Chapitre 4

Transformation de Fourier discrète

Sommaire du chapitre

Ce chapitre est un préliminaire aux chapitres qui suivent. Avant d'aborder les filtres, qui permettent de travailler et de modifier la transformée de Fourier, en d'autres termes le *spectre* d'un signal, il est nécessaire d'indiquer comment calculer effectivement ce spectre. Puisque les signaux auxquels on s'intéresse sont discrets (car numériques), on se concentre ici sur la transformée de Fourier discrète. Son calcul, si il est fait naïvement, peut être très coûteux. Il peut cependant être accéléré considérablement en utilisant une méthode due à Tuckey [1] et Cooley [2], le fameux algorithme *FFT* , dont le nom est l'acronyme correspondant à *Fast Fourier Transform.*

1. James Cooley (1926-2016) était un mathématicien americain.
2. John Wilder Tukey (16 juin 1915, New Bedford, Massachusetts - 26 juillet 2000, New Brunswick, New Jersey.) était un statisticien américain.

4.1 La TFD

On commence par rappeler le cadre discret où l'on se place.

4.1.1 Cadre et problèmes

On considère un signal échantillonné, fini :

$$s = (s_0, s_1, ..., s_{N-2}, s_{N-1}),$$

obtenu à partir d'un signal continu [3], également noté s, par une formule du type :

$$s_n = s(n\Delta t),$$

où Δt est le pas de l'échantillonnage, relié à la fréquence d'échantillonnage $2B$ du chapitre 1 par l'équation :

$$\Delta t = \frac{1}{2B}.$$

On ne tient pas compte ici du fait que le signal peut avoir été quantifié, c'est-à-dire que les s_n sont considérés a priori comme réels.

Définition 4.1 — Transformée de Fourier Discrète

On appelle Transformée de Fourier Discrète, en abrégé TFD, de la suite finie $s = (s_0, s_1, ..., s_{N-2}, s_{N-1})$, la suite finie

$$\hat{s} = (\hat{s}_0, \hat{s}_1, ..., \hat{s}_{N-2}, \hat{s}_{N-1})$$

définie par la formule :

$$\hat{s}_k := \sum_{n=0}^{N-1} s_n \mathrm{e}^{-2\mathrm{i}\pi k\frac{n}{N}}.$$

Comme toute Transformée de Fourier, on voit que la TFD est une application linéaire. Sa matrice est une *matrice pleine*, c'est-à-dire ne comportant pas de 0, de type matrice de Vandermonde. On voit de plus que le pas de temps utilisé n'apparaît pas dans la formule. On peut en quelque sorte considérer que celui-ci est égal à 1. Ceci est en fait naturel : un signal discret est une suite de nombres et seules des informations supplémentaires sur ce qu'elle représente permettent de connaître la fréquence qui a été utilisée pour obtenir l'échantillon. On va s'intéresser à deux questions dans la suite :

3. Autrement dit analogique.

1. Quel est le coût de calcul de la TFD ? et comment l'améliorer ?
2. Quelle est la qualité de l'approximation de la transformée de Fourier du signal analogique initial ?

4.1.2 Propriétés de la TFD et signaux périodiques discrets

Pour simplifier l'exposé, on introduit de nouvelles notations. Soit une suite finie $a = (a_0, a_1, \ldots, a_{N-2}, a_{N-1})$, et $\omega_N = \mathrm{e}^{-\mathrm{i}2\pi/N}$. Alors la suite $A = (A_0, A_1, \ldots, A_{N-2}, A_{N-1})$, définie par

$$A_m = \sum_{n=0}^{N-1} a_n \omega_N^{nm} \tag{4.1}$$

est la TFD de a. Sous forme matricielle, cela s'écrit :

$$\begin{pmatrix} A_0 \\ A_1 \\ \vdots \\ \vdots \\ A_{N-1} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & \cdots & \cdots & 1 \\ 1 & \omega_N^1 & \omega_N^2 & \cdots & \cdots & \omega_N^{N-1} \\ 1 & \omega_N^2 & \omega_N^4 & \cdots & \cdots & \omega_N^{2(N-1)} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \omega_N^N & \omega_N^{2(N-1)} & \cdots & \cdots & \omega_N^{(N-1)^2} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ \vdots \\ a_{N-1} \end{pmatrix},$$

et l'on notera la matrice de Vandermonde Ω_N.

Théorème 4.1 — Formule d'inversion de la TFD — En gardant les notations précédentes, on a :

$$a_n = \frac{1}{N} \sum_{m=0}^{N-1} A_m \omega_N^{-nm}.$$

Évidemment, on retrouve dans la TFD et sa formule d'inversion les analogies habituelles entre les transformées de Fourier et leurs inverses. Dans le cas présent, on retrouve simplement que l'inverse d'une matrice de Vandermonde est une matrice de Vandermonde [4], c'est-à-dire $\Omega_N^{-1} = 1/N \times \overline{\Omega_N}$.

Remarque 4.1 :

Cette formule reste valable si on considère *les extensions périodiques* de a et A, définies par les formules :

$$a_{n+kN} = a_n, \quad A_{m+kN} = A_m.$$

C'est maintenant comme cela que l'on envisagera les choses. Tous les signaux

4. proportionnelle à une matrice de Vandermonde, pour être plus précis.

considérés seront vus dorénavant comme des signaux discrets périodiques.

Par conséquent, l'ordre de sommation peut-être choisi arbitrairement. En supposant par exemple que $N = 2M + 1$, on obtient :

$$A_m = \sum_{n=0}^{N-1} a_n \omega_N^{nm}, \quad a_n = \frac{1}{2M+1} \sum_{m=-M}^{M} A_m \omega_N^{-nm}.$$

Pour simplifier encore, on notera dans la suite ces relations en abrégé par :

$$(a_n) \overset{\mathcal{F}_N}{\longleftrightarrow} (A_m).$$

Théorème 4.2 Si $(a_n) \overset{\mathcal{F}_N}{\longleftrightarrow} (A_m)$ et $(b_n) \overset{\mathcal{F}_N}{\longleftrightarrow} (B_m)$, alors :

$$(\sum_{k=0}^{N-1} a_k b_{n-k}) \overset{\mathcal{F}_N}{\longleftrightarrow} (A_m B_m).$$

Exercice 4.1 :

Montrer ce résultat.

Ce théorème est une variante de ceux que nous avions déjà vus au chapitre 1 sur le lien entre séries de Fourier et convolution.

4.2 L'algorithme FFT

En appliquant la formule (4.1), on est conduit à calculer la somme suivante :

$$A_m = a_0 + a_1 \omega_N^m + a_2 \omega_N^{2m} + \ldots + a_{N-1} \omega_N^{m(N-1)}.$$

Si on ne retient que les multiplications et en considérant que la suite $(\omega_N^{km})_{k=0,\ldots,N-1}$ a été pré-calculée [5], on est conduit à effectuer $N - 1$ opérations. Le calcul complet de la suite A nécessite donc $(N-1)^2$ opérations, ce qui n'est pas satisfaisant du tout dès lors que N devient « assez » grand.

5. c'est-à-dire calculée une fois pour toute, en supposant que l'entier N est fixé.

4.2.1 L'algorithme de Tuckey et Cooley

Publié en 1965 dans un article devenu depuis célèbre, Tuckey et Cooley ont donné une méthode permettant de réduire considérablement le temps de calcul de la suite A précédente.

Première itération

On va montrer comment réduire à $N \log_2 N$ le nombre de multiplications. On se place dans le cas où la taille des suites considérées est paire et on considère

$$(a_n) \overset{\mathscr{F}_{2N}}{\longleftrightarrow} (A_m).$$

On introduit alors deux suites $(b_n)_{n=0,\ldots,N-1}$ et $(c_n)_{n=0,\ldots,N-1}$ définies par $b_n = a_{2n}$ et $c_n = a_{2n+1}$ et on note :

$$b_n \overset{\mathscr{F}_N}{\longleftrightarrow} B_m, \quad c_n \overset{\mathscr{F}_N}{\longleftrightarrow} C_m.$$

On remarque alors que :
— $A_m = B_m + \omega_{2N}^m C_m$, pour $m = 0, \ldots, 2N-1$,
— $B_{m+N} = B_m$, $C_{m+N} = C_m$ et $\omega_{2N}^{m+N} = -\omega_{2N}^m$.
On en déduit :

$$A_m = B_m + \omega_{2N}^m C_m \qquad m = 0, ..., N-1 \tag{4.2}$$

$$A_{m+N} = B_m - \omega_{2N}^m C_m \qquad m = N, ..., 2N-1. \tag{4.3}$$

On a vu que le calcul de B et C nécessitait $2(N-1)^2$ opérations. Le calcul de A peut alors être fait à partir de B et C par $N-1$ opérations supplémentaires. De plus, on voit qu'ayant calculé tous les termes (4.2), on dispose sans calcul supplémentaire de tous les termes (4.3). Le coût total est donc :

$$2(N-1)^2 + N - 1 = (N-1)(2N-3),$$

au lieu de $(2N-1)^2$, ce qui fait qu'on a divisé le temps de calcul par, *grosso-modo*, deux.

Cas général

Mais on peut aller encore plus loin ! Il suffit pour cela d'appliquer récursivement le raisonnement précédent. Supposons pour ce faire que $N = 2^s$ et notons $F(N)$ le nombre d'opérations nécessaire au calcul de la TFD d'une suite de taille N, c'est-à-dire d'un signal N-périodique.

Théorème 4.3 Il existe un algorithme de calcul de TFD vérifiant :

$$F(N) \leq \frac{1}{2} N \log_2 N.$$

Preuve : En appliquant la méthode présentée dans l'exemple, on obtient :

$$F(2N) = 2F(N) + N - 1 \leq 2F(N) + N.$$

Supposons que $F(N) \leq \frac{1}{2} N \log_2 N$, alors

$$2F(N) + N \leq N(\log_2 N + 1) = \frac{1}{2} 2N \log_2(2N),$$

ce qui achève la démonstration par récurrence.

Pour illustrer ce résultat, considérons une suite de $N = 1024$ termes. Par la méthode naïve expliquée en introduction de cette section, le calcul nécessite $(N-1)^2 = 1\,046\,529$ opérations. Par l'algorithme FFT de Tuckey et Cooley, il nécessite moins de $\frac{1}{2} N \log_2 N = 5120$ opérations. Le rapport du temps de calcul entre les deux méthodes est donc d'à peu près 200. Autrement dit, ce qui nécessitait presque 7 mois ne requiert maintenant qu'une journée.

Si la taille de l'échantillon n'est pas une puissance de 2, ce qui n'arrive pas en traitement numérique du signal, on complète généralement l'échantillon par des zéros pour pouvoir appliquer l'algorithme.

4.3 Analyse numérique

À ÉCRIRE

4.4 Notes bibliographiques

Ce chapitre s'inspire de la présentation de la FFT faite dans la référence [3], qui ne contient pas l'application aux polynômes. On pourra également consulter les chapitres 7 et 8 de la référence [6] pour une autre présentation de l'algorithme, moins mathématique. Ici encore, de nombreux textes et documents concernant la FFT sont disponibles sur Internet.