TP3Sujet

December 10, 2018

1 TP3 : Filtrage et transformée de Fourier Rapide (FFT)

Dans ce TP, on propose d'explorer numériquement le concept de filtrage sur des cas simples. Dans la troisième partie, il s'agit de coder effectivement la transformée de Fourier discrète puis de coder la transformée de Fourier rapide.

Pour réaliser ce TP, on propose de charger les outils standards que sont :

```
In [1]: import matplotlib.pyplot as plt
        import numpy as np
```

Et on chargera les données fournies sous forme de fichiers `np.array` à télécharger et à charger dans la feuille de calcul `data.npy` et `fs.npy` qui contiennent les données et la fréquence d'échantillonnage d'un extrait de musique.

Important : on prendra un entier $M = 8000$ pour les transformée de Fourier et leur inverse pour les fonctions `np.fft`.

La feuille de calcul jupyter (ou autre tant que c'est commenté et qu'il y a des explications), sera à envoyer à la fin de TP à l'adresse chupin@ceremade.dauphine.fr, avec dans le nom de la feuille de TP, les nom des personnes composant le binôme

1.1 Fenêtrage et convolution

1.1.1 Réponse impulsionnelle au créneau

1. Construire dans dans le domaine fréquenciel (`np.fft.fftfreq` avec un pas de temps correspondant à la fréquence d'échantillonnage), une fonction créneau de fréquence de coupure $f_C = 1300$Hz. Cela sera notre *gabarit* pour notre filtre.

2. Par transformée de Fourier inverse (de la librairie `numpy` toujours), calculer la réponse impulsionnelle correspondant à la fonction créneau précédemment définie. Tracer la et commentez.

3. Puisque toutes les fonctions sont supposées périodiques, la réponse impulsionnelle obtenue est-elle causale ?

1.1.2 Fabrication de la réponse impulsionnelle finie (RIF)

4. À partir de cette réponse impulsionnelle non causale et infinie, transformez cette réponse en une réponse impulsionnelle finie causale. On prendra un support de 1000 échantillons pour cette réponse.

1.1.3 Filtrage par convolution

5. Définissez une fonction python qui réalise la convolution entre un signal (s_n) et une RIF (h_n) :

```
def convolutionRIF(s,h):
```

6. Calculez la convolution du signal `data` avec la RIF définie précédemment.

7. Vérifiez par l'étude du spectre du signal obtenu l'effet du filtre ainsi obtenu. Commentez !

1.2 Filtrage par modification du spectre

1. Si ce n'est pas déjà fait, représentez le spectre du signal `data` (on utilisera la taille du signal pour l'entier de définition de la transformée de Fourier)

2. Appliquer un *masque* de coupure de fréquence $[-f_c, f_c]$ avec $f_c = 1300Hz$.

3. Par transformée de Fourier inverse, obtenez le signal temporel correspondant au spectre modifié.

4. Comparez le signal à celui obtenu dans la partie précédente (il faudra sans doute légérement modifier le signal précédent pour comparer les signaux).

5. À partir du signal obtenu, recalculez le spectre pour vérifier que le signal temporel obtenu a bien le bon spectre.

1.3 Filtrage d'un signal de transport

Dans cette partie, on chargera le signal `signal.npy`. Celui-ci est le signal `data.npy` précédent qu'on a ajouté à un signal *porteur* de fréquence unique. C'est le principe général de la radio. On cherchera ici à récupérer le signal transporté.

1. Élaborez une stratégie pour filtrer ce signal et récupérer le signal transporté.

2. Comparez le signal ainsi extrait au signal de départ (spectre et signal temporel). Là encore, élaborez une stratégie pour comparez les éléments de façon mathématique.

1.4 Implémentation de la TFD et de la FFT

1. Implémentez la TFD en python, c'est-à-dire une fonction python qui à une suite (a_n) associe une suite (A_n) de même taille définie par le cours.

2. Implémentez la FFT par l'algorithme vu en cours.