TP1. Le traitement du signal audio en utilisant les fonctions Matlab

1. Lisez les materiaux : Introduction aux signaux audio du site:

http://mirlab.org/jang/books/audioSignalProcessing/audioIntro.asp?title=3-1 Introduction to Audio Signals (音訊基本介紹)

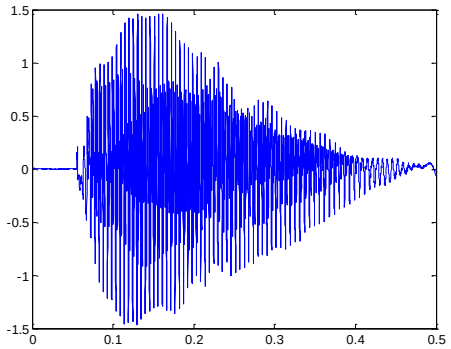
Ce TP présente quelqunes des caractéristiques les plus importantes des fonctions Matlab qui sont utilisés utilisé pour le traitement du signal audio. En particulier , nous couvrirons les sujets de base dans les fonctions suivantes :

1. Lire/read de fichiers audio, .wav;
2. Lecture/play aux signaux audio ;
3. Enregistrement/record d'un signal audio au microphone ;
4. Stockage/save des fichiers .wav .

2. Lire/read de fichiers audio, .wav;

MATLAB peut lire les fichiers wave via la commande " wavread " . L'exemple suivant lit le fichier wave " de sunday.wav » et afficher sa forme d'onde.

```
y, fs]=wavread('boy.wav');
sound(y, fs);                 % Playback of the sound data
time=(1:length(y))/fs;        % Time vector on x-axis
plot(time, y);                % Plot the waveform w.r.t. time
```



Dans l'exemple ci-dessus, " fs " est la fréquence d'échantillonnage qui est 16 000 dans ce cas . Cela indique qu'il existe 16 000 échantillons par seconde lorsque l'attache a été enregistrée. Le vecteur "y" est un vecteur de colonne contenant les échantillons des signaux de parole. Nous pouvons utiliser " (Y, fs) sonores " à jouer les signaux audio lues à partir du fichier. "temps" est un vecteur de temps dans lequel chaque élément correspond à la durée de chaque échantillon. Par conséquent nous pouvons tracer "y" contre " t " pour afficher directement la forme d'onde .

La plupart des signaux audio sont numérisés à une résolution binaire de 8 ou 16 bits. Si nous voulons savoir la résolution en bits du fichier d'onde d'entrée , nous pouvons utiliser un des arguments de sortie supplémentaires pour " wavread " pour obtenir les informations , telles que

```
[y , fs , nbits ] = wavread ( ' girl.wav ' ) ;
```

En outre , si nous voulons connaître la durée d'un flux de signaux audio , nous pouvons utiliser " longueur (y) / fs " directement . L'exemple suivant peut obtenir la plupart de l'information importante du fichier d'onde " de girl.wav

```
close all;
clear all;
[filename1, pathname1]=uigetfile('*.wav')
[y, fs, nbits]=wavread(filename1);
fprintf('Information of the sound file "%s":\n', filename1);
fprintf('Duration =  = %g sec\n', length(y)/fs);
fprintf(''Sampling rate = %g samples/sec\n', fs);
fprintf('Bit resolution = %g bits/sample\n', nbits);
```

Information of the sound file "girl.wav":

Duration = 1.45134 seconds

Sampling rate = 11025 samples/second

Bit resolution = 8 bits/sample

De l' exemple qui précède, on peut observer que tous les signaux audio sont compris entre -1 et 1. Cependant, chaque point d' échantillon est représenté par un nombre entier de 8 bits. Comment sont-ils liés ? Tout d'abord , nous devons savoir la convention suivante :

1. Si un fichier d'onde a une résolution de bits 8 bits , alors chaque point d' échantillon est stocké comme un entier non signé compris entre 0 et 255 (= 2 ^ 8-1) .
2. Si un fichier d'onde a une résolution de 16 bits , chaque point d' échantillon est stocké comme un entier non signé entre -32 768 (= 2 ^ 16/2) et 32 767 (= 2 ^ 16 / 2-1) .

Depuis presque toutes les variables dans MATLAB ont le type de "double" , donc tous les échantillons sont convertis en un nombre à virgule flottante comprise entre -1 et 1 pour la manipulation facile des données . Par conséquent, pour récupérer les valeurs entières d'origine des signaux audio , nous pouvons procéder comme suit .

1.Pour résolution de 8 bits , nous pouvons multiplier "y" (la valeur obtenue par wavread) par 128 , puis plus 128 .

2.Pour résolution de 16 bits , nous pouvons multiplier "y" (la valeur obtenue par wavread) par 32 768 .

Nous pouvons également utiliser la commande " wavread " pour lire un fichier wave stéréo . La variable retournée sera une matrice de deux colonnes contenant chacune les signaux audio à partir d'un seul canal. Exemple ci-après.

```
fileName='stereofile.wav';
[y, fs]=wavread(fileName);    % Read wave file
sound(y, fs);                 % Playback
left=y(:,1);                  % Left channel
right=y(:,2);                 % Right channel
subplot(2,1,1), plot((1:length(left))/fs, left);
subplot(2,1,2), plot((1:length(right))/fs, right);
```

3. Lecture/play aux signaux audio

Une fois que nous pouvons lire le fichier d'onde dans MATLAB , nous puissions commencer à traiter les signaux audio en modifiant leurs intensités , ou de modifier leurs taux d'échantillonnage , et ainsi de suite . Après les signaux audio sont traités , nous avons besoin de les jouer pour l'inspection auditive . Cette section présente les commandes MATLAB pour les signaux audio jeu .
La commande de base pour la lecture de signaux audio est " wavplay " . L'exemple suivant permet de charger un flux de signaux audio à partir du fichier " handel.mat " et de jouer le signal immédiatement.

```
load handel.mat          % Load the signals stored in handel.mat
wavplay(y, Fs);          % Playback of the signal
```

Comme le volume de lecture est déterminé par l'amplitude des signaux audio , on peut changer l'amplitude pour modifier le volume , comme suit .

```
[y, fs]=wavread('vowel.wav');
wavplay(1*y, fs, 'sync');      % Playback with original amplitude
wavplay(3*y, fs, 'sync');      % Playback with 3 times the original amplitude
wavplay(5*y, fs, 'sync');      % Playback with 5 times the original amplitude
```

Dans l'exemple ci-dessus, l'amplitude est augmentée progressivement, afin que nous puissions percevoir le volume croissant pendant la lecture. En particulier, "wavplay" supposent que les signaux d'entrée sont compris entre -1 et 1. Lorsque les signaux d'entrée sont hors de cette borne "son cassée" (trop grand ou trop petit), nous pouvons entendre. Pour essayer ceci, vous pouvez essayer "wavplay (100 * y, fs)" pour entendre le résultat. En outre, nous mettons un argument entrée 'sync' supplémentaire dans l'exemple ci-dessus. Cela rendra "wavplay" à jouer les signaux de manière synchrone. Ce est, la lecture ne démarre pas jusqu'à ce que la lecture précédente est terminée. Nous aurons plus de détails sur cette suite.
OBS.
Dans l'exemple ci-dessus, bien que nous ayons augmenter l'amplitude d'un facteur de 5, l'intensité perçue par l'oreille ne est pas par le facteur de 5. Cela sert à illustrer que la perception de volume ne est pas linéairement proportionnel à l'amplitude. En fait, il est proportionnel à l'amplitude de la logrithm.

Si nous changeons la fréquence d'échantillonnage en cours de lecture, il aura une incidence sur la durée ainsi que la hauteur/pitch perçue. Dans l'exemple suivant, nous allons augmenter les taux d'échantillonnage progressivement, de sorte que vous entendrez un son plus court avec de grand pas, similaire à le son du personnage de dessin animé Disney Donald Fauntleroy Duck.

```
[y, fs]=wavread('father.wav');
wavplay(y, 1.0*fs, 'sync');    % Playback at the original speed
wavplay(y, 1.2*fs, 'sync');    % Playback at 1.2 times the original speed
wavplay(y, 1.5*fs, 'sync');    % Playback at 1.5 times the original speed
wavplay(y, 2.0*fs, 'sync');    % Playback at 2.0 times the original speed
```

D'autre part , si nous baissons la fréquence d'échantillonnage progressivement , nous allons obtenir plus et sons graves . Finalement, il sonnera comme la moo de vache .

```
[y, fs]=wavread('father.wav');
wavplay(y, 1.0*fs, 'sync');     % Playback at the original speed
wavplay(y, 0.9*fs, 'sync');     % Playback at 0.9 times the original speed
wavplay(y, 0.8*fs, 'sync');     % Playback at 0.8 times the original speed
wavplay(y, 0.6*fs, 'sync');     % Playback at 0.6 times the original speed
```

Si nous voulons garder la même durée de temps, mais augmenter ou diminuer la hauteur des signaux audio , alors nous devons effectuer un décalage de hauteur ou pas mise à l'échelle . Il est au-delà de la portée de ce chapitre et sera expliqué dans les chapitres suivants .
Si on inverse les signaux audio en multipliant par -1 , la perception sera exactement le même que l'original. (Ce servent également à démontrer que la perception humaine de l'audio ne est pas affecté par sa phase.) Toutefois, si le les signaux audio inverser dans l'axe de temps , il sonnera comme une langue inconnue .

```
[y, fs]=wavread('father.wav');
wavplay(y, fs, 'sync');                 % Playback of the original signal
wavplay(-y, fs, 'sync');                % Playback of the up-down flipped signal
wavplay(flipud(y), fs, 'sync');         % Playback of the left-right flipped signal
```

MATLAB a une autre commande similaire " son " pour la lecture , qui peut être utilisé pour les deux plates-formes Windows et Unix.

4. Enregistrement/record d'un signal audio au microphone ;

Vous pouvez également utiliser la commande MATLAB " wavrecord " pour lire les signaux audio du microphone directement . Le format de la commande est

```
y = wavrecord ( n , fs ) ;
```

où «n» est le nombre d'échantillons à enregistrer, et " fs " est la fréquence d'échantillonnage . L'exemple suivant enregistre deux secondes de votre microphone .

```
fs=16000;                       % sampling rate
duration=2;                     % record duration
fprintf('Press any key to start %g sec...', duration); pause
fprintf('Recording...');
y=wavrecord(duration*fs, fs);   % Duration*fs is the no. of samples
fprintf('Finish Record'\n');
fprintf('Press any key to play...'); pause; fprintf('\n');
wavplay(y,fs);
```

Dans l'exemple ci-dessus, " durée * fs " est le nombre de points d'échantillonnage à enregistrer. Les points d'échantillonnage enregistrées sont stockées dans la variable "y" , qui est un vecteur de taille 32000x1 . Le type de "y" de données est double et l'espace mémoire prise par "y" est 256000 octets .

Dans l'exemple précédent , le nombre de canaux est égal à 1 et le type de données de points d'échantillonnage est double. Si nous voulons changer ces deux paramètres par défaut , nous pouvons introduire arguments d'entrée supplémentaires pour la commande " wavrecord " . Un format détaillé de " wavrecord " est :

```
y = wavrecord ( n , fs , le canal, dataType ) ;
```

où «canal» (généralement 1 ou 2) est le nombre de canaux d'enregistrement , et " dataType " (comme «double» , «unique» , « int16 », « uint8 ») est le type de points d'échantillonnage enregistrées des données. Différents types de données nécessiteront montant différent de l'espace mémoire . Exemple ci-après.

```
fs=16000;                                   % Sampling rate
duration=2;                                 % Recording duration
channel=1;                                  % Mono
fprintf('Press any key to start %g seconds of recording...', duration); pause
fprintf('Recording...');
y=wavrecord(duration*fs, fs, channel, 'uint8');% duration*fs  is the no. sample points
fprintf('Finished recording.\n');
fprintf('Pressy any key to hear the recording...'); pause; fprintf('\n');

wavplay(y,fs);
```

Cet exemple est presque le même que le précédente, sauf que le type de données est " uint8 ». Les points d'échantillonnage sont encore conservés dans la variable «y» avec le même 32000x1 de taille. Mais les éléments de "y" sont des nombres entiers compris entre 0 et 255. L'espace mémoire du "y" ne est plus qu'à 32 000 octets , ce qui est seulement 1/8 de celui de l'exemple précédent .

Le tableau suivant indique les types de données pris en charge par la commande wavrecord . Le tableau suivant indique les types de données pris en charge par la commande wavrecord .

Data types	Space requirement per sample	Range of the sample data
double	8 bytes/sample	Real number within [-1, 1]
single	4 bytes/sample	Real number within [-1, 1]
int16	2 bytes/sample	Integer within [-32768, 32767] or [-2^(nbits-1), 2^(nbits-1)-1]
uint8	1 byte/sample	Integer within [0, 255] or [0, 2^nbits-1]

Il semble que " wavrecord " se retire progressivement en fonction de l'aide en ligne lorsque vous tapez " wavrecord doc" . La nouvelle commande pour la lecture audio est " AudioRecorder " . Essayez " AudioRecorder doc" pour plus de détails .

```
fs=16000;               % Sampling rate
nbits=16;
nChannels=1;
duration=3;             % Recording duration
arObj=audiorecorder(fs, nbits, nChannels);
fprintf('Press any key to start %g seconds of recording...', duration); pause
fprintf('Recording...');
recordblocking(arObj, duration);
fprintf('Finished recording.\n');
fprintf('Press any key to play the recording...'); pause; fprintf('\n');
play(arObj);
fprintf('Plotting the waveform...\n');
y=getaudiodata(arObj);                  % Get audio sample data
plot(y);                                % Plot the waveform
```

Press any key to start 3 seconds of recording...Recording...Finished recording.

Press any key to play the recording...

Plotting the waveform...

5. Stockage/write des fichiers .wav .

Nous pouvons écrire des fichiers audio " .wav " en utilisant la commande de MATLAB " wavwrite " . Le format de la commande est:

```
wavwrite (Y, fs , nbits , WaveFile );
```

où «y» sont les données audio , " fs " est la fréquence d'échantillonnage , " nbits " est la résolution de bits, et " WaveFile " est le fichier .wav à écrire . Par exemple , nous pouvons suivre l'exemple suivant d'écrire mon enregistrement à un fichier "test.vaw".

Dans cet exemple , nous stockons les données audio dans le type de données « uint8 » et écrire les données dans le fichier .vaw test.wav . Nous invoquons alors l'application correspondante pour ".wav" pour la lecture du fichier . Depuis la variable «y» pour la commande " wavwrite " devrait être un double dans l'intervalle [-1 , 1] , nous avons besoin de faire un peu de conversion si les données enregistrées dans les autres types de données , tels que , « int16 'single' », ou« uint8 ». Voici le tableau de conversion .

Data types of "y"	How to convert it to 'double' within [-1, 1]
double	No conversion needed
single	y = double(y);
int16	y = double(y)/32768;
uint8	y = (double(y)-128)/128;

```
fs=11025;                                   % Sampling rate
duration=2;                                 % Recording duration
waveFile='test.wav';                        % Wav file to be saved
fprintf('Press any key to start %g seconds of recording...', duration); pause
fprintf('Recording...');
y=wavrecord(duration*fs, fs);
fprintf('Finished recording.\n');
fprintf('Press any key to save the sound data to %s...', waveFile); pause
nbits=8;                                    % Bit resolution
wavwrite(y, fs, nbits, waveFile);
fprintf('Finished writing %s\n', waveFile);
fprintf('Press any key to play %s...\n', waveFile);
dos(['start ', waveFile]);                  % Start the application for .wav file
```

MATLAB peut aussi écrire d'autres fichiers audio, tels que " .au " , qui sont les fichiers audio utilisés dans les postes de travail NeXT / Sun . La commande correspondante est " auwrite " . Se il vous plaît taper " help auwrite " dans les fenêtres de commande MATLAB pour plus d'informations sur l'utilisation de cette commande .

OBS.

Si vous voulez faire du traitement de signal audio avec MATLAB exclusivement , vous pouvez enregistrer/sauver vos données audio en utilisant la commande "save" pour les enregistrer dans des fichiers .mat . Vous pouvez ensuite charger ces fichiers audio .mat en utilisant la commande "load" directement .

6. Devoir

Basé sur les études précédentes cap.1-5 résoudre les problèmes et d'accomplir les exigences suivantes et de faire rapport des résultats, des captures, code, etc. Prennez les fichiers de travail wav.zip, du site du TP et sauvez-le dans le dosier de travail.

6.1. Obtenez d'information à partir d'un fichier audio mono.

Ecrire un script MATLAB qui peut lire le fichier "church.wav" et affichera les informations suivantes dans ce script:

- Le nombre de points d'échantillons
- Taux d'échantillonnage
- Résolution Bits

- Le nombre de canaux
- Temps d'enregistrement (en secondes)

6.2. Enregistrez fichier Wave.

Ecrire un script MATLAB pour enregistrer 10 secondes de la parole, comme : "Mon nom est X .... Y .... et je suis un étudiant à la maîtrise au Département de Communications, I PSI Université Technique de Cluj Napoca ". Sauvez le fichier comme myVoice.wav . Les paramètres d'enregistrement sont = 16 kHz fréquence d'échantillonnage, une résolution de 16 bits. Le script permettra obtenir des réponses aux questions suivantes dans la fenêtre MATLAB:

- Combien d'espace est occupé avec des données audio dans MATLAB espace de travail?
- Quel type de données sont les données audio?
- Comment calculez-vous la quantité de mémoire pour les paramètres prise de vue?
- Quelle est la taille du fichier myVoice.wav?
- Combien d'octets sont utilisés dans myVoice.wav pour enregistrer les données autres que des données audio lui-même?

6.3. Manipulation d'un signal audio.

Ecrire un script MATLAB pour enregistrer votre dicton «aujourd'hui c' est mon anniversaire." Essayez d'expliquer la reproduction observée après avoir essayé le traitement de signaux suivants:

- Multiplier par -1 signaux audio.
- Échanger les signaux audio sur l'axe du temps.
- Multiplier par 10 les signaux audio.
- Remplacer chaque échantillon avec sa racine carrée.
- Remplacer chaque échantillon avec sa place.

Signal decoupe / limite de sorte que les échantillons dans l'intervalle [-0,5, 0,5] sont mis à zéro.
Modifier la forme d'onde de sorte que les échantillons dans l'intervalle [-0,5, 0,5] sont mis à zéro, et des échantillons en dehors de la plage de [-0,5, 0,5] sont déplacés vers zéro, le montant de 0,5.

6.4. Expériences sur taux d'échantillonnage:

Ecrire un script MATLAB pour enregistrer votre coeur "mon nom est ***" avec un taux de 16 kHz et 8 bits de résolution échantillonnage ou utiliser un fichier existant. Essayez de ré-échantillonner les signaux audio en abaissant le taux d'échantillonnage à 11 kHz, 8 kHz, 4 kHz,....., et ainsi de suite. A quel taux d'échantillonnage commenceont à avoir de la difficultés à comprendre le contenu de la déclaration?

6.5. Expériences avec l'ajout de bruit.

Ecrire un script MATLAB pour enregistrer disant "Mon nom est ***" avec un taux échantillonnage de 8 kHz et une résolution de 8 bits. Nous pouvons ajouter du bruit à l'audio en utilisant la séquence suivante:

```
k = 0,1;
y2 + y = k * randn (longueur (y), 1);     % Bruit ajoute
Sound (y2, 8000);                         % Lecture / playback
plot (Y2);
```

Augmenter la valeur de k de 0,1 à chaque fois et de répondez aux questions suivantes:
a) À quelle valeur de K commenceons à avoir de la difficulté à comprendre le contenu en cours de lecture?
b) Dessinez les formes d'onde à différentes valeurs de k. À quelle valeur de k nous commençons à avoir des difficultés à identifier la période fondamentale?

Bibliographie

http://mirlab.org/jang/books/audioSignalProcessing/

http://mirlab.org/jang/matlab/toolbox/sap/

http://mirlab.org/jang/matlab/toolbox/utility/

http://neural.cs.nthu.edu.tw/jang/matlab/toolbox/sap/html/sap/wavReadInt.html