JANVIER 2023

RAPPORT DES TPS

Préparé par : Youssef ABERKANE
Groupe : IA

TP 1

INTRODUCTION :

Dans ce rapport, nous allons présenter une analyse fréquentielle d'un signal périodique constitué d'une somme de trois sinusoïdes de fréquences différentes. Nous utiliserons la commande fft pour calculer la TFD du signal d'origine et observer son spectre en amplitude. Nous introduirons ensuite un bruit blanc gaussien dans le signal pour étudier son effet sur le spectre de puissance. Enfin, nous examinerons les données audio collectées à partir de microphones sous-marins pour analyser le chant du rorqual bleu.

OBJECTIF DU TP :

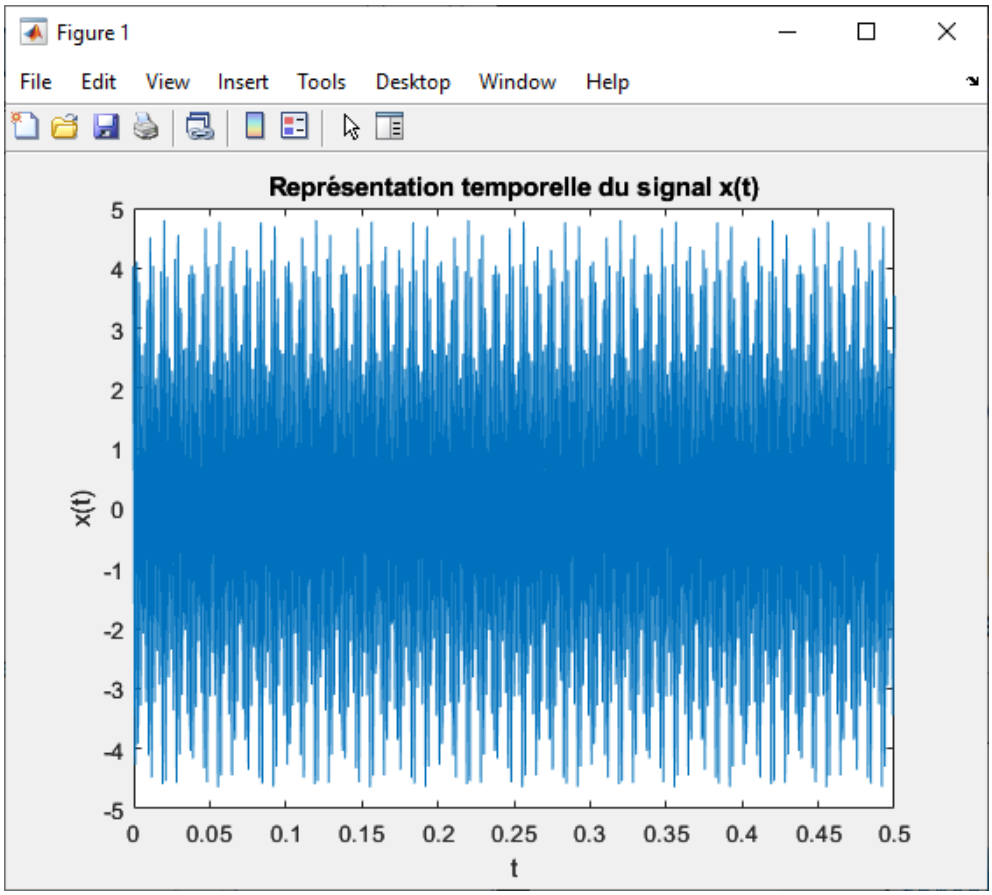
- REPRÉSENTATION DE SIGNAUX ET APPLICATIONS DE LA TRANSFORMÉE DE FOURIER DISCRÈTE (TFD) SOUS MATLAB.
- EVALUATION DE L'INTÉRÊT DU PASSAGE DU DOMAINE TEMPOREL AU DOMAINE FRÉQUENTIEL DANS L'ANALYSE ET L'INTERPRÉTATION DES SIGNAUX PHYSIQUES RÉELS.

REPRÉSENTATION TEMPORELLE ET FRÉQUENTIELLE :

- ## REPRÉSENTATION TEMPORELLE :

Dans la première partie, on essaie dee tracer le signal dans le domaine temporelle. On utilise un signal qui se compose d'une somme de trois sinusoÏdes de fréquences différentes : 440 Hz, 550 Hz et 2500 Hz.

```
fe = 10000; %Fréquence d'échantillonnage
te = 1/fe; %Période dééchantillonnage
N = 5000; %Nombre d'échantillons

t = 0:te:(N-1)*te; %Intervalle
x = 1.2*cos(2*pi*440*t+1.2)+ 3*cos(2*pi*550*t) + 0.6*cos(2*pi*2500*t); %signal périodique x(t)
```

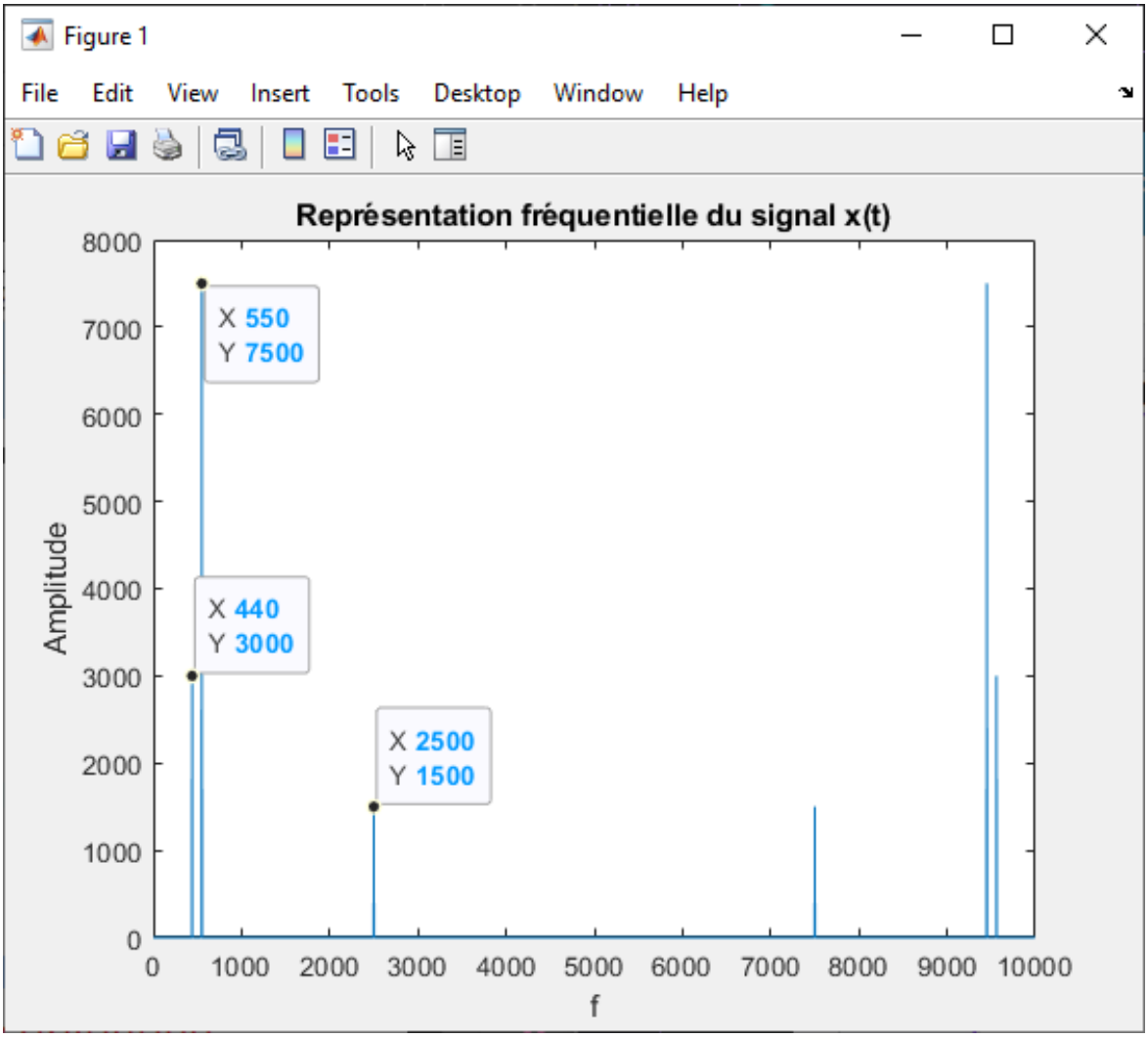


Représentation temporelle du signal

- ## REPRÉSENTATION FRÉQUENTIELLE :

La fonction **fft** genère une fonction complexe (spectre discret). La fonction a la possibilté d'effectuer une transformée de Fourier pour passer du domaine tomporel au domaine fréquentiel ce qui est l'objectif de cette partie.
Dans le graphe afficher ci-dessous, on remaque que les piques sont symétriques par rapport à la fréquence **fe/2** qui est égale à **5000Hz**. C'est **la symétrie conjuguée**

```
f = (0:N-1)*(fe/N);
y = fft(x);

plot(f,abs(y))
```

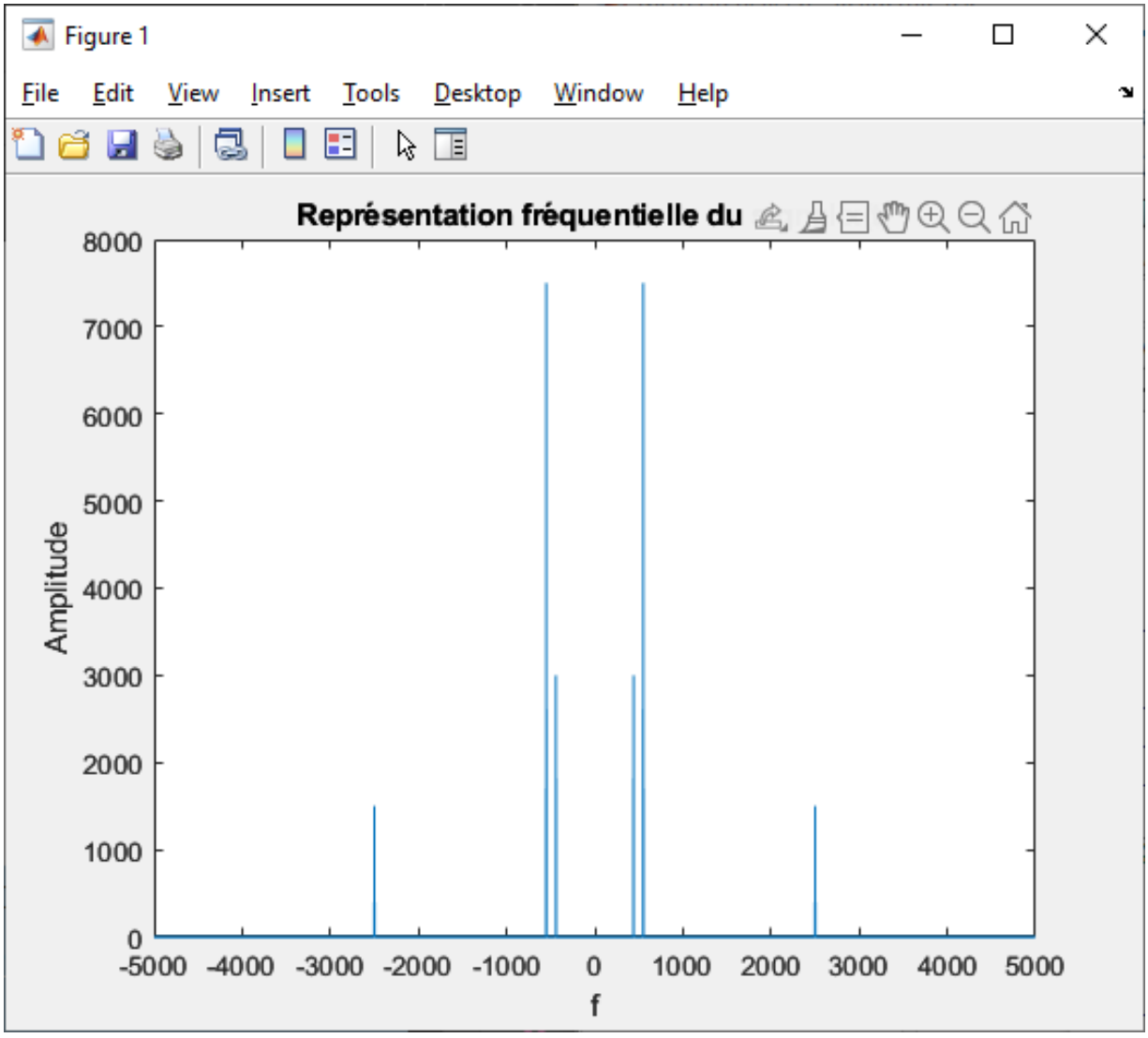


Représentation fréquentielle du signal

- ## REPRÉSENTATION FRÉQUENTIELLE AVEC FFTSHIFT :

La commande **fftshift** dans Matlab permet de décaler la fréquence nulle au centre de l'échantillon de fréquences pour une transformée de Fourier rapide (FFT) ou une transformation inverse de Fourier rapide (IFFT). Cela permet de faciliter l'interprétation des résultats de la FFT pour certains types d'analyses, comme les spectres d'amplitude.

```
fshift = (-N/2:N/2-1)*(fe/N);
y = fft(x);

plot(fshift,fftshift(abs(y)))
```

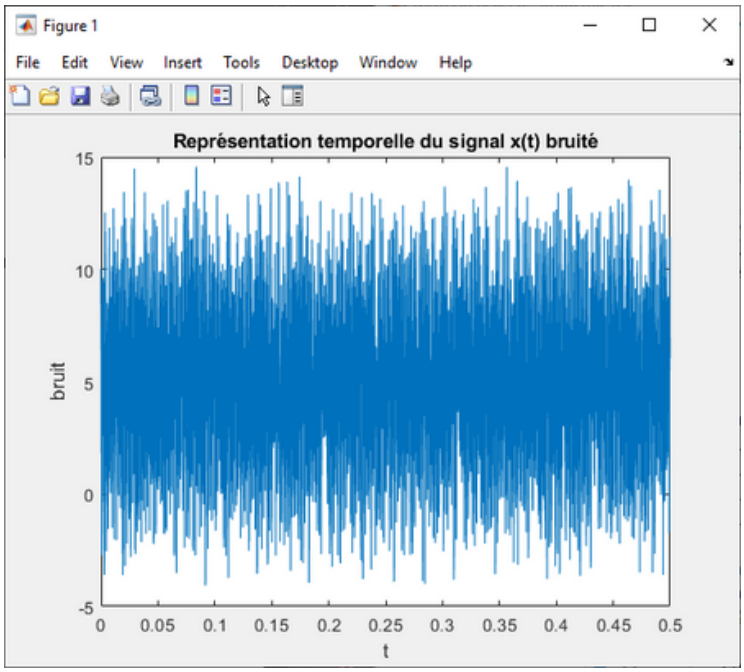


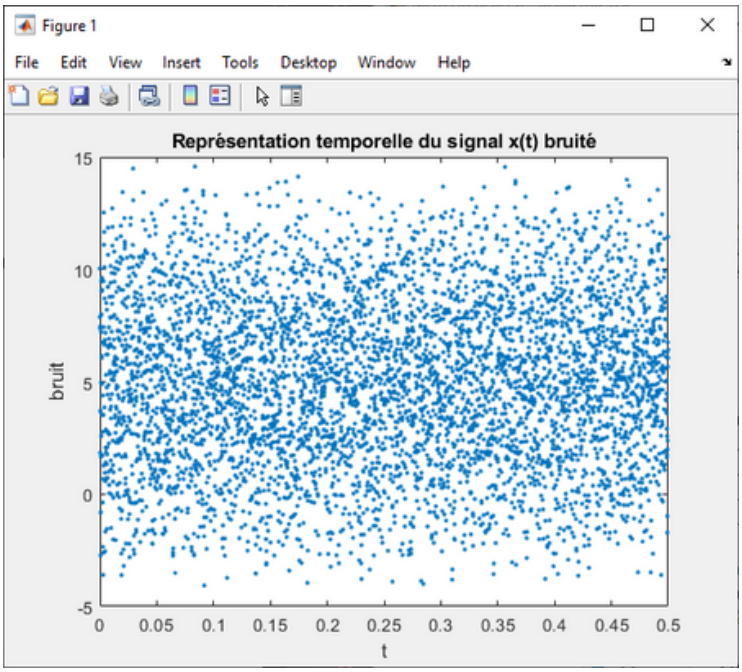
Représentation fréquentielle du signal (fftshift)

- **BRUIT GAUSSIEN :**

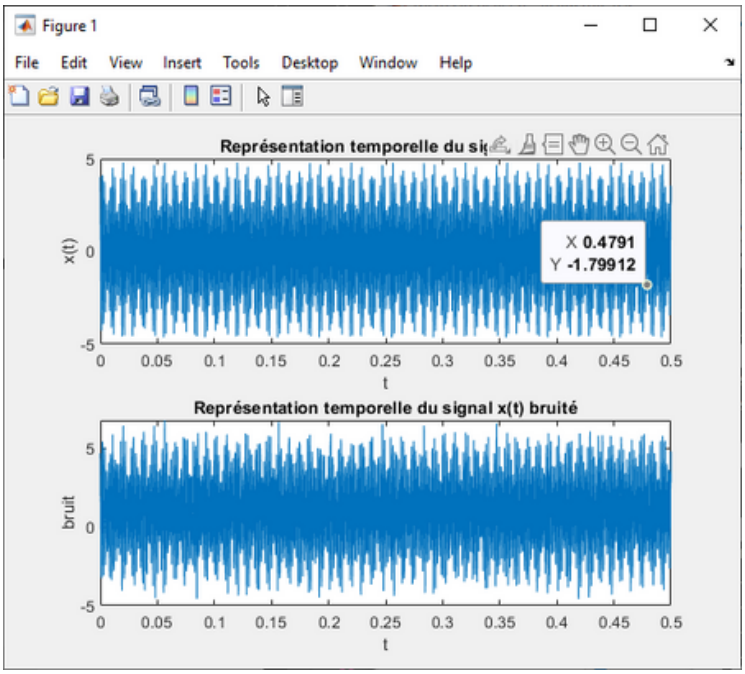
Dans cette partie, on crée un nouveau signal **xnoise** et on l'ajoute à notre signal d'origine pour visualiser la différence. La commande **sound** nous aide a écouté le bruit.

```
xnoise = x+2*rand(size(t));
sound(xnoise)

plot(t,xnoise)
```





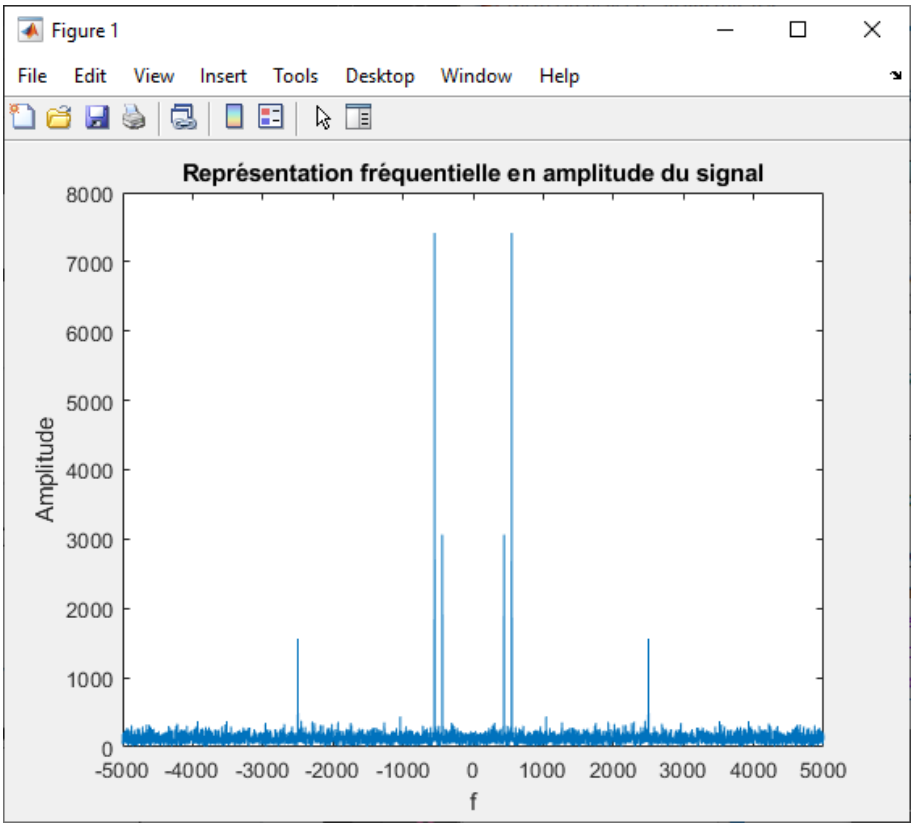
Représentation temporelle du signal bruité



Différence entre les deux signaux

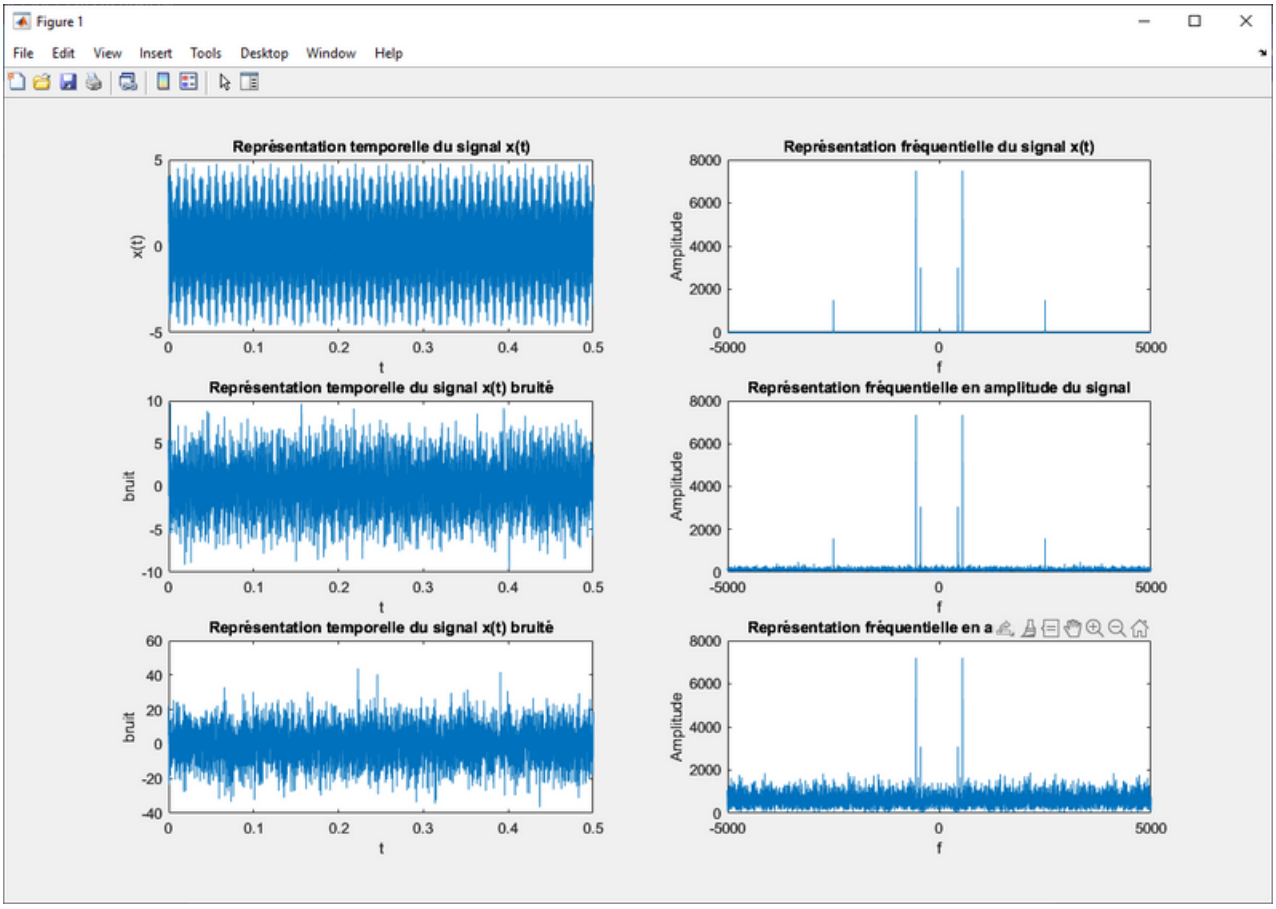
Nous constatons que le bruit présente des propriétés statiques, car la majorité des valeurs sont concentrées autour de 0 et varient entre -5 et 5, ce qui suggère que la moyenne demeurera constante.

```
fshift = (-N/2:N/2-1)*(fe/N);
ynoise = fft(xnoise);
plot(fshift,fftshift(abs(ynoise)));
```



Représentation fréquentielle en amplitude du signal

On remarque l'apparition des nouveaux piques dans le spectre fréquentielle générés par le bruit. L'information n'a pas desparu car onn peut visualiser les piques d'origine.
Dans le cas d'aufmentation de l'intensité du bruits, nous constatons que l'information devient de moins de moins visible, alors il sera impossible de détecter les piques de fréquence du signal initial.

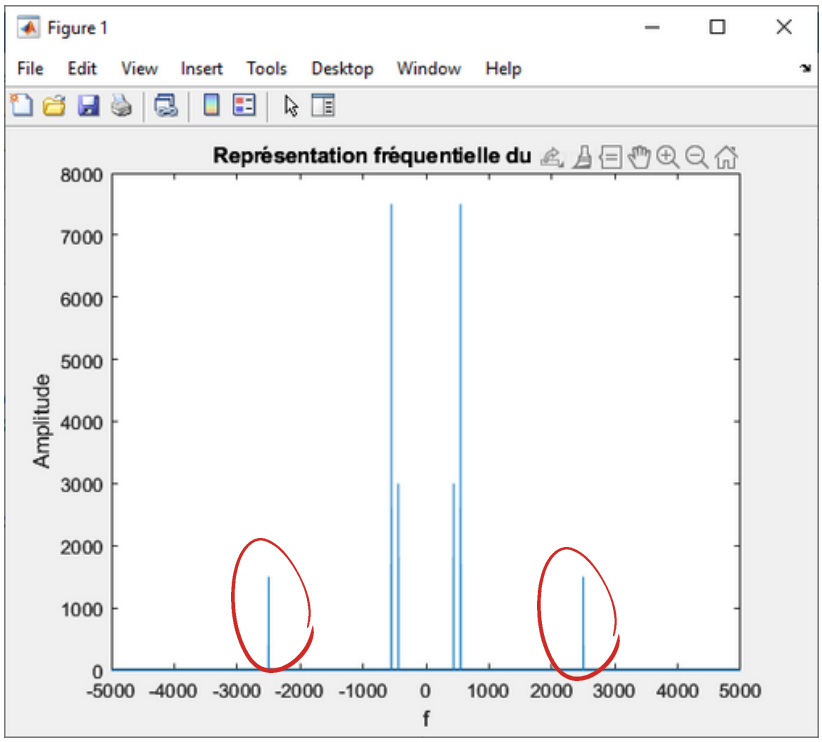


Représentation temporelle et fréquentielle des trois signaux traitées

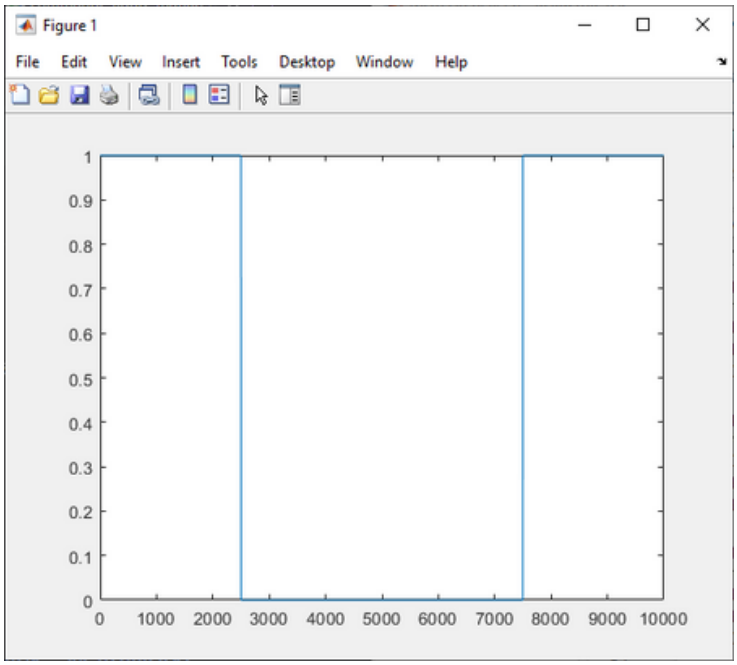
- # FILTRAGE DU SGINAL :

1. CONCEPTION DU FILTRE:

Dans cette étape, on essaiera de filtrer le signal x(t) des hautes fréquences supérieure à 2500Hz, donc on utilise un filtre passe-bas.

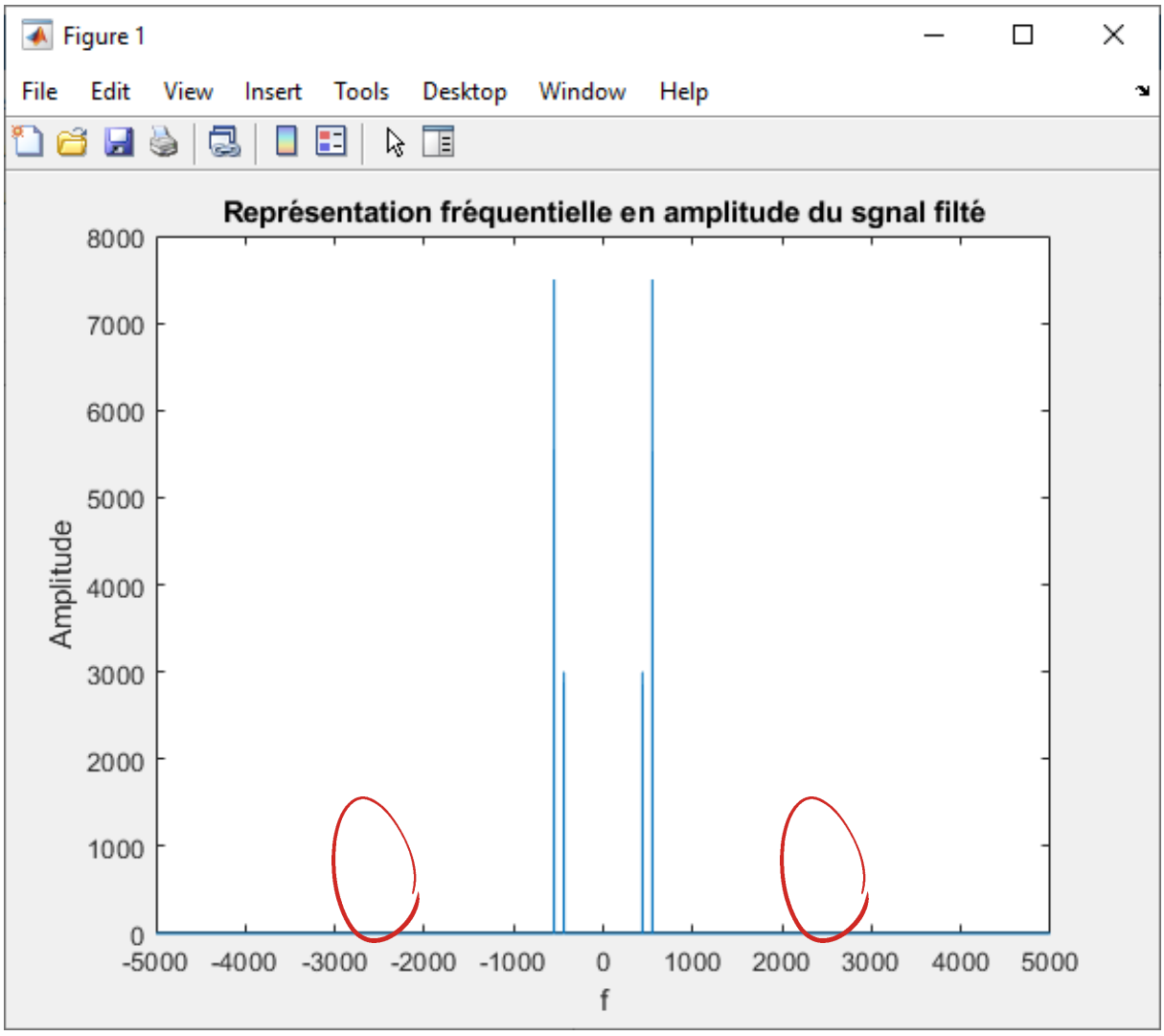



```
fc = 2500; %fréquence de coupure
pass_bas = zeros(size(x));
index_fc = ceil((fc*N)/fe);
pass_bas(1:index_fc) = 1;
pass_bas(N-index_fc+1:N) = 1;
xlabel('f')
ylabel('Amplitude')
title('Filtre pass-bas')

plot(f,pass_bas)
```



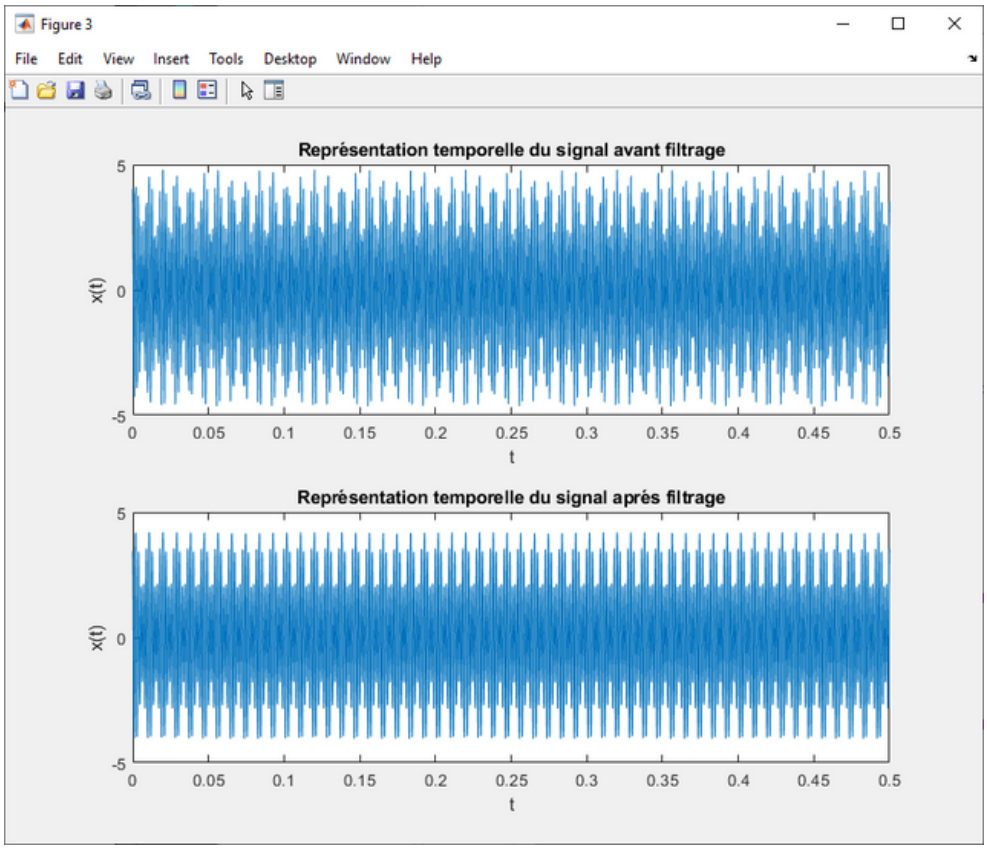
2. APPLICATION DU FILTRAGE:

```
z = pass_bas.*y;%filtre en fct de fréquence
sign_filtr = ifft(z,"symmetric");

plot(fshift,fftshift(abs(fft(sign_filtr))))
```



Aprés l'application du filtrage, on remarque que les piques de fréquence 2500 Hz on disparu.

3. COMPARAISON DU SIGNAL AVANT ET APRÉS LE FILTRAGE:



On peut remarquer la différence entre les deux signaux; le signal aprés filtrage est plus clair grâce au filtrage appliqué qui a supprimé le bruit causé par les hautes fréquences.

- ## CONCLUSION :

On peut conclure de ce TP que le spectre fréquentiel joue un rôle tres important dans la détection de bruit. Le passage du domaine temporel au domaine fréquentiel nous a permis de visualiser l'information essentiel plus les fréquences du bruit qu'on éleminer grâce à un filtre. Aprés l'application du filtre on peut bien voir la différence entre un signal bruité et un signal non-bruité.

TP 2

INTRODUCTION :

Dans ce TP, nous allons travailler sur une phrase enregistrée et une gamme musicale. Pour la phrase, nous allons charger le fichier audio, tracer son signal en fonction du temps, le jouer avec différentes fréquences d'échantillonnage et le segmenter en mots. Pour la gamme musicale, nous allons la synthétiser à l'aide d'un programme en utilisant des signaux sinusoïdaux. Puis, nous allons analyser son spectre à l'aide de l'outil signalAnalyzer et en utilisant la transformée de Fourier. L'objectif est de visualiser et comprendre les fréquences contenues dans ces signaux.

OBJECTIF DU TP :

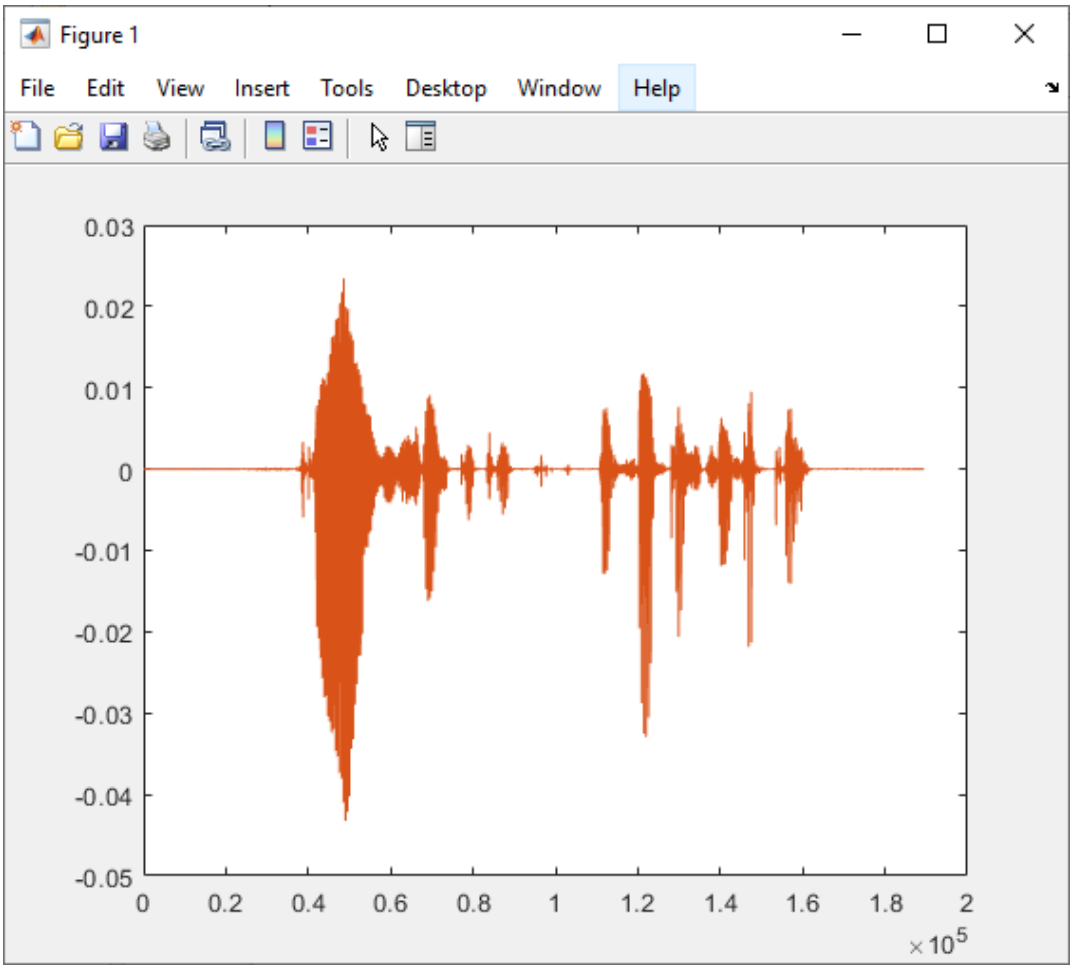
- COMPRENDRE COMMENT MANIPULER UN SIGNAL AUDIO AVEC MATLAB, EN EFFECTUANT CERTAINES OPÉRATIONS CLASSIUES SUR UN FICHIER AUDIO D'UNE PHRASE ENREGISTRÉE VIA UN SMARTPHONE.
- COMPRENDRE LA NOTION DES SONS PURS À TRAVERS LA SYNTHÈSE ET L'ANALYSE SPECTRALE D'UNE GAMME DE MUSIQUE.

On enregistre la phrase "**Rien ne sert de courir, il faut partir à point** " et on l'importe sur Matlab à l'aide du commande **audioread**.

```
x = 'phrase.wav';
[data,Fs] = audioread(x);
```

On trace le signal et on peut l'entendre en utilisant la commande **sound**.

```
plot(data)
sound(data,Fs)    %signal normal
```

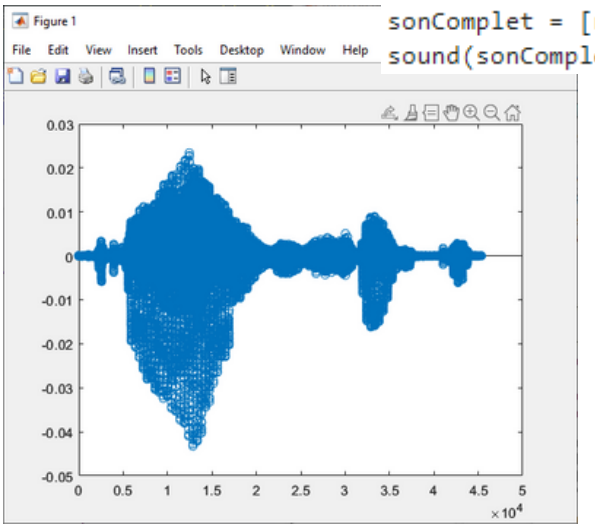


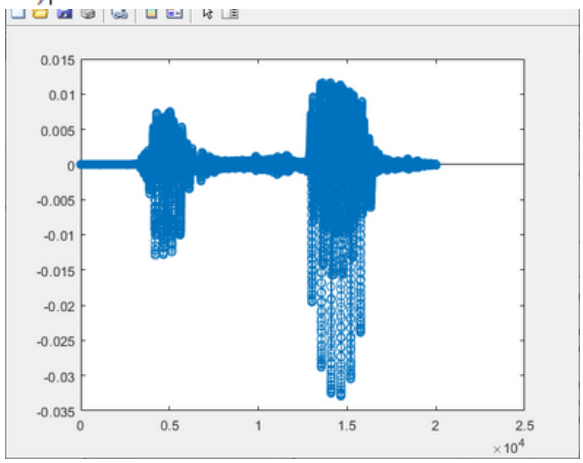
On peut changer la fréquence d'échantillage pour ralentir ou accélérer le signal.

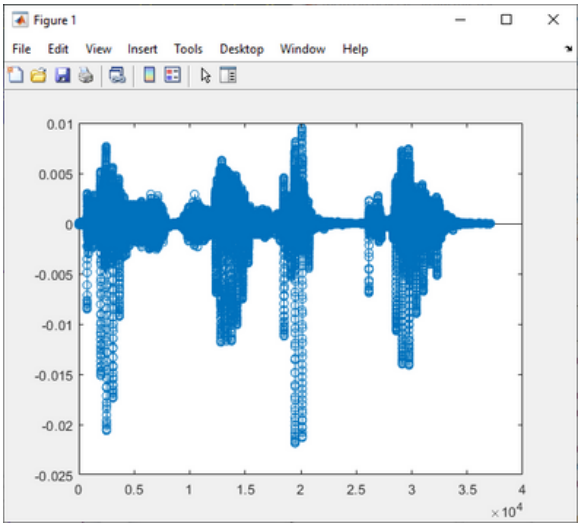
```
sound(data,Fs/2) %pour ralentir
sound(data,Fs*3) %por accélérer
```

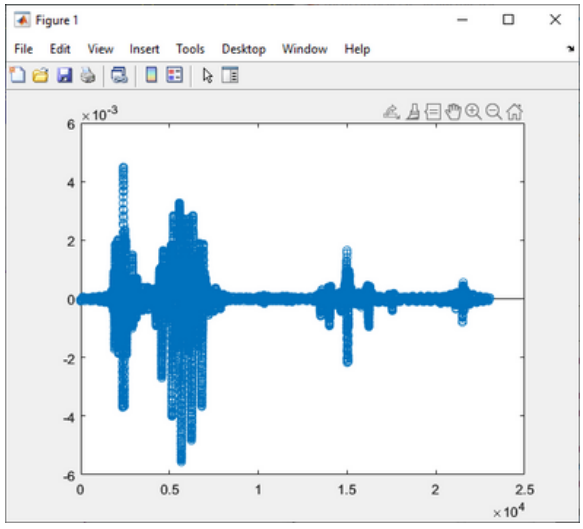
```
riennesertde = data(36175:81567);
stem(riennesertde)
sound(riennesertde,Fs)

courir = data(81567:104608);
stem(courir)
sound(courir,Fs/2)

ilfaut = data(107373:127405);
stem(ilfaut)
sound(ilfaut,Fs)

partirapoint = data(127405:164516);
stem(partirapoint)
sound(partirapoint,Fs)
```

```
sonComplet = [riennesertde, courir, ilfaut, partirapoint];
sound(sonComplet,Fs)
```







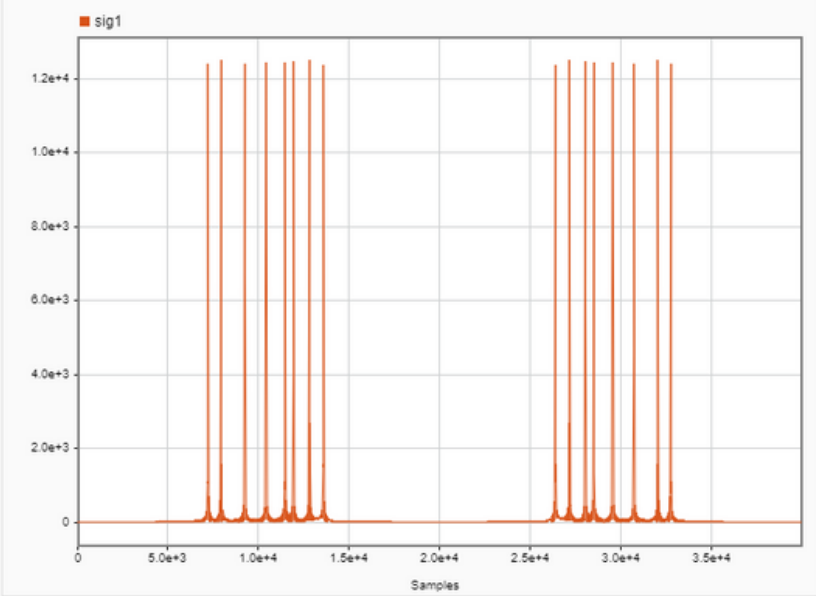


On peut visualiser et entendre chaque partir du signal seule.

```
sonComplet = [riennesertde, courir, ilfaut, partirapoint];
sound(sonComplet,Fs)
```

En concaténant les morceaux, on peur entendre le signal complet.

```
fe = 8192;
te = 5/fe;
N = 5000;
t = (0:N-5)*te;

do1 = 5*cos(2*pi*262*t);
sound(do1,fe)
re = 5*cos(2*pi*294*t);
sound(re,fe)
mi = 5*cos(2*pi*330*t);
sound(mi,fe)
fa = 5*cos(2*pi*349*t);
sound(fa,fe)
sol = 5*cos(2*pi*392*t);
sound(sol,fe)
la = 5*cos(2*pi*440*t);
sound(la,fe)
si = 5*cos(2*pi*494*t);
sound(do1,fe)
do2 = 5*cos(2*pi*523*t);
sound(do2,fe)

music = [do1, re, mi, fa, sol, la, si, do2];
sound(music,fe)
```



- ## CONCLUSION :

En conclusion, ce TP concerne la manipulation des signaux audio en utilisant Matlab. Les objectifs sont de comprendre les opérations courantes sur un fichier audio, ainsi que les concepts de sons purs en synthétisant et en analysant spectralement une gamme musicale.

TP 3

INTRODUCTION :

En gros, ce TP sur le traitement de signaux ECG nous a fourni une compréhension des étapes nécessaires pour nettoyer et analyser les signaux ECG. Nous avons commencé en enlevant le bruit autour du signal produit par l'électrocardiographe en utilisant des méthodes de filtrage. Ensuite, nous avons utilisé des techniques de détection de pics pour déterminer la fréquence cardiaque. Il est important de noter que la conversion du signal du temps continu au temps discret avec Matlab peut entraîner des différences de traitement et il est donc important de prendre en compte ces différences lors de l'analyse. Finalement, ce TP nous a permis de comprendre les principes fondamentaux de l'analyse des signaux ECG et comment utiliser les outils informatiques pour les mettre en pratique.

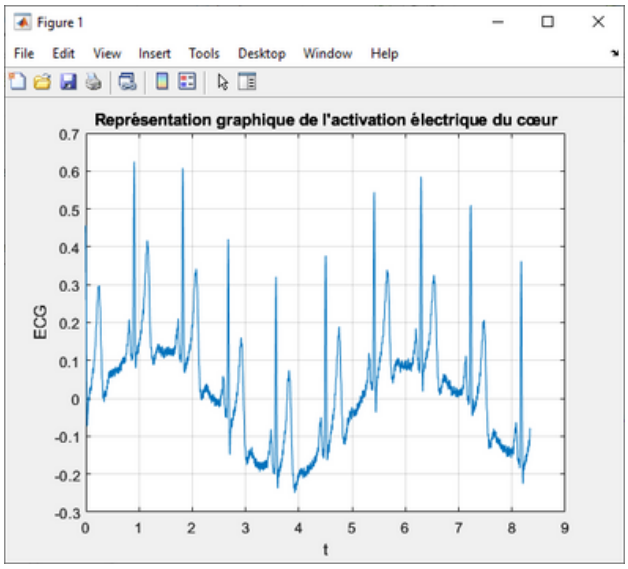
OBJECTIF DU TP :

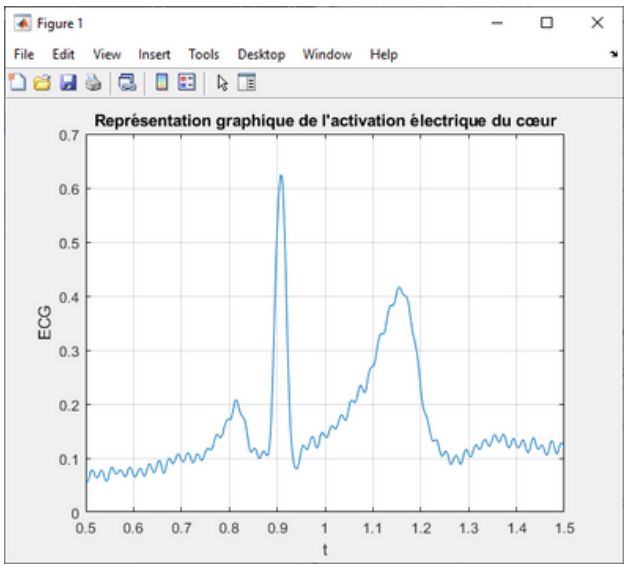
- SUPPRESSION DU BRUIT AUTOUR DU SIGNAL PRODUIT PAR UN ÉLECTROCARDIOGRAPHE.
- RECHERCHE DE LA FRÉQUENCE CARDIAQUE.

On trace le signal en fonction du temps avec une fréquence de 500 Hz, aprés on choisit l'intervalle [0.5 1.5] pour zoomer le signal.

```
load('ecg.mat');
fe = 500;
te = 1/fe;
N = length(ecg);
t = 0:te:(N-1)*te;

plot(t,ecg);
```

```
xlim([0.5 1.5]);
```

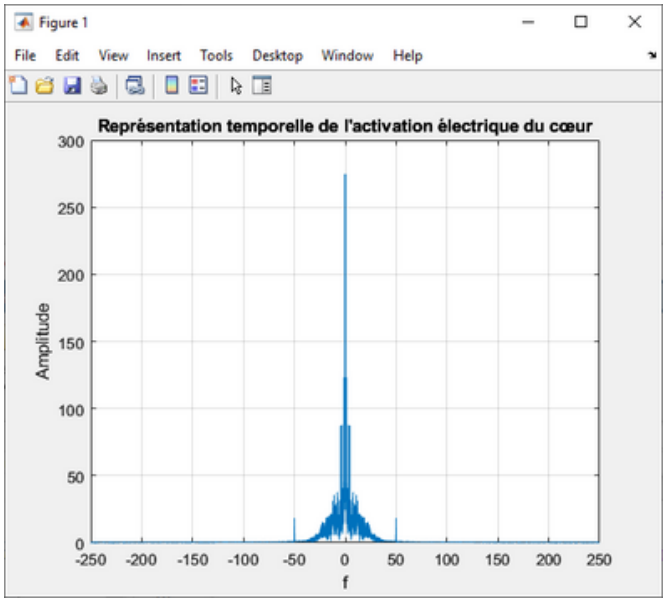




Représentation temporelle de signal avant et aprés le zoom

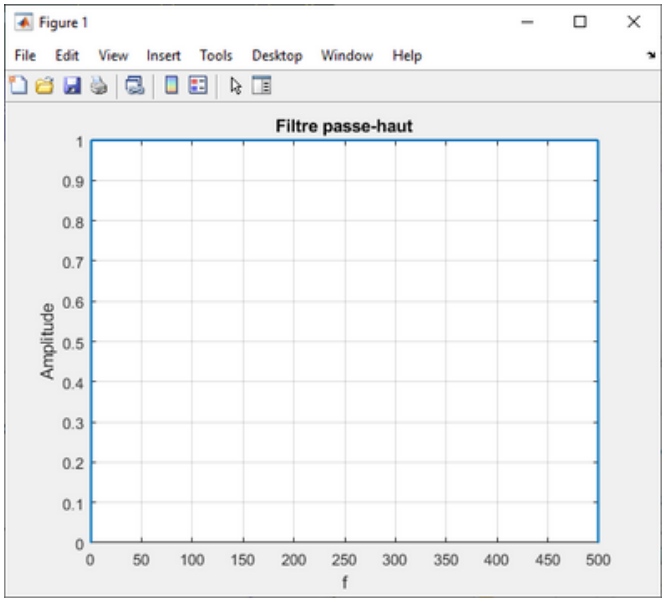
On peut bien remarquer que le signal representé est bruité. Les causes de ce bruit sont plusieurs, le signal peut être affecté par des bruits de l'environnement tels que des mouvements corporels, des mouvements respiratoires ou des bruits ambiants.

```
f = (0:N-1)*(fe/N);
fshift = (-N/2:N/2-1)*fe/N;
y = fft(ecg);
plot(fshift,fftshift(abs(y)))
```



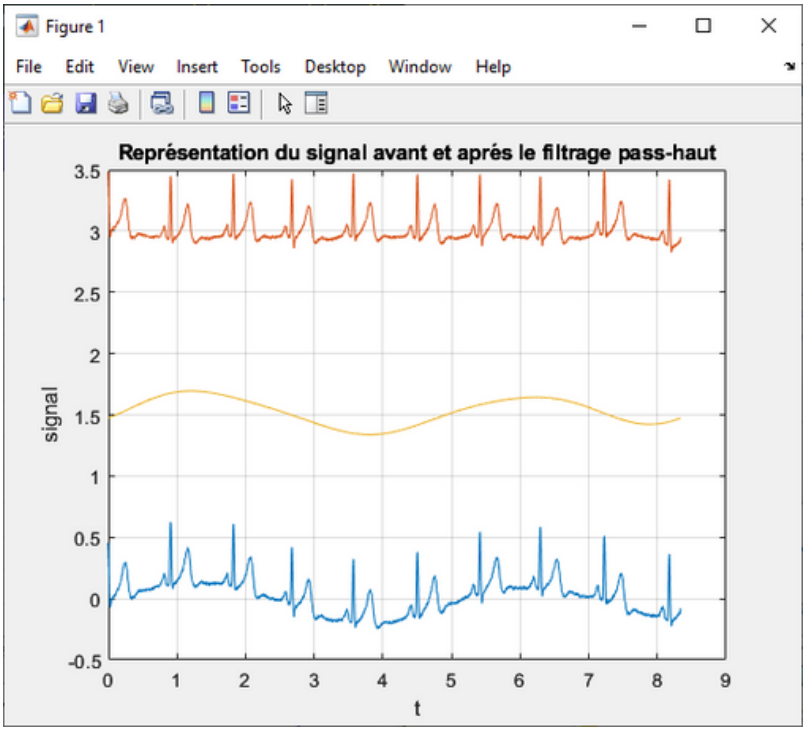
Représentation temporelle de l'activation électrique du cœur

```
filtre_ps_haut = ones(size(ecg));
fc = 0.5;
index_fc = ceil((fc*N)/fe);
filtre_ps_haut(1:index_fc) = 0;
filtre_ps_haut(N-index_fc+1:N) = 0;
plot(f,filtre_ps_haut,"linewidth",1.5)
```



Représentation du filtre passe-haut

Le filtre passe-haut est vecteur de 1 qui est nulle dans la fréquece de coupure **fc = 0.5 Hz** et son symétrique.

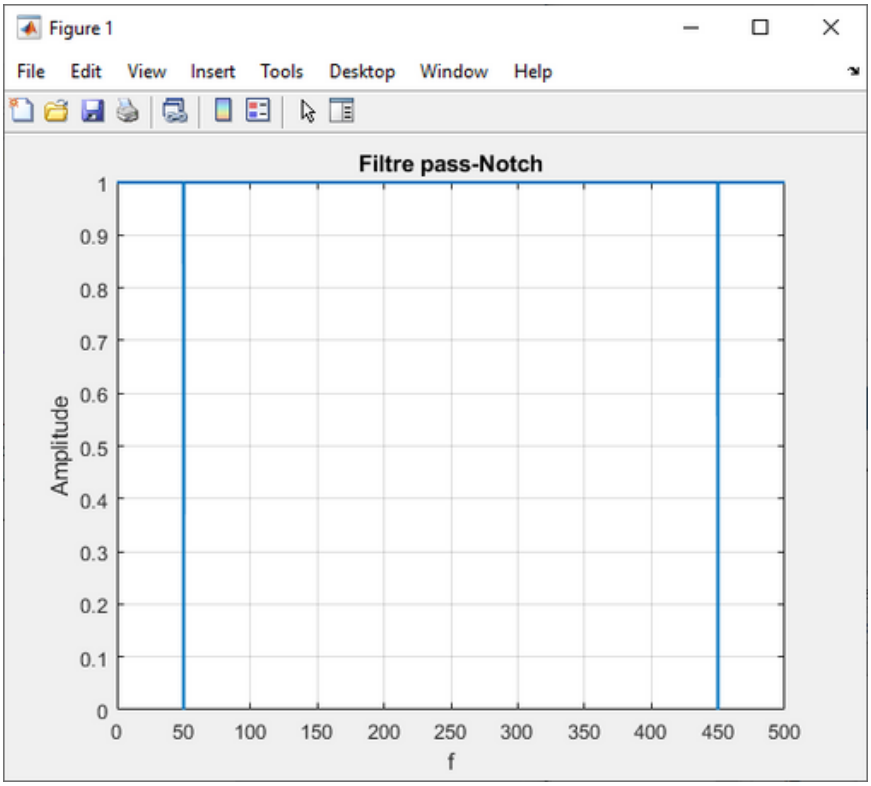


Représentation du signal avant et aprés le filtrage

Le filtrage se fait par la multiplication d'élément par élément du filtre passe-haut avec la transformé de fourié du signal ecg, ensuite en utilise la commande ifft pour pour visualiser le signal aprés filtrage dans le domaine temporel.
On remarue que il n'a plus de décalage au niveau de la hauteur aprés l'application du filtre passe-haut.

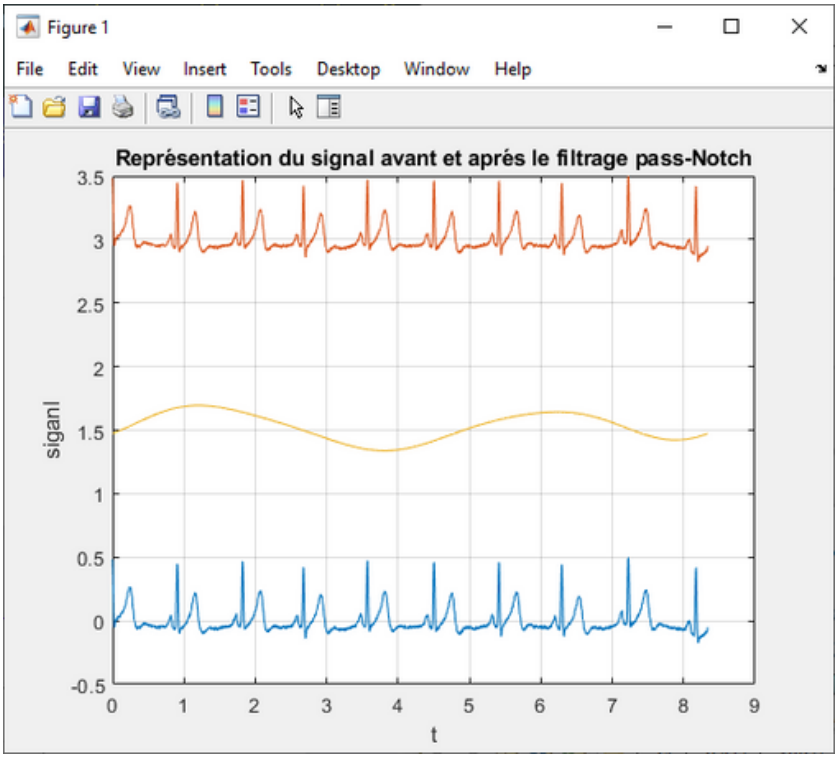
Le filtre pass-Notch est utilisé pour supprimé les fréquences de 50 Hz, c'et un vecteur de 1 qui prend la valeur 0 dans la fréquence 50 Hz et son symétrique 450.

```
filtre_passNotch = ones(size(ecg1));
fc1 = 50;
index_fc1 = ceil((fc1*N)/fe)+1;
filtre_passNotch(index_fc1) = 0;
filtre_passNotch(N-index_fc1+1) = 0;
plot(f,filtre_passNotch,"LineWidth",1.5)
```



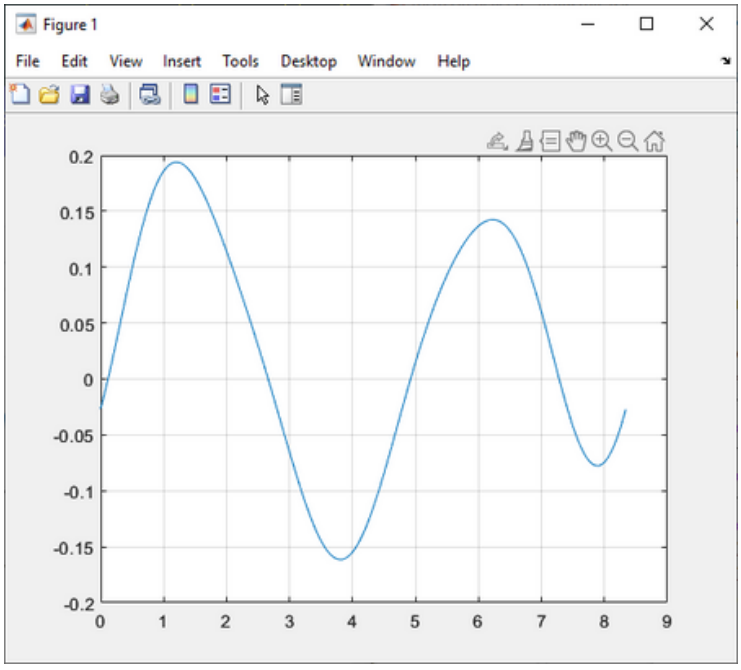
Représentation Du filtre pass-Notch

```
frq_ecg2 = filtre_passNotch.*fft(ecg1);
ecg2 = ifft(frq_ecg2, "symmetric");
plot(t,ecg1);
hold on
plot(t,ecg2+3);
hold on
plot(t,ecg-ecg2+1.5);
```

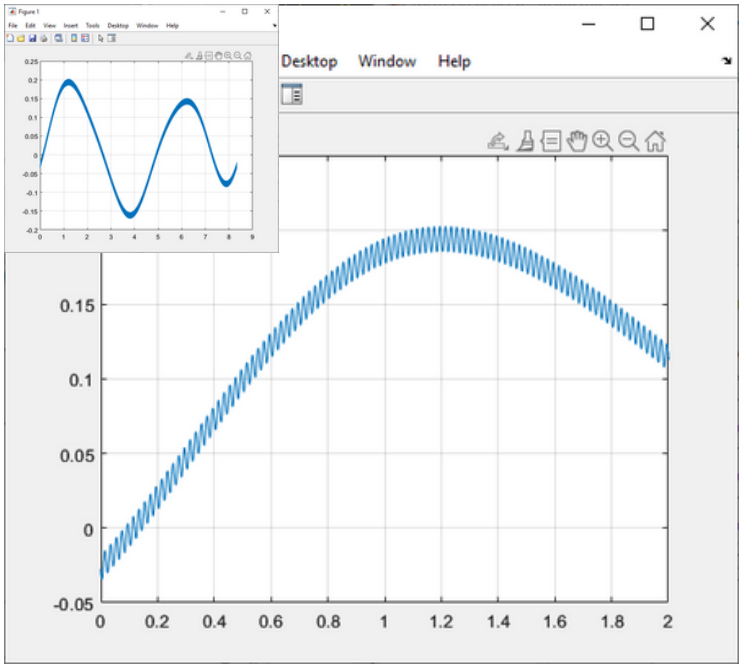


Représentation du signal avant et aprés le filtrage pass-Notch

```
plot(t,ecg-ecg1)
```



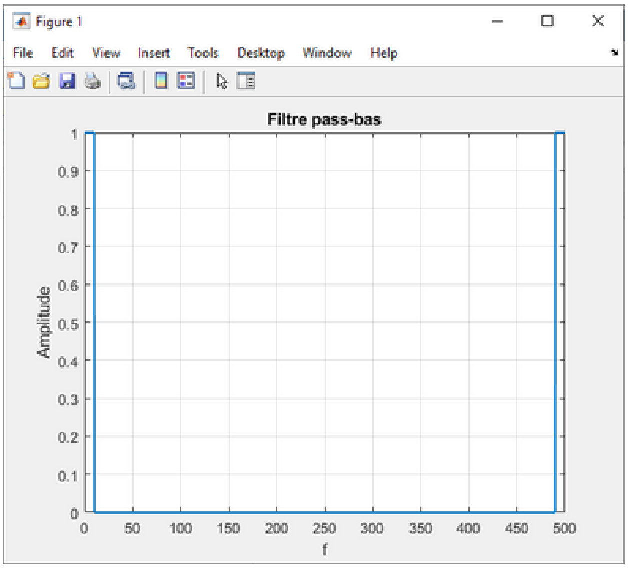

```
plot(t,ecg-ecg2)
```

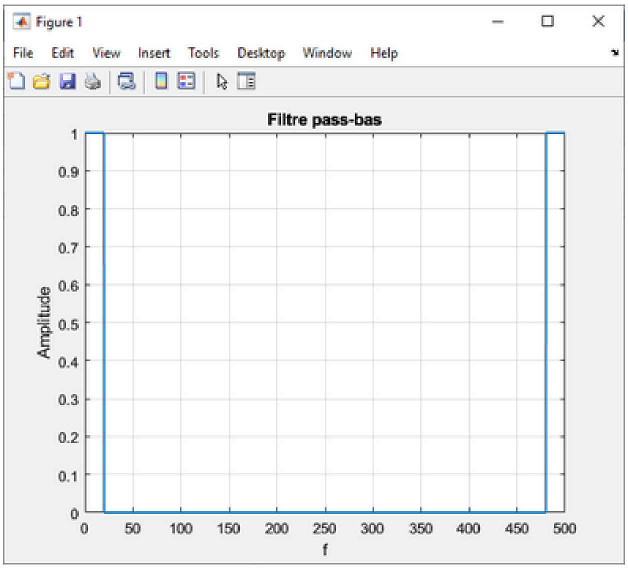


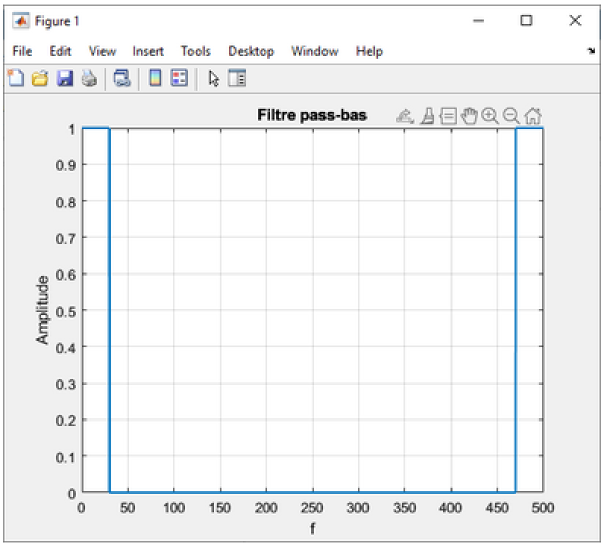
Partie supprimée après application des filtres

à l'aide du filtre pass-notch, on peur remarquer la suppression du signal sinusoïdale de 50 Hz.

Après avoir éliminé le bruit à basse fréquence et les interférences, il est constaté que le signal est toujours incorrect et que l'information n'a pas encore été restaurée. Il est donc nécessaire d'éliminer le bruit à haute fréquence.

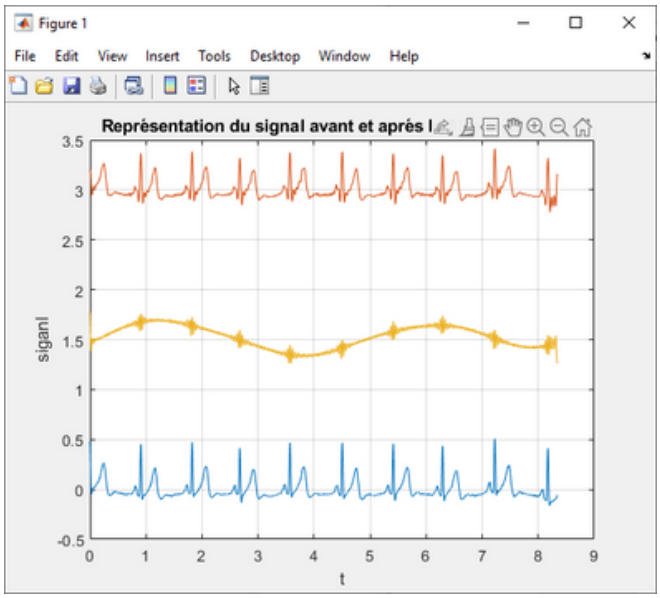


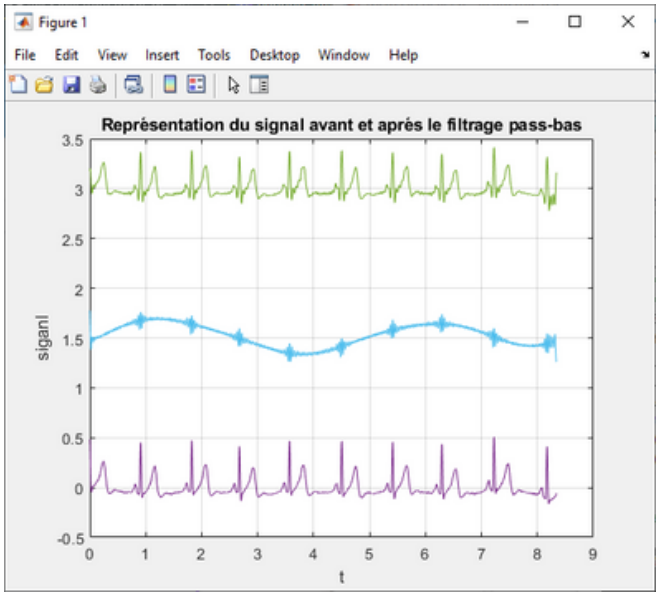


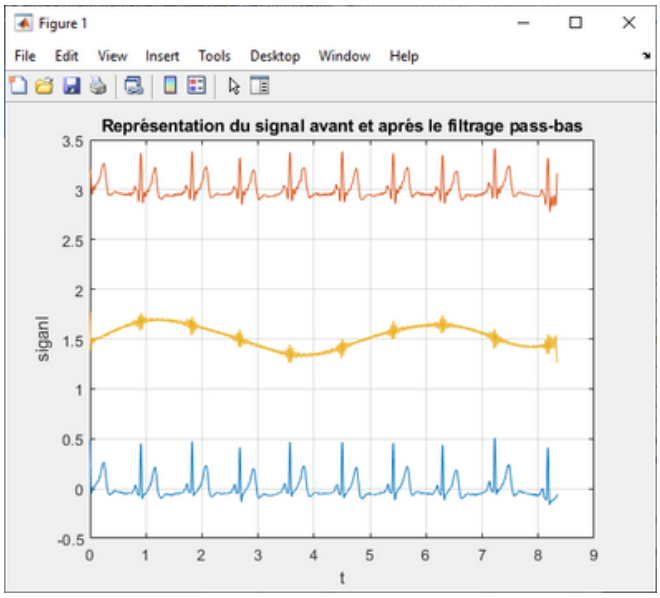


Représentation du filtre passe-bas avec des différences fréquences

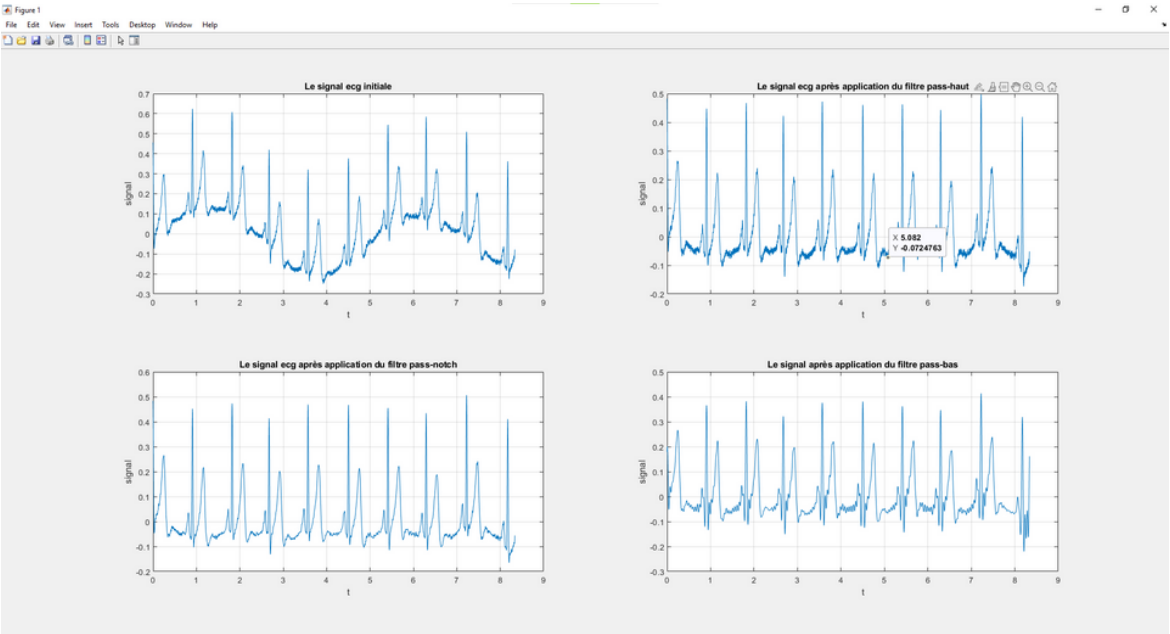
```
frq_ecg3 = filtre_passNotch.*fft(ecg2);
ecg3 = ifft(frq_ecg3, "symmetric");
plot(t,ecg2);
hold on
plot(t,ecg3+3);
hold on
plot(t,ecg-ecg3+1.5);
```







Représentation du signal avant et aprés le filtrage pass-bas



Représentation du signal initial et après filtrage

```
% Charger le signal ECG
load('ecg3.mat');

% Calculer la fonction d'autocorrelation
[acf, lags] = xcorr(ecg3, ecg3);

% Trouver le premier maximum local après le maximum global
[~, maxIndex] = max(acf);
peakIndex = maxIndex;
for i = maxIndex+1:length(acf)
    if acf(i) < acf(i-1)
        peakIndex = i;
        break;
    end
end

% Calculer la fréquence cardiaque en utilisant le temps entre les pics
heartRate = 1 / (lags(peakIndex) / Fs);

% Afficher la fréquence cardiaque trouvée
disp(['Fréquence cardiaque: ' num2str(heartRate) ' BPM']);
```

Le code charge un signal ECG nommé "ecg3.mat". Ensuite, il calcule la fonction d'autocorrélation du signal en utilisant la fonction Matlab "xcorr". Après avoir trouvé le maximum global de la fonction d'autocorrélation, le code trouve le premier maximum local suivant en parcourant la fonction d'autocorrélation à partir du maximum global et en arrêtant lorsqu'une valeur décroissante est trouvée. La fréquence cardiaque est alors calculée en utilisant le temps entre les pics trouvés dans la fonction d'autocorrélation et en divisant par la fréquence d'échantillonnage du signal (Fs). Enfin, le code affiche la fréquence cardiaque trouvée en battements par minute (BPM).

- ## CONCLUSION :

En gros, ce TP sur le traitement de signaux ECG nous a fourni une compréhension des étapes nécessaires pour nettoyer et analyser les signaux ECG. Nous avons commencé en enlevant le bruit autour du signal produit par l'électrocardiographe en utilisant des méthodes de filtrage. Ensuite, nous avons utilisé des techniques de détection de pics pour déterminer la fréquence cardiaque. Il est important de noter que la conversion du signal du temps continu au temps discret avec Matlab peut entraîner des différences de traitement et il est donc important de prendre en compte ces différences lors de l'analyse. Finalement, ce TP nous a permis de comprendre les principes fondamentaux de l'analyse des signaux ECG et comment utiliser les outils informatiques pour les mettre en pratique.

TP 4

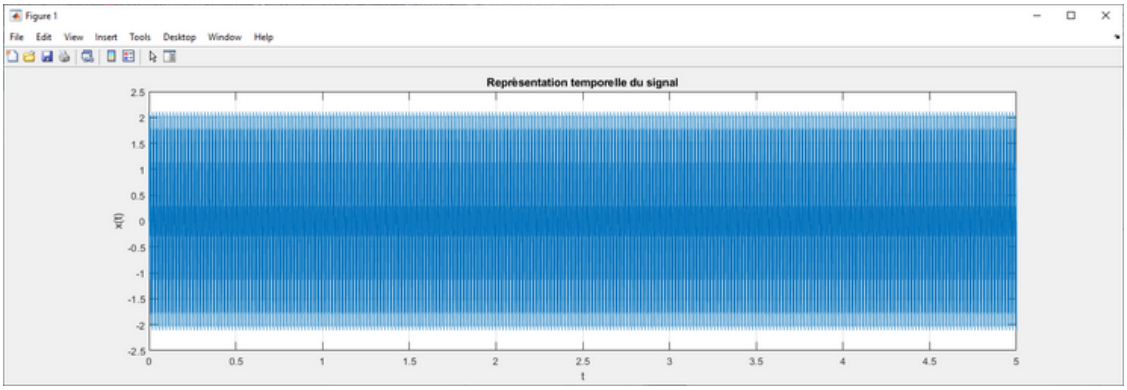
INTRODUCTION :

L'objectif de ce rapport est d'étudier différentes techniques de filtrage et de dé-bruitage sur des signaux électriques et sonores. Nous nous concentrons sur l'utilisation de filtres passe-haut pour supprimer les basses fréquences indésirables dans un signal d'entrée. Nous tracons également le diagramme de Bode pour observer la transmittance complexe du filtre, et choisissons la fréquence de coupure optimale. Nous mettons en œuvre ces techniques pour éliminer un bruit haute fréquence dans une musique enregistrée, et examinons l'influence du paramètre K et de l'augmentation de l'ordre du filtre sur la qualité de filtrage. En fin de rapport, nous présentons des conclusions sur les différentes méthodes utilisées.

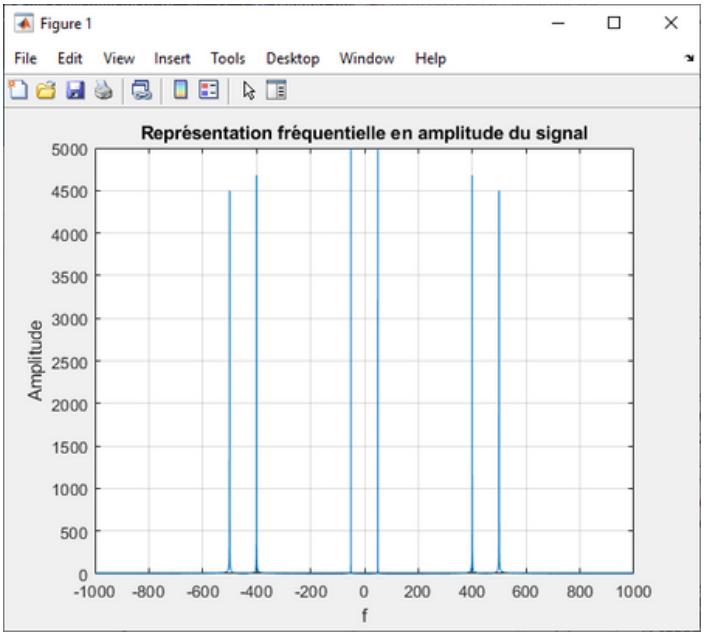
OBJECTIF DU TP :

- APPLIQUER UN FILTRE RÉEL POUR SUPPRIMER LES COMPOSANTES INDÉSIRABLES D'UN SIGNAL.
- AMÉLIORER LA QUALITÉ DE FILTRAGE EN AUGMENTANT L'ORDRE DU FILTRE.

```
Te=0.0005;
fe = 1/Te;
t = 0:Te:5;

f1=500;
f2=400;
f3=50;

N=length(t);
x=sin(2*pi*f1*t)+sin(2*pi*f2*t)+sin(2*pi*f3*t);
plot(t,x);
```

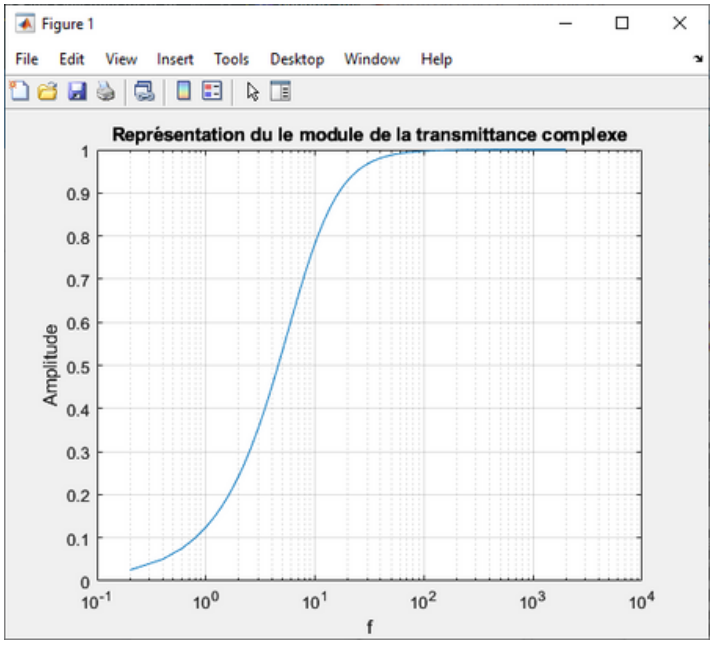



```
fshift=(-N/2:N/2-1)*fe/N;
y=fft(x);
plot(fshift,fftshift(abs(y)));
```

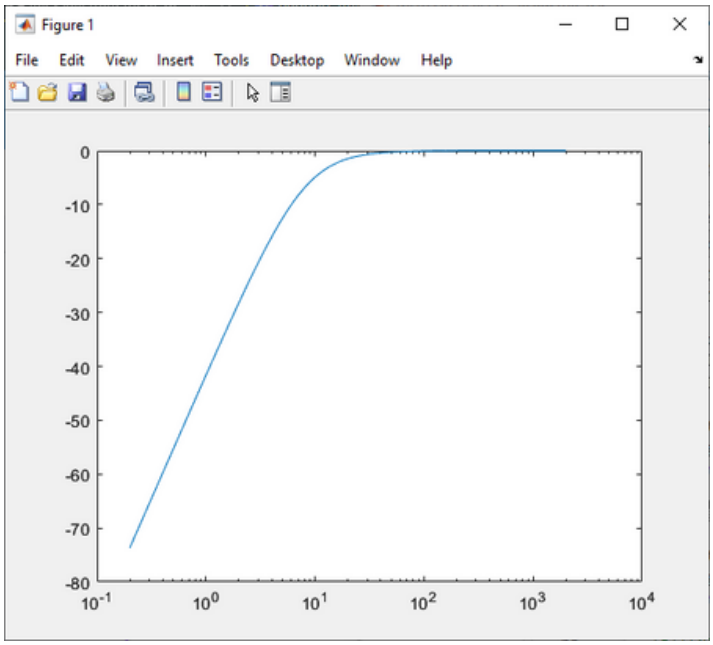


On peut remarquer que les basses fréquences ont une grande amplitude.

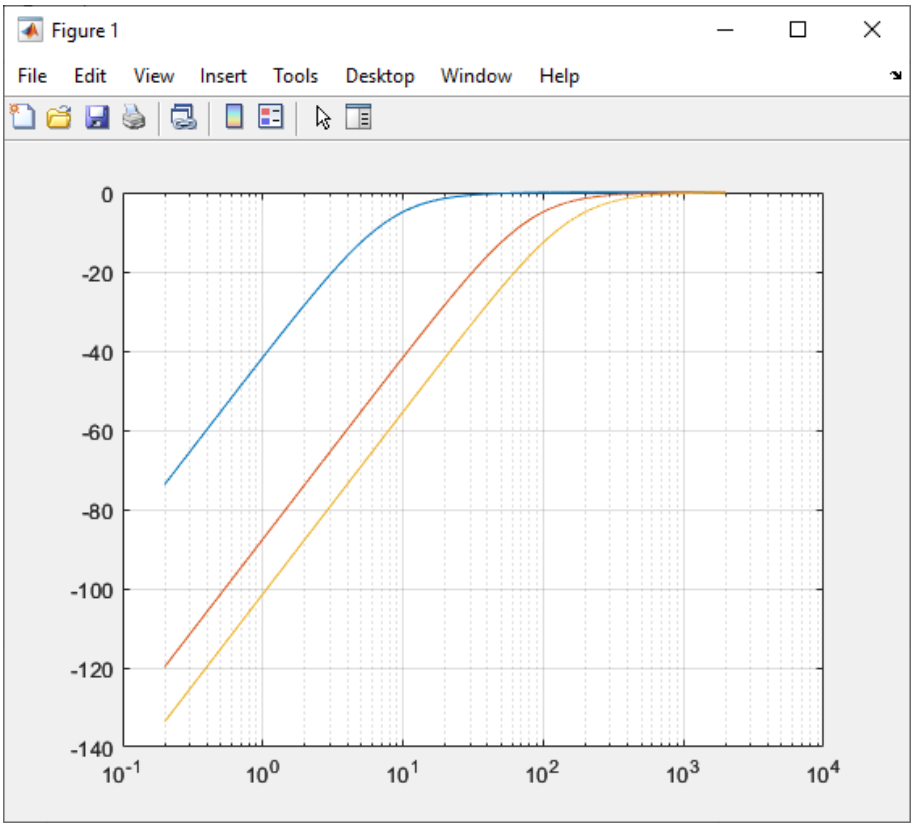
```
f = (0:N-1)*(fe/N);
K=1;
w=2*pi*f;
H=(K*1i*w/wc)./(1+1i*w/wc);
wc=50;

semilogx(f,abs(H));
```




```
G=20*log(abs(H));
semilogx(f,G)
```



```
wc1=500;
wc2=1000;

H1=(K*1i*w/wc1)./(1+1i*w/wc1);
H2=(K*1i*w/wc2)./(1+1i*w/wc2);

G=20*log(abs(H));
G1=20*log(abs(H1));
G2=20*log(abs(H2));

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



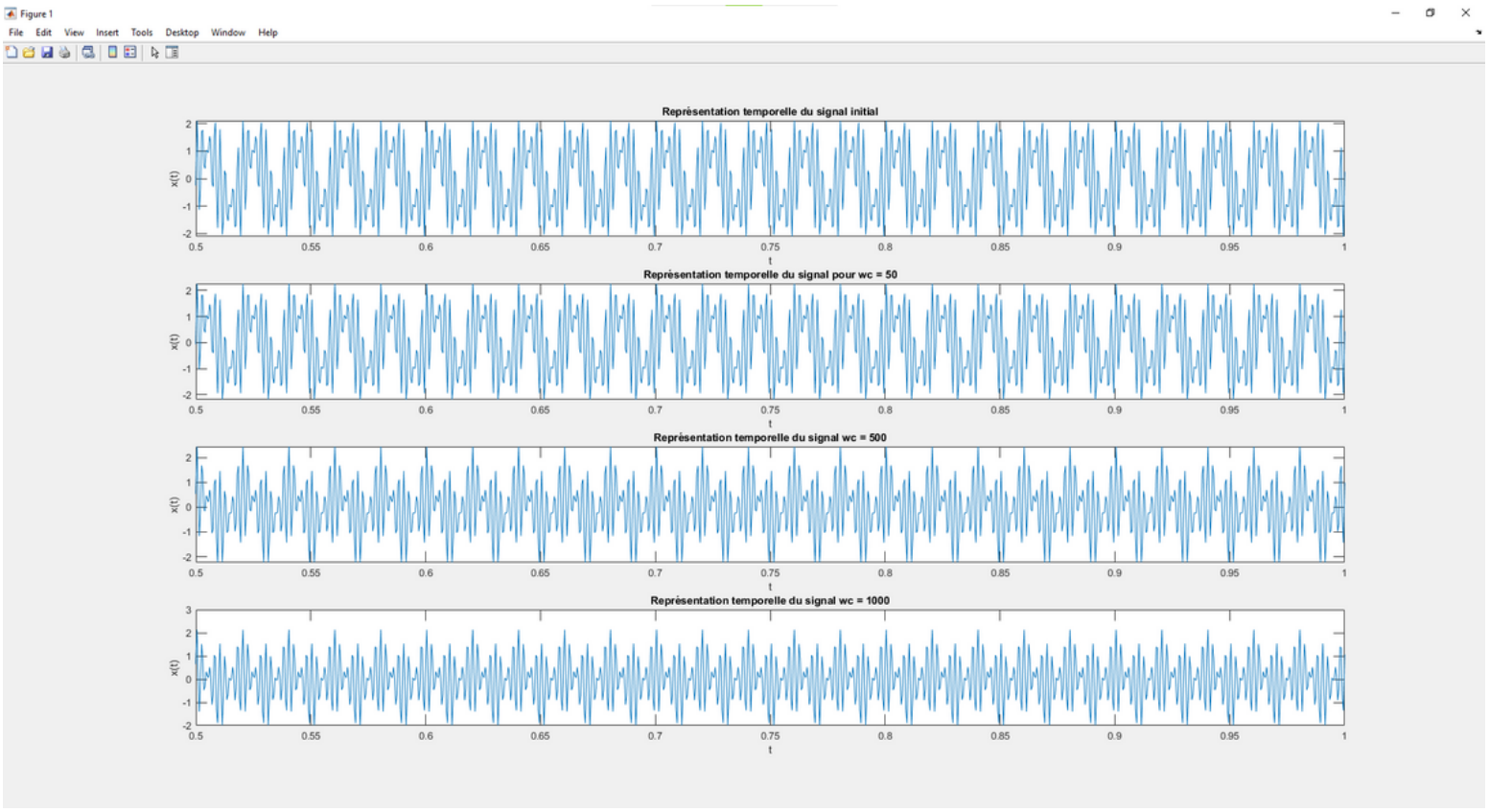
En dessinant le graphe de 20.log(|H(f)|) pour différentes fréquences de coupure telles que 50 Hz, 500 Hz, 1000 Hz, on peut constater que la pente de la courbe est de -20 dB par décade pour les fréquences supérieures à la fréquence de coupure. La fréquence de coupure (wc) détermine le point de départ de la pente de la courbe de -20 dB par décade.

```
filtre1 = H.*y;
filtre2 = H1.*y;
filtre3 = H2.*y;
```

Filtrage du signal fréquentiel

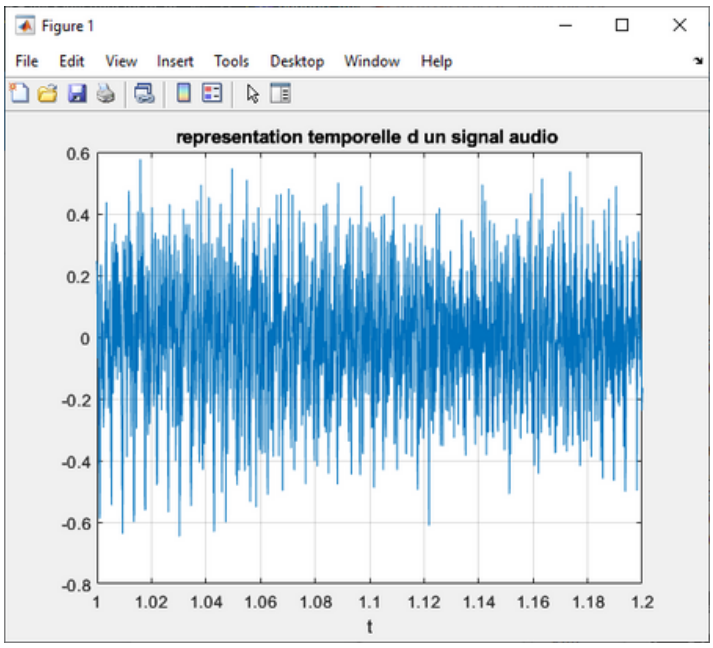
```
x1=ifft(filtre1,"symmetric");
x2=ifft(filtre2,"symmetric");
x3=ifft(filtre3,"symmetric");
```

Restitution du signal temporel filtré



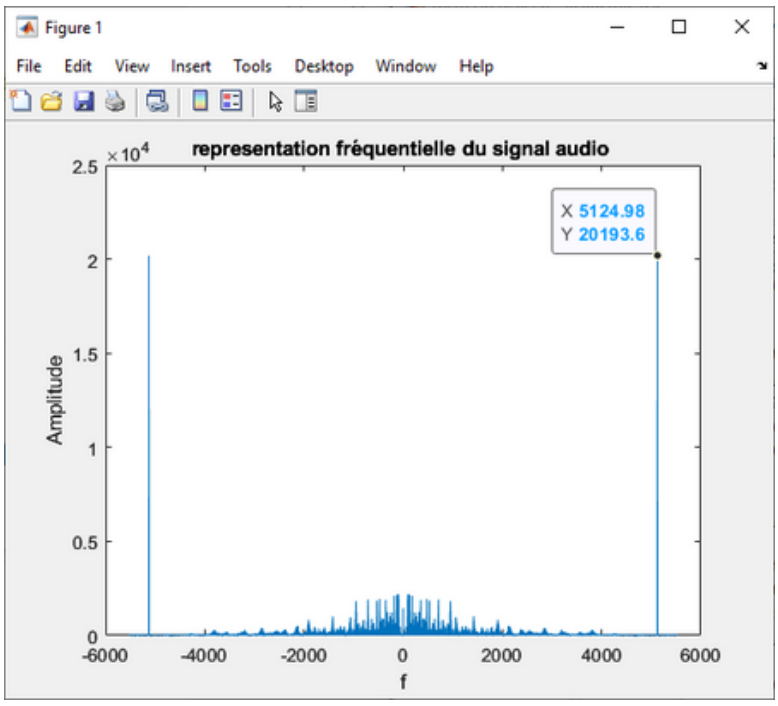
En utilisant différentes fréquences de coupure pour filtrer les signaux dans le domaine de fréquence, on peut remarquer que plus la fréquence de coupure est élevée, plus les signaux à haute fréquence seront atténués, tandis que si la fréquence de coupure est plus basse, plus les signaux à haute fréquence seront laissés passer.

```
[audio,fs]=audioread("test.wav"); %lecture de l'audio

Ts= 1/fs; % Période d'échantillonage
N=length(audio); % le nombre d'échantillons égal à la taille du vecteur musi
t=0:Ts:(N-1)*Ts;
plot(t,audio)
```



Lors de l'analyse du signal, on constate que la musique est perturbée par un bruit qu'il est nécessaire d'éliminer.

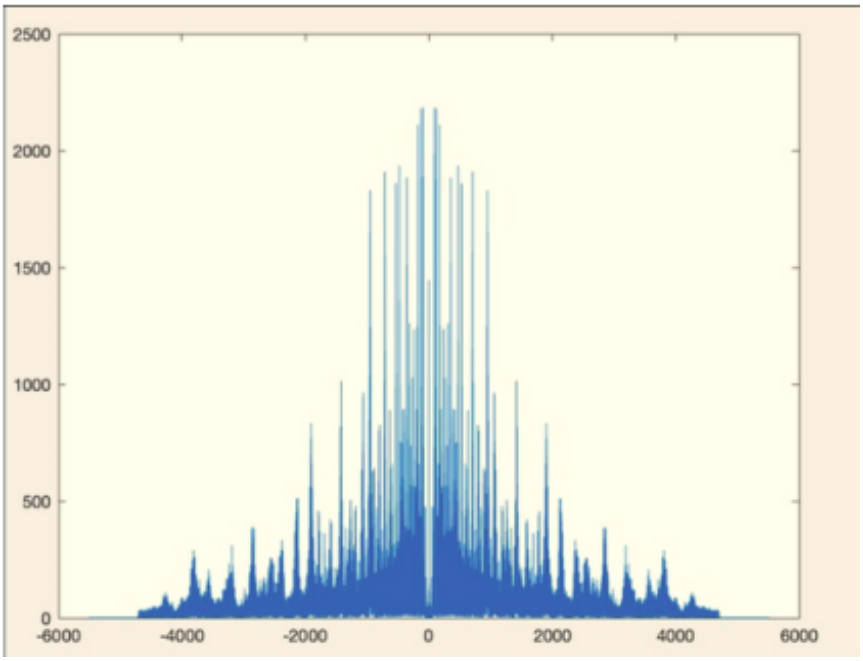
```
fshift = (-N/2:N/2-1)*(fs/N); % le pas de discrétisation : fe/N
transfF=fft(audio); % transformée de fourirer
plot(fshift,fftshift(abs(transfF)));
```



En examinant le spectre de fréquence, on constate la présence d'une fréquence particulière à 5124 Hz qui doit être enlevée.

```
fc=4700; %fréquence de coupre, 4700 car l'atténuation est de 0.7
f=(0:N-1)*(fs/N);
k=1;

%Transmittance complexe
H = k./(1+1i*(f/fc).^1000);

%Conception du filtre
H_filter = [H(1:floor(N/2)), flip(H(1:floor(N/2)))];

%Filtrage
f = transfF(1:end-1).*H_filter;

%Signal filtré
filtered_sign=ifft(f,"symmetric");
plot(fshift(1:end-1), fftshift(abs(fft(filtered_sign))));
```

Nous ne choisissons pas la fréquence de coupure comme étant 5124 Hz, car nous utilisons un filtre analogique qui comporte une réduction du gain et de la phase à un certain niveau, contrairement à un filtre idéal.



On a conçu un filtre passe-haut avec une fréquence de coupure définie à 4700 Hz pour atténuer rapidement les signaux de fréquence inférieure. Le paramètre K, qui représente le gain du signal, a un impact direct sur l'amplitude du signal après le filtrage. Si K est élevé, l'amplitude du signal sortant sera élevée. Après l'application du filtre, on peut constater que la qualité sonore s'est améliorée en éliminant une partie du bruit, sans perdre les informations importantes. On peut améliorer encore plus la qualité du signal en augmentant l'ordre du filtre.

- ## CONCLUSION :

En résumé, ce travail pratique sur le filtrage analogique a permis de mettre en application les connaissances théoriques apprises en cours. Nous avons utilisé un filtre passe-haut pour supprimer une fréquence spécifique dans un signal d'entrée x(t) qui comprenait plusieurs fréquences différentes. Nous avons tracé le diagramme de Bode pour différentes fréquences de coupure pour comprendre comment cela affecte la transmittance complexe H(f). Nous avons choisi une fréquence de coupure optimale et avons appliqué le filtrage pour obtenir le signal désiré y(t). Enfin, nous avons également utilisé nos connaissances pour éliminer les bruits haute fréquence d'un signal sonore en utilisant un filtre passe-bande. En conclusion, ce TP a permis de comprendre l'importance de choisir un filtre approprié pour améliorer la qualité d'un signal et d'appliquer les concepts théoriques étudiés en cours.