TP2 : Transformée de Fourier Rapide (FFT) et filtrage

1 Position du problème

On considère un signal de parole enregistré puis numérisé. Le but de ce TP est de constater l'effet de deux modifications du signal et leur visualisation par les méthodes de Fourier. Ces deux modifications sont

- un filtrage passe bas "dur" éffectué directement sur le spectre du signal
- un sous échantillonnage du signal.

Dans le premier cas, le but est de constater par l'écoute la modification causée par le filtrage. Le filtrage peut provenir du fait que le son est par exemple envoyé dans un canal à faible bande passante, mais il peut aussi être désiré pour éliminer les bruits de fond haute fréquence comme certains souffle.

Dans le deuxième cas, c'est le théorème de Shannon qui est en action. On cherchera à visualiser les conséquences sur le spectre du sous échantillonnage, puis on essaiera d'écouter le résultat de cette opération pour se faire une idée concrète du problème.

2 Manipulations

2.1 Récupération et création de fichiers .wav

Les fichiers de données que nous allons considérer sont des fichiers au format **.wav**.

Exemple :

> *Écouter le fichier pharyn.wav avec le lecteur de votre choix*

Il faut pouvoir transformer ces fichiers en un vecteur dont les composantes sont les valeurs du signal aux instants $(k-1)T/m$, $k = 1, \ldots, m$. Pour cela, on utilise la commande **wavread**.

Exemple : La ligne de commande

> *[s,fs]=wavread('pharyn.wav')*

permet d'obtenir le vecteur s recherché et la fréquence d'échantillonnage fs avec laquelle la voix a été numérisée, c'est-à-dire combien de valeurs par seconde ont été recueillies.

Supposons à l'inverse que nous ayons synthétisé un signal s et que nous cherchions à le transformer en fichier **.wav** pour l'écouter. La commande est alors **wavwrite**

> *wavread(s,fs,'newpharyn.wav')*

permet de transformer le vecteur s en un fichier *newpharyn.wav* avec la fréquence fs désirée.

2.2 FFT et filtrage passe-bas

On s'intéresse maintenant à la tranformée de Fourier discrète du signal obtenu dans le vecteur s. Cette transformée de Fourier est une opération linéaire sur le vecteur/signal s de la forme :

$$TF_{2^n} = \begin{bmatrix} e^{-2i\pi 0\frac{0}{m}} & e^{-2i\pi 0\frac{1}{m}} & \cdots & e^{-2i\pi 0\frac{m-1}{m}} \\ e^{-2i\pi 1\frac{0}{m}} & e^{-2i\pi 1\frac{1}{m}} & \cdots & e^{-2i\pi 1\frac{m-1}{m}} \\ \vdots & \vdots & \vdots & \vdots \\ e^{-2i\pi (m-1)\frac{0}{m}} & e^{-2i\pi (m-1)\frac{1}{m}} & \cdots & e^{-2i\pi (m-1)\frac{m-1}{m}} \end{bmatrix} \begin{bmatrix} s_0 \\ s_1 \\ \vdots \\ s_{n-1} \end{bmatrix} \quad (1)$$

Bien sûr, comme pour toute opération linéaire bijective sur $\mathbb{R}^m$, la transformée de Fourier discrète inverse est obtenue en prenant l'inverse de la matrice de transformée de Fourier. Cette matrice étant orthogonale, il suffit de prendre la transposée puis la conjuguée.

Comme vous le savez maintenant, la meilleure façon de caluler cette transformée est l'algorithme de transformée de Fourier rapide qui fonctionne en $\mathcal{O}((3n-1)2^n)$ opérations. Il existe de la même façon un algorithme de FFT inverse. La fonction scilab pour calculer la FFT est tout simplement **fft**.

Exemple : La ligne de commande

$> S = fft(s, -1)$

calcule la FFT du vecteur/signal s et

$> s = fft(S, 1)$

calcule la FFT inverse. Le paramètre 1 ou -1 à donner pour déclarer que l'on souhaite une fft ou une fft inverse fait référence au signe devant de $2i\pi$ dans les exponentielles selon que l'on prenne la FFT (-1) ou son inverse pour lequel la transposée conjuguée donne un signe positif (et donc 1) dans les exponentielles complexes.

Je suggère que vous preniez un signal tel que *'pharyn.wav* et que vous calculier sa FFT. Vous devez obtenir un signal de la forme donnée par la figure 1.

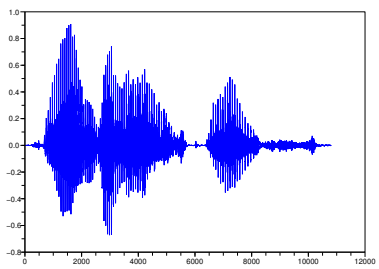

FIGURE 1 – Signal s échantillonné.

puis affichez la valeur absolue de la fft de s (**abs(S)**) et obtenez après traitement un résultat semblable à la figure 2 ci dessous.

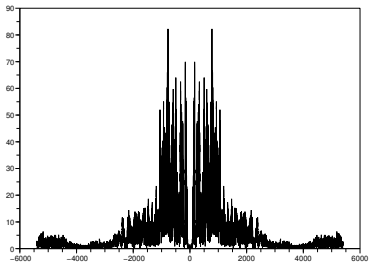

FIGURE 2 – FFT S du signal s.

Attention, la fft donne un vecteur qui commence à la fréquence 0 et au lieu de symétriser le spectre par rapport à 0 comme il est usel de représenter les signaux réels. On fera donc l'opération

$> N = size(S, 2)$ *si* S *est un vecteur colonne*
$> S = S([[N/2 + 1 : N], [1 : N/2]])$

On peut maintenant filtrer avec un passe bas pour voir ce que cela donne comme genre de résultat. Comme tout est numérique, c'est très simple à réaliser : il suffit de mettre les B premières composantes du spectre à zéro ainsi que les B dernières.

$> PBS = S$ *si* S *est un vecteur colonne*
$> PBS(1 : B) = zeros(1, B)$
$> PBS(N - B + 1 : N) = zeros(1, B)$

On obtient alors le spectre donné par la figure 3.

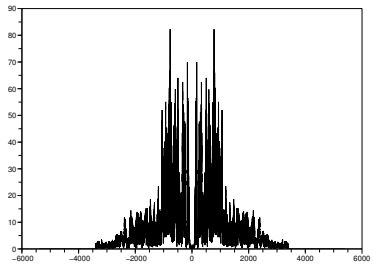

FIGURE 3 – FFT S du signal s après filtrage.

Si l'on prend la FFT inverse de PBS[1], on obtient le signal de la forme donnée par la figure 4.

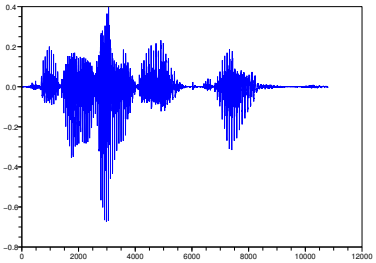

FIGURE 4 – Signal *pbs* filtré par le passe bas "dur".

Reste à écouter la voix obtenue en reconvertissant le vecteur/signal *pbs* en un fichier **.wav** à l'aide de la fonction **wavwrite**.

1. en n'oubliant pas de remettre le spectre dans son état initial c'est à dire en commençant par la fréquence zero