q_{[a,b]}(f)| \leqslant M_{n+1,[a,b]}(f) \int_a^b |(x-x_0)(x-x_1)...(x-x_n)|\, dx.$$

5.5 Corollaire : Si $f \in \mathcal{C}^2([c,d])$ alors

$$|I_{[c,d]}(f) - Q_N^{Tr}(f)| \leqslant \frac{(d-c)^3}{6N^2} M_{2,[c,d]}(f).$$

5.6 Corollaire : Si $f \in \mathcal{C}^3([c,d])$ alors

$$|I_{[c,d]}(f) - Q_N^{Si}(f)| \leqslant \frac{(d-c)^4}{64\,N^3} M_{3,[c,d]}(f).$$

Moralité : l'estimation d'erreur est proportionnel à une puissance de $1/N$, "petite" pour N grand.

NB1 : on peut montrer que $I_{[c,d]}(f) - Q_N^{Mi}(f)| \leqslant \frac{(d-c)^3}{12\,N^2} M_2(f)$ si $f \in \mathcal{C}^2([c,d])$.
NB2 : on peut montrer que $Q_N^{Mi}(f) \leqslant I_{[c,d]}(f) \leqslant Q_N^{Tr}(f)$ pour f convexe sur $[c,d]$.

Un dernier exemple

Comment choisir N pour calculer $\int_{-1}^{1} \exp(-x^2)\,dx$ à l'aide de la méthode des trapèzes avec une erreur $\leqslant 10^{-6}$?

Ici $f(x) = \exp(-x^2)$, $[c,d] = [-1,1]$,

$$Q_N^{Tr}(f) = \frac{1}{N}\frac{1}{e} + \sum_{j=1}^{N-1} e^{-(2j/N)^2}.$$

Calculons d'abord $M_{2,[-1,1]}(f)$

$$f'(x) = -2xe^{-x^2}, \quad f''(x) = (4x^2-2)e^{-x^2}, \quad f'''(x) = (-8x^3+12x)e^{-x^2}$$

et donc $f''(-x) = f''(x)$, $f'''(x) \geqslant 0$ pour $x \in [0,1]$, et alors $M_{2,[-1,1]}(f) = \max\{|f''(0)|, |f''(1)|\}/2 = |f''(0)|/2 = 1$. Il suffit donc de choisir

$$10^{-6} \geqslant \frac{(d-c)^3}{6N^2} M_{2,[c,d]}(f) = \frac{4}{3}\frac{1}{N^2}$$

ou $N \geqslant \sqrt{\frac{4}{3}10^6} \approx 1154.7$.

Algorithme de Gauss