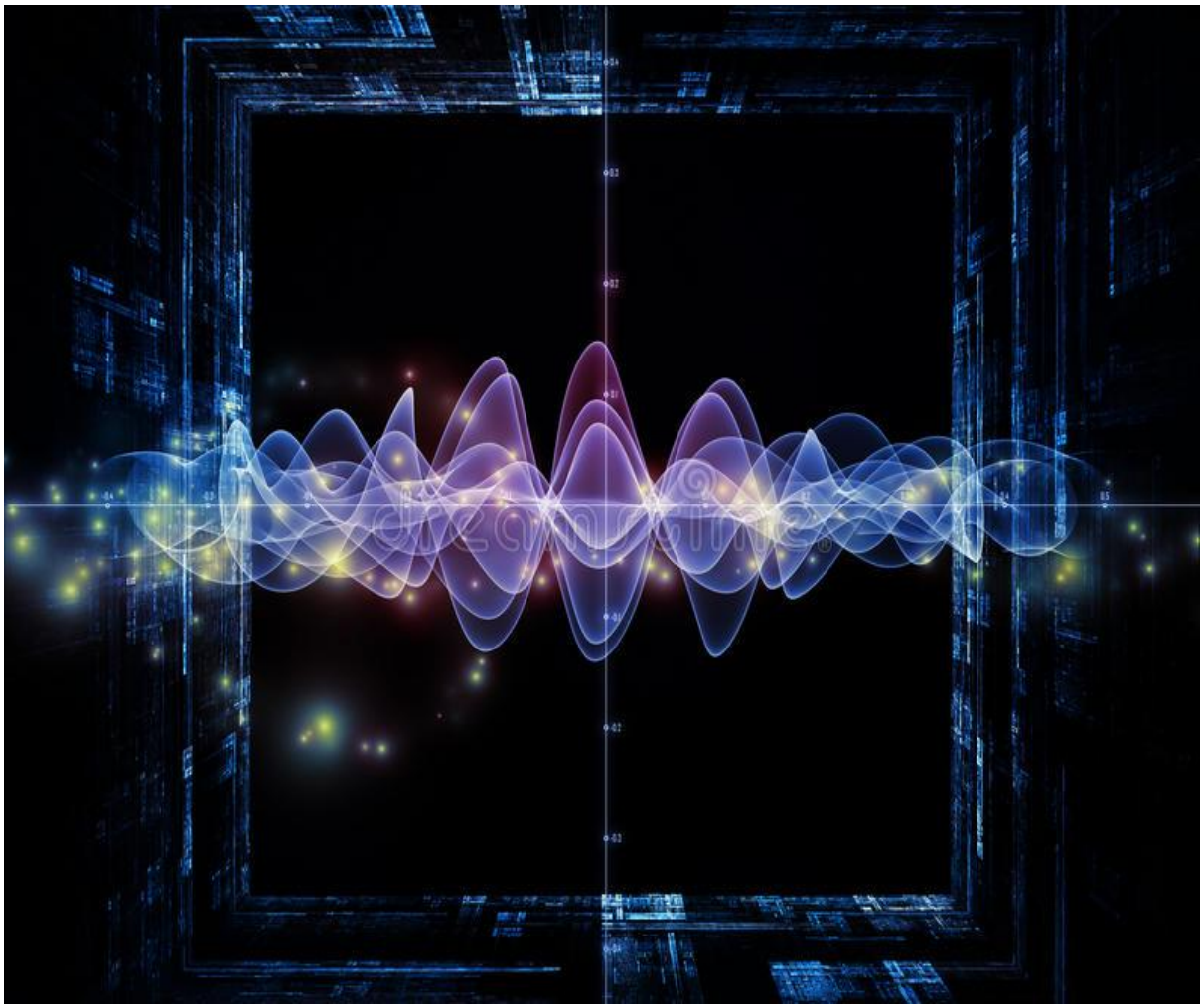
TP2 : jeux de mots :



Réaliser par : Youssef laamari

Module : Traitement de signal

Prof : AMMOUR Alae

Objectifs :

***Comprendre comment manipuler un signal audio avec Matlab, en effectuant certaines opérations classiques sur un fichier audio d'une phrase enregistrée via un smartphone.**
***Comprendre la notion des sons purs à travers la synthèse et l'analyse spectrale d'une gamme de musique.**

Commentaires : Il est à remarquer que ce TP traite en principe des signaux continus. Or, l'utilisation de Matlab suppose l'échantillonnage du signal. Il faudra donc être vigilant par rapport aux différences de traitement entre le temps continu et le temps discret. Tracé des figures : toutes les figures devront être tracées avec les axes et les légendes des axes appropriés. Travail demandé : un script Matlab commenté contenant le travail réalisé et des commentaires sur ce que vous avez compris et pas compris, ou sur ce qui vous a semblé intéressant ou pas, bref tout commentaire pertinent sur le TP.

1- Sauvegardez ce fichier sur votre répertoire de travail, puis charger-le dans MATLAB à l'aide de la commande « audioread ».

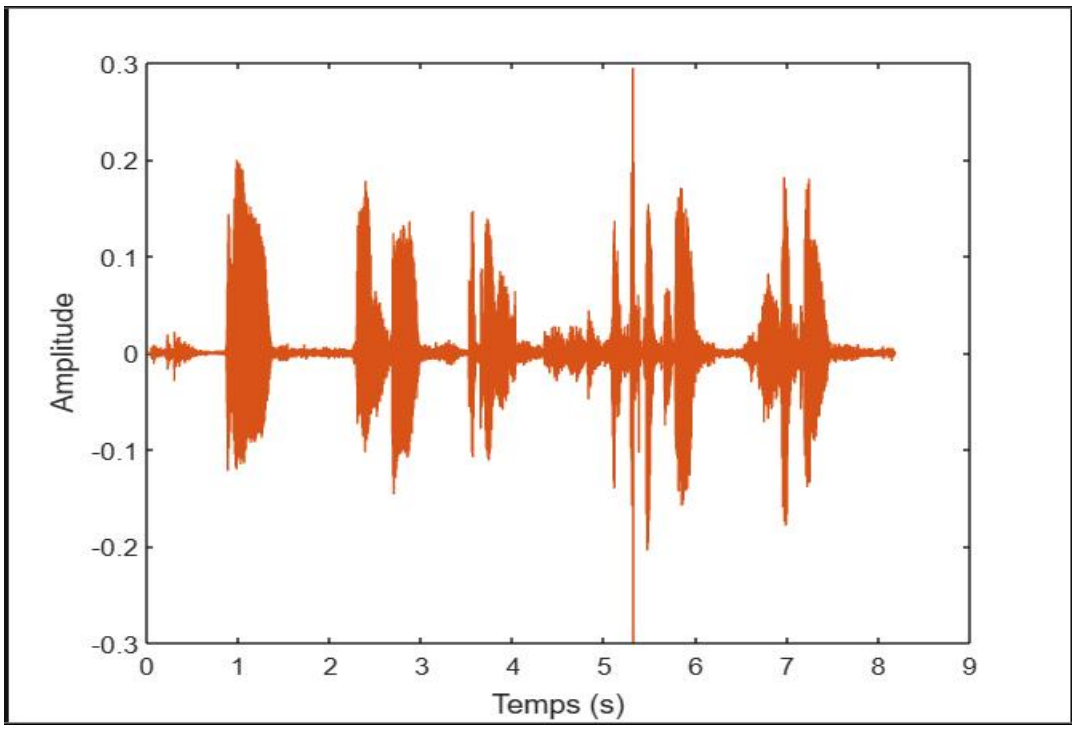
**********1**********

```
clear all
close all
Clc
% charger le fichier audio
[y, fs] = audioread('phrase.wav');
```

2- Tracez le signal enregistré en fonction du temps, puis écoutez-le en utilisant la commande « sound » .

**********2**********

```
clear all
close all
clc

% charger le fichier audio
[y, fs] = audioread('phrase.wav');

% plot le signal audio en fonction du temps
t = (1:length(y))/fs; % créer un vecteur temporel
plot(t, y);
xlabel('Temps (s)');
ylabel('Amplitude');

% jouer l'audio en utilisant la fonction son
sound(y,fs);
```



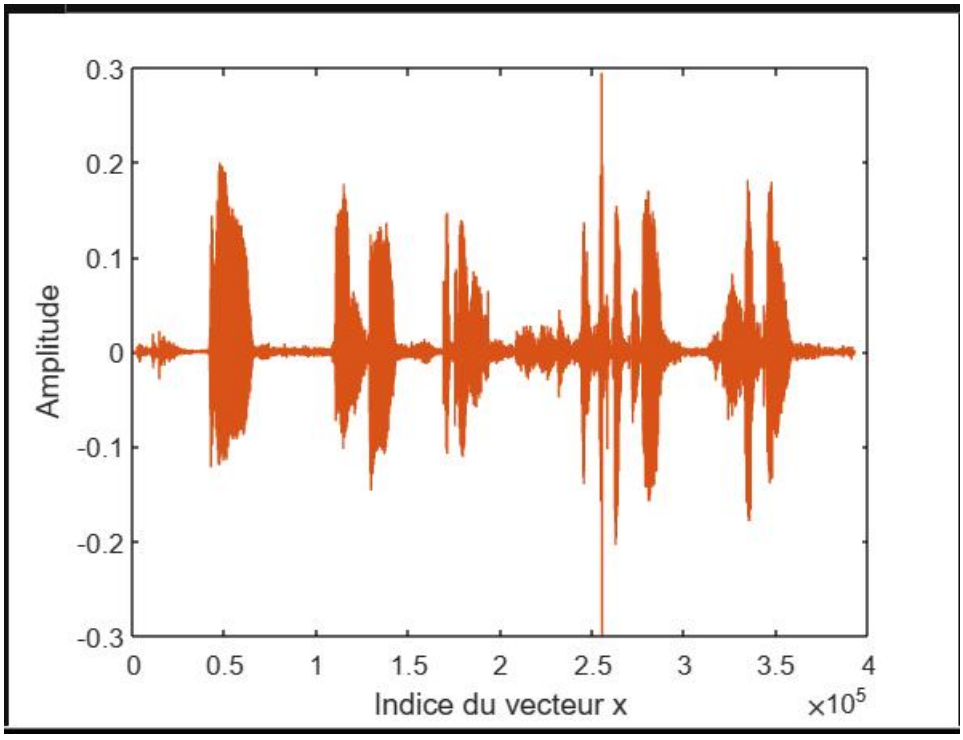
3- Cette commande permet d'écouter la phrase à sa fréquence d'échantillonnage d'enregistrement. Écoutez la phrase en modifiant la fréquence d'échantillonnage à double ou deux fois plus petite pour vous faire parler comme « Terminateur » ou « Donald Duck ». En effet, modifier la fréquence d'échantillonnage revient à appliquer un changement d'échelle y(t) = x(at) en fonction de la valeur du facteur d'échelle, cela revient à opérer une compression ou une dilatation du spectre initial d'où la version plus grave ou plus aigüe du signal écouté.

**********3**********

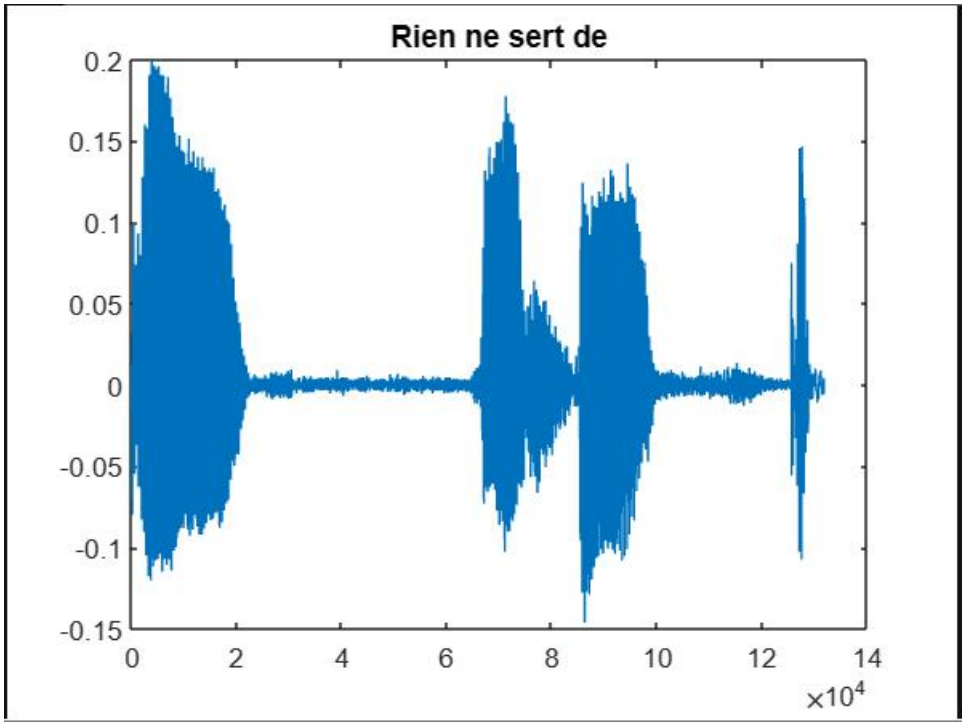
```
% jouer l'audio en utilisant la fonction son
sound(y,fs*2); % Terminator
sound(y,fs/2); % Donald Duck
```

4- Tracez le signal en fonction des indices du vecteur x, puis essayez de repérer les indices de début et de fin de la phrase « Rien ne sert de ».

```
clear all
close all
clc

% charger le fichier audio
[y, fs] = audioread('phrase.wav');
n = length(y);
plot(1:n,y)
xlabel('Indice du vecteur x')
ylabel('Amplitude')
sound(y,fs)
```




```
clear all
close all
clc

% charger le fichier audio
[y, fs] = audioread('phrase.wav');

riennesertde = y(43325:175100);
plot(riennesertde);
title('Rien ne sert de');
```

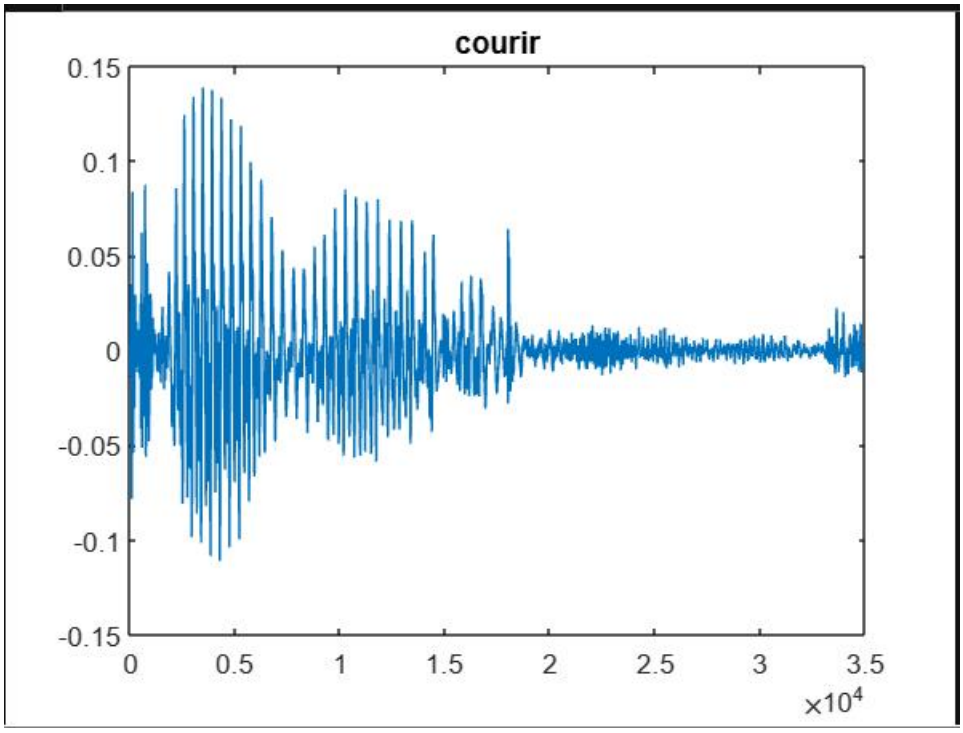


5-Pour segmenter le premier mot, il faut par exemple créer un vecteur « riennesertde » contenant les n premières valeurs du signal enregistré x qui correspondent à ce morceau. Créez ce vecteur, puis écoutez le mot segmenté .

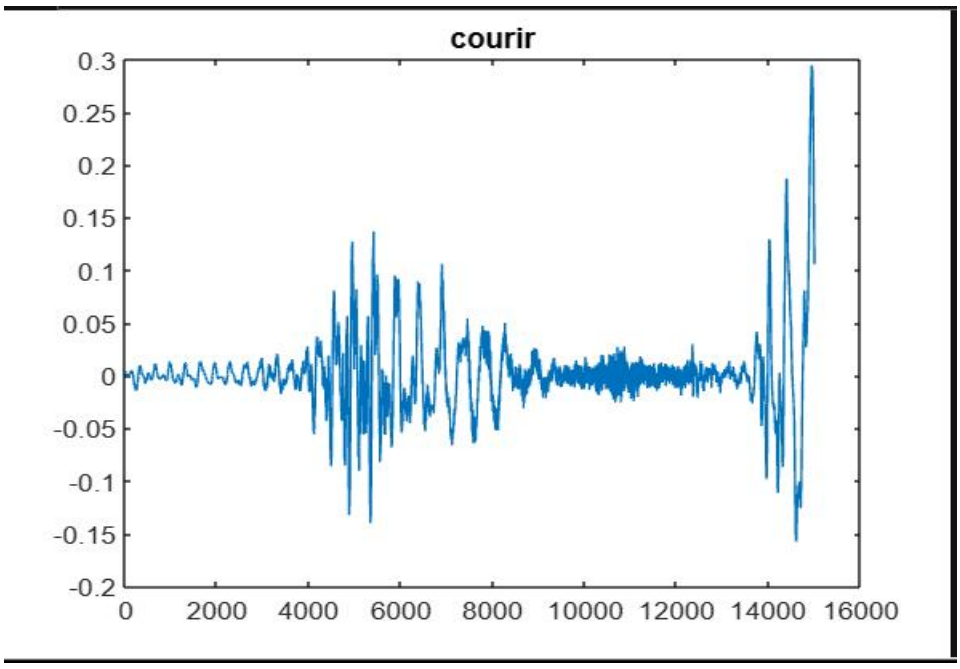
```
clear all
close all
clc

% charger le fichier audio
[y, fs] = audioread('phrase.wav');

riennesertde = y(43325:175100);
plot(riennesertde);
title('Rien ne sert de');
sound(riennesertde,fs);
```

6- Segmentez cette fois-ci toute la phrase en créant les variables suivantes : riennesertde, courir, ilfaut, partirapoint ,

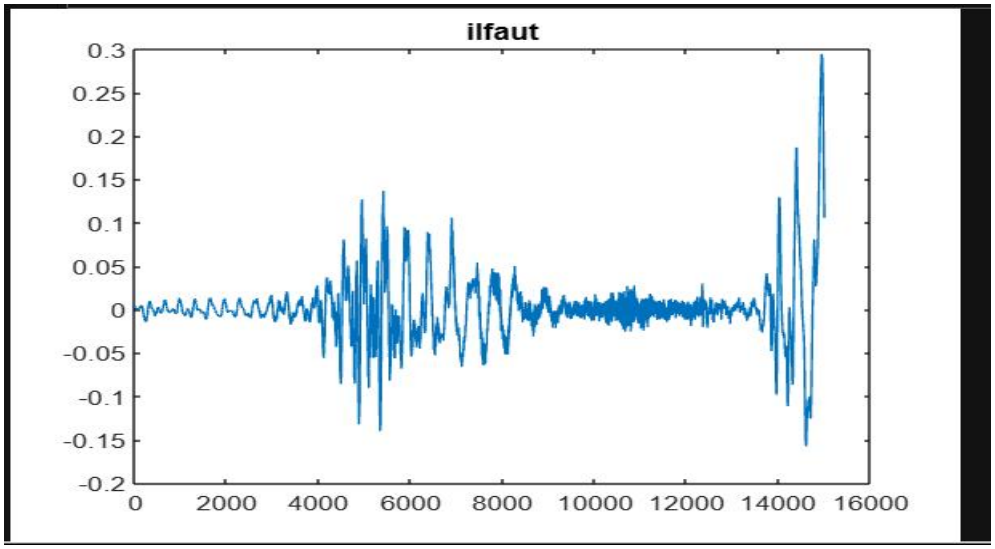
```
clear all
close all
clc

% charger le fichier audio
[y, fs] = audioread('phrase.wav');

courir = y(175100:210000);
plot(courir);
title('courir');
sound(courir,fs);
```



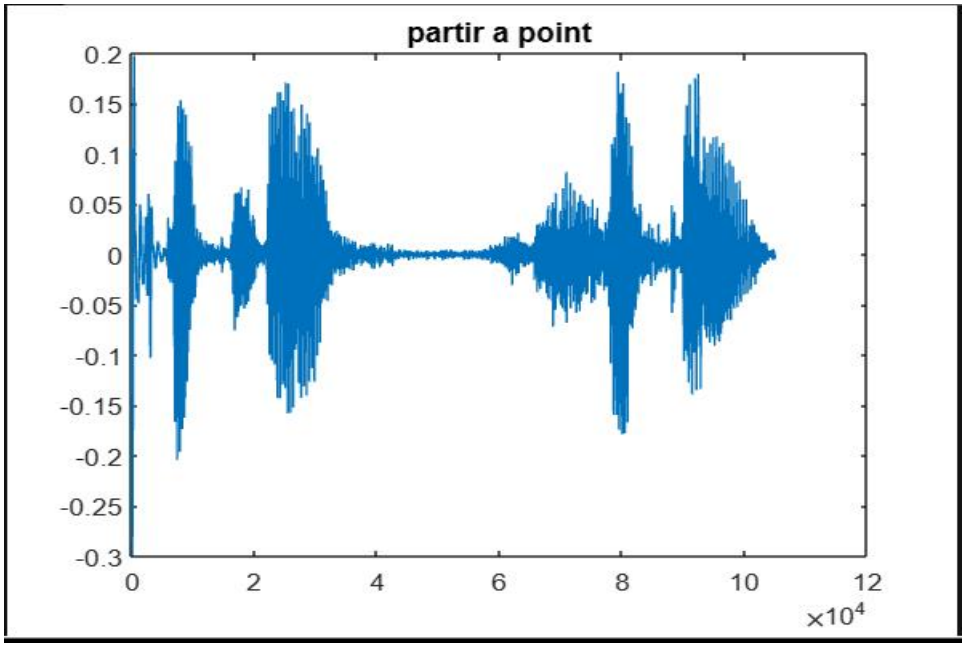

```
clear all
close all
clc

% charger le fichier audio
[y, fs] = audioread('phrase.wav');

ilfaut = y(240000:255000);
plot(ilfaut);
title('ilfaut');
sound(ilfaut,fs);
```



```
clear all
close all
clc
% charger le fichier audio
[y, fs] = audioread('phrase.wav');
ilfaut = y(240000:255000);
plot(ilfaut);
title('ilfaut');
sound(ilfaut,fs);
```

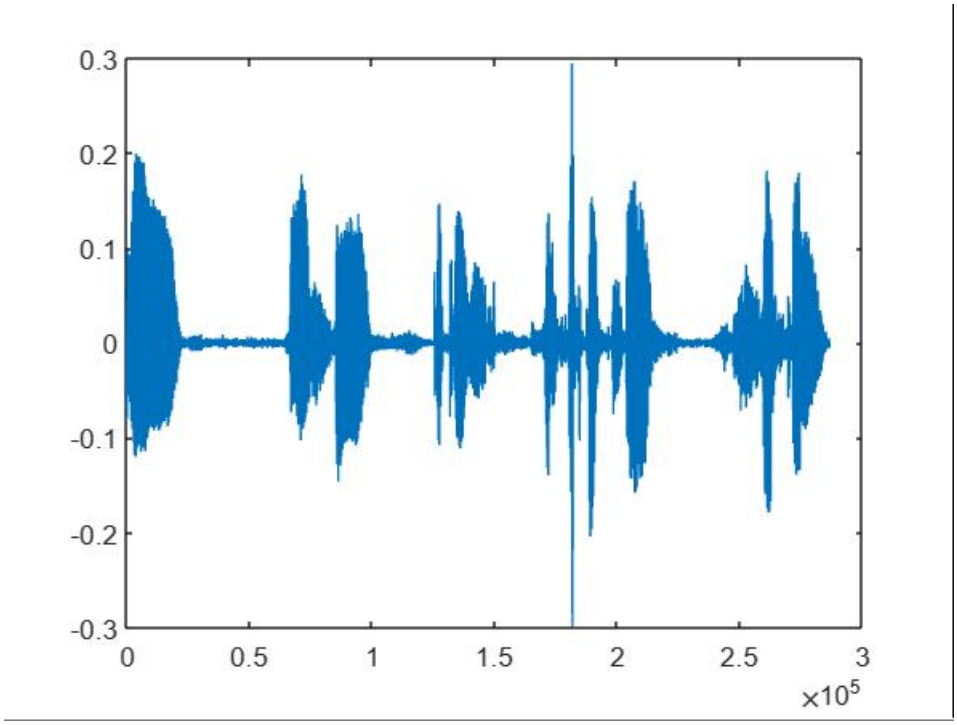



```
clear all
close all
clc
% charger le fichier audio
[y, fs] = audioread('phrase.wav');
partirapoint = y(255000:360000);
plot(partirapoint);
title('partir a point');
sound(partirapoint,fs);
```



7- Notez que le signal initial de parole est un vecteur colonne contenant un certain nombre de valeurs (length(x)). Réarrangez ce vecteur pour écouter la phrase synthétisée « Rien ne sert de partir à point, il faut courir ».

```
clear all
close all
clc
% charger le fichier audio
[y, fs] = audioread('phrase.wav');
riennesertde = y(43325:175100);
courir = y(175100:210000);
ilfaut = y(240000:255000);
partirapoint = y(255000:360000);
nouveau_signal = [riennesertde,courir,ilfaut,partirapoint];
plot(nouveau_signal);
sound(nouveau_signal,fs);
```



Synthèse et analyse spectrale d'une gamme de musique :

- Synthèse d'une gamme de musique Les notes

de musique produites par un piano peuvent être synthétisées approximativement numériquement. En effet, chaque note peut être considérée comme étant un son pur produit par un signal sinusoïdal. La fréquence de la note « La » est par exemple de 440 Hz. 1- Créez un programme qui permet de jouer une gamme de musique. La fréquence de chaque note est précisée dans le tableau ci-dessous. Chaque note aura une durée de 1s. La durée de la gamme sera donc de 8s. La fréquence d'échantillonnage fe sera fixée à 8192 Hz.

```
clear all
close all
clc
% fréquence d'échantillonnage
fe = 8192;
% durée de chaque note
duree_de_note = 1;
% fréquences des notes
frequences_des_notes = [262, 294, 330, 349, 392, 440, 494, 523];

% Génération des signaux de chaque note
signaux = cell(1, length(frequences_des_notes));
for i = 1:length(frequences_des_notes)
t = 0:1/fe:duree_de_note ;
signaux{i} = sin(2*pi*frequences_des_notes(i)*t);
end

% Concaténation des signaux pour jouer masikha
masikha = [signaux{:}];

% Lecture de masikha
sound(masikha, fe);
```

- Spectre de la gamme de musique

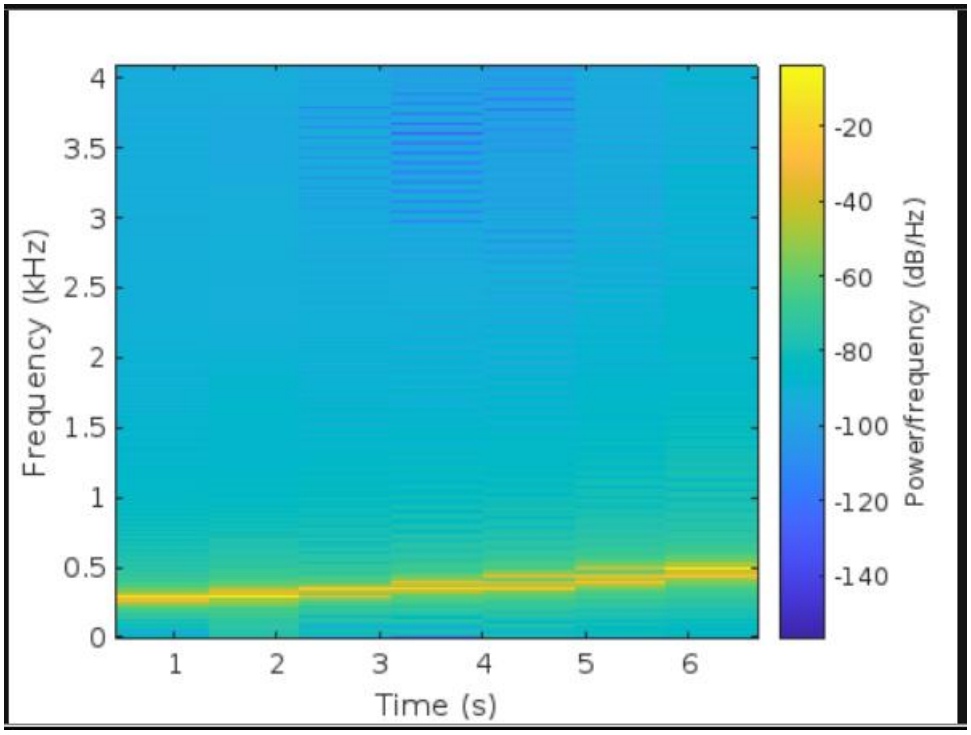
2- Utilisez l'outil graphique d'analyse de signaux signalAnalyzer pour visualiser le spectre de votre gamme. Observez les 8 fréquences contenues dans la gamme et vérifiez leur valeur numérique à l'aide des curseurs.

----------2----------

```
signalAnalyzer(masikha, fe);
```

3- Tracez le spectrogramme qui permet de visualiser le contenu fréquentiel du signal au cours du temps (comme le fait une partition de musique) mais la précision sur l'axe des fréquences n'est pas suffisante pour relever précisément les 8 fréquences.

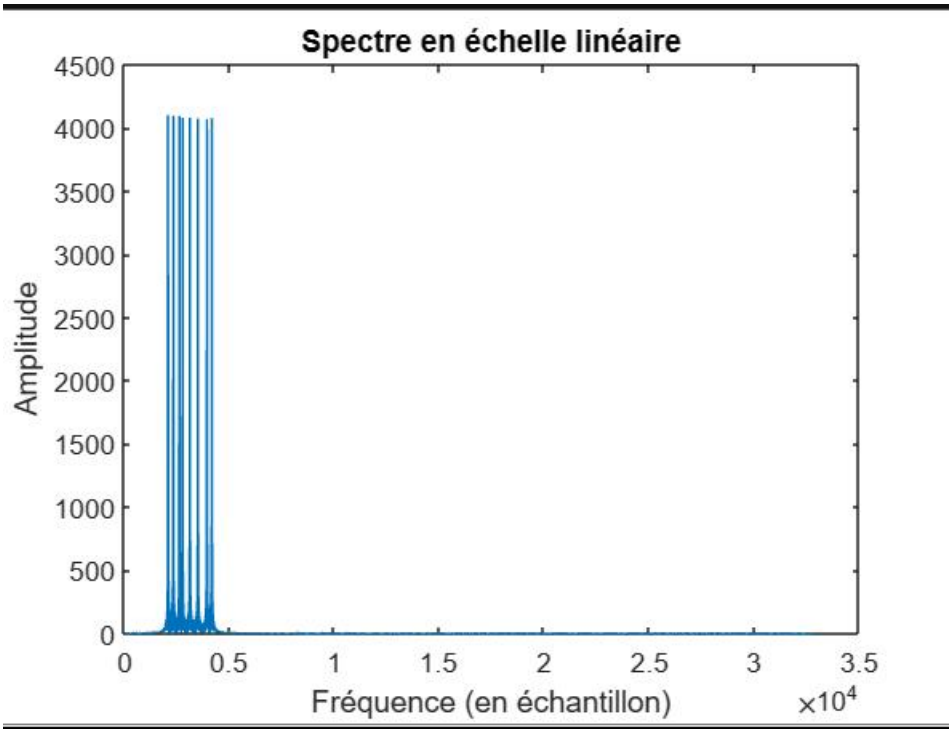
```
figure(1)
spectrogram(masikha, [], [], [], fe, 'yaxis');
```

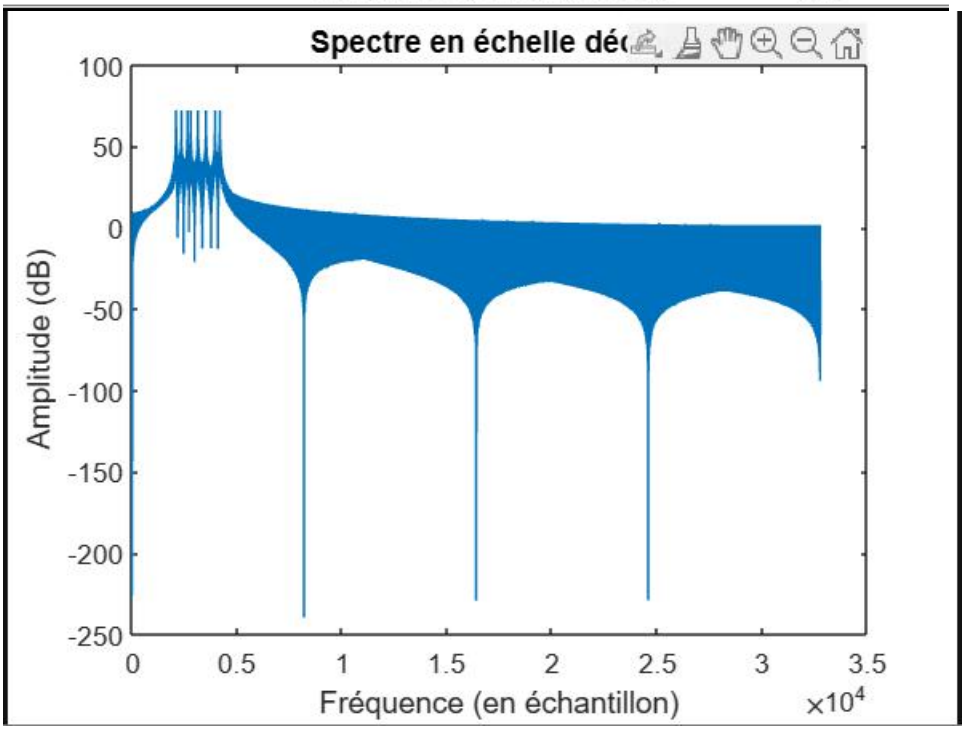


- Approximation du spectre d'un signal sinusoïdal à temps continu par FFT

4-Le spectre d'un signal à temps continu peut être approché par transformée de Fourier discrète (TFD) ou sa version rapide (Fast Fourier Transform (FFT). Afficher le spectre de fréquence de la gamme musicale crée en échelle linéaire, puis avec une échelle en décibels.

```
clear all
close all
clc



fe = 8192; % fréquence d'échantillonnage
duree = 1; % durée de chaque note
% fréquences des notes
frequences_des_notes = [262, 294, 330, 349, 392, 440, 494, 523];

% Génération des signaux de chaque note
signaux = cell(1, length(frequences_des_notes));
for i = 1:length(frequences_des_notes)
t = 0:1/fe:duree;
signaux{i} = sin(2*pi*frequences_des_notes(i)*t);
end

% Concaténation des signaux pour jouer la gamme
masikha = [signaux{:}];

% Calcul de la FFT
fft_masikha = fft(masikha);
N = length(fft_masikha);

% Affichage du spectre en échelle linéaire
figure;
plot(abs(fft_masikha(1:N/2)));
xlabel('Fréquence (en échantillon)');
ylabel('Amplitude');
title('Spectre en échelle linéaire');

% Affichage du spectre en échelle de décibels
figure;
plot(20*log10(abs(fft_masikha(1:N/2))));
xlabel('Fréquence (en échantillon)');
ylabel('Amplitude (dB)');
title('Spectre en échelle décibel')
```


Spectre en échelle linéaire
Amplitude
Fréquence (en échantillon)
×10^4


Spectre en échelle dé
Amplitude (dB)
Fréquence (en échantillon)
×10^4