TP 4 – Transformée de Fourier 2

S.Gibet

Année 2019-2020

L'objectif est ici de bien comprendre les bases de la transformée de Fourier appliquée à des signaux réels. Le programme en Python *fft-oiseaux* est fourni sur Moodle. L'idée est de jouer avec, de modifier les paramètres et de visualiser les résultats.
Documentation : `https://docs.scipy.org/doc/scipy/reference/signal.html`

Transformée de Fourier appliquée à des signaux sonores réels

Vous utiliserez dans cet exercice des signaux sonores de votre choix. Vous pourrez en particulier trouver dans le répertoire DATA quelques fichiers son. Vous utiliserez les librairies numpy, matplotlib et scipy (pour lire un fichier son, traiter le signal et tracer le spectrogramme) :

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.io.wavfile import read
from scipy.fftpack import fft
from scipy import signal
```

1. Lire un signal (chant d'oiseau par exemple). Affichez sa taille (en nombre d'échantillons), sa fréquence d'échantillonnage, ainsi que la durée du signal en secondes.

2. Ecrire le programme qui permet d'effectuer la transformée de Fourier de votre signal pour une fenêtre de Nf échantillons (prendre 512 ou 1024) et un Offset que vous choisirez (cela dépend du signal que vous utilisez). Tracez le spectre en amplitude et en phase pour différentes fenêtres temporelles. Qu'observez-vous ?

3. Tracez le spectrogramme du signal (encore appelé sonagramme pour le son). Vous pourrez utiliser les lignes de code suivantes :

   ```
   f, t, Sxx = signal.spectrogram(x, fs)
   plt.pcolormesh(t, f, np.log(Sxx))
   ```

4. On souhaite calculer le spectre du signal précédent à certains instants spécifiques d'intérêt (zones du spectrogramme où le son est présent). On se propose alors d'extraire un morceau du signal correspondant à un fichier son donné. Ce morceau est défini par un Offset (intervalle de temps à partir du début du fichier son) et un nombre d'échantillons dans le temps (winSize). On souhaite également appliquer une fenêtre temporelle qui permet d'éviter les bords non linéaires de la fenêtre rectangulaire. Dans ce cas, on multiplie le signal par cette fenêtre temporelle "arrondie" (winType).

Ecrire une fonction *calculerSpectre(echantillons, ...)* qui retourne le vecteur fréquentiel et le vecteur du spectre calculé sur le signal *échantillons*. Vous pourrez visualiser l'amplitude du spectre en dB.

5. Ecrire une fonction Python *movingFFT* qui appelle la fonction *calculerSpectre* pour différentes parties du signal. Cette fonction prend pour paramètres :
   fileName : Le nom du fichier signal à charger
   winSize : la taille de la fenêtre de signal à traiter
   offset : la position de la fenêtre de signal à traiter
   winType : le type de la fenêtre à utiliser (rectangulaire, hamming, hanning)
   Testez cette fonction pour différents signaux, offsets et différents types de fenêtres temporelles.