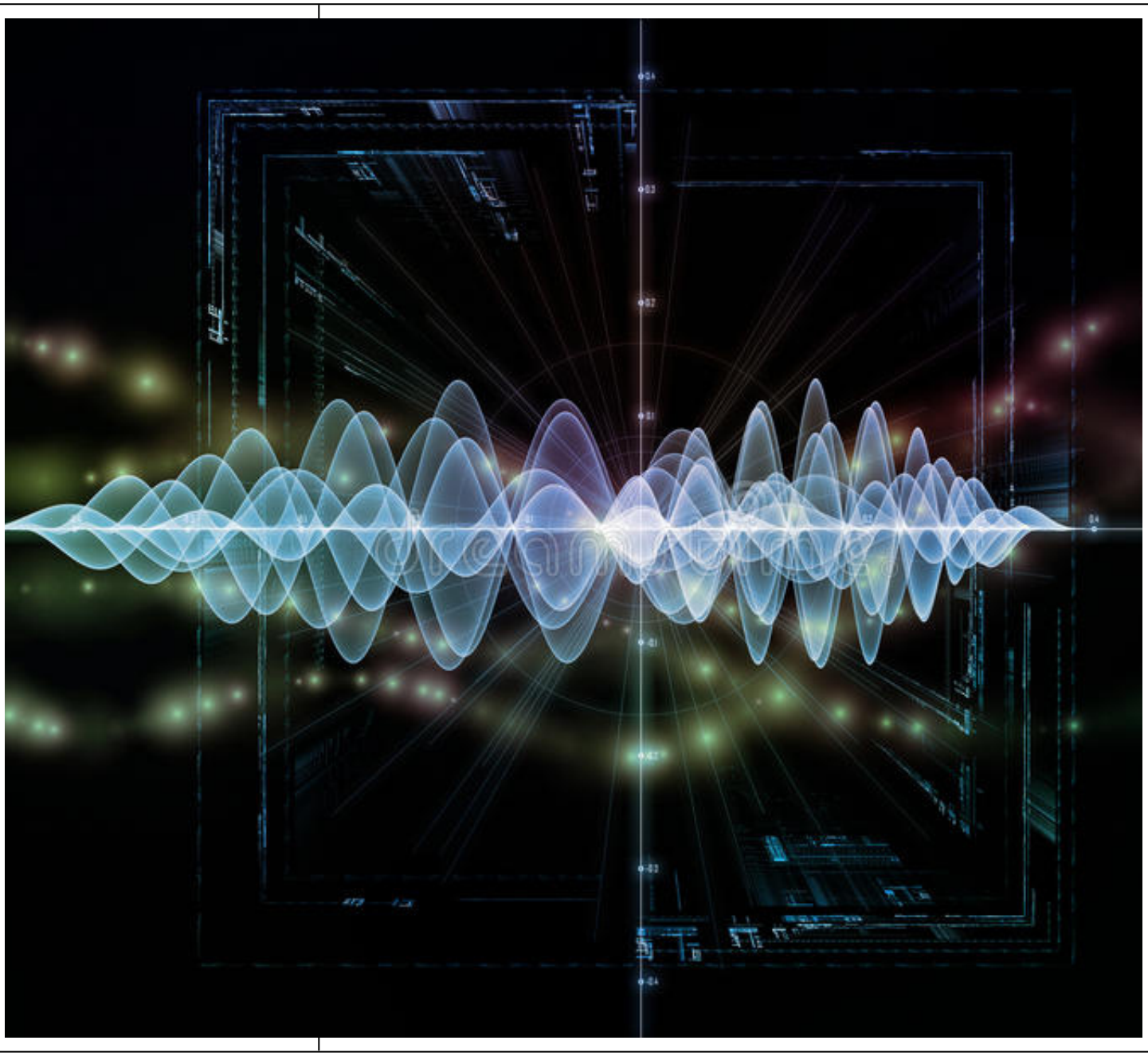
TP2

Synthèse et analyse spectrale d'une gamme de musique



BENACHOUR Ayoub

partie 1 :

qst 1

```
[x,fe]=audioread('phrase.wave');
```

la fonction audioread() retourne les données audio sous forme de vecteur, ainsi que la fréquence d'échantillonnage du fichier audio.

qst 2

```
sound(signal,fe)
te=1/fe; %te: periode d echantillonnage
t = (0:length(signal)-1) / fe;
plot(t,signal)
xlabel('Temps (s)');
ylabel('signal(t)');
```

Ce code crée des vecteurs de temps et d'amplitude en utilisant la fréquence d'échantillonnage fe, puis tracer le signal audio en utilisant la fonction plot(). Le signal audio est affiché en fonction du temps, avec l'amplitude du signal sur l'axe des ordonnées et le temps sur l'axe des abscisses.

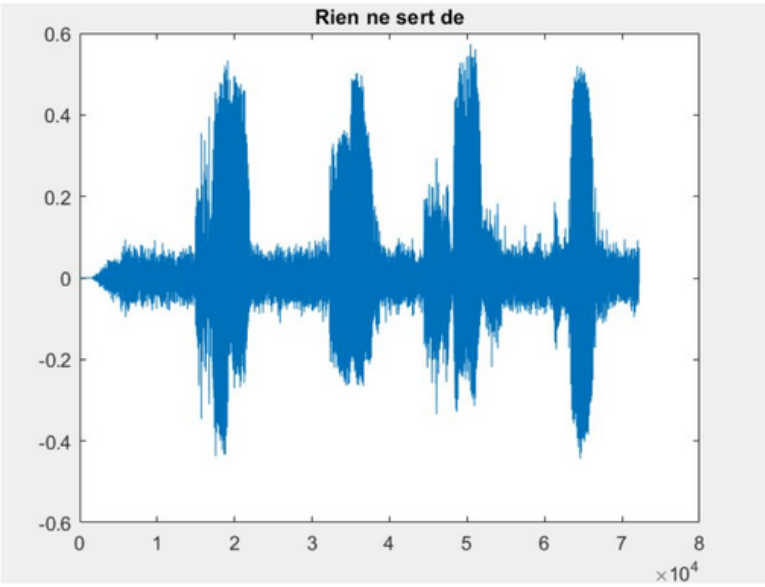
qst 3

```
sound(x,2*fe); %Donald Duck
sound(x,fe/2); %Terminator
```

La compression du spectre consiste à réduire l'écart entre les fréquences les plus hautes et les plus basses du signal, tandis que la dilatation du spectre consiste à augmenter cet écart. Si le spectre est comprimé, les fréquences les plus hautes et les plus basses du signal seront plus proches l'une de l'autre, ce qui peut entraîner une version plus grave du signal. Si le spectre est dilaté, les fréquences les plus hautes et les plus basses seront plus éloignées l'une de l'autre, ce qui peut entraîner une version plus aigüe du signal.

qst 4

```
%Tracez le signal en fonction des indices

plot(signal);
xlabel('indices');
title('signal')

% Trouver les indices du vecteur "rien ne sert de" dans le signal
rien_ne_sert_de_indice_debut = 5000;
rien_ne_sert_de_indice_fin = 100000;
```



-plot(signal) sert de tracer le signal en fonction des indices et non pas en fonction de temps, alors on peut maintenant clairement visualiser les mots constituant le signal .

-Alors , pour detecter la phrase "rien ne sert de" , il faut prendre l indice de debut du mot "rien" et l indice de fin du mot "de" pour construire l intervalle des indices du vecteur "rien ne sert de" .

qst 5

```
rien_ne_sert_de=signal(rien_ne_sert_de_indice_debut:rien_ne_sert_de_indice_fin);

% ecouter le vecteur "rien ne sert de"

 sound(rien_ne_sert_de,fe)

%Tracez le vecteur "rien ne sert de" en fonction des indices

plot(rien_ne_sert_de);
xlabel('indices');
title('rien ne sert de')
```

-sound(rien_ne_sert_de,fe) envoie le signal audio y au haut-parleur à la fréquence d'échantillonnage fe , puis , plot(rien_ne_sert_de); pour tracer le vecteur "rien ne sert de" en fonction des indices , on peut clairement voir les 4 mots qui contituent ce vecteur a partir des pics.

qst 6

```
% Trouver les indices du vecteur "courir" dans le signal
courir_indice_debut = 100000;
courir_indice_fin = 120000;
courir=signal(courir_indice_debut:courir_indice_fin);
%sound(courir,fe)

% Trouver les indices du vecteur "il faut" dans le signal
il_faut_indice_debut = 120000;
il_faut_indice_fin = 160000;
 il_faut=signal(il_faut_indice_debut:il_faut_indice_fin);
% sound(il_faut,fe)

% Trouver les indices du vecteur "partir a point" dans le signal
partir_a_point_indice_debut = 160000;
partir_a_point_indice_fin = 217000;
partir_a_point=signal(partir_a_point_indice_debut:partir_a_point_indice_fin);
%sound(partir_a_point,fe)
```

-de la meme maniere , on peut detecter facilement les autres vecteurs a partir des pics .

qst 7

```
reconstitution_signal =[rien_ne_sert_de;partir_a_point;il_faut;courir];
sound(reconstitution_signal,fe);
```

-on peut rearranger le vecteur initial pour ecouter differentes phrases en creant un vecteur colonne constituant des vecteurs du signal decomposés .

partie 2 :

qst 1

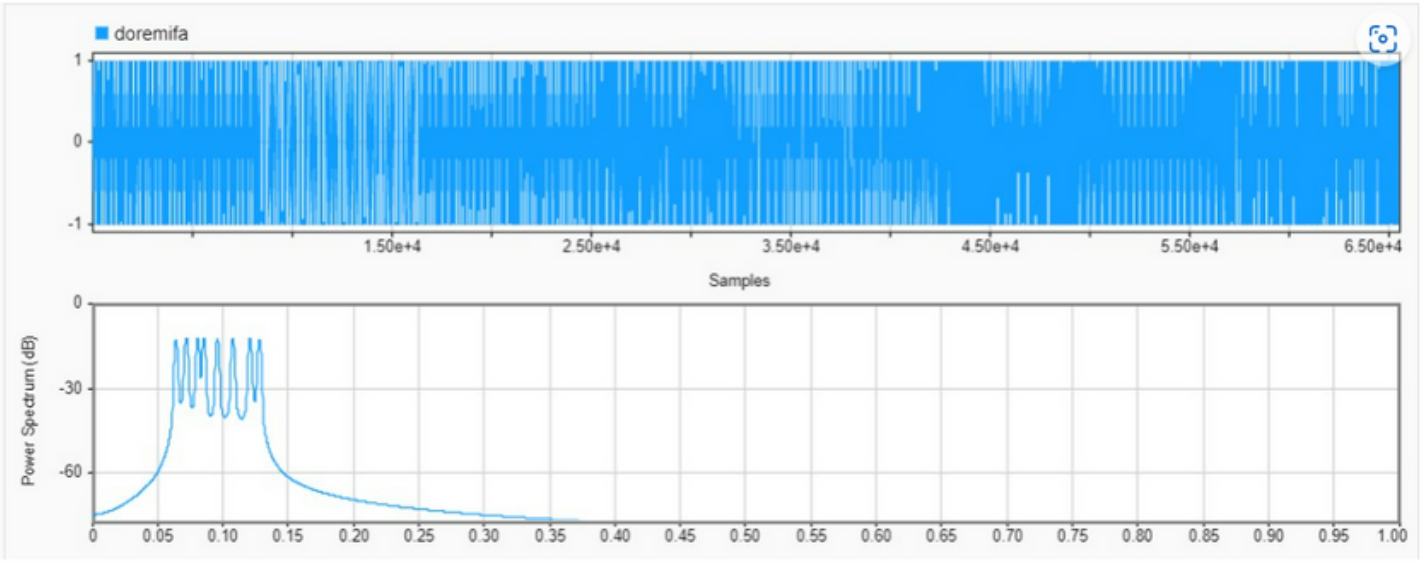
```
m_Fs=8192;
Ts=1/m_Fs;
t=[0:Ts:1];


F_dol=262;
F_re=294;
F_m=330;
F_fa=349;
F_sol=392;
F_si=494;
F_do2=523;
F_la=440;

Dol=sin(2*pi*F_dol*t);
re=sin(2*pi*F_re*t);
mi=sin(2*pi*F_m*t);
fa=sin(2*pi*F_fa*t);
so=sin(2*pi*F_sol*t);
la=sin(2*pi*F_la*t);
si=sin(2*pi*F_si*t);
do=sin(2*pi*F_do2*t);

gamme_de_musique= [Dol,re,mi,fa,so,la,si,do];
sound(gamme_de_musique,m_Fs);
```

- le vecteur"gamme_De_musique" est une gamme de musique constituee de 8 notes qui sont representées comme des signaux sinusoidaux avec leurs frequences correspondantes .

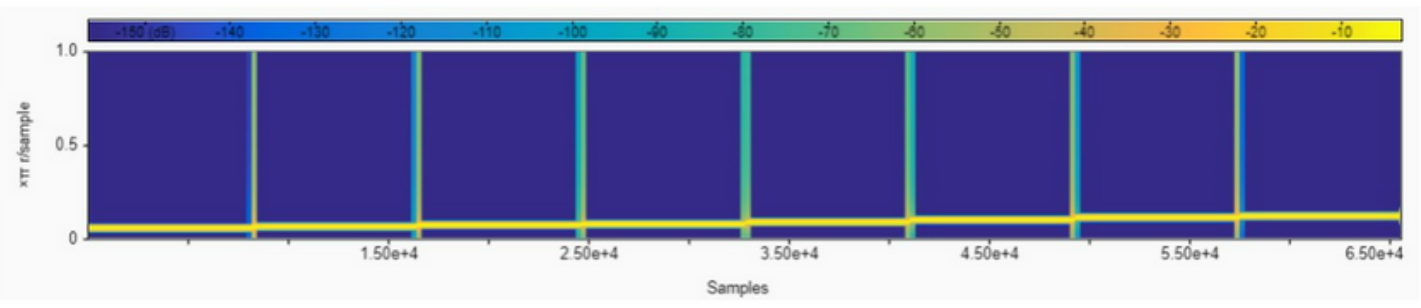
qst 2

-Le Signal Analyzer est un outil de visualisation de signal dans MATLAB. Il permet de visualiser et d'analyser des signaux numériques en utilisant diverses techniques d'analyse, telles que la transformée de Fourier, la décomposition en ondelettes, l'analyse de corrélation, etc.

qst 3

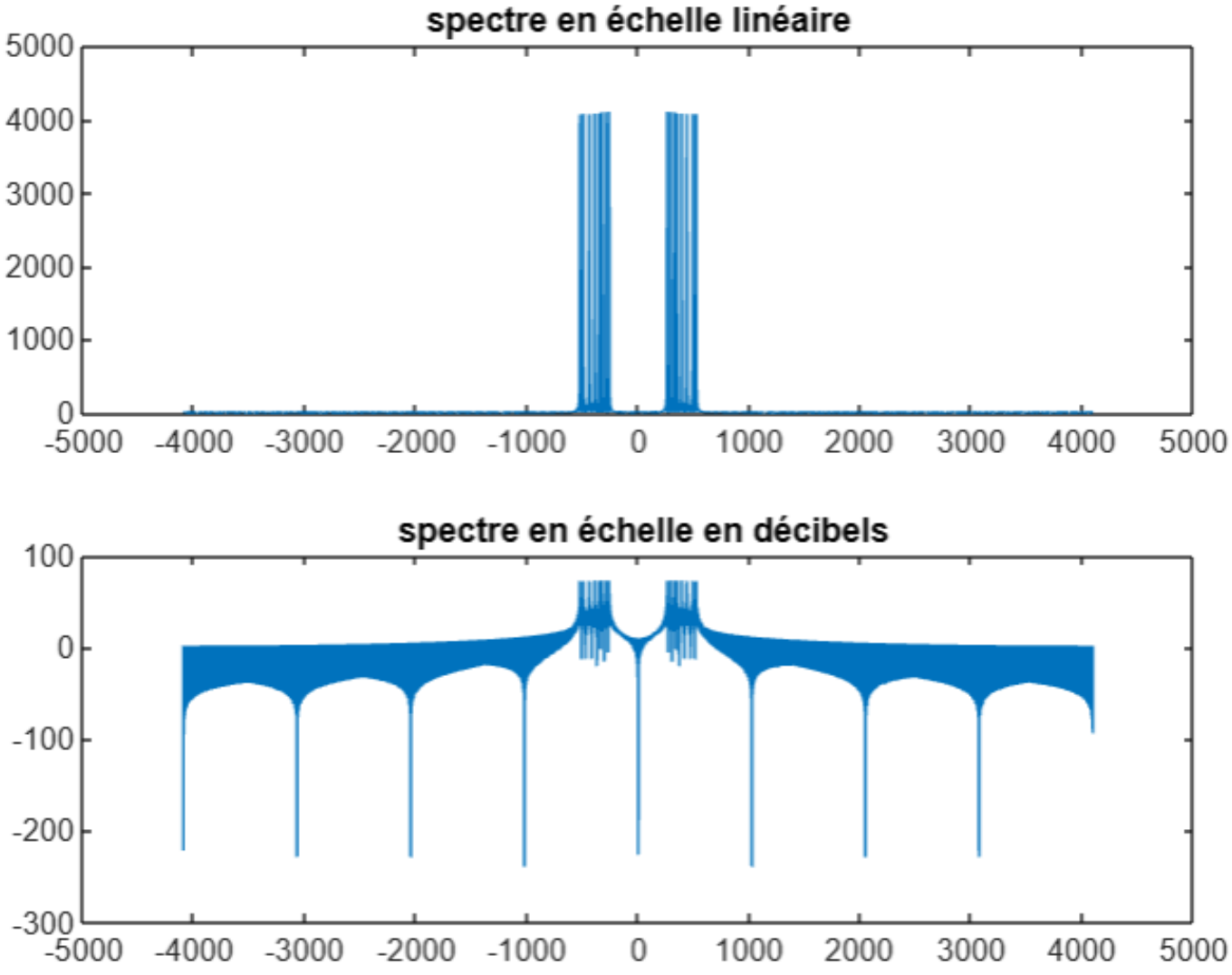


spectrogramme doremifa



qst 4

```
%tracer le spectre en echelle lineaire
spectre_lineaire=abs(fft(gamme_de_musique));
subplot(2,1,1);
fshift=(-length(gamme_de_musique)/2:length(gamme_de_musique)/2 -1 )*m_Fs/length(gamme_de_musique);
plot(fshift,fftshift(spectre_lineaire));
title('spectre en échelle linéaire');

%tracer le spectre en echelle en decibels
subplot(2,1,2);
spectre_db = 20 * log10(spectre_lineaire); % convertion en dB
plot(fshift,fftshift(spectre_db));
title(' spectre en échelle en décibels');
```



-L'échelle en décibels (dB) est une échelle logarithmique utilisée pour mesurer le niveau de puissance d'un signal. En revanche, l'échelle linéaire est une échelle qui représente les valeurs d'un signal de manière proportionnelle, sans utiliser de transformation logarithmique.