Transformée de fourier rapide et convolution

Jean-Christophe Toussaint

10 juin 2025

La technique usuelle pour déterminer les fréquences caractéristiques ainsi que l'amplitude des harmoniques qui composent un signal, consiste à calculer sa transformée de fourier. Dans une approche numérique, cette opération est effectuée sur un ensemble de valeurs échantillonnées.

Le but de l'exercice est dans un premier temps, d'utiliser ici le module scipy.fft pour calculer les transformées de fourier rapides directe et inverse dont la complexité est en $\vartheta(N \ln N)$, puis d'effectuer des produits de convolution en utilisant le module scipy.signal.

1 Définitions mathématiques

La transformée directe d'un signal périodique s échantillonné en N points est définie par :

$$ft[k] = \sum_{n=0}^{N-1} s[n] \exp\left(-i\frac{2\pi k}{N}n\right) \tag{1}$$

où $k \in [\![0, N-1]\!]$ avec $i^2 = -1$.

Sa transformée inverse est définie par :

$$s[n] = \frac{1}{N} \sum_{k=0}^{N-1} ft[n] \exp\left(+i\frac{2\pi k}{N}n\right) \tag{2}$$

C'est en 1965 que James Cooley et John Tukey redécouvrent une méthode de factorisation de ces sommes discrètes, déjà imaginée par Gauss en 1805, qui permettent de réduire à $\vartheta(N \ln N)$, le nombre d'opérations. Cette technique porte le nom de transformée de fourier rapide (Fast Fourier Transforms).

Le produit de convolution de deux signaux échantillonnés en N points, est définie par :

$$(f * g)[p] = \sum_{m=0}^{n-1} f_m . g_{p-m} \tag{3}$$

où $p - m \equiv p - m + n \mod n$ avec $p \in [0, n[$.

On montre aussi que la transformée de fourier d'un produit de convolution est égale au produit des transformées de fourier.

2 Exercice : transformée de Fourier d'un sinus

On veut calculer la transformée de fourier discrète d'un signal sinusoïdal périodique $G(t) = sin(\frac{2\pi}{T}t)$. On fixera $t \in [0, T[$.

Il est nécessaire d'importer les modules numpy, matplotlib et scipy.fft :

```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
from scipy.fft import fft, fftfreq
```

1. Echantillonner le domaine $[0, T = 1[$ en $N = 100$ points en utilisant la fonction numpy `np.linspace`. Utiliser la commande help(cmd) pour obtenir une aide en ligne sur tout objet ou fonction ou module.
2. Utiliser les fonctions `fft.fft` et `fft.ifft` pour calculer les transformées directe et inverse.
3. Tracer le signal et les composantes réelle et imaginaire de sa transformée de fourier.
4. Vérifier que l'on reconstruit bien le signal initial $G(t)$ en appliquant la transformée inverse sur le spectre de fourier.

3 Exercice : transformée de Fourier d'une gaussienne

Faites de même sur un signal gaussien $G(t) = \exp(\frac{-t^2}{2\sigma^2})$. On prendra $\sigma = 0.1$ et on fixera $t \in [-T/2, T/2[$.

1. Echantillonner le domaine $[-T/2, T/2[$ en $N = 100$ points en utilisant la fonction numpy `np.linspace`.
2. Avant d'appliquer la transformée de fourier, il est nécessaire de décaler vers la gauche le tableau contenant G de $N/2$ cases en utilisant la fonction `np.roll`. Une explication vous êtes donnée en annexe I.
3. Utiliser les fonctions `fft.fft` et `fft.ifft` pour calculer les transformées directe et inverse.
4. Tracer le signal et les composantes réelle et imaginaire de sa transformée de fourier.

4 Convolution de deux signaux

On définit ci-après un signal carré comme une liste de 1 complétée à droite d'une même quantité de zéros. On utilise une méthode de zero-padding pour doubler la taille du tableau tout en le complétant de zéros. Cette technique a peu d'intérêt en 1D mais devient très utile pour les dimensions supérieures à 1.

```
sq=np.ones(5)
sq=np.pad(sq, (0, len(sq)), mode='constant') # (before, after)
```

1. Développer une fonction python `conv(f, g)` permettant de faire le calcul du produit de convolution discret d'après (3).
2. La fonction `numpy.convolve(sq, sq, mode='full')` effectue une convolution discrète au sens mathématique. Comparer avec le résultat précédent.
3. La fonction `scipy.ndimage.convolve1d` effectue le calcul du produit de convolution au sens du traitement d'image, sans retournement du noyau, ce qui ne correspond pas à la définition mathématique. Il faut donc retourner le noyau qui est passé en deuxième argument de `convolve1d(sq, sq[::-1], mode='constant', cval=0)` .
4. La fonction `scipy.signal.fftconvolve(sq, sq, mode='full')` du module
   `scipy.signal` permet d'obtenir directement le produit de convolution. Le zero-padding sur `sq` est appliqué en interne.
5. Vérifier que l'on obtient aussi la même résultat en utilisant la propriété sur la transformée de fourier d'un produit de convolution.

5 Convolution d'un signal par une gaussienne

Dans le cadre d'un modèle simpliste unidimensionnel, on imagine ici qu'une grandeur ϕ (nommée ci-après potentiel) est obtenue par le produit de convolution suivant :

$$\phi(x) = \int_{-L}^{L} q(y)G(x-y)dy = (q * G)(x) \tag{4}$$

où $q(x)$ est l'équivalent d'une distribution de charges et $G(x)$ est appelé le noyau. Pour fixer les idées, on prendra dans la suite un noyau gaussien $G(x) = \exp(\frac{-x^2}{2\sigma^2})$, défini partout.

Lorsque la distribution de charges est la somme de distributions de dirac, $q(x) = \sum_i q_i \delta(x - x_i)$, l'expression du potentiel ϕ s'écrit comme la somme de gaussiennes centrées en x_i :

$$\phi(x) = \sum_i q_i \exp\left(\frac{-(x-x_i)^2}{2\sigma^2}\right) \tag{5}$$

Le but de l'exercice est de reproduire ce résultat, en plaçant les charges q_i aux noeuds x_i d'une grille unidimensionnelle.

1. Discrétisée $[-L, L = 1]$ en $N = 100$ points d'espace en utilisant la fonction `linspace`
2. Définir le tableau numpy q de dimension N dans lequel on placera quelques charges non nulles.

3. Défnir le tableau numpy G de dimension N contenant l'échantillonnage de la fonction de Green. On prendra $\sigma = 0.1$.
4. Utiliser d'abord votre propre fonction `conv` calculant le produit de convolution sans optimisation.
5. Utiliser ensuite `filter.convolve1d(q, G, mode='constant')`. Comparer vos résultats sous la forme de graphiques.

On désire maintenant développer notre propre routine de convolution, utilisant les FFT du module `fft`. Montrer qu'il est nécessaire

1. de mettre en oeuvre la technique du zéro-padding aux tableaux q et G.
2. d'appliquer sur le tableau G un décalage à gauche de $N/2$ termes. Tracer G et commenter.
3. de calculer les FFT directes sur q et G.
4. d'appliquer une FFT inverse sur leur produit.
5. et finalement de ne garder que les N premiers éléments du tableau précédent, pour obtenir le produit de convolution désiré.

6 Annexe I : décalage vers la gauche

Son origine est liée à la définition de la transformée de fourier fenêtrée :

$$\hat{f}(\omega) = \int_{-T/2}^{+T/2} s(t) \exp(-i\omega t)\, dt \tag{6}$$

avec $\omega_k = \frac{2\pi}{T} k$ avec $k \in [\![0, N-1]\!]$.

La discrétisation en temps impose que $t = nT/N - T/2$ avec $n \in [\![0, N-1]\!]$. Le pas de temps dt est donc égal à T/N.

On obtient :

$$\hat{f}(\omega_k) = \frac{T}{N} \exp(+i\omega_k(T/2)) \sum_{n=0}^{N-1} s[n] \exp\left(-i\frac{2\pi k}{N} n\right) = \frac{T}{N} \exp(+i\omega_k(T/2))\, ft[k] \tag{7}$$