Transformée de Fourier

Année 2007

Table des matières

1 Définitions

Signal : On appelle *signal* une fonction du temps, à support compact. Représentation du signal physique : tension, intensité, ...

Transformée de Fourier : Sous réserve d'existence, on définit l'application suivante, nommée *Transformée de Fourier* :

$$\mathfrak{F}(x) \text{ aussi notée } \widehat{x} : \nu \longmapsto \int_{-\infty}^{\infty} x(t) * e^{-2i\pi\nu t}\, dt \tag{1}$$

$\rightsquigarrow$ Lourdeur du calcul : nécessite de connaître les valeurs du signal *à chaque instant.*

2 Transformée de Fourier Discrète

On montre que l'on peut *discrétiser* cette transformation :

$$\forall k \in [|0..N-1|], \quad \widehat{x_k} = X_k = \sum_{j=0}^{N-1} x_j e^{-i\frac{2\pi jk}{N}} \tag{2}$$

Remarque : Elle est inversible, on montre que :

$$\forall k \in [|0..N-1|], \quad x_k = \frac{1}{N} \sum_{j=0}^{N-1} X_j e^{i\frac{2\pi jk}{N}} \tag{3}$$

Moyens de calcul de cette transformation ?

3 Algorithmes

3.1 Décomposition naïve

On utilise la définition (2).

Complexité : $O(N^2)$: Algorithme long dès que N est grand.

3.2 Calcul récursif

On décompose la somme de N termes en deux sommes de $N/2$.

Complexité : $O(N \log_2(N))$.

Incovénient : Nécessite une mémoire conséquente pour N grand (appel récursif).

3.3 Calcul itératif - Exemple pour $N = 2^3$

Idée Décomposition des indices en base 2.
On note : $\omega = \mathrm{e}^{-i\frac{2\pi}{8}}$.

$$\forall k \in [|0..N-1|], \quad X_k = \sum_{j=0}^{N-1} x_j \omega^{jk}$$

On décompose j et k en base 2 :

$$\begin{cases} j &= j_2 \times 2^2 + j_1 \times 2^1 + j_0 \times 2^0 \\ k &= k_2 \times 2^2 + k_1 \times 2^1 + k_0 \times 2^0 \end{cases} \quad \forall i \in \{0,1,2\},\ j_i \in \{0,1\} \text{ et } k_i \in \{0,1\}$$

Alors :

$$\omega^{jk} = \omega^{4k_0 j_2} \cdot \omega^{(4k_1+2k_0)j_1} \cdot \omega^{(4k_2+2k_1+k_0)j_0}$$

On reécrit X_k sous la forme :

$$X_k = X(4k_2+2k_1+k_0) = \sum_{j_0=0}^{1} [\sum_{j_1=0}^{1} [\sum_{j_2=0}^{1} x(4j_2+2j_1+j_0)\omega^{4k_0 j_2}]\omega^{2j_1(2k_1+k_0)}]\omega^{j_0 k}$$

Notations

- $x_1(4k_0+2j_1+j_0) = \sum_{j_2=0}^{1} x(4j_2+2j_1+j_0)\omega^{4k_0 j_2}$
  $= x(2j_1+j_0) + x(4+2j_1+j_0)\omega^{4k_0}$
- $x_2(4k_0+2k_1+j_0) = \sum_{j_1=0}^{1} x_1(4k_0+2j_1+j_0)\omega^{2j_1(2k_1+k_0)}$
  $= x_1(4k_0+j_0) + x_1(4k_0+2+j_0)\omega^{2(2k_1+k_0)}$
- $x_3(4k_0+2k_1+k_2) = \sum_{j_0=0}^{1} x_2(4k_0+2k_1+j_0)\omega^{j_0 n}$

$$\boxed{X(4k_2+2k_1+k_0) = x_3(4k_0+2k_1+k_2)}$$

Complexité : $O(N\log_2(N))$, identique à l'algorithme récursif.

Avantage : Ne nécessite pas autant de mémoire. En théorie, il semble être le plus efficace.

4 Résultats

5 Application : Multiplication de polynômes

5.1 Notations

P et Q, polynômes de degré $n-1$, et $R = PQ$; R est donc de degré $2n-2$.

$$P = \sum_{i=0}^{n-1} p_i X^i \,;\; Q = \sum_{i=0}^{n-1} q_i X^i \text{ et } R = \sum_{i=0}^{2n-1} r_i X^i \text{ avec } r_{2n-1} = 0$$

Idée : On va calculer deux transformées de Fourier à $N = 2n$ éléments.

$$\begin{cases} p = (p_0, p_1, \ldots, p_{n-1}, \underbrace{0, \ldots, 0}_{n \text{ termes}}) \\ q = (q_0, q_1, \ldots, q_{n-1}, \underbrace{0, \ldots, 0}_{n \text{ termes}}) \\ r = (r_0, r_1, \ldots, r_{n-1}, r_n, \ldots, r_{2n-2}, 0) \end{cases}$$

$\omega = \mathrm{e}^{-i\frac{2\pi}{N}}$, racine primitive *N-ième* de l'unité.

5.2 Calcul des $\widehat{p_i}$, $\widehat{q_i}$

D'après la formule (2), on a :

$$\triangleright \quad \forall k \in [|0 \ldots 2n-1|], \quad \widehat{p_k} = \sum_{j=0}^{2n-1} p_j \omega^{jk} = P(\omega^k)$$

$$\triangleright \quad \forall k \in [|0 \ldots 2n-1|], \quad \widehat{q_k} = \sum_{j=0}^{2n-1} q_j \omega^{jk} = Q(\omega^k)$$

$$\triangleright \quad \forall k \in [|0 \ldots 2n-1|], \quad \widehat{r_k} = \sum_{j=0}^{2n-1} r_j \omega^{jk} = R(\omega^k)$$

Il vient :

$$\forall k \in [|0 \ldots 2n-1|], \quad \widehat{r_k} = (P \times Q)(\omega^k) = P(\omega^k) \times Q(\omega^k) = \widehat{p_k} \times \widehat{q_k}$$

On retrouve ensuite les coefficients de R par application de la Transformée Inverse.

Avantage : Un calcul classique se fait en $O(n^2)$. Ici, on peut réduire à $O(N \log_2(N))$!