Feuille de TP n°2

UF Analyse Hilbertienne et Fourier

L'objectif de cette séance de TP est d'illustrer numériquement les notions de transformée de Fourier et de convolution vues en cours et voir leurs applications possibles pour l'analyse de signaux permanents. L'outil de base est la transformée de Fourier discrète et l'algorithme de transformée de Fourier rapide qui lui est associé.

Pour cette séance, nous travaillerons en une dimension et analyserons des signaux sonores. 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 donn\'ee freq permet de connaÃ®tre le
    nombre d'\'echantillons 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 \'a 44.1 kHz
    contenant 88200 \'echantillons \'a z\'
    ero. On obtiendra donc deux secondes de
    silence.
```

Une alternative au module wavtools est d'utiliser la librairie de scipy : scipy.io, scipy.io.wavfile. Cette alternative est utilisée dans les deux fichiers .py où sont programmés l'analyse d'un signal sonore (Fourier, Fourier à fenêtre).

1 Introduction : transformée de Fourier à fenêtre

Principe : On cherche à étudier un signal $x : [0, T_0] \to \mathbb{R}$. On travaille avec une fenêtre $h_T[-T, T] \to \mathbb{R}$ où T est à définir et on cherche à représenter le spectre à des instants $t_k = T + k\,\delta T$ avec $k \in [0, n]$ et $T + n\delta T \leq T_0$. On calcule la transformée de Fourier discrète de $h_T\, x(. - t_k)$: les fréquences associées sont des multiples de $1/T$.

Mise en oeuvre Dans ce TP, on utilisera les fenêtres de Hann et de Hamming : ces fenêtres sont déjà implémentées dans NumPy. Voir la documentation à l'adresse suivante :
http ://docs.scipy.org/doc/numpy/reference/generated/numpy.hanning.html

La sortie machine est un diagramme $2d$ dans le plan temps-fréquence représentant la valeur absolue des coefficients de Fourier (attention de ne retourner que la première moitié du spectre) et qu'on appelle spectrogramme.

Exercice 1 – Analyse d'un signal périodique par morceaux.

1) Fabriquer un signal contenant la fréquence $F_1 = 440Hz$ pendant $2s$ et $F_2 = 600Hz$ pendant $3s$ avec une fréquence d'échantillonnage $Fe = 2400Hz$.

2) Déterminer la taille minimale de la fenêtre temporelle permettant de capturer les deux fréquences présentes dans le signal.

3) Représenter le spectrogramme pour différentes tailles de fenêtres et étudier la dépendance vis à vis de la résolution temporelle et de la résolution fréquencielle.

4) A partir du spectrogramme, extraire les lignes de crêtes (les endroits où les coefficients de Fourier sont de plus grande amplitude.

Exercice 2 – Fréquences instantanées. Certains des signaux que l'on va analyser sont implémentés dans Scipy (librairie SCIPY.SIGNAL). La documentation est disponible à cette adresse :

http ://docs.scipy.org/doc/scipy/reference/signal.html

1) Chirps linéaires Faire l'analyse temps fréquence d'un signal de la forme $x(t) = \alpha \cos(a\,t + b\,t^2)$.

2) Faire l'analyse temps-fréquence de la somme de deux *chirps* linéaires.

3) Chirps hyperboliques Faire l'analyse temps fréquence d'un signal de la forme $x(t) = \alpha \cos(\frac{a}{b-t})$ pour $t < b$. Expliquer ce qui se passe lorsqu'on approche de $t = b$.

4) Faire l'analyse temps-fréquence de la somme de deux *chirps* hyperboliques.

5) Signaux sonores Représenter le spectrogramme pour les signaux sonores fournis (fichiers .wav).

2 Vocoder de phase

Si on souhaite ralentir un son, une approche naive consiste à lire le vecteur plus lentement. Idem pour accelerer un son en modifiant la fréquence d'échantillonnage du signal considéré : tester ces différentes possibilités avec les outils proposés. Cette modification de la vitesse de lecture induit une modification du contenu fréquentiel du son. Le vocoder de phase est un outil qui permet de modifier la longueur du son sans altérer le contenu fréquentiel.

2.1 Contenu du vocoder

Pour construire un vocoder de phase

1. On fenêtre le signal en trames.
2. On stocke dans deux tableaux, le module et la phase des fft des trames. Pour dilater ou comprimer un son, on interpole séparérement le module et la phase sur chaque nouvelle trame.
3. On reconstruit le signal à partir des nouveaux tableaux de module et de phase.

Le vocoder que nous proposons de construire est une fonction vocoder qui fait ainsi appel à 4 sous programmes :

1. ANALYSE qui calcule le spectrogramme et la phase d'un signal S.
2. SYNTHESE qui reconstruit un signal Srec à partir d'un spectrogramme et d'un tableau de phase.
3. INTERPSPEC qui interpole le spectrogramme initial.
4. INTERPPHASE qui interpole la phase.

Dans le programme vocoder, on choisira le signal et le facteur de compression ou de dilatation. vocoder crée un fichier .wav directement lisible.

2.2 Les programmes

2.2.1 Analyse du son

Ecrire une fonction ANALYSE qui prend pour entrée un vecteur colonne S et un entier N et qui renvoie, deux tableaux Spec et Phase. La première ligne du programme est

```
def Spec,Phase=Analyse(S,N):
```

Dans l'analyse comme dans la synthèse on utilisera des fenêtres de taille N, décalées de $N/8$. En chaque point passeront ainsi 8 fenêtres. On tronquera le signal à un nombre entier de fenêtres, en pratique on prendra souvent $N = 1024$. On pourra utiliser les notations suivantes

— NS taille de S.
— Nf nombre de fenêtres d'analyse.
— H fenêtre de Hanning de taille N.

Remarques :

— $Nf = floor(8 * NS/N) - 7$.

— La fenêtre d'indice k commence à l'indice $1 + (k-1)N/8$ et se termine à l'indice $(k-1)N/8 + N$.
— Ce programme peut être écrit en moins de 15 lignes.

2.3 Synthèse du son

Ecrire une fonction SYNTHESE qui prend en entrée deux tableaux, un de module $Spec$ et un de phase $Phase$ et qui renvoie un vecteur $Srec$. La première ligne du programme est

```
def synthese(Spec,Phase):
```

Remarques
— Il est plus facile de partir d'un vecteur $Srec$ nul et d'y ajouter chaque fenêtre que de parcourir les points un à un et de sommer les contributions des 8 fenêtres associées au point.
— On n'oubliera pas de fenêtrer avec une fenêtre de Hanning, les trames reconstruites.
— Ce programme peut également être écrit en moins de 15 lignes.

2.4 Interpolation du spectre

Ecrire une fonction InterpSpec qui prend en entrée, un spectrogramme $Spec$ et un vecteur colonne T contenant les "indices réels" des nouvelles fenêtres à construire et qui renvoie le tableau $Spec2$ du spectrogamme interpolé. Les valeurs de T doivent varier entre 1 et $Nf - 1$. La première ligne du programme est

```
def InterpSpec(Spec,T):
```

Remarques :
— Le nombre de fenêtres de $Spec2$ est égal à la longueur de T
— La colonne d'indice k de $Spec2$ correspond à la colonne d'indice réel $T(k)$ de $Spec$. Elle se calcule à partir des colonnes de $Spec$ d'indices $floor(T(k))$ et $floor(T(k)) + 1$.

2.5 Calcul de la phase

Ecrire un programme InterpPhase qui prend en entrée un tableau de phase $Phase$ et un vecteur T contenant les indices "réels" des nouvelles fenêtre et qui renvoie le tableau de phase $Phase2$ associé à ce jeu de fenêtre. La première ligne du programme est

```
def InterpPhase(Phase,T):
```

Le calcul de la phase est le point le plus délicat du vocoder de phase. Pour que le son reconstruit soit le plus fidèle possible au son original, la phase associée à chaque fréquence doit varier à la même vitesse que sur le son original.
Comme dans le programme précédent, la colonne d'indice k de $Phase2$ correspond à la colonne d'indice réel $T(k)$ dans le tableau $Phase$. Cette colonne $Phase2(:,k)$ doit être construite de telle sorte que la variation de phase locale entre les deux tableaux soit proche :

$$Phase2(:,k) - Phase2(:,k-1) = Phase(:,floor(T(k))) - Phase(:,floor(T(k)) - 1) \quad (1)$$

2.6 Le Vocoder

On appellera par exemple Vocoder.m le programme principal. Il prend en entrée un fichier .wav et un facteur de dilatation $alpha$.

— On analyse le son grâce au programme Analyse.m .
— On crée un vecteur T qui contiendra les "indices réels" des fenêtres associées au signal reconstruit.
```
T=[1:alpha:Nf-1];
```
— On interpole le spectre et on calcule la phase avec InterpSpec.m et InterpPhase.m .
— On reconstruit le son avec Synthese.m .
— Grâce à la commande wavwrite on crée un fichier .wav associé au vecteur reconstruit.