Feuille de TP n°2

Licence 3 MApI3, module Signal, Fourier, image

1 Transformée de Fourier et signaux audio

On donne le fichier `wavtools.py`, qui fournit quelques fonctions utiles pour la manipulation de fichiers audio au format WAV. On peut le trouver à l'adresse `http://www.math.univ-toulouse.fr/~vfeuvrie/l3mapi3`.

Dans ce qui suit, des données audio seront représentées sous la forme d'un tableau à une dimension contenant des nombres compris entre -1 et 1. Ces nombres décrivent une pression dans l'air (exprimée comme la différence par rapport à la pression au repos) au cours du temps. L'oreille perçoit ces variations de pression comme un son.

Listing 1 – Jouer un fichier wave

```
from wavtools import *

play_wave('test.wav')
```

Listing 2 – Chargement d'un fichier audio

```
signal, freq = open_wave(path)

#La donnée freq permet de connaître le
    nombre d'échantillons par seconde
    contenu dans le signal
```

Listing 3 – Enregistrement d'un fichier audio

```
from numpy import *

sound = zeros(88200)
save_wave(sound, 'test.wav', 44100)

#On enregistre un fichier WAVE à 44.1 kHz
    contenant 88200 échantillons à zéro. On
    obtiendra donc deux secondes de silence.
```

Exercice 1 – Son sinusoïdal. Dans ce qui suit, on travaille avec une fréquence d'échantillonnage des données audio de 30 kHz.

1) Générer un tableau contenant l'échantillonnage d'une fonction sinusoïdale à 440 pulsations (périodes) par seconde pendant 3 secondes.

2) Sauvegarder et écouter le fichier audio obtenu.

3) On rappelle que la transformée de Fourier discrète d'un signal v à N échantillons permet d'obtenir ses coordonnées dans la base $(\omega^k)_{0\leq k<N}$, avec $\omega_n^k = e^{2ik\pi\frac{n}{N}}$. Avec ces notations, le *mode* associé au coefficient $\hat{v}(k)$ est, par définition, l'indice k.

a) La fonction `fft(signal)` du paquetage `numpy.fft` renvoie la transformée de Fourier discrète d'un signal à une dimension. La fonction `abs` de `numpy` renvoie le module d'un tableau de complexes. Représenter le module de la transformée de Fourier du signal en ordonnée en fonction du mode en abscisse.

b) La fonction `fftfreq(N, d)` de la bibliothèque `numpy.fft` renvoie un tableau contenant les fréquences associées à la transformée de Fourier pour un signal contenant N échantillons espacés d'un intervalle d (d est donc l'inverse de la fréquence d'échantillonnage). Représenter les fréquences du signal sinusoïdal en ordonnée en fonction du mode en abscisse.

c) La fonction `fftshift` de `numpy.fft` effectue une permutation des éléments d'un tableau. Après l'avoir testée sur un tableau renvoyé par `fftfreq`, expliquer son intérêt pour la manipulation des coefficients de la transformée de Fourier.

4) Représenter le module de la FFT du signal en ordonnée en fonction des fréquences en abscisse.

5) Même question en remplaçant le signal sinusoïdal par un signal carré (on pourra utiliser l'expression $\text{sign}(\sin x)$, qui renvoie un signal carré de période 2π, compris entre -1 et 1).

6) Même question en remplaçant le signal sinusoïdal par un signal triangulaire (on pourra utiliser l'expression $\frac{2}{\pi}\arcsin(\sin x)$, qui renvoie un signal triangulaire de période 2π, compris entre -1 et 1).

7) Décrire qualitativement les résultats obtenus.

Exercice 2 – Deux sons sinusoïdaux.

1) Sur le modèle de l'exercice précédent, générer un fichier audio de deux secondes, contenant :
—un signal sinusoïdal de fréquence 440Hz pendant la première seconde ;
—un signal sinusoïdal de fréquence 600Hz pendant la deuxième seconde.

2) Représenter le module de la FFT du signal en fonction des fréquences.

3) Décrire qualitativement les résultats obtenus.

Exercice 3 – Identifier des notes de musique. Voici un tableau contenant les fréquences de quelques notes de musique.

Note	do	ré	mi	fa	sol	la	si	do	ré
Fréquence (Hz)	261,6	293,7	329,6	349,2	392,0	440	493,9	523,2	587,3

1) Ouvrir les trois fichiers fournis : `flute.wav`, `basson.wav` et `compose.wav`, et représenter le module de leur FFT en fonction des fréquences.

2) Essayer d'identifier graphiquement la fréquence fondamentale (la plus petite fréquence non nulle) de chacun des fichiers, et en déduire la note perçue par l'oreille.

3) Décrire qualitativement les résultats obtenus.

2 Transformée de Fourier à deux dimensions

Les questions qui suivent peuvent être réalisées avec des boucles `for`. Toutefois, `python` offre de nombreuses façons de « vectoriser » l'effet d'une ou plusieurs boucles, notamment avec la fonction `meshgrid` de `numpy`. L'avantage est une exécution plus rapide et une lisibilité accrue du code. Dorénavant, lorsque c'est possible, on préférera l'usage de telles constructions.

L'exemple qui suit présente l'équivalence entre un code itératif classique et sa version vectorisée pour remplir un tableau à deux dimensions.

Listing 4 – Remplissage d'un damier itératif

```
T = zeros((30, 20))

for x in range(30):
  for y in range(20):
    T[x, y] = (-1) ** (x + y)

display_image(T)
```

Listing 5 – Remplissage d'un damier vectorisé

```
U = zeros((30, 20))

[X, Y] = meshgrid(range(30), range(20))
U[X, Y] = (-1) ** (X + Y)

display_image(U)
```

Exercice 4 – Visualiser une FFT à deux dimensions. Une image de $N \times M$ pixels encodée avec 256 niveaux de gris peut être interprétée comme une matrice à N lignes et M colonnes dont les composantes prennent des valeurs dans $[0, 255] \subset \mathbb{N}$.

1) En utilisant la librairie `numpy` de Python, générer une matrice carrée $Z \in \mathcal{M}_{N,N}(\mathbb{R})$ dont les composantes vérifient la relation

$$Z_{n,m} = \cos\left(\frac{2\pi}{N} f\, X_{n,m}\right)$$

où $X_{n,m} = m$. On prendra pour commencer $N = 32$ et $f = 1$.

2) Éxecuter la commande `display_images(Z)`.
Comme on peut le constater, Z n'est pas en l'état une image. Cependant, les commandes d'affichage font la translation ici de l'image de la fonction cos, soit l'intervalle $[-1, 1]$, vers l'espace d'affichage $[0, 255] \subset \mathbb{N}$. Créer une deuxième matrice $Z2$ qui prend directement en compte cette translation.

3) Calculer la FFT de Z. Utiliser pour cela la fonction `fft2` de `numpy.fft`.

4) Représenter le module de cette FFT à l'aide de la fonction fournie `display_image`. On pourra utiliser le paramètre `vmax = 10000` de manière à augmenter la luminosité des valeurs situées en-dessous. À la place, on pourra aussi afficher $\log_{10}(1 + |\hat{Z}|)$ afin « d'aplatir » l'intervalle des valeurs affichées. On pourra également observer l'effet de la fonction `fftshift` sur les données à visualiser.

5) La fonction `display_image` affiche une image (ici de taille 512×512). Les axes des abscisses et des ordonnées proposent par défaut des valeurs minimales et maximales liées à la taille de l'image. Ceci n'est pas correct lors de l'utilisation de la FFT. On lui préférera par la suite la fonction `aff_img` qui permet de changer les valeurs par défaut sur les axes. Une exemple d'utilisation est fourni ci-dessous.

```
i=np.zeros((128,128))
i[54:74,54:74]=255
aff_img(i,[-64,64,-64,64])
aff_img(i,xv=range(-64,64),yv=range(-128,0))
```

Utiliser cette nouvelle fonction pour reproduire l'étude de la question précédente

6) Décrire qualitativement les résultats obtenus.

7) Reproduire la question pour Z_2.

8) Créer une nouvelle matrice Z_3 vérifiant la même relation que Z mais pour $f = 8$. Reproduire les questions précédentes et interpréter.

9) Générer une matrice carrée $Z_4 \in \mathcal{M}_{N,N}(\mathbb{R})$ dont les composantes vérifient la relation

$$Z_{4,n,m} = \cos\left(\frac{2\pi}{N}(f_x X_{n,m} + f_y Y_{n,m})\right)$$

où $X_{n,m} = m$ et $Y_{n,m} = n$. On prendra $N = 32$, $f_x = 4$ et $f_y = 8$. Reproduire les questions précédentes et interpréter.

10) Créer la matrice Z_5 vérifiant

$$Z_{5,n,m} = Z_{4,n,m} \exp(-R_{n,m}^2)$$

où $R_{n,m} = \sqrt{\left(\frac{2\pi}{N} X_{n,m} - \pi\right)^2 + \left(\frac{2\pi}{N} Y_{n,m} - \pi\right)^2}$. Répéter les questions précédentes pour cette nouvelle matrice.

11)Répéter les mêmes questions pour $Z_6 = \text{sign}(Z_1)$ où on aura pris $f = 2$ et $Z_7 = \text{sign}(Z_4)$ pour $f_x = 2$ et $f_y = 2$.

12) Ouvrir l'image `barbara.png` et la stocker dans la variable `i`. Nous souhaitons voir dans cette question l'effet d'une « troncature » en fréquence. Créer pour cela les cinq matrices carrées suivantes de taille $N \times N$

$$\begin{aligned}
R_{1,n,m} &= (X_{n,m} - 256)^2 + (Y_{n,m} - 256)^2, \\
R_{2,n,m} &= (X_{n,m} - 183)^2 + (Y_{n,m} - 318)^2, \\
R_{3,n,m} &= (X_{n,m} - 338)^2 + (Y_{n,m} - 192)^2, \\
R_{4,n,m} &= (X_{n,m} - 205)^2 + (Y_{n,m} - 211)^2, \\
R_{5,n,m} &= (X_{n,m} - 305)^2 + (Y_{n,m} - 311)^2.
\end{aligned}$$

À l'aide de la FFT inverse, créer l'image Z_8 vérifiant la relation

$$Z_8 = \text{ifft2}\Big(\exp(-R_1/200) \cdot \text{fft}(i)\Big).$$

La notation $\cdot$ désigne ici la multiplication terme à terme. Afficher Z_8 et interpréter le résultat.
Reproduire la question pour

$$Z_9 = \text{ifft2}\Big((\exp(-R_1/200) + \sum_{k=2}^{5} \exp(-R_k/50)) \cdot \text{fft}(i)\Big).$$