EIDIA
École d'Ingénierie Digitale et d'Intelligence Artificielle

UEMF

TP2 -Jeux de mots Synthèse et analyse spectrale d'une gamme de musique

Fait Par

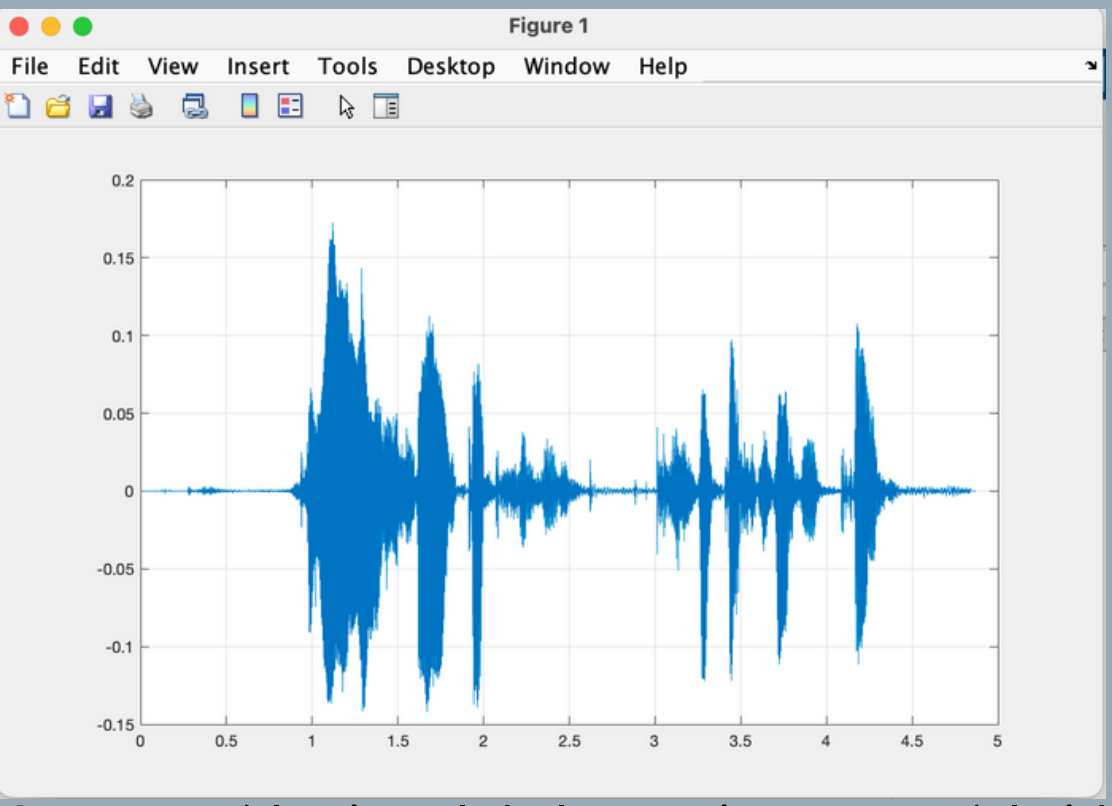
BENDIDI OUMAIMA

Introduction

Dans ce TP, nous allons manipuler un signal audio avec Matlab en effectuant certaines opérations classiques sur un fichier audio d'une phrase enregistrée via un smartphone. Nous allons également comprendre la notion de sons purs en effectuant la synthèse et l'analyse spectrale d'une gamme de musique. Nous devrons être attentifs aux différences de traitement entre le temps continu et le temps discret dans le traitement des signaux en utilisant Matlab. Les figures doivent être tracées avec les axes et les légendes des axes appropriés. Au final, nous remettrons un script Matlab commenté qui reflète notre travail réalisé ainsi que des commentaires sur ce que nous avons compris et pas compris, ce qui nous a semblé intéressant ou pas, bref tout commentaire pertinent sur le TP.

Jeux de mots

```
clear all
close all
clc
%on enregistre l'audiodans une variable nommée "data" et la fréquence d'échantillonnage dans une variable nommée "fs"
[data,fs]=audioread("Centre.mp3");
%On trace le signal
Te=1/fs;
N=length(data);
t = 0:Te:(N-1)*Te;
plot(t,data)
grid on
%pour ecouter l'audio
sound(data,fs)
```

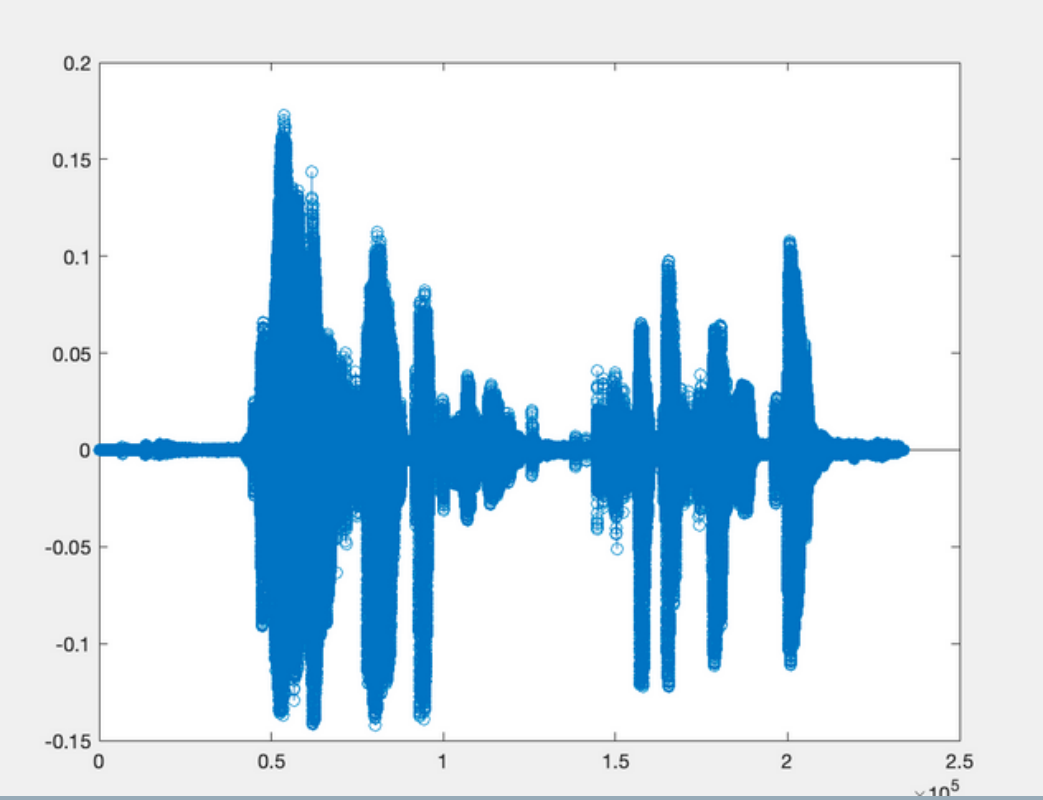


On a tracé le signal de l'enregistrement à l'aide de Matlab

```
%pour ecouter l'audio
% sound(data,fs)
%On modifie la frequence d'echantillonage pour acceler/ ou ralentir l'audio

audio_ralentie = resample(data,fs*4,fs);
sound(y2,fs*4);
audio_accelere= resample(data,fs/2,fs);
sound(y2,fs/2);
```

On a pu entendre l'enregistrement en ralentie et en accéléré

```
%% diviser l'enregistrement puis le restiteuer


stem(data)
%pour obsever du signal enregistré x qui correspondent à chaque morceau.


riennesertde=data(40992:98000)
%on segmente le signal
stem(riennesertde)
% sound(riennesertde,fs)

 courir=data(98000:140000);
%  sound(courir,fs)

 ilfaut=data(140000:160999);
%   sound(ilfaut,fs)

 partirapoint=data(160999:233384);
%  sound(partirapoint,fs)

parole=[riennesertde;courir;ilfaut;partirapoint];
sound(parole,fs)
```



Tout d'abord, la fonction "stem" est utilisée pour tracer les données enregistrées originales, permettant à l'utilisateur de vérifier visuellement le signal.

Ensuite, le signal est segmenté en quatre parties distinctes en utilisant le tableau "data" et les indices 40992:98000, 98000:140000, 140000:160999, et 160999:233384. Ces segments sont respectivement affectés aux variables "riennesertde", "courir", "ilfaut", et "partirapoint".

Enfin, les quatre segments sont concaténés ensemble dans un nouveau tableau appelé "parole" et joués à l'aide de la fonction "sound" avec la fréquence d'échantillonnage d'origine "fs".

Synthèse et analyse spectrale d’une gamme de musique

Ce code crée un programme qui joue une gamme de musique. La fréquence de chaque note de l'échelle est définie au début du code en utilisant le tableau "noteFreq". Chaque note est générée en utilisant un signal sinusoïdal synthétisé numériquement. La fréquence d'échantillonnage "fe" est fixée à 8192 Hz et la durée de chaque note est de 1 seconde.

Les lignes suivantes utilisent la fonction "cos" pour générer des signaux sinusoïdaux pour chaque note de la gamme en utilisant les fréquences correspondantes. Le signal est ensu
ite stocké dans des variables distinctes: "do", "re", "mi", "fa", "sol", "la", "si", "do2".

Ensuite, tous les signaux sont concaténés dans un tableau "musique" et joués à l'aide de la fonction "sound" avec la fréquence d'échantillonnage "fe".

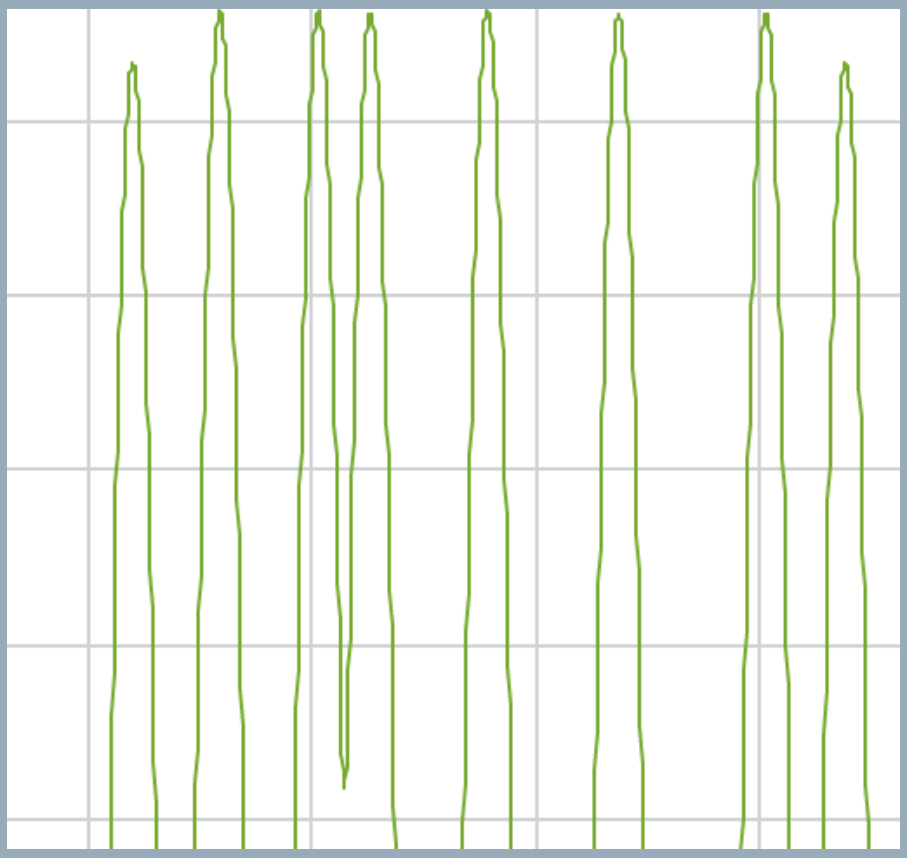
```
fe=8192;
te=1/fe;
N=5000;
t=(0:N-1)*te;
do=10*cos(2*pi*262*t);
%sound(do,fe)
re=10*cos(2*pi*294*t);
%sound(re,fe)
mi=10*cos(2*pi*330*t);
%sound(mi,fe)
fa=10*cos(2*pi*349*t);
%sound(re,fa)
sol=10*cos(2*pi*392*t);
%sound(sol,fe)
la=10*cos(2*pi*440*t);
%sound(la,fe)
si=10*cos(2*pi*494*t);
%sound(si,fe)
do2=10*cos(2*pi*523*t);
%sound(do2,fe)
musique=[do,re,mi,fa,sol,la,si,do2];
 sound(musique,fe)
```

La fréquence de chaque note est précisée dans le tableau ci-dessous

Do1	Ré	Mi	Fa	Sol	La	Si	Do2
262 Hz	294 Hz	330 Hz	349 Hz	392 Hz	440 Hz	494 Hz	523 Hz

Spectre de la gamme de musique

```
%Spectre de la gamme de musique


f=(0:N-1)*(fs/N);
signalAnalyzer(musique);


spectre_musique=fft(musique);
```

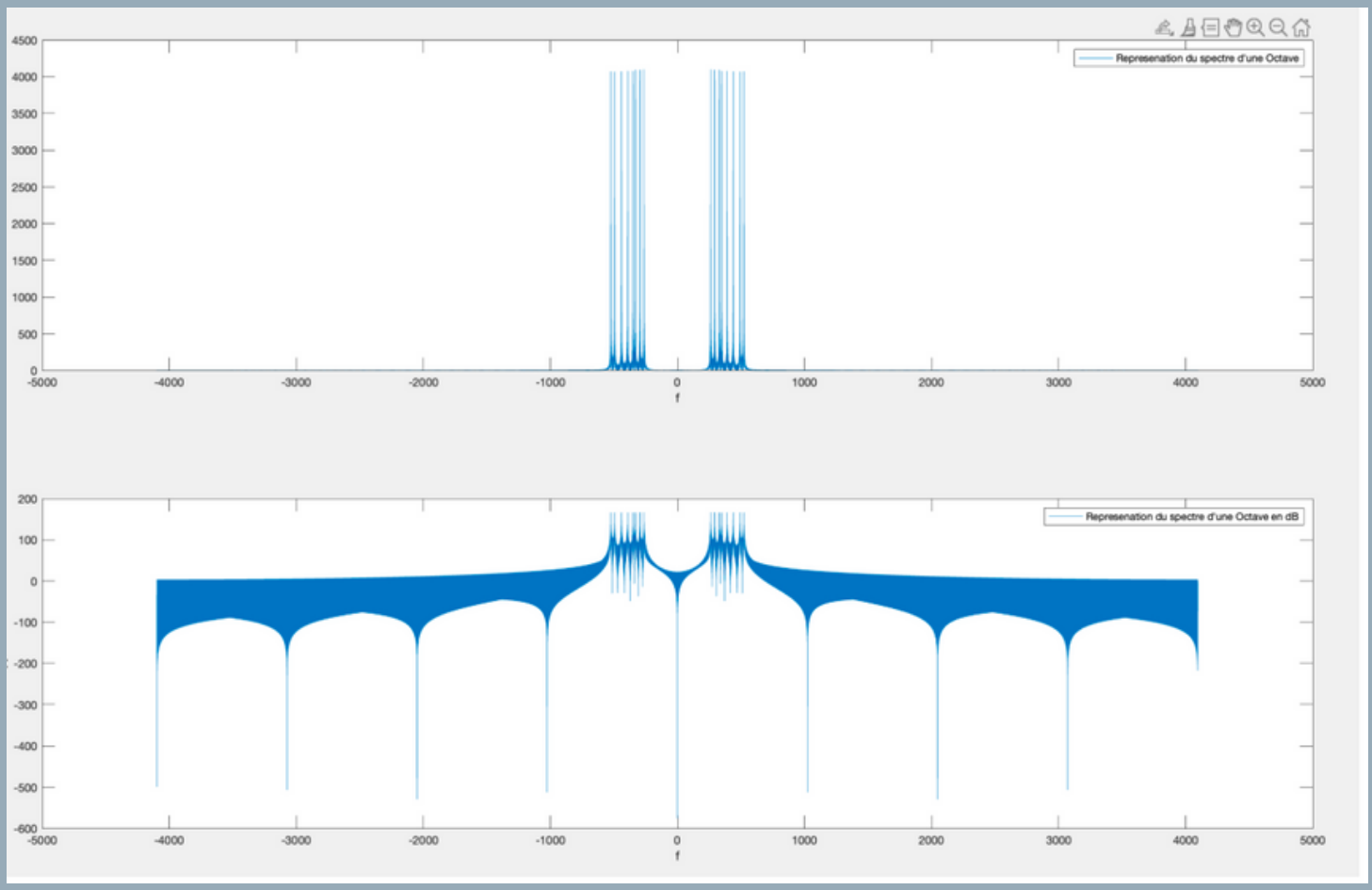


On peut bel et bien voir 8 fréquences

Approximation du spectre d’un signal sinusoïdal à temps continu par FFT

```
%% Approximation du spectre d'un signal sinusoïdal à temps continu par FFT

subplot(2,1,1)
    plot(fshift,fftshift(abs(y)));
    legend("Represenation du spectre d'une Octave");
    xlabel("f");
    ylabel("A");
    subplot(2,1,2)
    sig = 20*log(fftshift(abs(y)));
```

Le premier sous-graphique montre la représentation du spectre d'une octave en utilisant la fonction de plot et en affichant les valeurs absolues de y après l'application de la fonction fftshift. Le second sous-graphique montre la même représentation, mais en dB, en utilisant la fonction 20 * log pour convertir les valeurs absolues en dB. Les légendes, les étiquettes d'axe et les labels sont également ajoutés pour une meilleure visualisation.



Conclusion

En conclusion, ce TP a permis de comprendre les fondamentaux de la transformation de Fourier et de son utilisation pour visualiser le contenu fréquentiel d'un signal au cours du temps en traçant un spectrogramme. Le code matlab montré a permis de représenter le spectre d'une octave et de le représenter en dB pour une meilleure visualisation. Ce TP a donné une base solide pour continuer à explorer les applications de la transformation de Fourier en analyse de signaux.