

TP3 : Traitement d'un signal ECG

Réalisé par : IKEN Fatima

Encadre par : Mr AMMOUR ALAE

Filière : big data

Année universitaire : 2022/2023

Objectifs

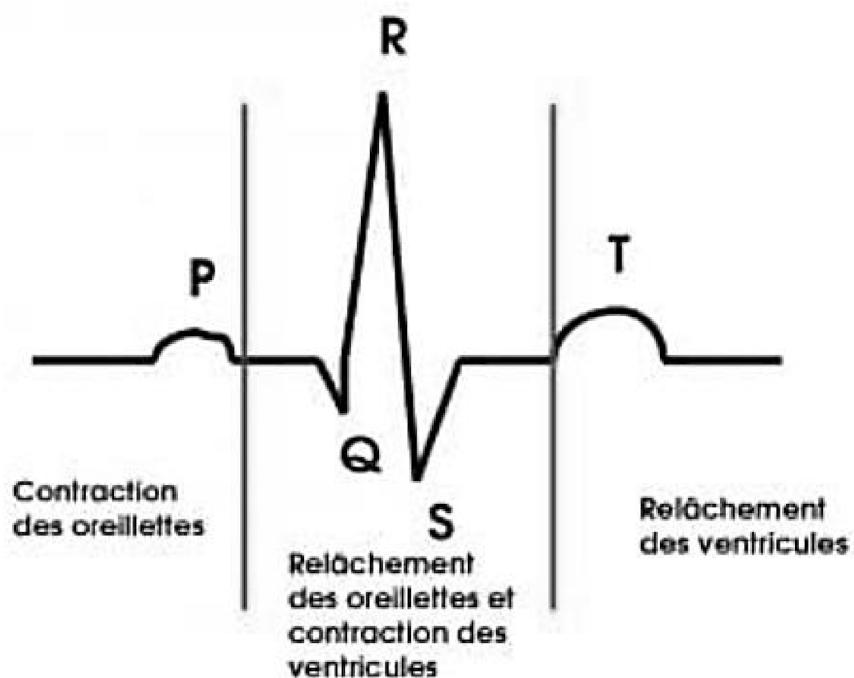
Suppression du bruit autour du signal produit par un électrocardiographe.

Recherche de la fréquence cardiaque.

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

=====>c'est quoi un signal ECG?!



Un électrocardiogramme (ECG) est une représentation graphique de l'activation électrique du cœur à l'aide d'un électrocardiographe. Cette activité est recueillie sur un patient allongé, au repos, par des électrodes posées à la surface de la peau. L'analyse du signal ECG est utile dans le but de diagnostiquer des anomalies cardiaques telles qu'une arythmie, un risque d'infarctus, de maladie cardiaque cardiovasculaire ou encore extracardiaque.

Suppression du bruit provoqué par les mouvements du corps

1:Sauvegarder le signal ECG sur votre répertoire de travail, puis charger-le dans Matlab à l'aide la commande load

=>code:

C:\Users\fatim\OneDrive\Desktop\3er

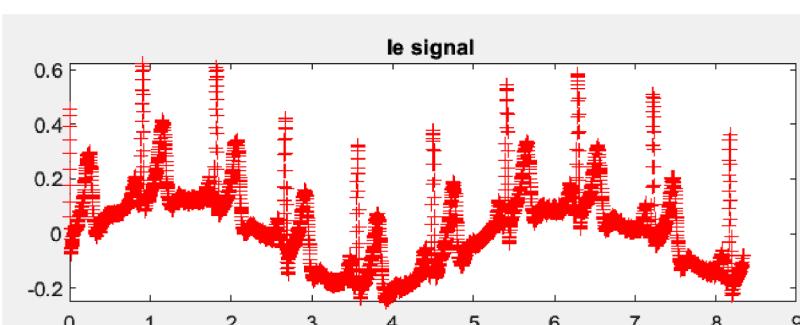
3 tp4.m +

```
clear all  
close all  
clc
```

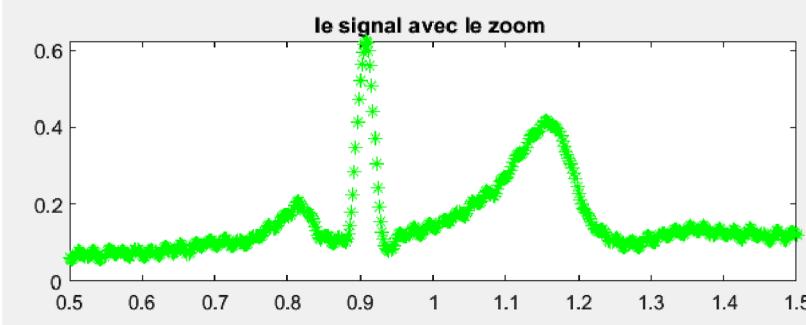
```
load("ecg.mat");  
x=ecg;
```

2:Ce signal a été échantillonné avec une fréquence de 500Hz. Tracer le en fonction du temps, puis faire un zoom sur une période du signal.

=>représentation graphique



=>code

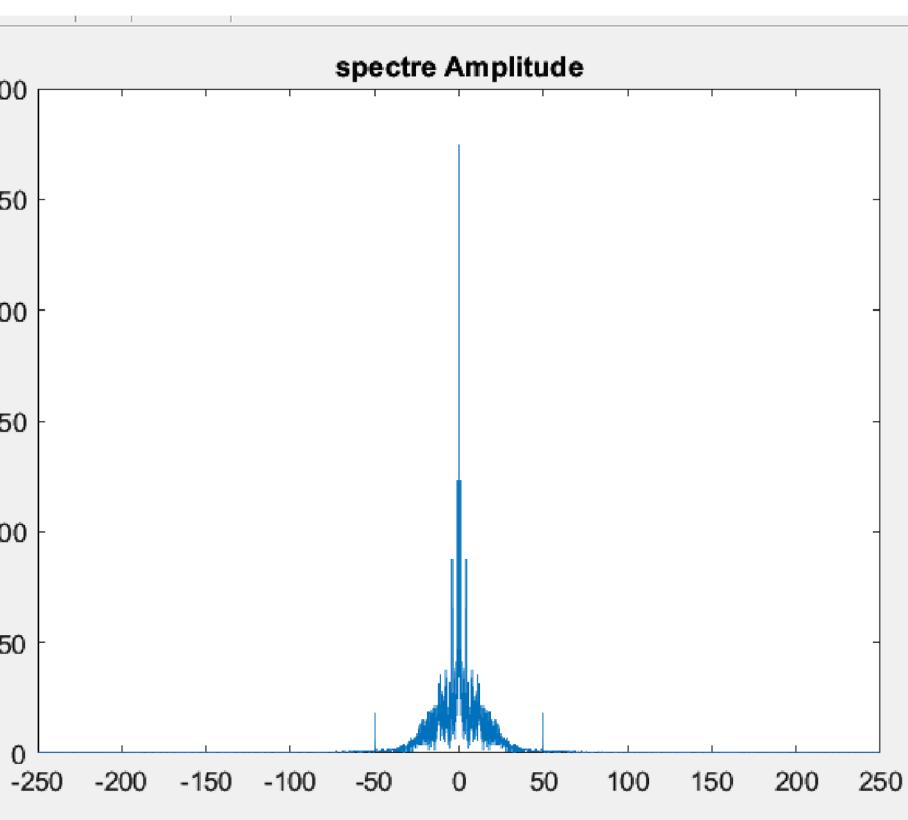


```
clear all  
close all  
clc  
load("ecg.mat");  
x=ecg;  
fs=500;  
N=length(x);  
ts=1/fs;  
t=(0:N-1)*ts;  
subplot(2,1,1)  
plot(t,x,'r +');  
title("le signal ");  
subplot(2,1,2)  
plot(t,x,'g *');  
xlim([0.5, 1.5])  
title("le signal avec le zoom ");
```

3: Pour supprimer les bruits à très basse fréquence dues aux mouvements du corps, on utilisera un filtre idéal passe-haut. Pour ce faire, calculer tout d'abord la TFD du signal ECG, régler les fréquences inférieures à 0.5Hz à zéro, puis effectuer une TFDI pour restituer le signal filtré.

=>spectre d'amplitude centré dans zero

==>représentation graphique



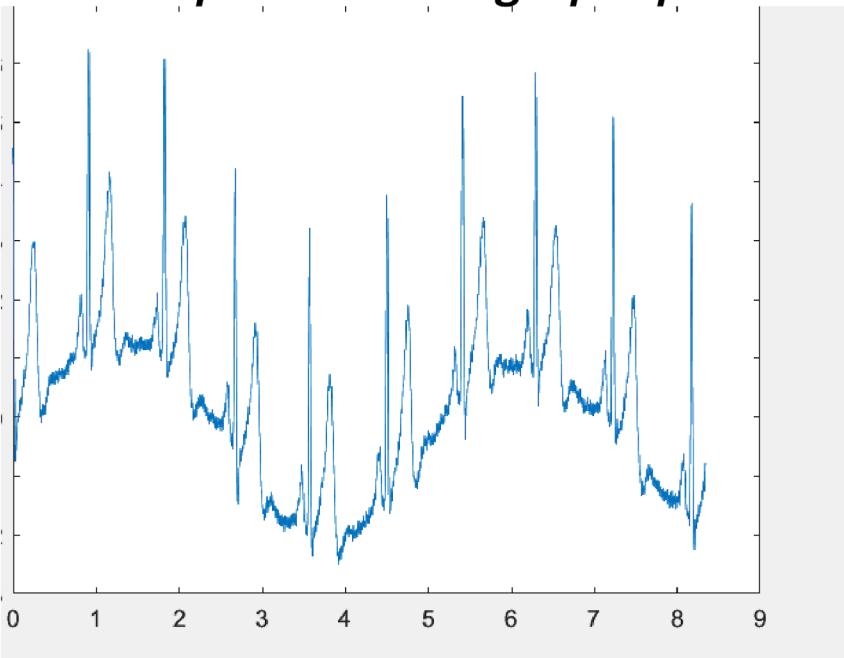
=>code

```
r - C:\Users\fatim\OneDrive\Desktop\3eme année\s5\signal\TPP 3
tled3 tp4.m +
clear all
close all
clc
load("ecg.mat");
x=ecg;
fs=500;
N=length(x);
ts=1/fs;
t=(0:N-1)*ts;

y = fft(x);
f = (0:N-1)*(fs/N);
fshift = (-N/2:N/2-1)*(fs/N);

plot(fshift,fftshift(abs(y)))
title("spectre Amplitude")
```

==>représentation graphique



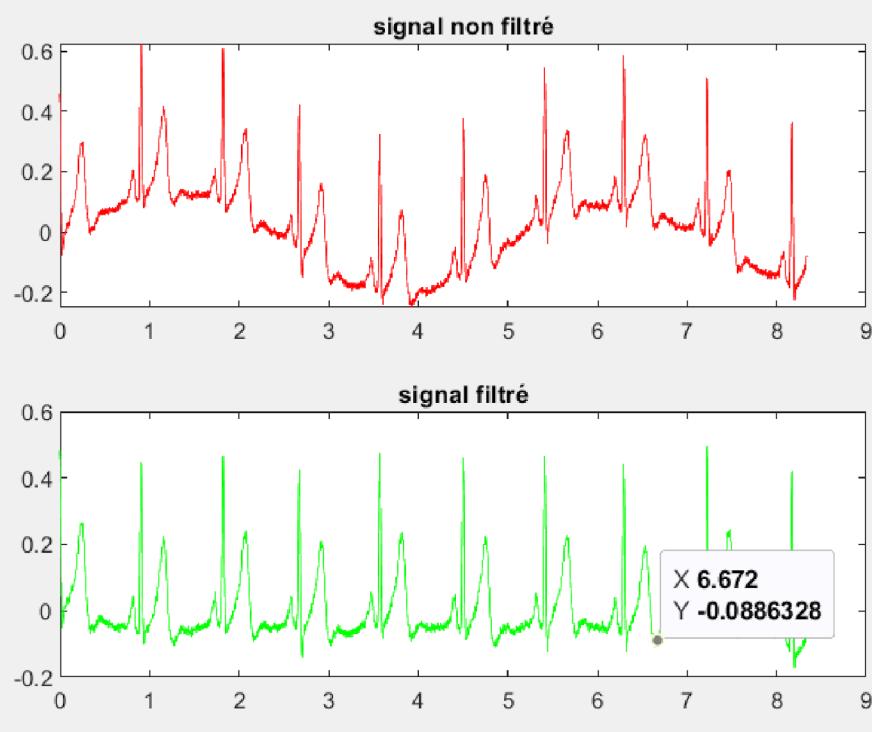
=>code

```
I3 tp4.m x untitled4.m x +  
clear all  
close all  
clc  
load("ecg.mat");  
x=ecg;  
fs=500;  
N=length(x);  
ts=1/fs;  
t=(0:N-1)*ts;  
y = fft(x);  
f = (0:N-1)*(fs/N);  
fshift = (-N/2:N/2-1)*(fs/N);  
h = ones(size(x));  
fh = 0.5;  
index_h = ceil(fh*N/fs);  
h(1:index_h)=0;  
h(N-index_h+1:N)=0;  
ecg1_freq = h.*y;  
ecg1 =ifft(ecg1_freq,"symmetric");  
plot(t,ecg)
```

4: Tracer le nouveau signal *ecg1*, et noter les différences par rapport au signal d'origine.

==>représentation graphique

=>code



```

Users\fatim\OneDrive\Desktop\3eme ann
tp4.m      untitled4.m      +
clear all
close all
clc
load("ecg.mat");
x=ecg;
fs=500;
N=length(x);
ts=1/fs;
t=(0:N-1)*ts;
y = fft(x);
f = (0:N-1)*(fs/N);
fshift = (-N/2:N/2-1)*(fs/N);
h = ones(size(x));
fh = 0.5;
index_h = ceil(fh*N/fs);
h(1:index_h)=0;
h(N-index_h+1:N)=0;
ecg1_freq = h.*y;
ecg1 =ifft(ecg1_freq,"symmetric");
%plot(t,ecg)
subplot(2,1,1)
plot(t,ecg,'r');
title("signal non filtré")
subplot(2,1,2)
plot(t,ecg1,'g');
title("signal filtré")

```

Suppression des interférences des lignes électriques 50Hz

5:Appliquer un filtre Notch idéal pour supprimer cette composante. Les filtres Notch sont utilisés pour rejeter une seule fréquence d'une bande de fréquence donnée

```

clear all
close all
clc
load("ecg.mat");
x=ecg;
fs=500;
N=length(x);
ts=1/fs;
t=(0:N-1)*ts;
y = fft(x);
f = (0:N-1)*(fs/N);
fshift = (-N/2:N/2-1)*(fs/N);
h = ones(size(x));
fh = 0.5;
index_h = ceil(fh*N/fs);
h(1:index_h)=0;
h(N-index_h+1:N)=0;
ecg1_freq = h.*y;
ecg1 =ifft(ecg1_freq,"symmetric");
Notch = ones(size(x));
fcn = 50;
index_hcn = ceil(fcn*N/fs)+1;
Notch(index_hcn)=0;
Notch(index_hcn+2)=0;
ecg2_freq = Notch.*fft(ecg1);
ecg2 =ifft(ecg2_freq,"symmetric");

```

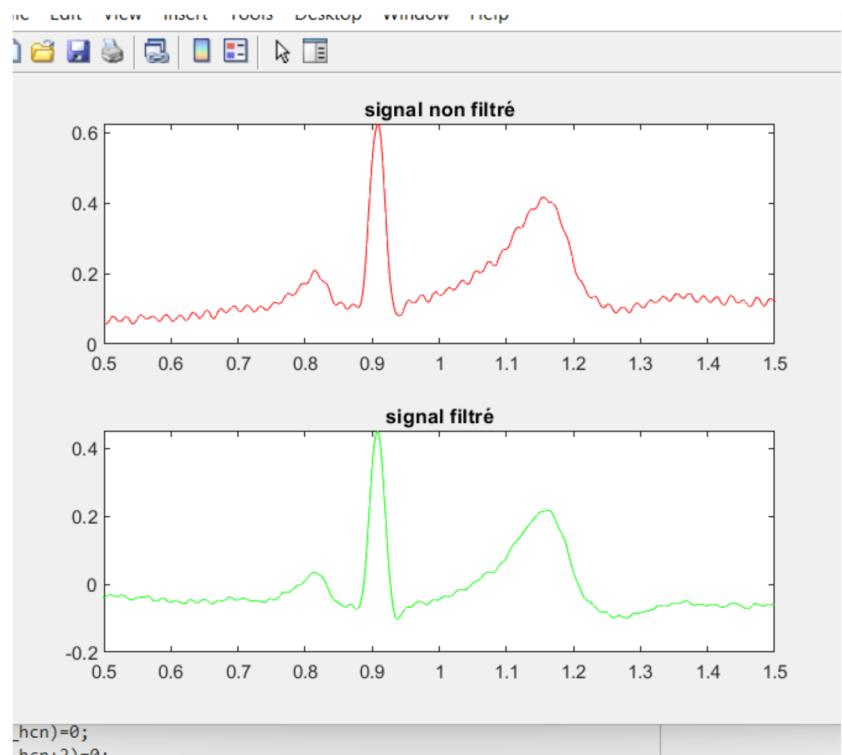
=>code

6: Visualiser le signal ecg2 après filtrage.

=>code

```
untitled3  tp4.m  untitled4.m  +
clc
load("ecg.mat");
x=ecg;
fs=500;
N=length(x);
ts=1/fs;
t=(0:N-1)*ts;
y = fft(x);
f = (0:N-1)*(fs/N);
fshift = (-N/2:N/2-1)*(fs/N);
h = ones(size(x));
fh = 0.5;
index_h = ceil(fh*N/fs);
h(1:index_h)=0;
h(N-index_h+1:N)=0;
ecg1_freq = h.*y;
ecg1 =ifft(ecg1_freq,"symmetric");
Notch = ones(size(x));
fcn = 50;
index_hcn = ceil(fcn*N/fs)+1;
Notch(index_hcn)=0;
Notch(index_hcn+2)=0;
ecg2_freq = Notch.*fft(ecg1);
ecg2 =ifft(ecg2_freq,"symmetric");
subplot(2,1,1)
plot(t,ecg,'r')
xlim([0.5 1.5])
title("signal non filtré")
subplot(2,1,2)
plot(t,ecg2,'y');
title("signal filtré")
xlim([0.5 1.5])
```

==>représentation graphique



Amélioration du rapport signal sur bruit

7: Chercher un compromis sur la fréquence de coupure, qui permettra de préserver la forme du signal ECG et réduire au maximum le bruit. Tester différents choix, puis tracer et commenter les résultats.

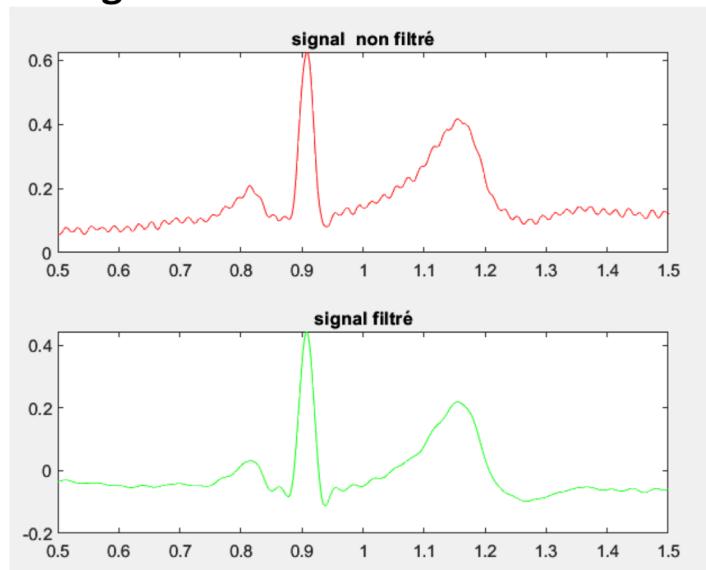
```
x=ecg;
fs=500;
N=length(x);
ts=1/fs;
t=(0:N-1)*ts;
y = fft(x);
f = (0:N-1)*(fs/N);
fshift = (-N/2:N/2-1)*(fs/N);
h = ones(size(x));
fh = 0.5;
index_h = ceil(fh*N/fs);
h(1:index_h)=0;
h(N-index_h+1:N)=0;
ecg1_freq = h.*y;
ecg1 =ifft(ecg1_freq,"symmetric");
Notch = ones(size(x));
fcn = 50;
index_hcn = ceil(fcn*N/fs)+1;
Notch(index_hcn)=0;
Notch(index_hcn+2)=0;
ecg2_freq = Notch.*fft(ecg1);
ecg2 =ifft(ecg2_freq,"symmetric");
pass_bas = zeros(size(x));
fcb = 30;
index_hcb = ceil(fcb*N/fs);
pass_bas(1:index_hcb)=1;
pass_bas(N-index_hcb+1:N)=1;
ecg3_freq = pass_bas.*fft(ecg2);
ecg3 =ifft(ecg3_freq,"symmetric");
```

=>code

indow

8: Visualiser une période du nouveau signal filtré $ecg3$ et identifier autant d'ondes que possible dans ce signal

=>représentation graphique



=>code

```
tshift = (-N/2:N/2-1)*(1/s/N);
h = ones(size(x));
fh = 0.5;
index_h = ceil(fh*N/fs);
h(1:index_h)=0;
h(N-index_h+1:N)=0;
ecg1_freq = h.*y;
ecg1 =ifft(ecg1_freq,"symmetric");
Notch = ones(size(x));
fcn = 50;
index_hcn = ceil(fcn*N/fs)+1;
Notch(index_hcn)=0;
Notch(index_hcn+2)=0;
ecg2_freq = Notch.*fft(ecg1);
ecg2 =ifft(ecg2_freq,"symmetric");
pass_bas = zeros(size(x));
fcb = 30;
index_hcb = ceil(fcb*N/fs);
pass_bas(1:index_hcb)=1;
pass_bas(N-index_hcb+1:N)=1;
ecg3_freq = pass_bas.*fft(ecg2);
ecg3 =ifft(ecg3_freq,"symmetric");

subplot(2,1,1)
plot(t,ecg,'r');
title("signal non filtré")
xlim([0.5 1.5])
subplot(2,1,2)
plot(t,ecg3,'g');
title("signal filtré")
xlim([0.5 1.5])
```

Identification de la fréquence cardiaque avec la fonction d'autocorrélation

9: Ecrire un programme permettant de calculer l'autocorrélation du signal ECG, puis de chercher cette fréquence cardiaque de façon automatique. Utiliser ce programme sur le signal traité *ecg3* ou *ecg2* et sur le signal ECG non traité. NB : il faut limiter l'intervalle de recherche à la plage possible de la fréquence cardiaque.

=>code

```
tcb = 50;
index_hcb = ceil(fcb*N/fs);
pass_bas(1:index_hcb)=1;
pass_bas(N-index_hcb+1:N)=1;
ecg3_freq = pass_bas.*fft(ecg2);
ecg3 =ifft(ecg3_freq,"symmetric");
[c,lags] = xcorr(ecg3,ecg3);

E = length(c);
Vector = [0];
for n = 1:E
    if c(n) > 10
        Vector(end+1) = c(n);
    end
end

M = max(Vector);
in = find(c == M);
s = lags(in);
if s <12
    Vector(Vector == M) = [];
end
end

frequence = (s/fs)*60;
frequence_min=30;
frequence_max=160;
if frequence > frequence_min & frequence < frequence_max
    fprintf('la fréquence cardiaque est :%f ',frequence);
end
```