TP1

Traitement numérique du son

1 Introduction

Le but de ce TP est de mettre en pratique les notions de traitement numérique vues en cours, TDs et dans le précédent TP. On se focalisera sur le traitement de fichiers sons. Avant de commencer, télécharger la fiche de TP sur le site

http://www.u-bourgogne.fr/monge/e.busvelle/teaching.php

ainsi que tous les fichiers sons et le script Matlab qui vont avec et qui serviront au cours de ce TP. On lancera ensuite Matlab et on se placera dans le répertoire où ont été placés les fichiers sons.

On rappelle qu'un son est produit par une variation de la pression de l'air à une fréquence comprise entre 20 Hz et 20 kHz. Le son se déplace dans l'air à une vitesse approximative de $V_{son} = 331 + 0.6T\ m/s$ où T désigne la température de l'air en degrés Celsius. A $20°C$, $V_{son} \simeq 343\ m/s$. Cette vitesse est aussi influencée par le taux d'humidité (le son se déplace dans l'eau à une vitesse d'environ 1480 m/s, et encore plus vite dans un solide). La vibration de l'air est captée par l'oreille qui la transforme en sensation auditive.

Usuellement, les fichiers sons non-compressés sont des fichiers avec l'extensions `.wav`. Ils contiennent essentiellement :

- F_e : la fréquence d'échantillonnage des signaux
- B : le nombre de bits considérés pour la quantification des signaux
- Une ou deux "pistes" (suivant que le signal est mono ou stéréo) constituées d'une liste de nombres représentant la valeur du signal sonore en fonction du temps (multiple de la période d'échantillonnage).

Dans Matlab, on peut charger un fichier audio par la commande

```
>> [y,Fe,B] = wavread('nom_du_fichier');
```

et y contient les données échantillonnées, en une colonne (son mono) ou deux colonnes (son stéréo). On peut savoir le nombre de colonnes (de pistes, noté p) et le nombre d'échantillons N par

```
>> [N,p]=size(y)
```

On peut extraire une piste par

`>> y_piste=y( :,1) ;`

On peut extraire une partie d'une piste audio par

`>> y_extrait=y_piste(k1 :k2) ;`

On peut représenter graphiquement un signal y (mono) en fonction des indices, par

`>> plot(y)`

ou en fonction du temps (où T_e est la fréquence d'échantillonnage) par

`>> t=(0 :N-1)*Te ;`

`>> plot(t,y) ;`

Enfin, on peut écouter le signal par

`>> wavplay(y,Fe,'sync') ;`

L'option 'sync' est facultative, voir sa signification dans la documentation Matlab.

Pour terminer cette introduction, citons la commande `wavwrite` pour sauvegarder un fichier (après traitement numérique) et `wavrecord` pour enregistrer un son.

2 Échantillonnage et quantification

2.1 Échantillonnage

Charger dans Matlab le fichier "nbouvier.wav".

Question 1 *Décrire la source audio contenue dans ce fichier, préciser :*

- *s'il sagit d'un son mono ou stéréo*
- *échantillonné avec quelle fréquence*
- *de quelle durée précise (en millisecondes)*
- *avec quel nombre de bits les données sont-elles stockées*

Représenter graphiquement le signal en fonction des indices.

Pour sous-échantillonner le signal à la fréquence $\frac{F_e}{2}$. Il suffit d'écrire

`>> y2=y(1 :2 :N) ;`

ce qui revient à prendre les échantillons de 1 à N par pas de 2. Pour jouer ce signal, il suffira de taper

`>> wavplay(y2,Fe/2,'sync') ;`

On peut représenter $y2$ en fonction du temps dans le même graphique que y (lui aussi en fonction du temps) grâce à la fonction `hold on` :

```
>> t2=t(1 :2 :N) ;
>> hold on,
>> plot(t2,y2,'r') ;
```

Question 2 *Sous-échantillonner successivement aux fréquences $\frac{F_e}{2}$, $\frac{F_e}{4}$, $\frac{F_e}{8}$. Dans chaque cas, tracer le signal original et le signal sous-échantillonné. Ecouter le signal sous-échantillonné. Préciser dans chaque cas :*

- *la fréquence maximale restituable (grâce au théorème de qui vous savez)*
- *le facteur de compression, si on remplace le signal original par le signal sous-échantillonné (un facteur 3 signifie que le fichier compressé est 3 fois plus petit que l'original)*
- *si le signal est propre, dégradé, ou incompréhensible.*
  Quelle est la fréquence maximale de la parole (googlez !) ?

2.2 Quantification

On continue avec le son du fichier "nbouvier.wav". Représenter graphiquement le signal contenu dans ce fichier.

Question 3 *Déterminer le quantum d'abord visuellement (en zoomant) puis calculant la différence entre deux échantillons consécutifs* $y(n+1) - y(n)$ *dont les valeurs ne diffèrent que d'un quantum (et repérés graphiquement). Que vaut le quantum ? Est-ce cohérent avec la valeur* B *rendue par la fonction* `wavread` *? Que valent les valeurs minimales et maximales codables dans le fichier wav (au delà, il y aurait saturation) ?*

Toutes les fonctions de traitement du son peuvent être effectuées depuis Matlab. Pour un traitement plus complexe (en particulier avec du filtrage), on peut utiliser Simulink. Il faut simplement pouvoir lire un signal depuis une variable Matlab, et écrire le résultat de la simulation dans une variable Matlab.

Les blocs utiles sont les blocs "From workspace" et "To workspace", le terme "workspace" désignant l'environnement de travail Matlab. A titre d'exemple, réalisez le schéma de la Figure 1.

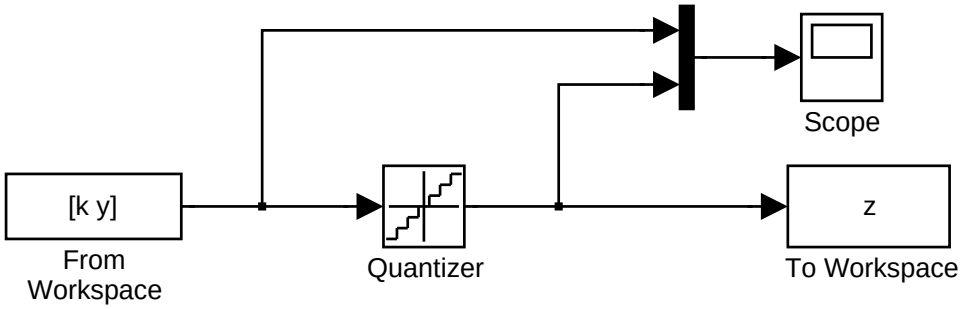

FIGURE 1 – Quantification avec Simulink

Dans le bloc "From Workspace", on doit entrer un tableau dont la première colonne est le temps et les colonnes suivantes sont les valeurs des signaux pour les temps donnés en première colonne. Dans notre cas, on défini k par `k=(0 :N-1)'` (ne pas oublier le prime) où N est la longueur de l'échantillon, qu'il faudra aussi entrer comme "Stop time" dans le schéma Simulink. **On a donc choisi d'utiliser le temps d'échantillonnage comme unité de temps**, ceci afin de rester dans le domaine de la simulation numérique. Celà permet aussi d'entrer dans les paramètres de configuration, dans le champ "Solver" : "Discrete (no continuous state)", ce qui accélère la simulation.

Finalement, il suffit d'entrer `[k,y]` dans le champ "Data" du bloc "From Workspace" et 1 dans le champ "Sample time" (temps d'échantillonnage).

Le bloc "To Workspace" est encore plus simple à configurer. Il suffit d'entrer le nom de la variable de sortie et de laisser "Sample time" à -1 (Figure 2). On n'oubliera pas cependant de choisir le format de sauvegarde correct ("Save format" à "Array").

Régler le bloc "Quantizer" et lancer une simulation. Vous devez voir dans le "scope" deux signaux assez semblables mais avec un quantum différent. D'aupre part, vous devez avoir dans

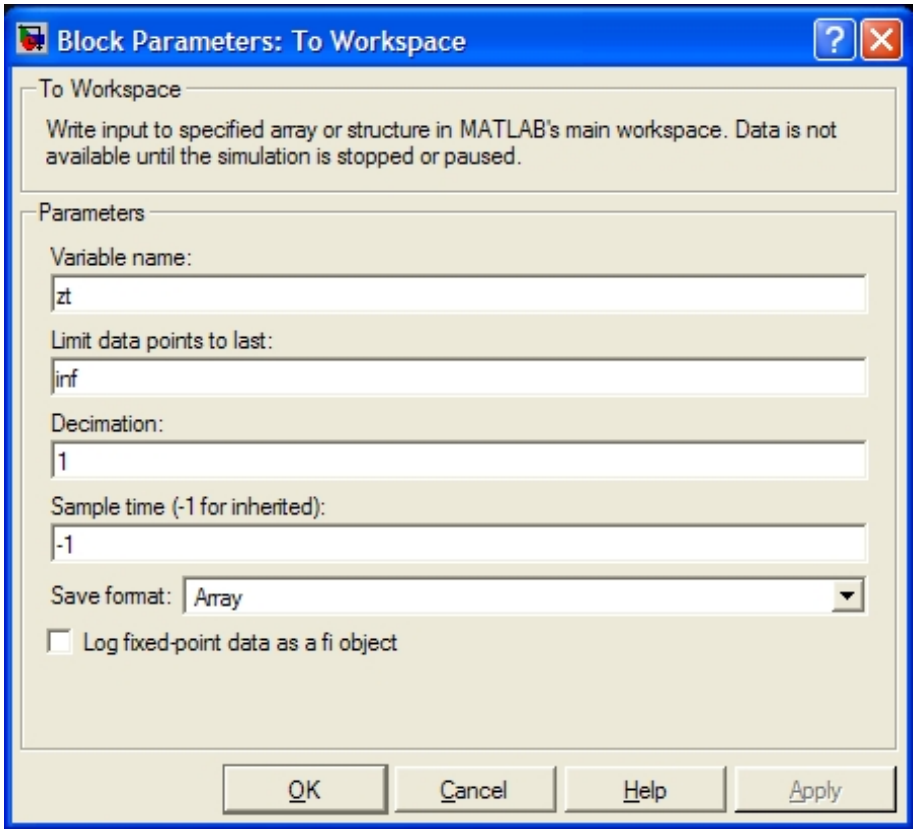

FIGURE 2 – Paramètres du bloc "To Workspave"

l'environnement Matlab une variable z que vous pouvez tracer pour vérifier qu'elle correspond bien à y quantifié.

Question 4 *Tester des quantifications sur* 7 *à* 3 *bits (il faut calculer le quantum correspondant, lancer la simulation, regarder le scope et surtout, écouter le signal produit, z, par* `wavplay`*). Commentez le résultat. En fonction du nombre de bits utilisés, dire si le signal est détérioré, incompréhensible ? Préciser dans chaque cas le facteur de compression, si on remplace le signal original par le signal avec la nouvelle quantification.*

3 Compression MP3 (simplifiée)

3.1 Principe

Comme vous avez pu le voir (l'entendre) dans la section précédente, les paramètres d'échantillonnage et de quantification ne permettent pas de gagner beaucoup de place sans détériorer grandement la qualité de l'écoute. La compression mp3 (plus généralement mpeg-1 à 3) est basée sur la représentation fréquentielle du signal. Nous allons en illustrer le principe de base.

Essentiellement, il s'agit de calculer la représentation fréquentielle du signal et "d'oublier" les fréquences inaudibles car trop peu puissantes en regard des autres fréquences audibles au même moment. On travaille sur chaque piste indépendamment.

3.2 Fonctions Matlab utiles

La fonction de base est évidemment la transformée de Fourier discrète, intitulée `fft` (pour Fast Fourier Transform, évidemment). Pour un signal temporel y, on calcule la transformée de Fourier par

>> `z=fft(y) ;`

Si le signal est de longueur N, il en sera de même de sa transformée de Fourier qui sera constituée de N nombres complexes. Si le signal y a été échantillonné avec une fréquence F_e, la plage de fréquence utile est $\left[0, \frac{F_e}{2}\right]$ (merci qui ?) avec une résolution entre les raies de $\frac{F_e}{N}$. On peut donc tracer la représentation spectrale par

>> `f=(0 :N-1)/N*Fe ;`

>> `plot(f,abs(z))`

ou

>> `plot(f(1 :100),abs(z(1 :100)))`

pour ne tracer que les 100 premières raies (par exemple, et si $N \geq 200$)

3.3 Mise en oeuvre

On continue à travailler avec le signal "nbouvier.m". On notera r le facteur de compression souhaité. On va commencer par traiter le signal complet d'un seul coup.

Question 5 *Pourquoi est-il stupide de traiter le fichier d'un seul coup ? On répondra à cette question quand on aura terminé cette section...*

On calcule donc la transformée de Fourier z de y. Après cela, tapez

>> whos y z

Vous devez avoir deux signaux de même longueur, z étant complexe. Le plus difficile maintenant est de calculer un seuil s tel que

- toutes les fréquences de puissance inférieure à s seront négligées
- les fréquences de puissance supérieures à s seront gardées
- le pourcentage de fréquence gardées sera de $\frac{100}{r}\%$ ce qui correspond bien à un facteur de compression de r.

Le calcul se fait en trois étapes :

1. on commence par trier les modules de la transformée de Fourier z de y

   >> `zs=sort(abs(z)) ;`

2. on cherche la valeur s en fonction du facteur de compression r

   >> `s=zs(round(N*(1-1/r)) ;`

3. enfin, on met à 0 les valeurs inférieures au seuil s.

   >> `z(abs(z)<s)=0 ;`

Dans une réalisation pratique, il conviendrait maintenant de ne stocker que les coefficients non nuls. Un fichier mp3 contient dont des informations de nature fréquentielles.

Pour reconstituer le signal (c'est ce que fait un lecteur mp3), on fait la transformée de Fourier inverse :

>> `y_comp=ifft(z) ;`

et on peut écouter le résultat...

Question 6 *Faire un scrpt Matlab qui lit le fichier et le compresse. Essayer plusieurs facteurs de compression. Déterminer, à votre oreille, le facteur de compression à partir duquel le résultat devient inacceptable (qualité d'écoute trop dégradée, c'est subjectif).*

3.4 Amélioration : représentation temps-fréquence

On note toujours N la longueur du signal. Maintenant, et c'est beaucoup moins stupide (penser à répondre à la question 5), on va découper le signal en morceaux. Par exemple, on peut découper en morceaux de 1024 échantillons (donc d'une durée de $1024\,T_e$ secondes ce qui est petit) et faire le traitement précédent sur chaque morceaux. Le programme est le suivant, dans lequel `nf` représente le nombre de morceaux et `lf` la longueur de chaque morceaux :

```
zf=zeros(N,1) ;
nf=2^6 ;                        % Nombre de fenêtres
lf=round(N/nf) ;          % Longueur des fenêtres
k=0 ;
for f=1 :nf,
            zf(k+(1 :lf))=fft(y(k+(1 :lf))) ;
            k=k+lf ;
end,
zs=sort(abs(zf)) ;
seuil=zs(round(N*(1-1/r))) ;
zf(abs(zf)<seuil)=0 ;
k=0 ;
for f=1 :nf,
            ycf(k+(1 :lf))=ifft(zf(k+(1 :lf))) ;
            k=k+lf ;
end,
```

Vous devez être à même de comprendre chaque ligne de ce programme. Le script correspondant peut-être téléchargé sur le site du TP.

Question 7 *Tester plusieurs valeurs de r. Déterminer la valeur de r à partir de laquelle on entend une dégradation, celle à partir de laquelle le signal devient franchement désagréable. Comparer avec le facteur de compression d'un mp3 classique.*

4 Analyse spectrale comparative

4.1 Song of the black lizard

Le but de cette section est de s'habituer à manipuler des signaux sonores : découper, calculer des puissances spectrales, interpréter les raies spectrales (pics de fréquences). Pour cela, on va charger le fichier "BlackLizardExtract.wav", qui est un extrait d'une chanson de *Pink Martini*, "Song of the Black Lizard".

Question 8 *Quelles sont les caractéristiques de ce fichier wav (mono/stéréo, fréquence d'échantillonnage, nombre d'échantillons, longueur en secondes, nombe de bits pour la quantification).*

Ecouter attentivement cette introduction musicale : on y entend la chanteuse, China Forbes, et plusieurs instruments dont une trompette. A un certain moment, la voix de la chanteuse laisse la place à la trompette sans que l'on puisse déceler de rupture. C'est une performance assez remarquable car la voix et l'instrument de musique se distinguent par plusieurs facteurs, qui sont d'ailleurs les facteurs distinctifs d'un son :

- la hauteur : c'est la fréquence fondamentale (celle qui a la plus grande puissance) de la note,
- l'intensité : c'est la puissance, l'énergie contenue dans le son,
- le timbre : il est donné par les fréquences qui sont émises, en plus de la fréquence fondamentale, et que l'on appelle les harmoniques. c'est le timbre qui permet de distinguer plusieurs instruments, ou même des voix différentes.
- la dynamique (enveloppe ADSR : c'est l'évolution de la puissance de la note en fonction du temps, caractérisée elle-même par quatre paramètres (c'est arbitraire) :
  - **A**ttack : l'attaque, le temps de montée en puissance de la note
  - **D**ecay : le déclin, correspondant à un court laps de temps précedent le phase stationnaire
  - **S**ustain : le maintien, c'est le temps pendant lequel la note est maintenue
  - **R**elease : le relâchement, chute de la puissance sonore

Question 9 *Extraire la partie du signal, qui dure environ trois secondes, correspondant à la transition entre la voix de la chanteuse et la trompette. Calculer et tracer le spectre de la voix de China Forbes (en extrayant à nouveau une partie du signal). Sur le même graphique, tracer le spectre de la trompette. Comparer, interpréter.*

5 Filtrage numérique

A partir de cette section, vous maitrisez à peu près toutes les fonctions utiles pour faire du traitement numérique de son :

- Conversion analogique/numérique et effets de quantification d'un signal analogique, conséquences pratiques du théorème de Shannon.
- Lecture, écriture, enregistrement et restitution de fichiers audio
- Analyse spectrale et interprétation des pics de fréquences
- Traitement numérique du signal avec Simulink
- Filtrage numérique (filtres passe-bas, passe-haut) : de tels filtres ont été calculés en TD et sont facilement utilisables dans Simulink, de la même façon que lors de la séance de TP numéro 2
- Filtrage avancé : filtres de Wiener. De tels filtres ont aussi été calculés en TD pour plusieurs utilisations : déconvolution, estimation paramétrique, débruitage

En conséquence, si vous en êtes arrivé là, c'est maintenant à vous de vous débrouiller tout seuls.

5.1 Mixage d'un son et de bruit

Reprendre le fichier "nbouvier.wav" et lui ajouter un bruit blanc que l'on peut générer par "Random Number". Régler la puissance du bruit (c'est-à-dire sa variance, souvenez-vous des TDs) de sorte que le bruit gène la compréhension de la phrase.

Question 10 *Quelle variance σ^2 avez vous choisi ?*

5.2 Filtre passe-bas

On veut maintenant rendre la phrase à nouveau intelligible. Pour cela, il va falloir réduire la puissance du signal bruité, ce qui ne peut être fait qu'en supprimant les fréquences qui ne correspondent pas au signal. Sans information sur le signal d'origine, il est impossible de supprimer totalement le bruit.

Pour faire un filtre passe-bas numérique, on peut utiliser ce que l'on a fait en TD ou utiliser les fonctions Matlab qui existent pour cela. Par exemple, on peut taper :

```
>> fc=0.3 ;
>> f = [0 fc fc 1] ;
>> m = [0 0 1 1] ;
>> [b,a] = yulewalk(8,f,m) ;
```

ce qui permet de calculer les coefficients d'un filtre à réponse impulsionnelle infinie (IIR) stable, de fréquence numérique de coupure égale à $0,3$ (ce sera le paramètre à régler) : **b** et **a** sont des vecteurs qui s'interprètent comme

$$H(z) = \frac{b_1 + b_2 z^{-1} + b_3 z^{-2} + \cdots + b_9 z^{-8}}{1 + a_2 z^{-1} + a_3 z^{-2} + \cdots + a_9 z^{-8}}$$

(car $a_1 = 1$). On peut représenter graphiquement le filtre obtenu par :

```
[h,w] = freqz(b,a,128) ;
plot(f,m,w/pi,abs(h),'--')
```

La fonction de transfert n'a plus qu'a être utilisée dans Simulink, à l'aide d'un bloc "Discrete filter".

Question 11 *Réaliser un filtre numérique passe-bas et déterminer la fréquence de coupure la plus adéquate pour améliorer le signal bruité et le rendre compréhensible.*