Astérisque

J. MORGENSTERN

Transformations de Fourier discrètes

Astérisque, tome 38-39 (1976), p. 159-167

<http://www.numdam.org/item?id=AST_1976__38-39__159_0>

Société Mathématique de France
Astérisque 38-39 (1976) p.159-167

TRANSFORMATIONS DE FOURIER DISCRÈTES

par

J. MORGENSTERN

Ce bref exposé ne constitue en aucun cas une étude exhaustive de l'état de ces questions qui sont en perpétuelle évolution mais plutôt une introduction, pour mathématiciens non spécialistes, sur un exemple, aux résultats et techniques de la théorie des algorithmes et de leur complexité.

1.- RAPPELS SUR LA TRANSFORMATION DE FOURIER "CLASSIQUE"

Si on se place dans $L^2(R)$ par exemple, on définit $\hat{f}(t) = \frac{1}{\sqrt{2\pi}} \int_{\mathbb{R}} f(s) e^{-2i\pi st}\, ds$, ainsi que $f(t) = \frac{1}{\sqrt{2\pi}} \int_{\mathbb{R}} \hat{f}(s) e^{2i\pi st} ds$ en particulier pour $n \in \mathbb{Z}$, cela donne :

$$f(n) = \frac{1}{\sqrt{2\pi}} \int_{\mathbb{R}} \hat{f}(s) e^{2i\pi ns} ds \ . \qquad (1)$$

Si on développe $\hat{f}$ en série de Fourier, on obtient : $\hat{f}(t) = \sum_{-\infty}^{+\infty} a_n e^{-2i\pi nt}$ sur l'intervalle $(0,1)$ avec $a_n = \frac{1}{\sqrt{2\pi}} \int_{\mathbb{R}} \hat{f}(s) e^{2i\pi ns}\, ds$ en comparant avec (1), il vient $a_n = f(n)$, donc $\hat{f}(t) = \sum_{-\infty}^{+\infty} f(n) e^{-2i\pi nt}$.

Supposons maintenant que f soit à support borné, par exemple $f(t) = 0$ pour $|t| > N$, on obtient $\hat{f}(t) = \sum_{-\infty}^{+\infty} f(n) e^{-2i\pi nt} = \sum_{|n| \leqslant N} f(n) e^{-2i\pi nt}$. Si on échantillonne $\hat{f}$ aux points d'abscisses $\frac{k}{2N}$, il vient le théorème d'échantillonnage de La Vallée Poussin (attribué à Shannon) :

$$f(\frac{k}{2N}) = \sum_{-N}^{+N} f(n) e^{2i\pi nk/2N} , \qquad (2)$$

d'où on peut tirer les formules permettant de calculer exactement $\hat{f}(t)$ en un point t quelconque à partir des échantillons en $\frac{k}{2N}$. Ceci fait apparaître :

2.- LA TRANSFORMATION DE FOURIER DISCRÈTE

(2) peut aussi s'écrire : $A_k = \sum_{n=1}^{N} a_n w^{nk}$ avec $w = e^{2i\pi/N}$, k étant égal à 1,2,...,N. Si le vecteur de départ est $\underline{a} = (a_1, a_2, \ldots, a_N)$, le vecteur transformé est lui égal à $\underline{A} = (A_1, A_2, \ldots, A_N)$. Cette transformation jouit de propriétés analogues à la transformation ordinaire, à savoir :

- elle possède une inverse : $a_n = 1/N \sum_{k=1}^{N} A_k w^{-nk}$,
- elle diagonalise les translations de $\mathbb{C}^N$: soit $a' = (a_2, a_3, \ldots, a_N, a_o)$, on a $A'_k = w^{-k} A_k$,
- propriété de convolution : si $\underline{a}$ et $\underline{b}$ sont deux vecteurs et $\underline{A}, \underline{B}$ les transformés correspondants et si $\underline{C} = (C_1, C_2, \ldots, C_N)$ est tel que $C_k = A_k \cdot B_k$ alors $C_k = \sum_{n=1}^{N} c_n w^{nk}$ et on a $c_k = \sum_{n=1}^{N} a_n * b_{k-n} = a \cdot b$ c'est une convolution où les indices sont pris modulo N.

Les physiciens ont souvent besoin de calculer les transformées de Fourier de certaines fonctions, notamment en astrophysique pour l'analyse spectrale ou autres ...

Dans la pratique électronicienne il est une opération fort répandue, c'est le filtrage qui lui nécessite des calculs de convolutions circulaires ou non. A priori, le calcul de la convolée de deux suites ne semble pas simplifié par la transformation de Fourier discrète, en effet, pour chaque composante un calcul direct nécessite N multiplications scalaires et N-1 additions, c'est-à-dire de l'ordre de N^2 opérations au total, cependant que la transformation de Fourier discrète $\underline{A} = W\underline{a}$ où

$$W = \begin{pmatrix} w & w^2 & \ldots\ldots & w^{N-1} & 1 \\ w^2 & w^4 & \ldots\ldots & w^{2(N-1)} & 1 \\ \ldots\ldots & \ldots\ldots & \ldots\ldots & \ldots\ldots & \ldots\ldots \\ 1 & 1 & \ldots\ldots & 1 & 1 \end{pmatrix}$$

c'est-à-dire $W = ((w^{ij}))_1$ nécessite, par un calcul direct déjà N^2 opérations ce qui semble exclure un algorithme où on calculerait les deux transformées de Fourier des deux suites que l'on multiplierait terme à terme, puis où on obtiendrait le résultat en calculant la transformée inverse de ce produit et pourtant ...

3.- TRANSFORMATION DE FOURIER RAPIDE (Fast Fourier Transform)

Si nous cherchons à calculer le produit de W par le vecteur **a** en moins d'opérations, on peut regarder s'il n'y aurait pas dans W des lignes ou des morceaux de ligne proportionnels; pour cela recherchons les mineurs d'ordre deux nuls de W, il vient : $w^{i_1 j_1} w^{i_2 j_2} - w^{i_1 j_2} w^{i_2 j_1} = 0$.

Cette condition devient, sachant que $w^N = 1$: $(i_2 - i_1).(j_2 - j_1) = 0 \pmod N$. Donc si N est non premier, ils existent .

Exemple : si $N = r.s$, on peut prendre $i_2 = i_1 + m\,r$ et $j_2 = j_1 + p\,s$, c'est-à-dire que l'on prend des sous-suites de lignes et de colonnes régulièrement espacées.

Etudions plus en détail le cas particulier où $N = 2^q$:

$$A_k = \sum_{n=1}^{N} (a_{2n} w^{2nk} + a_{2n-1} w^{(2n-1)k})$$

$$A_k = \sum_{n=1}^{N} a_{2n} w^{2nk} + w^{-k} \sum_{n=1}^{N} a_{2n-1} w^{2nk} .$$

Ici $j_2 - j_1 = 2$; on prend donc $i_2 - i_1 = 2^{N/2}$, on obtient :

$$A_{k+N/2} = \sum_{n=1}^{N/2} a_{2n} w^{2n(k+N/2)} + w^{-(k+N/2)} \sum_{1}^{N/2} a_{2n-1} w^{2n(k+N/2)}$$

$$= \sum_{n=1}^{N/2} a_{2n} w^{2nk} + w^{-(k+N/2)} \sum_{n=1}^{N/2} a_{2n-1} w^{2nk} \quad .$$

Ici, on remarque que w^2 est racine N/2ème de 1. Par conséquent calculer la transformée de Fourier en N points se ramène au calcul de deux transformées de Fourier en N/2 points. En effet k varie de 1 à N/2 ainsi que n.

Si T(N) est le nombre d'opérations nécessaires au calcul de la transformation de Fourier discrète à l'ordre N, on obtient pour $N = 2^q$ la récurrence : $T(N) = 2T(N/2) + (3/2)N$ car il faut 2 additions et 1 multiplication pour chaque couple de lignes :

$$\begin{cases} A_k = A'_k + w^k A''_k & \text{et} \\ A_{k+N/2} = A'_k - w^k A''_k & \text{pour } k = 1,2,\ldots,N/2. \end{cases}$$

Ecrivons les récurrences :

$$\begin{aligned} T(N) &= 2T(N/2) + \frac{3N}{2} \\ &\cdots\cdots \\ T(\tfrac{N}{2}p) &= 2T(\tfrac{N}{2}p + 1) + \tfrac{3}{2}N/2^p \\ &\cdots\cdots \\ T(1) &= 0 \end{aligned}$$

ce qui donne, en multipliant par des puissances de deux bien choisies : $T(N) = (3/2) \sum_{1}^{q} N = (3/2)q \cdot N$, c'est-à-dire : $T(N) = (3/2)N \cdot \log N$, soit $(1/2)N \cdot \log N$ multiplications et $N \log N$ additions. En général, si N n'est pas premier et égal à $P_1^{\alpha_1} \cdot P_2^{\alpha_2} \cdots P_r^{\alpha_r}$, on trouve par un algorithme analogue dû à Cooley et Tukey un nombre d'opérations de $3N(\sum_{1}^{r} \alpha_i (P_i - 1))$. On peut écrire, en se ramenant éventuellement au cas particulier pour l'échantillonnage, que pour N quelconque, on a $T(N) = O(N \log N)$ par la transformation de Fourier rapide. Cela change tout.

Exemples :

$N = 1.000$

$N^2 = 1.000.000$

$N \log N \not\approx 10.000$

$N = 16.000$

$N^2 = 256.10^6$

$N \log N = 230.000.$

Le temps pris par la transformation de Fourier peut être négligeable par exemple pour les calculs de convolution, de multiplications de polynômes, de multiplications de nombres où on compte les opérations par "bit".

4.- BORNES INFÉRIEURES

La question se pose de savoir si le calcul de la Transformation de Fourier Discrète peut se faire plus rapidement ou si l'algorithme rapide ci-dessus est optimal.

Les théorèmes généraux connus sur les bornes inférieures de Winograd et Fiduccia ne donne qu'une minoration en O(N).

Algorithmes linéaires.- Si nous considérons des algorithmes de la famille suivante (ce qui dans ce cas n'est pas une restriction) où chaque étape est de la forme :

$$F_{i+1} = F_i \cup (\lambda_i f + \mu_i g)$$

dans lequel F_i est une famille finie de formes linéaires f et g sont dans F_i , λ_i et μ_i sont dans $\mathbb{C}$, au départ nous avons F_o = (les constantes + les coordonnées), à l'arrivée $A \subseteq F_m$, à la même étape, la famille A est calculée. En règle générale on cherche une fonction $\ell(F)$ telle que :

(1) $\ell(F_o) = 0$

(2) $\ell(F') = \ell(F)$ si $F' = F \cup (f)$ pour $f \in F$

(3) $\ell(F') \geqslant \ell(F)$ si $F' = F \cup (g)$, g quelconque

(4) $\ell(F_{i+1}) \leqslant \ell(F_i) + k$ où k est une constante positive.

De cela on déduit simplement par récurrence que :

$$m\,k \geqslant \ell(F_m) \geqslant \ell(A),$$

ce qui donne une borne inférieure au nombre d'opérations m de l'algorithme calculant A.

Considérons le déterminant de plus grand module que l'on peut extraire du tableau des coefficients de la famille F; soit $\Delta(F)$ son module; on a $\Delta(F_o) = 1$. Il est invariant et non décroissant selon les propriétés (2) et (3). De plus on a, soit :

$$\Delta(F_{i+1}) = \Delta(F_i)$$

soit, comme

$$\begin{aligned}\Delta(F_{i+1}) &= |\det(\ldots, \lambda_i f + \mu_i g)| \\ &= |\lambda_i \det(\ldots, f) + \mu_i \det(\ldots, g)| \\ &\quad 2\,\mathrm{Max}(|\lambda_i|, |\mu_i|)\,\Delta(F_i)\end{aligned}$$

d'où : $\Delta(F_{i+1}) \leqslant 2M\,\Delta(F_i)$.

De là on conclut que la fonction $\log(\Delta(F))$ satisfait aux propriétés (1), (2), (3) et (4) avec :

$$\log \Delta(F_{i+1}) \leqslant \log 2M + \log \Delta(F_i)$$

ce qui donne :

$$m \geqslant \frac{\log \Delta(F_m)}{\log 2M} .$$

Dans le cas de la transformation de Fourier $|\det W| = (\sqrt{N})^N$

$\log \Delta(A) = (1/2)N \log N$, si $M < N^s$ pour tout $s < \ell$, on obtient une borne inférieure <u>non linéaire</u>. En particulier pour $M = 1$, on a $O(N \log N)$.

Dans le cas général on ne sait pas si M est borné indépendamment de N, on peut essayer d'utiliser des propriétés analogues aux précédentes en prenant comme fonction $\ell(F)$ la fonction

$$\delta = \log \mathrm{Max} \frac{|\det(f_1, f_2, \ldots, f_n)|}{\|f_1\| . \|f_2\| \ldots\ldots \|f_n\|} \quad \text{où } \|f_i\| = \sup_j |f_{ij}|$$

où f_{ij} est le coefficient de la $j^{ième}$ coordonnée dans la forme f_i. On remarque alors que ℓ est invariante par multiplication d'une forme f_k par un scalaire ; en somme on a normalisé les formes. Mais à ce moment la condition de linéarité du déterminent n'est plus satisfaite et la condition (4) non plus en général. Néanmoins si :

$$* \qquad \|\lambda f + \mu g\| \geqslant \begin{matrix} \|\lambda f\| \\ \text{et} \\ \|\mu g\| \end{matrix}$$

on a :

$$\delta_{i+1} = \frac{|\det(\ldots, \lambda f + \mu g)|}{\ldots \|\lambda f + \mu g\|} \leqslant \frac{|\det(\ldots, f)|}{\ldots \|f\|} + \frac{|\det(\ldots, g)|}{\ldots \|g\|}$$

c'est-à-dire $\delta_{i+1} \leqslant 2\ \delta_i$, ce qui entraîne $m \geqslant \log \Delta(F)$ où F est la famille à calculer.

Quels sont les cas où * est satisfaite ? Par exemple :

- si f et g n'ont pas de variable en commun,
- si les coefficients sont positifs,
- ...

sinon on a seulement $\|\lambda f + \mu g\| \geqslant |\ \|\lambda f\| - \|\mu g\|\ |$ et on doit chercher les conditions où le second terme est supérieur ou égal à $\alpha\|f\|$ pour une constante α par exemple. Ces questions sont ouvertes.

5.- TRANSFORMÉE DE MERSENNE

Soient p un nombre premier et F_p le corps premier correspondant. Soit g une racine primitive de p : $g^{p-1} = 1$ et $g^k \neq 1$ pour $0 < k < p-1$. On peut définir la transformée aux racines primitives (de Fourier en fait) par :

$$A_k = \sum_{n=1}^{p} a_n\, g^{nk}$$

et la transformée inverse

$$a_n = (p-1) \sum_{k=1}^{p} A_k\, g^{-nk}$$

(sachant que $(p-1)^2 = 1 (\text{mod}.p)$).

Cette transformée jouit des propriétés de convolution mod(p - 1) : si $c_m = \sum_{n=1}^{p} a_n \cdot b_{(m-n)} = a * b$ où $(m - n) = m - n \bmod(p)$ et si $A = W \cdot a$ et $B = W \cdot b$ et si $\in = W \cdot c$ alors : $C_k = A_k \cdot B_k$. Par exemple si $p = 7$ et $g = 3$, on obtient :

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

Les nombres de Mersenne

Ce sont les nombres premiers de la forme $p = 2^q - 1$, ce qui entraîne que q est premier, d'ailleurs considérons la représentation binaire d'un entier $a < 2^q$: $a = \sum_{o}^{q-1} a_i 2^i$ avec $a_i = 0$ ou 1 ; si $\bar{a}_i$ est le complément de a_i $\bar{a} = \sum_{o}^{q-1} \bar{a}_i \, 2^i$ vérifie $a + \bar{a} = 0 \pmod p = 2^q - 1$ ce qui entraîne que la retenue de l'addition se met en tête car $2^q = 1$. Cela est très simple à câbler dans une machine.

Les multiplications par une puissance de 2 correspondent à des décalages dans la représentation binaire.

On sait qu'il existe g racine primitive telle que $g^{(p-1)/q} = 2$, d'où $p - 1 = 2^q - 2$ est divisible par q (th de Fermat). On peut donc chercher $A_k = \sum_{n=1}^{q} a_n (g^{(p-1)/q})^{nk}$, c'est-à-dire $A_k = \sum_{1}^{q} a_n 2^{nk}$ et $a_n = R \sum_{k=1}^{q} A_k 2^{-nk}$ avec $R \cdot q = 1$. Cette transformée requiert $q(q - 1)$ additions dont q seulement ne pourraient être simplifiées par un bon câblage de l'ordinateur spécifique éventuel et elle ne requiert aucune multiplication. Par contre, comme q est premier il ne correspond pas de technique de transformation de Fourier Rapide. Signalons que cette transformation peut s'effectuer sur un anneau.

VI.- TRANSFORMÉE DE FERMAT

Les nombres de Fermat sont de la forme $p = 2^{2^t} + 1$

t	0	1	2	3	4
p	3	5	17	257	65537

ceux-ci sont premiers. L'arithmétique est un peu moins simple que dans le cas de Mersenne. On a toujours $A_k = \sum_1^p a_n g^{nk}$. On peut aussi définir $A_k = \sum_1^{2q} a_n (g^{(p-1)/2q})^{nk}$ avec $q = 2^t$. La longueur de la suite est $2q$.

Soit aussi $A_k = \sum_1^{2q} a_n 2^{nk}$. Dans ce cas on peut utiliser des techniques de Transformation de Fourier Rapide et on aboutit à :

$q \log 2q$ décalages

$2q \log 2q$ additions

0 multiplication.

Ces techniques sont en voie de réalisation pratique.

*
* *

Lire : AHO HOPCROFT ULLMAN : The Design and Analysis of Computer Algorithms.

Addison Wesley 1974.

Jacques MORGENSTERN
Département de Mathématiques
Université de Nice
Parc Valrose
06034 NICE CEDEX