TRAITEMENT DE SIGNAL :

RAPPORT DU TP1 ,TP2 ,TP3,TP4

Nom prénom : **OMAR HABOUB**

Tp1

Représentation temporelle et fréquentielle :

1- Tracer le signal x(t) :

$$x(t) = 1.2\cos(2pi440t + 1.2) + 3\cos(2pi550t) + 0.6\cos(2pi2500t)$$

Nous déclarons les variables :

```
%% Représentation temporelle et fréquentielle

x = 1.2*cos(2*pi*440*t+1.2)+3*cos(2*pi*550*t)+0.6*cos(2*pi*2500*t);
figure;
```

nous avons utiliser la fonction plot pour tracer le signal

2-la TFD du signal x(t) :

-Nous utilisons la commande fft pour faire la transforméede fourier discret d'une manière rapide
- On fait abs pour avoir le module et pour afficher le spectre d’amplitude

3-

Nous utilisons la fonction fftshift pour faire le décalage
On doit créer une autre variable

- "fe" représente la fréquence d'échantillonnage et la variable
- "N" représente la longueur de l'échantillon.
Le décalage de fréquence est calculé en utilisant l'expression (-N/2:N/2-1)*(fe/N)* .

4-

On va créer un nouveau signal xnoise

```
%Introduction de bruit
```

On Utilise la commande randn pour générer ce bruit
La fonction "randn" génère des nombres aléatoires à partir d'une distribution normale avec une moyenne de 0 et une déviation standard de 1.
La fonction "size" prend en entrée un tableau (dans ce cas "x") et renvoie la taille du tableau sous la forme (lignes, colonnes).

5-

```
%sound(noise)
xnoise = x+noise;
```

On utilise la commande sound pour écouter le signal

6-

REMARQUE :

Ce code effectue une transformée de Fourier rapide (FFT) sur le signal« xnoise » et stocke le résultat dans une variable « ybruit ».
La FFT est utilisée pour convertir un signal du domaine temporel au domaine fréquentiel.
La fonction fftshift est utilisée pour déplacer la composante de fréquence nulle vers le centre du spectre et la fonction abs est utilisée pour prendre l'amplitude de la sortie complexe de la FFT.
La ligne suivante, plot(fshift,fftshift((abs(ybruit)*2)/N)), trace la FFT normalisée et décalée, et le titre est réglé sur « Le signal noise ».

7-

Ce code trace le résultat en utilisant les valeurs de décalage de fréquence sur l'axe des x et la valeur absolue de la fft, échelle par un facteur de 2/N.
Il est important de noter que le graphique montre l'amplitude de la FFT, qui est une valeur complexe, donc l'amplitude est élevée au carré et échelle par un facteur de 2/N, comme c'est courant en traitement du signal. La fonction fftshift est également utilisée pour centrer le contenu de fréquence du signal

En augmentant l'intensité de bruit, on s'attend à voir une augmentation de la puissance du bruit dans toutes les bandes de fréquences, ce qui se traduit par une diminution du rapport signal à bruit (SNR) et une diminution de la qualité du signal. On peut observer dans le graphique obtenu une augmentation de la puissance du bruit dans toutes les bandes de fréquences, ce qui se traduit par une diminution de la clarté du signal et une difficulté à distinguer les différentes composantes fréquentielles du signal d'origine.

Analyse fréquentielle du chant du rorqual bleu :

1-

Ce code utilise la fonction "audioread" pour lire un fichier audio nommé "bluewhale.au" et le stocker dans la variable "whale", ainsi que la fréquence d'échantillonnage dans la variable "fe". Il crée également une nouvelle variable "chant" qui contient un échantillon de la variable "whale" en utilisant les valeurs de l'index 2.45e4 à 3.10e4. Cela permet de sélectionner une portion spécifique du fichier audio pour analyse ou utilisation ultérieure.

2-

```
sound(chant,fe)
```

3-

Ce code calcule la longueur de l'échantillon de la variable "chant" et la stocke dans la variable "N".

Il calcule également le temps d'échantillonnage en utilisant la variable "fe" et le stocke dans la variable "te", et crée ensuite un tableau de temps "t" en utilisant les valeurs de l'index de 0 à N-1 multipliées par (10*te). Il crée une nouvelle figure, et dans la première sous-figure, il trace le signal "chant" sur l'axe des x et des y, avec le titre "Le signal whant".

Il calcule ensuite la transformée de Fourier (FFT) du signal "chant" et le stocke dans la variable "y" et calcule la densité spectrale de puissance, ensuite il crée un tableau de fréquences "f" en utilisant les valeurs de l'index de 0 à la partie entière de (N/2) multipliées par (fe/N)/10, Dans la deuxième sous-figure, il trace la densité spectrale de puissance sur l'axe des x et des y, avec le titre "Le signal densité spectrale de puissance du signal".

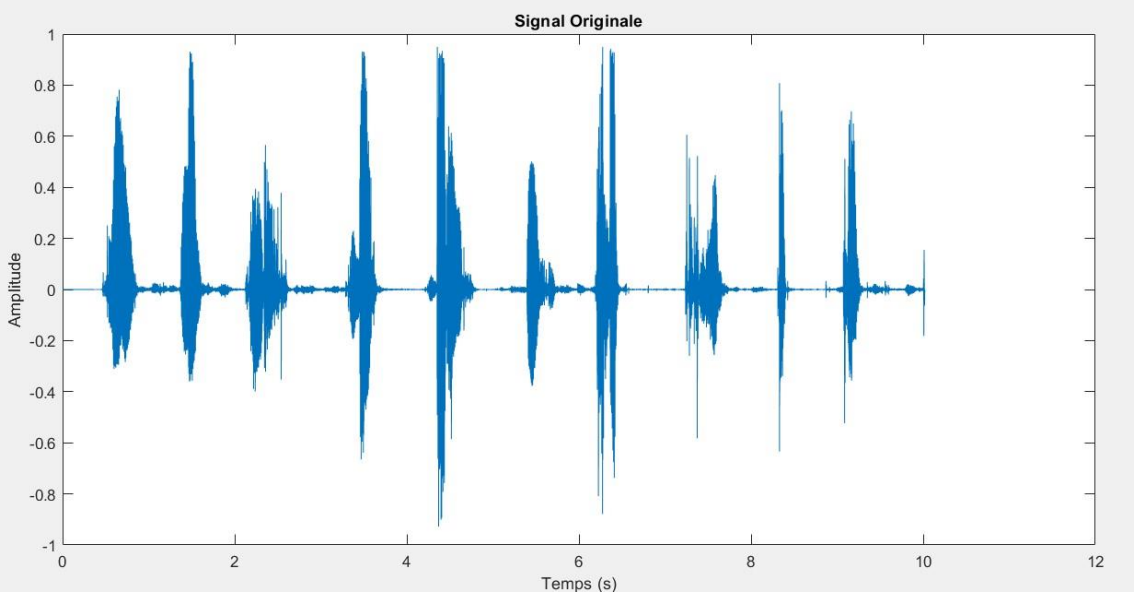
TP2

1-

```
[y,Fs]=audioread('phrase.wav');
dt = 1/Fs;
t = 0:dt:(length(y)-1)*dt;
```

2-

```
sound(y,Fs);
```



3-

Pour jouer le fichier audio chargé dans MATLAB,on utilise la fonction "sound" avec les paramètres appropriés. La première ligne de code "sound(y, 2*fs)" jouera le fichier audio à une fréquence d'échantillonnage deux fois plus élevée que celle d'origine. La seconde ligne de code "sound(y, fs/2)" jouera le fichier audio à une fréquence

d'échantillonnage deux fois plus faible que celle d'origine. Ainsi, vous pourrez entendre une différence dans la qualité sonore et la vitesse de lecture de l'audio.

4-

```
rien_ne_sert_de = y(5055:77249);
```

Cela crée une nouvelle variable appelée "rien_ne_sert_de" qui contient les échantillons audio correspondant aux mots "Rien ne sert de" dans le fichier d'origine.

```
plot(rien_ne_sert_de);
title('Rien ne sert de');
```

5-
Pour jouer la portion de la phrase "Rien ne sert de" de l'audio chargé dans MATLAB
pour jouer ces échantillons audio, on utilise la commande sound

```
rien_ne_sert_de = y(5055:77249);
sound(rien_ne_sert_de,Fs);
```

6-
Cela crée des variables distinctes pour chaque partie de la phrase.

7-
Cela crée une nouvelle variable appelée "new_phrase" qui contient toutes les échantillons audio des différentes parties de la phrase

Cela jouera les échantillons audio contenus dans la variable "new_phrase" à la fréquence d'échantillonnage d'origine (Fs).

Synthèse et analyse spectrale d’une gamme de musique

1-

Pour combiner ces ondes pour créer une mélodie, on utilise la concaténation comme suit

```
doremifasol_solfamiredo= [Dol,re,mi,fa,so,la,si,do,do,si,la,so,fa,mi,re,Dol];
faded =[fa,fa,fa,si,mi,mi,re,si,si,si,si,fa,fa,fa,mi];
doremifa =[Dol,re,mi,fa,so,la,si,do];
inv =[do,si,la,so,fa,mi,re,do];
sound(doremifasol_solfamiredo,m_Fs);
```

2-

```
rien_ne_sert_de = y(5055:77249);
partir_a_point  = y(121652:168300);
il_faut = y(95393:121652);
courir  = y(76613:95393);
```

4-

```
Sp=abs(fft(doremifa));
u=mag2db(Sp);
```

Cela calcule la transformation de Fourier rapide (FFT) du signal audio contenu dans la variable "doremifa", puis convertit cette FFT en une échelle de dB pour une meilleure visualisation.

```
subplot(2,1,1);
fshift=(-length(doremifa)/2:length(doremifa)/2 -1 )*Fs/length(doremifa);
plot(fshift,fftshift(Sp));
title('signal(doremi) spectre');
subplot(2,1,2);
plot(fshift,fftshift(u));
title('signal(doremi) spectre shifted');
```

Cela crée un sous-graphique divisé en deux parties, la première affiche le spectre du signal audio dans la variable "doremifa" et la deuxième affiche le même spectre, mais décalé pour une meilleure visualisation.

TP3

1-

% Charger le fichier .mat contenant le signal ECG

load('ecg.mat');

2-

La commande "load("ecg.mat")" permet de charger les données du fichier "ecg.mat" dans le workspace de MATLAB. Les données dans ce fichier doivent être enregistrées sous la forme d'une variable nommée "ecg".

Ensuite, la commande "fs = 500" définit la fréquence d'échantillonnage à 500 Hz. La commande "t = (0:length(ecg)-1)/fs" crée un vecteur de temps correspondant à chaque échantillon dans le signal "ecg" en utilisant la fréquence d'échantillonnage définie.

La commande "plot(t, ecg)" permet de tracer le signal "ecg" en fonction du temps. Les commandes "xlabel('Temps (s)')" et "ylabel('Amplitude')" ajoutent des étiquettes à l'axe des x et des y respectivement.

Ces commandes permettent de tracer le signal électrocardiogramme (ECG) en fonction du temps et de visualiser les variations de l'amplitude du signal par rapport au temps. Il serait possible de visualiser des anomalies ou des caractéristiques particulières dans le signal comme des ondes P,Q,R,S,T.

3-

```
% Faire un zoom sur une période du signal
xlim([t_start t_end])
```

La commande "xlim([t_start t_end])" permet de zoomer sur une période spécifique du signal en limitant l'affichage de l'axe des x entre les valeurs "t_start" et "t_end" en secondes. Il est important que les valeurs t_start et

t_end soient comprises dans l'intervalle de temps représenté par le vecteur t créé précédemment, sinon il y aurait une erreur. Cette commande permet de zoomer sur une période spécifique du signal pour une meilleure visualisation des détails, par exemple pour observer une zone d'anomalie ou une caractéristique particulière.

4-

```
% Tracer les deux signaux côte à côte pour faciliter la comparaison
```

La commande "subplot(2,1,1)" permet de créer un sous-graphique divisé en 2 lignes et 1 colonne, et de sélectionner le premier sous-graphique pour y tracer le signal "ecg_signal". La commande "subplot(2,1,2)" permet de sélectionner le deuxième sous-graphique pour y tracer le signal "ecg_filtered".

Ces commandes permettent de tracer les deux signaux côte à côte pour faciliter la comparaison entre le signal d'origine et le signal filtré. Il est possible de comparer les différences entre les deux signaux et de visualiser les effets du filtrage sur celui-ci. La commande "title" permet d'ajouter des titres aux graphiques pour une meilleure identification. Ces commandes sont utiles pour visualiser les effets d'un filtrage sur un signal, vérifier l'efficacité d'un filtre ou encore comparer les différences entre les signaux d'origine et filtrés.

Suppression des interférences des lignes électriques 50Hz

```
% Calculer la TFD du signal ECG
```

```
% Régler la fréquence de bruit du secteur 50Hz à zéro
```

```
% Effectuer une TFDI pour restituer le signal filtré
```

```
% Tracer les deux signaux côte à côte pour faciliter la comparaison
```

La commande "ecg_tfd = fft(ecg_signal)" permet de calculer la transformation de Fourier rapide (FFT) du signal "ecg_signal".

La commande "notch_freq = 50" définit la fréquence de bruit du secteur à 50 Hz. La commande "notch_index = round(notch_freq*length(ecg_signal)/fs)" calcule l'indice de la fréquence de bruit dans le vecteur FFT en utilisant la fréquence d'échantillonnage "fs" et la longueur du signal "ecg_signal". Ensuite, la commande "ecg_tfd(notch_index-5:notch_index+5) = 0" régle à zéro les valeurs du vecteur FFT autour de l'indice de la fréquence de bruit pour éliminer le bruit de secteur de la bande passante.

La commande "ecg_filtered = ifft(ecg_tfd)" effectue une transformation de Fourier inverse (IFFT) sur le vecteur FFT modifié pour restituer le signal filtré.

7-

```
% Essayer différentes fréquences de coupure
 % Tracer le signal filtré pour chaque fréquence de coupure
```

La commande "cutoff_freq_list = [5, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100]" crée une liste de fréquences de coupure pour lesquelles le signal sera filtré.

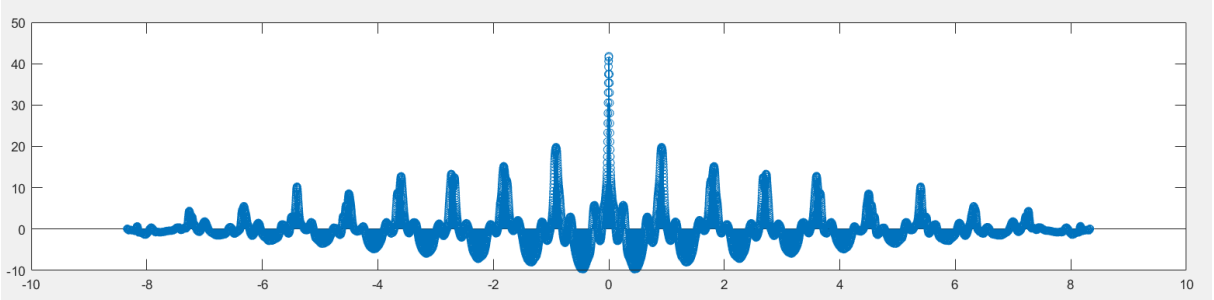
La boucle "for" parcourt la liste de fréquences de coupure, pour chaque itération, la commande "cutoff_freq = cutoff_freq_list(i)" récupère la fréquence de coupure en cours. La commande "cutoff_index = round(cutoff_freq*length(ecg_signal)/fs)" calcule l'indice de la fréquence

de coupure dans le vecteur FFT en utilisant la fréquence d'échantillonnage "fs" et la longueur du signal "ecg_signal". Ensuite, la commande "ecg_tfd(1:cutoff_index) = 0" régle à zéro les valeurs du vecteur FFT en dessous de l'indice de la fréquence de coupure pour éliminer les fréquences plus basses que la fréquence de coupure.

La commande "ecg_filtered = ifft(ecg_tfd)" effectue une transformation de Fourier inverse (IFFT) sur le vecteur FFT modifié pour restituer le signal filtré.

Enfin, les commandes "figure", "plot" permettent de tracer le signal filtré pour chaque fréquence de coupure. Le titre du graphique est généré par la commande "sprintf" qui permet de créer une chaîne de caractères formatée qui contient la fréquence de coupure en cours.

9-



TP4

Filtrage et diagramme de Bode

1-

```
Te = 5*1e-4;   % Pas d'échantillonnage
f1 = 500;  % frequence 1
f2 = 400;   %frequence 2
f3 = 50;    %frequence 3
t = 0:Te:5-Te;   Vecteur de temps
fe = 1/Te;
N = length(t);

fshift = (-N/2:N/2-1)*(fe/N);
f = (0:N-1)*(fe/N);

x = sin(2*pi*f1*t)+sin(2*f2*pi*t)+sin(2*pi*f3*t);
y = fft(x);
```

2-Traçons le signal x(t) et sa transformé de Fourrier

```
subplot(2,1,1)
plot(t,x)

subplot(2,1,2)
plot(fshift, fftshift(abs(y)));
```

H(f) = (K.j.w/wc) / (1 + j. w/wc)

1-Tracer le module de la fonction H(f)

```
h = (k*1j*((2*pi*f)/wc))./(1+1j*((2*pi*f)/wc));

semilogx(abs(h))

plot(abs(h))

legend("Module de h(t)")
```

2-Tracer 20.log(|H(f)|) pour différentes pulsations de coupure wc

```
h = (k*1j*((2*pi*f)/wc))./(1+1j*((2*pi*f)/wc));

G = 20*log(abs(h));
```

3-

```
h = (k*1j*((2*pi*f)/wc))./(1+1j*((2*pi*f)/wc));

h1 = (k*1j*((2*pi*f)/wc1))./(1+1j*((2*pi*f)/wc1));

h2 = (k*1j*((2*pi*f)/wc2))./(1+1j*((2*pi*f)/wc2));

G = 20*log(abs(h));

G1 = 20*log(abs(h1));

G2 = 20*log(abs(h2));

wc = 50;

wc1 = 500;

wc2 = 1000;

semilogx(f,G,f,G1,f,G2);

title("Diagramme de Bode")

xlabel("rad/s")

ylabel("decibel")

legend("G : wc=50","G1 : wc=500","G2 : wc=1000")
```

4-