TD 4 : FFT avec Python

28 Octobre 2020

1 Exemple simple de DFT

Dans cet exemple, nous allons utiliser les fonctions fft (DFT), ifft (DFT inverse) et fftfreq (abscisse dans l'espace réciproque) de numpy.

L'objectif de cet exercice est d'étudier numériquement la transformée de Fourier d'une Gaussienne : $G(x) = \exp(-\frac{x^2}{2\sigma^2})$. Analytiquemet, la T.F. de G vaut $\tilde{G}(\omega) = \sqrt{2\pi}\sigma \exp(-\frac{\omega^2\sigma^2}{2})$

1. Écrire une fonction Python qui retourne une Gaussienne $\exp(-\frac{x^2}{2\sigma^2})$ sur [-L/2, L/2] et son axe des `x = L/N*np.arange(N) - L/2`, prenant comme paramètres σ, L et N.
2. Tracer la Gaussienne pour $\sigma = 0.5$ et $N = 1000$ et $L = 5$.
3. Tracer la DFT de la Gausienne pour $\sigma = 0.15$ pour une plage de fréquences inférieures à 20 : (a) Quel taux d'échantillonnage $\Delta x = L/N$ faut-il utiliser ? (b) Quelle valeur minimum de L faut-il choisir pour obtenir une résolution en fréquence de 0.04 ?
4. Comparer la courbe obtenue avec la formule analytique ? Pourquoi la DFT de la Gaussienne n'est pas possitive ?
5. Appliquer la transformée de Fourier inverse et vérifier que l'on obient bien le signal initial.

2 Les verres musicaux

Un verre d'eau partiellement rempli peut émettre une note distincte lorsque l'on frotte son bord avec un doigt mouillé. C'est l'ingrédient de base du Verrillon. Mais que cache son spectre ?

La vidéo Singing_Glass.MOV à été faite avec un verre à vin, de l'eau et un smartphone. Sa bande son à été extraite dans le fichier SingingGlass.wav.

1. Télécharger, puis importer la piste sonore SingingGlass.wav :

```
from scipy.io.wavfile import read
samplerate, amplitude = read('SingingGlass.wav')
```

2. Afficher le nombre de points N contenus dans ce signal, la durée de l'intervalle entre deux points et la durée totale de l'enregistrement.
3. Afficher le signal. Quelle est approximativement la fréquence du son ?
4. L'intensité du spectre à une fréquence donnée est proportionnel au carré du module de la transformée de Fourier. Affichez le spectre de SingingGlass.wav.
5. Zoomer autour de la fréquence du son. Que remarquez-vous ?
6. On a phénomène de battement que l'on entend à l'oreille. Qu'elle est la fréquence du battement ?

Il s'agit d'un phénomène méconnu mais relativement commun avec les verres musicaux, où deux modes de vibration orthogonaux sont excités. Ces deux modes n'ont pas exactement la même énergie due aux imperfections dans la symétrie du verre. La non-dégénérescence de ces modes dépend du niveau d'eau. Voir par exemple: https://newt.phys.unsw.edu.au/music/people/publications/Jundtetal2006.pdf

3 FFT 2D : reconstruire la phase d'une fonction d'onde

La transformée de Fourier peut être généralisée aux fonctions à n-dimension. Dans le cas 2D, on utilisera les fonctions fft2 et ifft2 du module fft de numpy (from numpy.fft import fft2, ifft2).
Attention : comme pour la T.F. 1D, la fréquence nulle se retrouve dans les coins. Il est plus commode pour la visualisation d'avoir le zéro de fréquence au centre de l'image, ce qui est réalisé avec les fonctions fftshift() et ifftshift(). Une transformée de Fourier avec le zéro des fréquence en son centre utilise donc fftshift(fft2()). Son inverse est ifft2(ifftshift()).

Dans ce TD, nous allons exploiter la transformée de Fourier pour retrouver la *phase* d'une fonction complexe (ex. un champ électromagnétique) à partir de données sur l'amplitude. On étudie une fonction a géométrie circulaire de type Laguerre-Gauss :

$$F(r,\phi) = e^{-\frac{r^2}{w_0}} \left(\frac{r\sqrt{2}}{w_0}\right)^{|\ell|} e^{i\phi\ell}, \tag{1}$$

où r et ϕ sont les coordonnées polaires et w_0 est la taille caractéristique du faisceau. Le nombre ℓ est un entier. Pour projeter l'information de la phase sur une intensité (mesurée par exemple avec une caméra), il est nécessaire de faire interférer $F(r,\phi)$ avec une onde plane, de forme mathématique:

$$P(x,y) = e^{i(k_x x + k_y y)}, \tag{2}$$

où x et y sont les coordonnées cartésiennes et k_x et k_y déterminent le gradient de phase selon x et y.

A - Interférence de deux ondes à moment angulaire non-nul. L'objectif de cette partie est de simuler le signal issu de l'interférence en F et P.

Pour les calculs on prendra $w_0 = 1$, $\ell = 1$, $k_x = \frac{2\pi}{0.1}$ et $k_y = 0$. On tracera les images sur une grille de 1024x1024 points centrée en 0 et de largeur $\pm 2w_0$. Cette grille est obtenue de la façon suivante :

```
w0 = 1
L=4*w0
N=1024
x0 = np.arange(N)*L/N - L/2
x,y = np.meshgrid(x0, x0)
```

1. Écrire une fonction python `Laguerre_Gauss(x, y, l, w)` pour le faisceau Laguerre et une fonction `plane_wave(x, y, kx, ky)` pour l'onde plane. La phase en coordonnée polaire peut s'obtenir à partir de `phi = np.arctan2(y,x)`.
2. Afficher l'intensité du signal d'interférence de ces deux ondes. On peut observer une figure d'interférence "en fourche" signifiant un déphasage de 2π autour du centre. Que se passe-t-il si ℓ est plus grand ?
3. Pour $\ell = 1$, affichez la DFT 2D de la figure d'interférence. **Attention** : la transformée de Fourier retourne un tableau complexe. Pour l'observer, il faudra ne regarder que l'amplitude (np.abs()). On utilisera la fonction plt.imshow pour tracer l'image en fausse couleur : la couleur de chaque pixel varie du bleu au rouge suivant la valeur du point. Par défaut, le bleu correspond au minimum et le rouge au maximum. Ces valeur peuvent être modifiées à l'aide des arguments optionnels `vmin` et `vmax`. On choisira 10^4 :

```
plt.imshow(np.abs(image_tilde), vmin=0, vmax=1E4)
```

Les deux pics à droite et à gauche du centre sont les fréquences (positives et négatives) venant du cosinus des franges d'interférence.

L'objectif est de retrouver l'amplitude du faisceau. L'idée est la suivante, que l'on va étudier à 1D, on note $a(x)$ l'amplitude complexe de l'image. La figure d'interférence sera donc $|a(x) + e^{ik_0 x}|^2$ soit $|a(x)|^2 + 1 + a(x)e^{ik_0 x} + \bar{a}(x)e^{-ik_0 x}$. Mutiplier par e^{ikx} revient dans l'espace de Fourier à décaler les fréquences. La TF comportera donc un terme en $\tilde{a}(k - k_0)$ et $\tilde{a}(k + k_0)$. Nous avons donc autour de k_0 et $-k_0$ la transformée de Fourier de a et donc toute l'information sur a y compris sa phase. Le but de la partie suivante est de reconstituer cette phase.

B - Nous allons maintenant utiliser une image expérimentale. Télécharger le fichier *Rotating_Superfluid.tif* de la page Github et chargez-le dans un array 2D numpy (ndarray) avec :

```
import matplotlib.image
image = matplotlib.image.imread("Rotating_Superfluid.tif")
plt.imshow(image)
```

Il s'agit d'un fluide quantique 2D en rotation, un vortex, possédant une phase $\phi(x, y)$. Il porte une charge topologique dont la valeur (un entier) correspond au nombre de fois où la phase 2π est accumulée le long d'un chemin tournant autour du coeur de ce vortex (dans l'exemple précédant, la charge topologique était ℓ). Pour pouvoir extraire la phase $\phi(x, y)$, on a fait interférer l'image du vortex avec un faisceau plan de référence.

1. Affichez les fréquences spatiales de cette image. Utilisez fftshift() pour que le centre de cette matrice corresponde aux fréquences nulles. Stockez cette DFT dans une variable nommée `image_tilde`. Pour réduire le contraste et rendre le signal plus visible à l'écran, on affichera préférentiellement le logarithme (np.log()) de la valeur absolue du signal.

2. Pour reconstruire la phase du fluide, on isolera l'un des deux pics latéraux en copiant la zone $[415 \pm 40, 500 \pm 40]$ de `image_tilde` dans un nouveau tableau. Placez ensuite ces valeurs au centre d'un tableau de nombres complexes peuplé de 0 et ayant la même taille que `image_tilde`.

3. Effectuez la FFT 2D inverse. Isolez l'amplitude (np.abs()) et la phase (np.angle()) dans deux tableaux de valeurs séparés et affichez-les.

4. Quel est l'enroulement de la phase autour du centre ? En déduire la charge topologique du vortex.

5. Tester aussi votre code de reconstruction de phase sur l'exemple numérique construit en **A**.

6. (Optionnel) La phase est définie modulo 2π avec la fonction angle(). Utilisez la fonction `unwrap_phase()` du module `restoration` du package `skimage` (`from skimage.restoration import unwrap_phase`) pour "déplier" la phase. Observez l'enroulement, affichez la charge topologique (amplitude crête-crête de la phase).