RAPPORTDU TP3: TRAITEMENT D’UN SIGNAL ECG

Fait par: Oumayma Bennouna

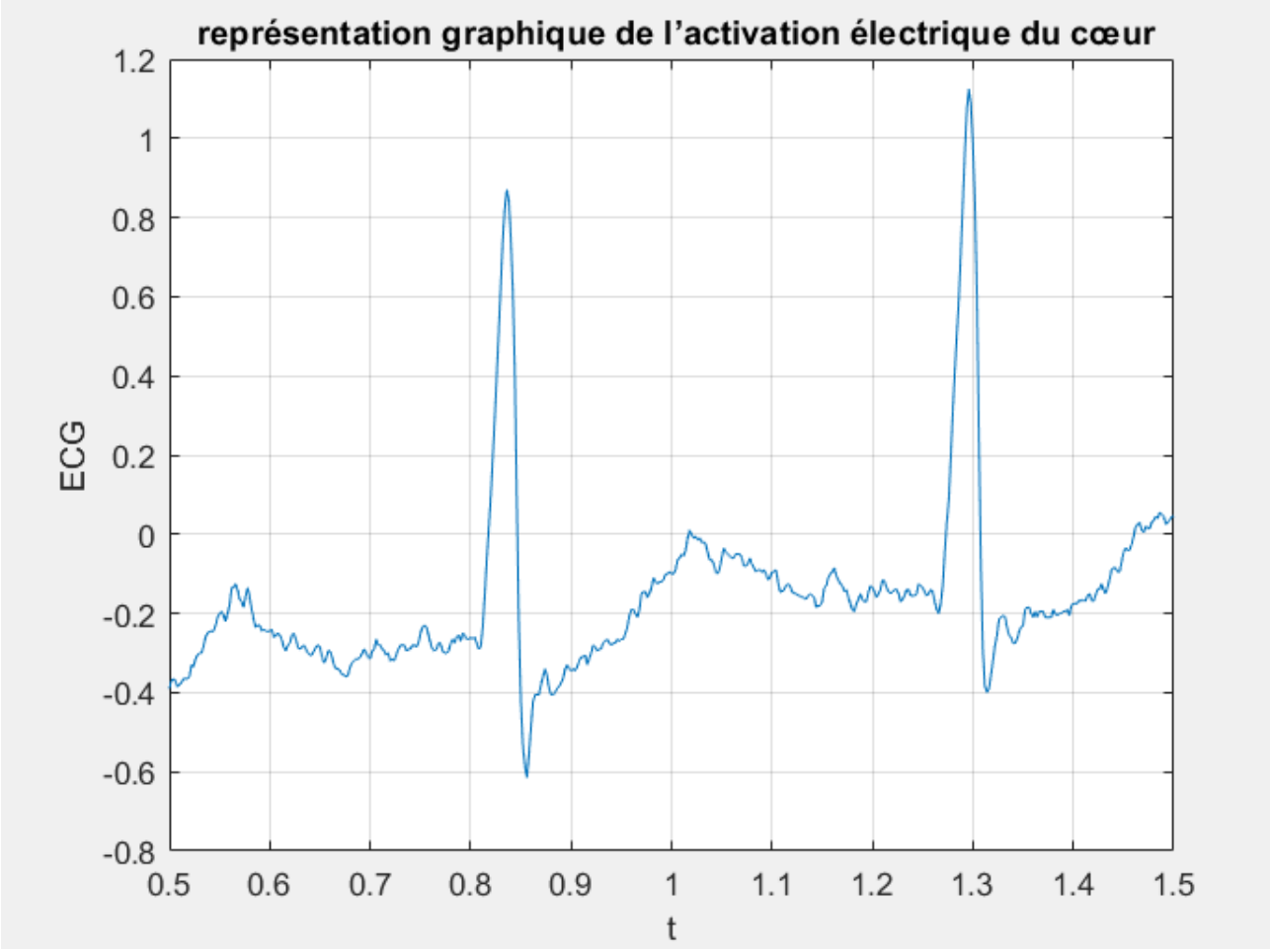
Introduction

Le TP3 de traitement d'un signal ECG a pour objectifs la suppression du bruit autour du signal produit par un électrocardiographe et la recherche de la fréquence cardiaque. Il est important de noter que le traitement de ce type de signal nécessite de prendre en compte les différences entre le temps continu et le temps discret. L'ECG est un outil diagnostique important pour détecter les anomalies cardiaques telles qu'une arythmie ou un risque d'infarctus. Les signaux ECG sont souvent contaminés par différentes sources de bruit, et il est donc important de filtrer ces bruits pour obtenir un diagnostic précis.

SUPPRESSION DU BRUIT PROVOQUÉ PAR LES MOUVEMENTS DU CORPS

2- Représentation temporelle du signal échantilloné

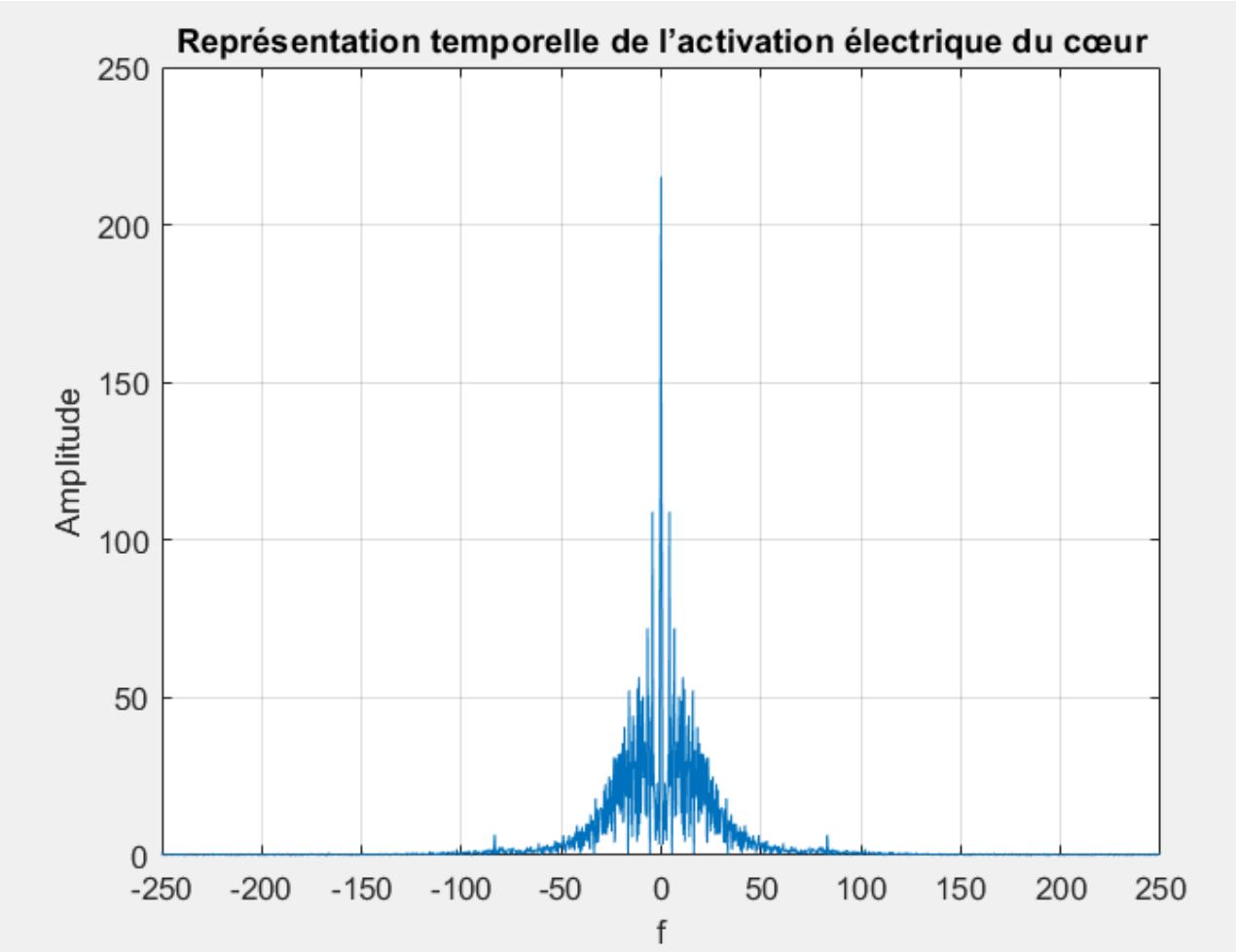
```
load('ecg.mat');
Fe=500;
Te=1/Fe;
N=length(ecg);
t=0:Te:(N-1)*Te;
plot(t,ecg)
grid on
title(" représentation graphique de l'activation électrique du cœur")
xlabel("t")
ylabel("ECG")
xlim([0.5 1.5]);
```



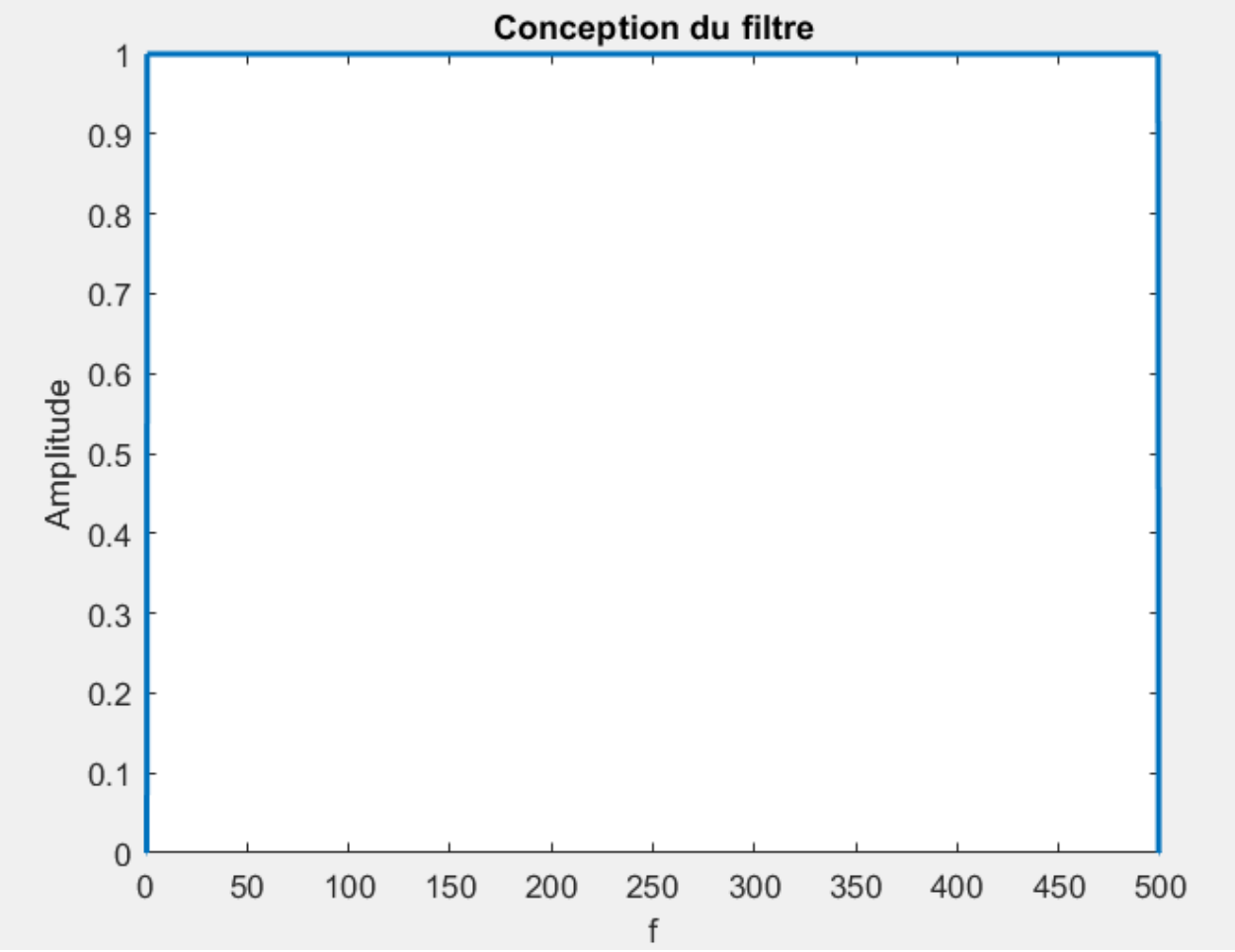
D'après la figure représentée on remarque que le signal est bruitée , c'est à dire que l'information est altérée par des interférences, des bruits internes musculaires et des bruits provoqués par les mouvements du corps résultant de l'activité respiratoire; d'où l'importance du filtrage.

3-Représentation fréquentielle du signal échantilloné

```
% Représentation Fréquentielle

f=(0:N-1)*(Fe/N);
fshift=(-N/2:N/2-1)*Fe/N;
y = fft(ecg);
plot(fshift,fftshift(abs(y)));
grid on
xlabel('f');
ylabel('Amplitude')
title('Représentation temporelle de l'activation électrique du cœur');
```



```
% Conception du fitltre pass_haut

pass_haut = ones(size(ecg));
fc1=0.5;
index_fc = ceil((fc1*N)/Fe);
pass_haut(1:index_fc)=0;
pass_haut(N-index_fc+1:N)=0;
plot(f,pass_haut,"linewidth",1.5)
xlabel('f');
ylabel('Amplitude')
title('Conception du filtre');
```



Le filtre passe haut est un vecteur de 1 qui prend une valeur nulle dans la fréquence de coupure fc1 = 0.5 Hz et son symétrique.

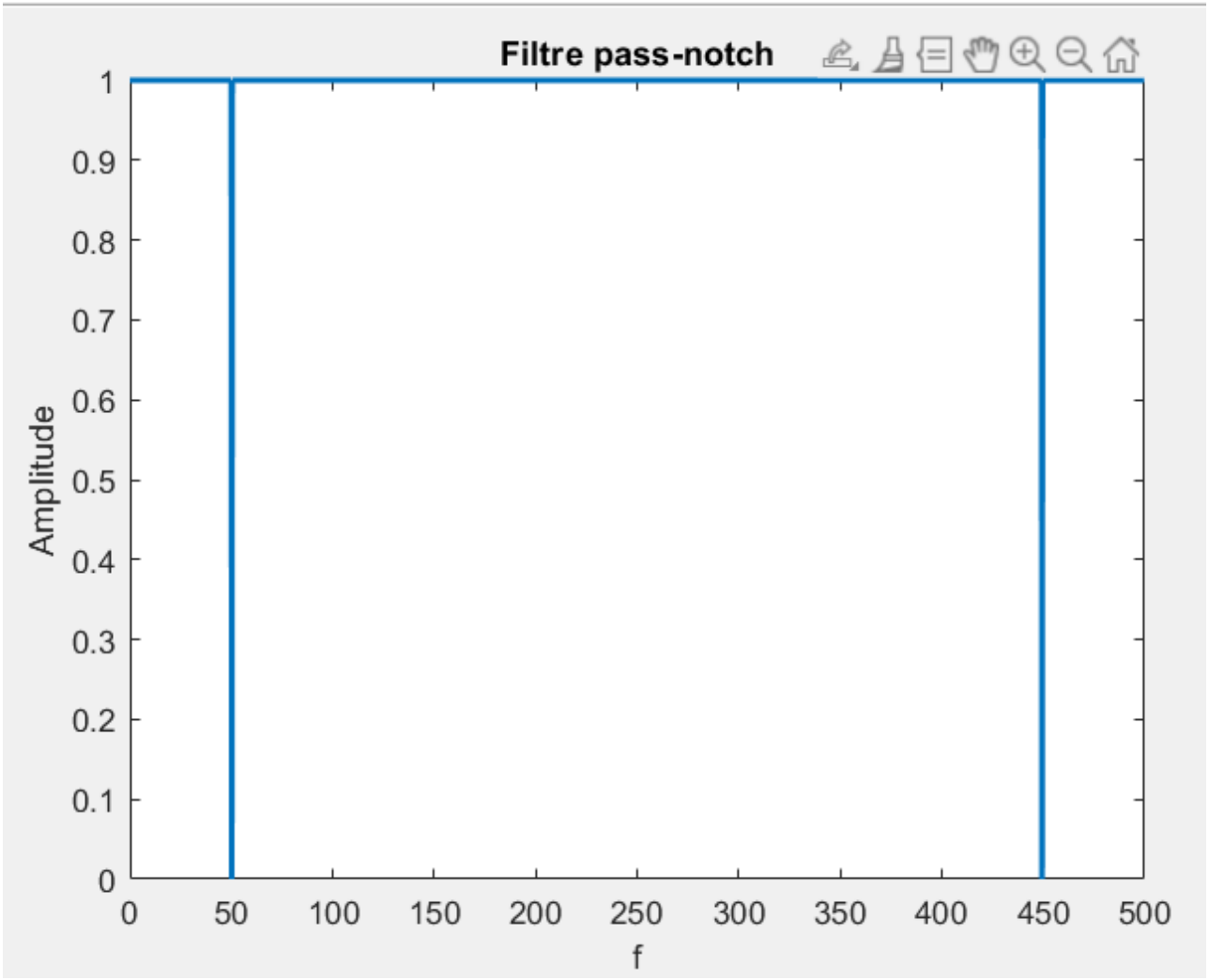
```
% Filtrage
ecg1_freq = pass_haut.*y;
ecg1=ifft(ecg1_freq,"symmetric");
plot(t,ecg);
hold on
plot(t,ecg1+3);
hold on
plot(t,ecg-ecg1+1.5);
grid on
xlabel('t');
ylabel('signal')
title('Représentation du signal avant et après le filtre passe-haut');
```

Le filtrage se fait par la multiplication d'élément par élément du filtre passe-haut avec la TFD du signal ecg, et l'application de la transformée de Fourier inverse par la fonction "ifft" pour restituer le signal temporel filtré.

Après l'application du filtre passe-haut on remarque un changement au niveau du signal où il n'y a plus de décalage au niveau de la hauteur.

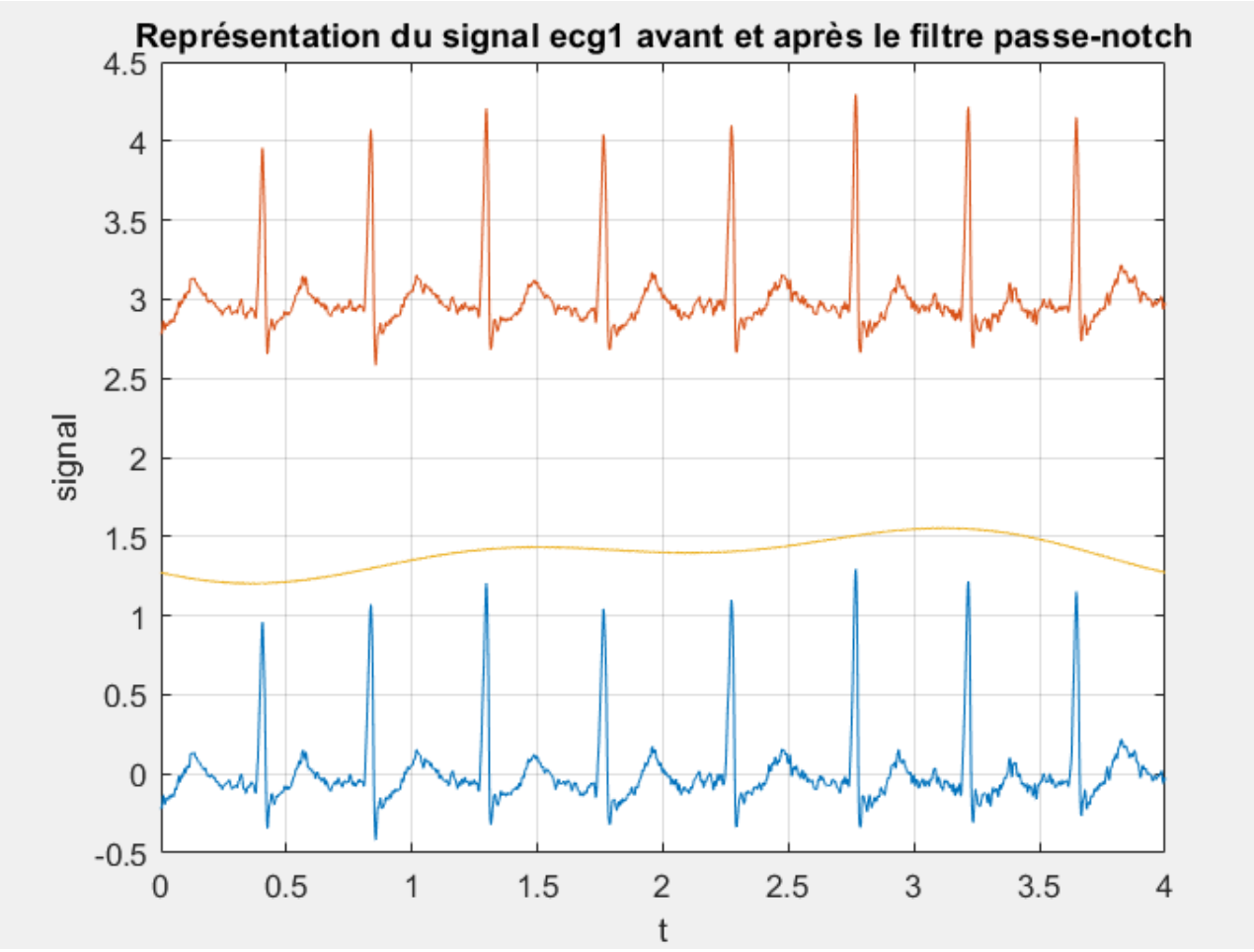
SUPPRESSION DES INTERFÉRENCES DES LIGNES ÉLECTRIQUES 50HZ

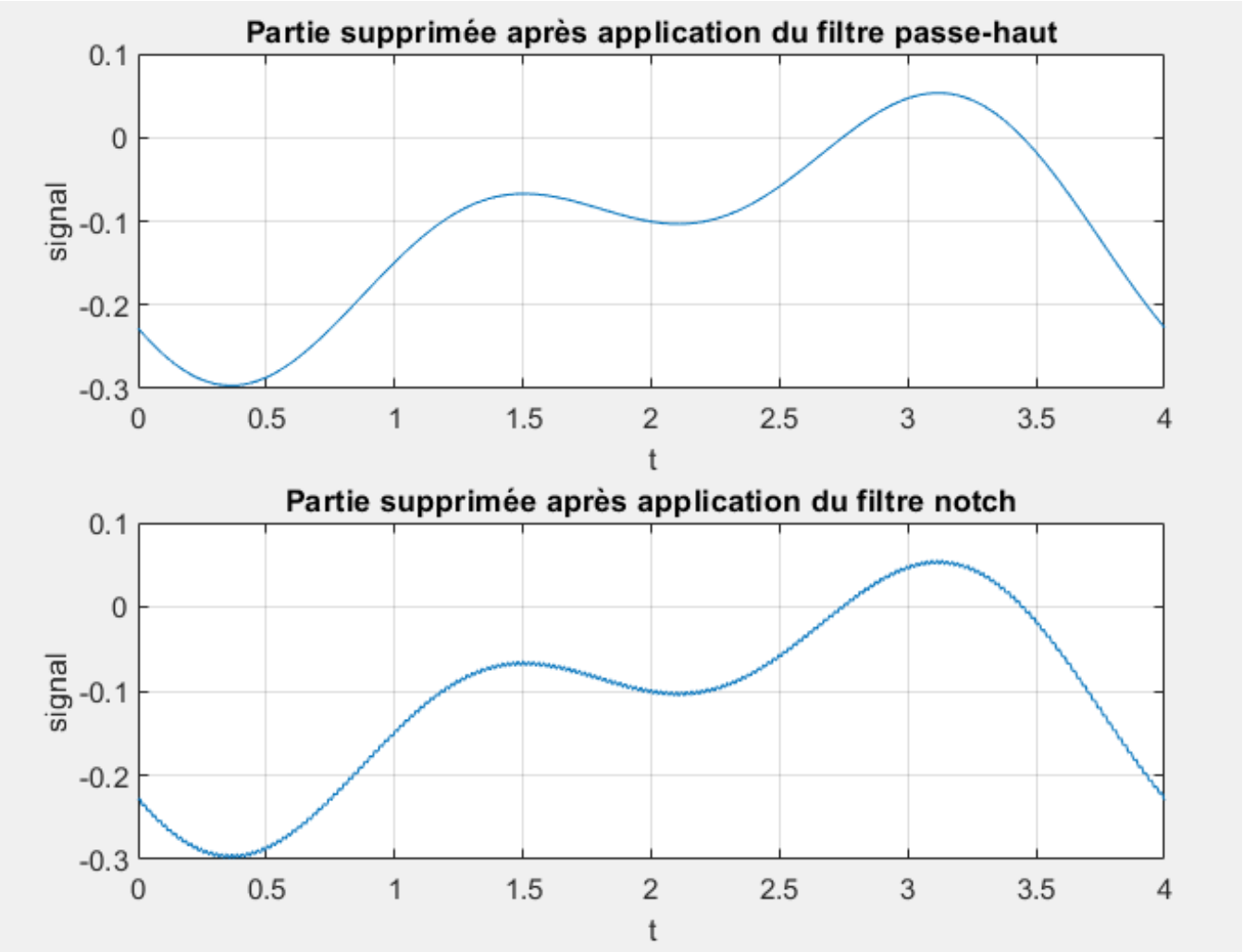
```
% Conception du fitltre pass_notch

pass_notch = ones(size(ecg));
fc2=50;
index_fc2 = ceil((fc2*N)/Fe)+1;
pass_notch(index_fc2)=0;
pass_notch(N-index_fc2+1)=0;
plot(f,pass_notch,"linewidth",1.5)
```



On utilise un filtre pass-notch pour supprimer les interférences de 50Hz, ce dernier est un vecteur de 1 qui prend la valeur 0 dans la fréquence 50HZ et son symétrique 450.

```
% Filtrage 2
ecg2_freq = pass_notch.*fft(ecg1);
ecg2=ifft(ecg2_freq,"symmetric");

plot(t,ecg1);
hold on
plot(t,ecg2+3);
hold on
plot(t,ecg-ecg2+1.5);
grid on
xlabel('t');
ylabel('signal')
title('Représentation du signal ecg1 avant et après le filtre passe-notch');
```





On constate que le signal sinusoïdale de 50Hz a été aussi supprimée après l'application du filtre pass-notch.

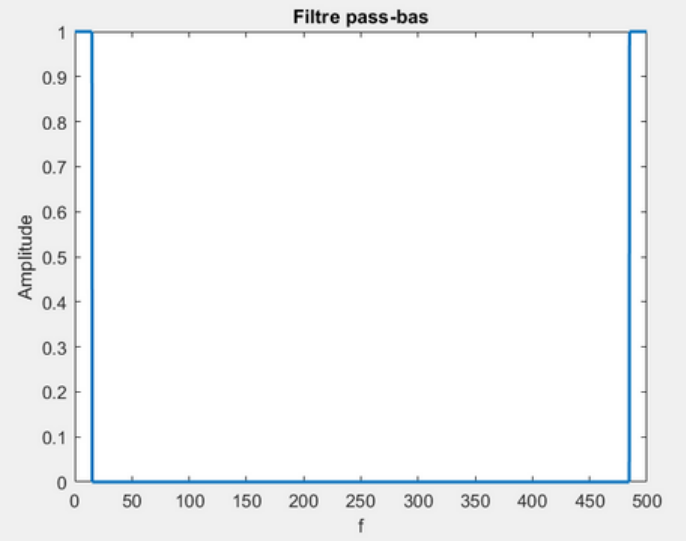
AMÉLIORATION DU RAPPORT SIGNAL SUR BRUIT

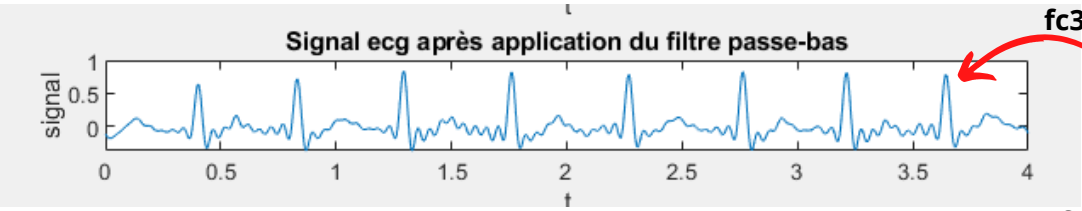
Après la suppression des des bruits de basses fréquences et des interférences, on remarque que le signal est toujours erroné et que l'information n'a pas encore été restituée. Il faut donc supprimer les bruits de hautes fréquences.

AMÉLIORATION DU RAPPORT SIGNAL SUR BRUIT

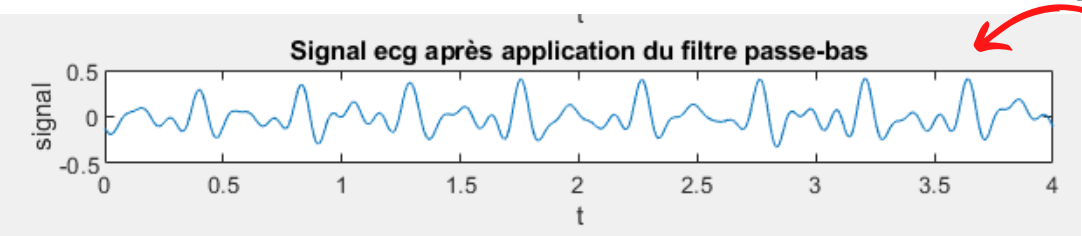
```
% Conception du filtre pass_bas

pass_bas=zeros(size(ecg));
fc3 = 10;
index_fc3 = ceil((fc3*N)/Fe);
pass_bas(1:index_fc3)= 1;
pass_bas(N-index_fc3+1:N) = 1;
xlabel('f');
ylabel('Amplitude')
title('Filtre pass-bas');

% Filtrage 3

ecg3_freq = pass_bas.*fft(ecg2);
ecg3 = ifft(ecg3_freq,"symmetric");
```

```
subplot(311)
plot(t,ecg)
xlabel('t');
ylabel('signal')
title('Signal ecg original');

subplot(312)
plot(t,ecg2)
xlabel('t');
ylabel('signal')
title('Partie supprimée après application du filtre notch');


subplot(313)
plot(t,ecg3)
xlabel('t');
ylabel('signal')
title('Signal ecg après application du filtre passe-bas');
```

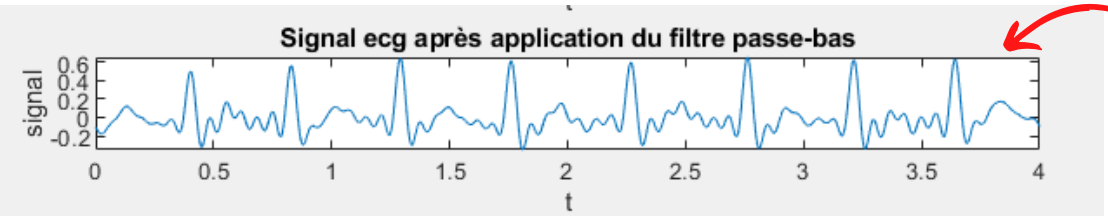




fc3 =20 Hz



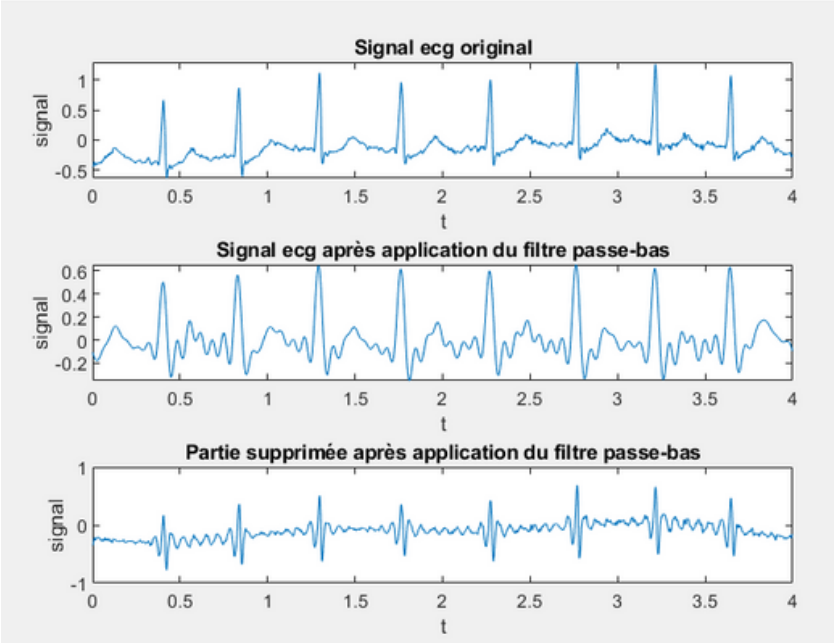
fc3 =10 Hz



fc3 =15 Hz

Nous avons essayé 3 types de fréquences de coupure
pour le filtre passe-bas , pour voir celle qui ne nous fera pas perdre l'information et nous pouvons remarquer que fc3= 15 est la meilleure pour appliquer le filtre passe-bas.

On observe qu'après l'application des 3 filtres, passe haut ,passe-bas , pass-notch, nous avons pu éliminer les différents bruits et interférences et restituer le signal avec l'information voulue.



```
%Définir l'intervalle de recherche pour la fréquence cardiaque
min_hr = 40; % battements par minute
max_hr = 220; % battements par minute

% Calculer l'autocorrélation du signal ECG
[acf,lags] = xcorr(ecg3,ecg3);

% Trouver la fréquence cardiaque en se basant sur l'autocorrélation
[max_corr, max_index] = max(acf);
heart_rate = 60*Fe/(lags(max_index))

% Vérifier si la fréquence cardiaque est dans l'intervalle de recherche
if heart_rate > min_hr && heart_rate < max_hr
    disp(['Fréquence cardiaque : ', num2str(heart_rate), ' battements par minute']);
else
    disp('Fréquence cardiaque non détectée');
end
```

Ce code utilise des données d'un signal ECG pour calculer la fréquence cardiaque de la personne. Il utilise d'abord la méthode de l'autocorrélation pour calculer la corrélation entre une copie décalée du signal ECG et lui-même. Il utilise ensuite la valeur maximale de cette autocorrélation pour déterminer le décalage temporel qui correspond à la fréquence cardiaque. Il utilise ensuite cette information pour calculer la fréquence cardiaque en battements par minute. Enfin, Il vérifie si la fréquence cardiaque est dans un intervalle de recherche (min_hr = 40 bpm et max_hr = 220 bpm) et si c'est le cas, il affiche la fréquence cardiaque. S'il ne se trouve pas dans cet intervalle, il affiche un message indiquant que la fréquence cardiaque n'a pas été détectée.

Conclusion

En conclusion, ce TP sur le traitement d'un signal ECG nous a permis de comprendre les différentes étapes nécessaires pour nettoyer et analyser un signal ECG. Nous avons commencé par supprimer les bruits autour du signal produit par un électrocardiographe en utilisant des techniques de filtrage. Ensuite, nous avons utilisé des méthodes de détection de pics pour trouver la fréquence cardiaque. Il est important de noter que l'échantillonnage du signal avec Matlab implique des différences de traitement entre le temps continu et le temps discret, et il est donc important de faire attention à ces différences lors de l'analyse. En fin de compte, ce TP nous a permis de comprendre les concepts fondamentaux derrière l'analyse des signaux ECG et comment utiliser des outils informatiques pour mettre en pratique ces concepts.