In The Name Of God The Compassionate & The Merciful

CA#3 SIGNAL

Design by SAM78

4/3/1398

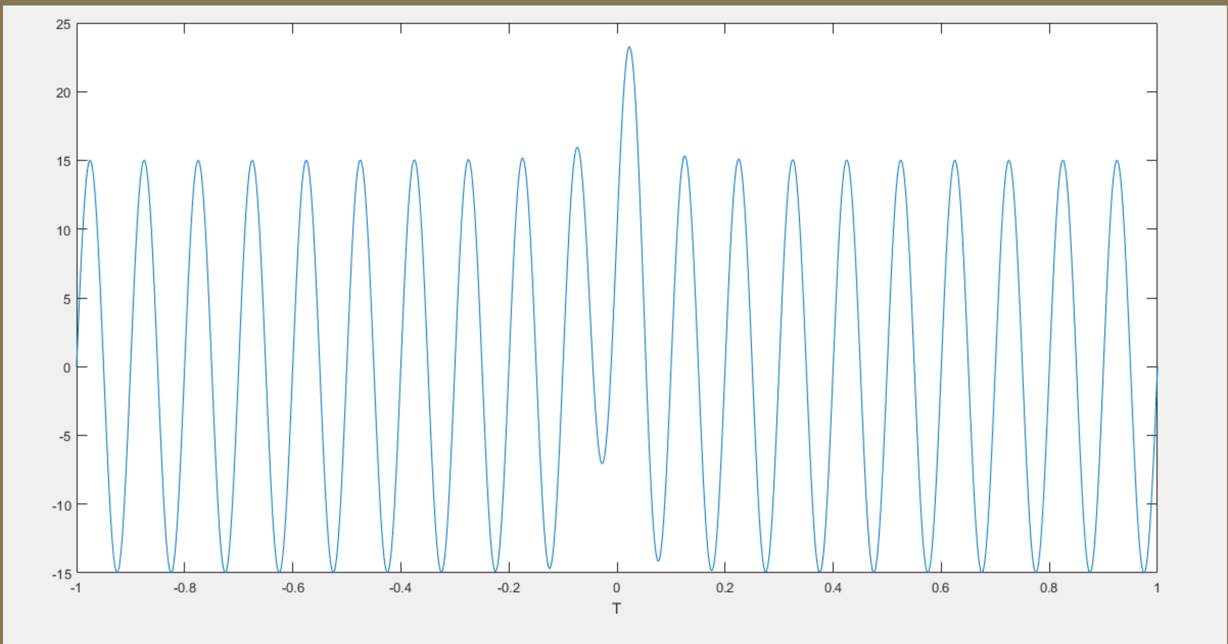
SUMMER TERM

810196405

- First Section: Sampling

This part includes working with arrays and function Fourier single sided and double sided with stem and plot tools.

1st part is about to plotting x function and finding out its length in time domain.



The image above is achieved after having the following code run:

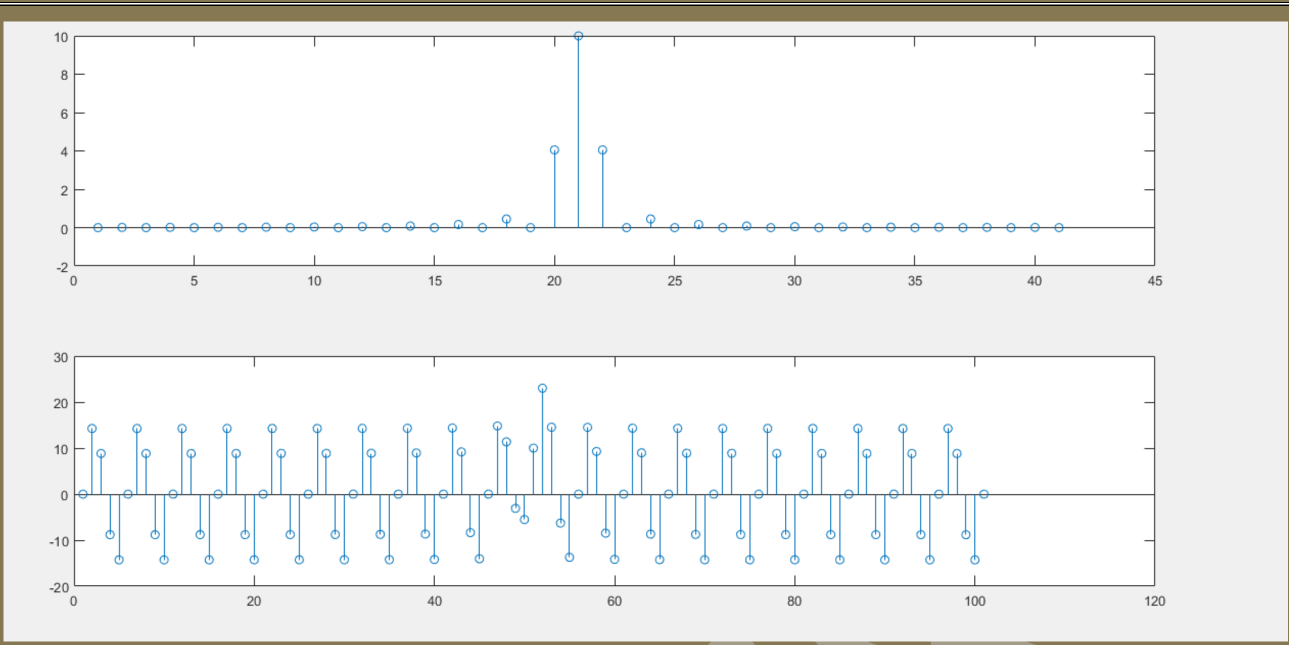
```
clear;
clc;
f0=10;
fs=1000;
fs1=20;
fs2=50;
t=linspace(-10/f0,10/f0,2*fs+1);
x=f0*(sinc(f0*t)).^2+15*sin(2*pi*f0*t);
figure
plot(t,x,'DisplayName','x(t)');
xlabel('T');
```

The length of X is 2*fs+1 which 2 comes from time domain which is from -1 to 1.

For the next part we need to have Fs1=20 and Fs2=50 in the same Time domain.

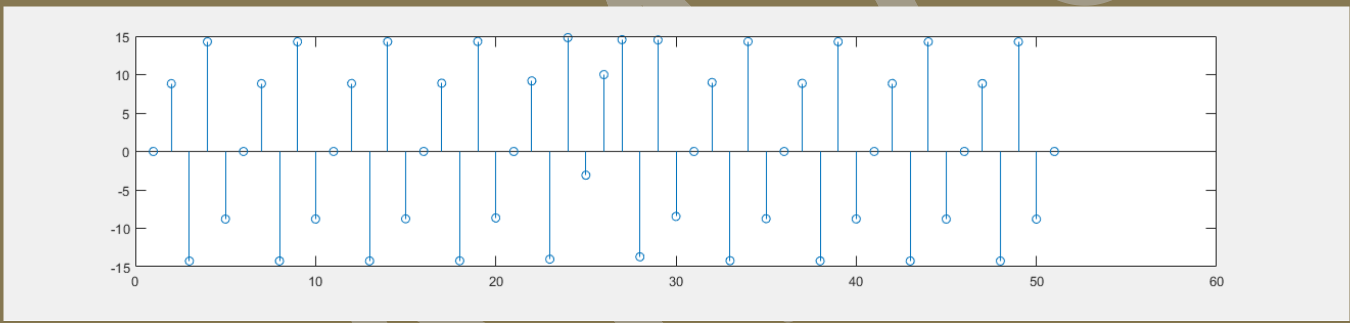
```
x1=x(1:fs/fs1:end);
x2=x(1:fs/fs2:end);

figure
subplot(2,1,1);
stem(x1);
subplot(2,1,2);
stem(x2);
```

As ordered we sampled x and draw it by stem function ;The result comes ahead:



The above picture shows how important fs is to contain enough of information and not to delete some of them. The more the fs the better the result.

Having fs=25 even makes it much more better as shown below:

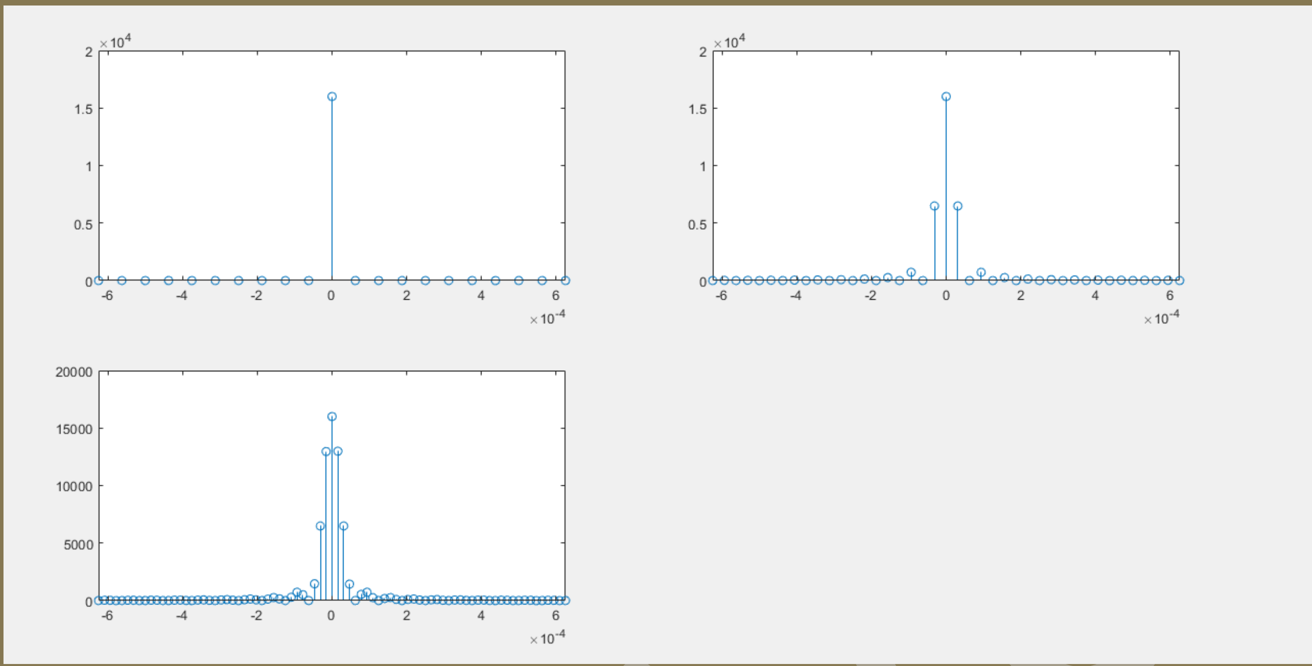


So in the former image Nyquist rate is not considered as it is in the latter.

Now for the next step we will have f0=16000 and alternative fs which vary from 16000 to32000 and 64000.

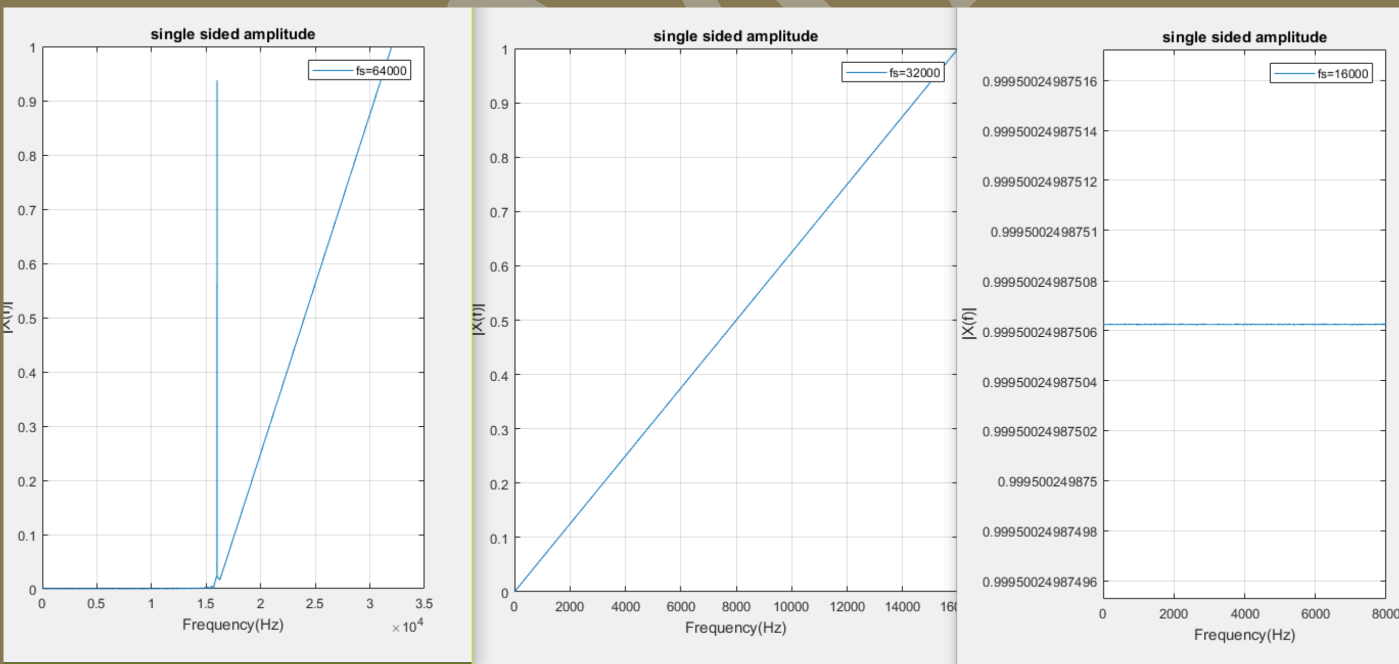
```
    f01=16000;
    fs1=16000;
    fs2=32000;
    fs3=64000;

    t1=-1000/f01:1/fs1:1000/f01;
    t2=-1000/f01:1/fs2:1000/f01;
    t3=-1000/f01:1/fs3:1000/f01;

    x1=f01*(sinc(f01*t1)).^2+15*sin(2*pi*f01*t1);
    x2=f01*(sinc(f01*t2)).^2+15*sin(2*pi*f01*t2);
    x3=f01*(sinc(f01*t3)).^2+15*sin(2*pi*f01*t3);


     figure
     subplot(2,2,1);
    stem(t1,x1);
    xlim([-10/f01,10/f01]);
     subplot(2,2,2);
    stem(t2,x2);
    xlim([-10/f01,10/f01]);
     subplot(2,2,3);
    stem(t3,x3);
    xlim([-10/f01,10/f01]);
```

And the result:



As you see it comes much more better if we consider $fs \geq 2 * f0 + 1$ which means here it should be more than 32000 + 1 .

Frequency analysis:



My Fourierfunction comes below:

I added some options as given name and choosing whether it is double sided or the single one.

This function is called Fourier func as presented here and gives us Fast Fourier Transform of the signal and will plot it as desired.

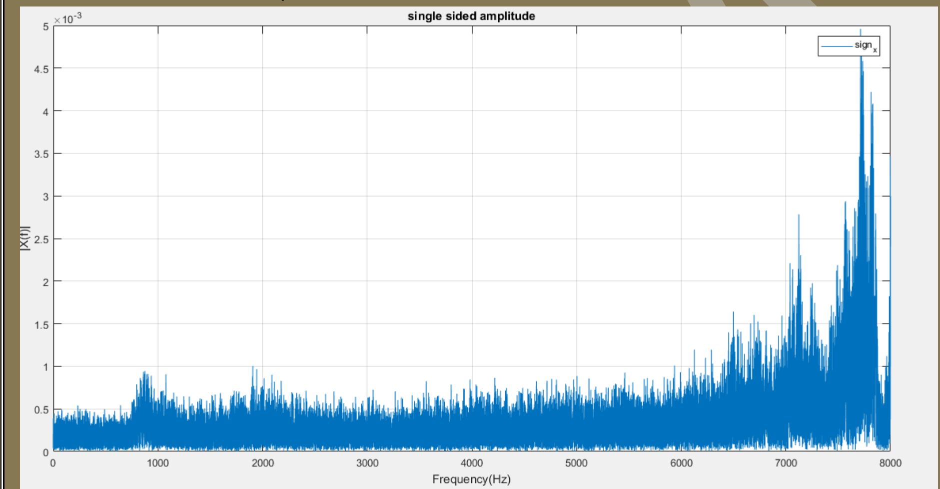
```matlab
function z=Fourierfunc(a,x,fs,p)
if a==1
    b=0;
else if a==2
        b=-1;
    end
end

    name=p;
L=length(x);
NFFT=2.^nextpow2(L);
X=fftshift(fft(x,NFFT)/(16*L));
if a==1
f=(fs/2)*linspace(b,1,NFFT/2+1);
figure
plot(f,2*abs(X(1:NFFT/2+1)),'DisplayName',name);
title('single sided amplitude');
else if a==2
f=(fs/2)*linspace(b,1,NFFT);
figure
plot(f,abs(X(1:NFFT)),'DisplayName',name);
title('double sided amplitude');
    end
end

xlabel('Frequency(Hz)');
ylabel('|X(f)|');
grid on
```

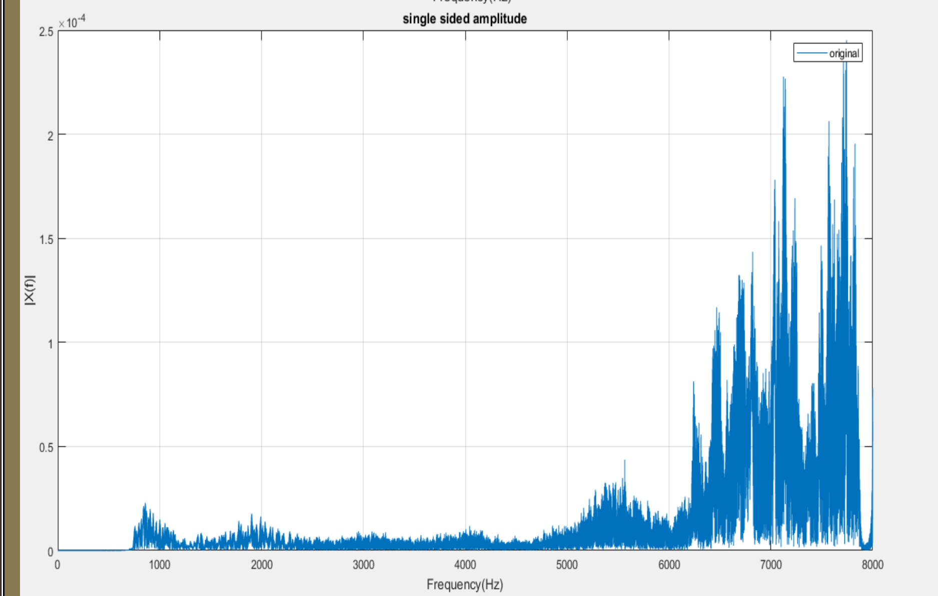
Fourierfunc

Last step in this section is to have audio file listened twice one without change and the other will be changed using sign function.

As we did we were the witness of some changes eventually in the voice getting some noises maybe or unclear in some places but still understandable .

The reason maybe :having sign function operated on the sing cause square wave signals with amp=1 and different duty cycles with sharp edges at high frequencies which result in some noises but the lower frequencies are untouched.

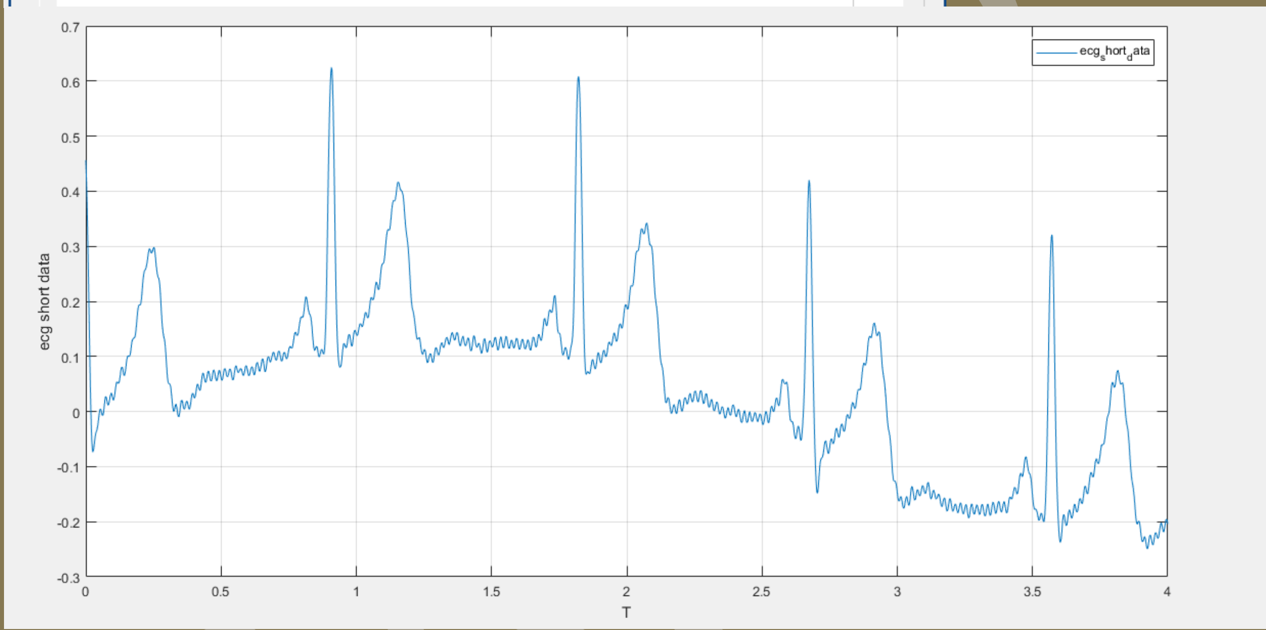




- Second Section: Noise Filtration

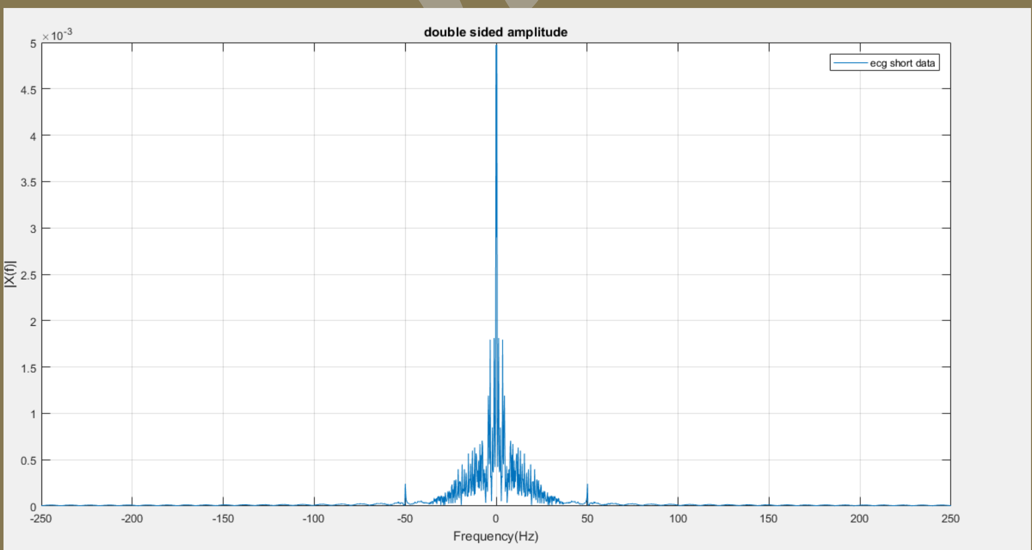
Limiting the Time Domain to 0 to4 and then taking action will cause the signal to have such spectrum :

```
clc;
clear;
ecg_data=load('ecg.dat');
fs=500;
t=0:1/fs:4;

ecg_short_data=ecg_data(1:4*fs+1);


figure
plot(t,ecg_short_data);
xlabel('T');
ylabel('ecg short data');
grid on
legend('show');
```



We still can watch over the fluctuations as it goes up in the frequency and the we see maybe 5 of its period.

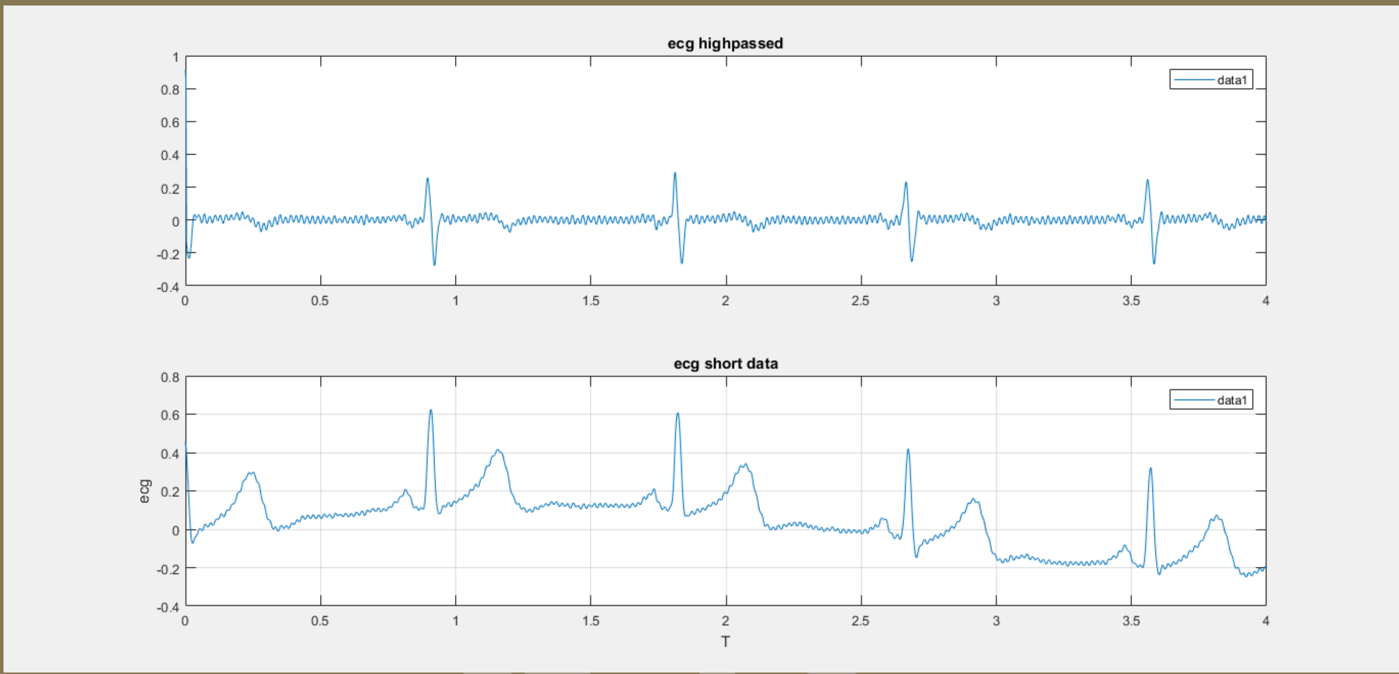
Now we have the Fourier transform of the signal below:we have fftshift but the electrical noise would be seen at the 50 and -50 hz .



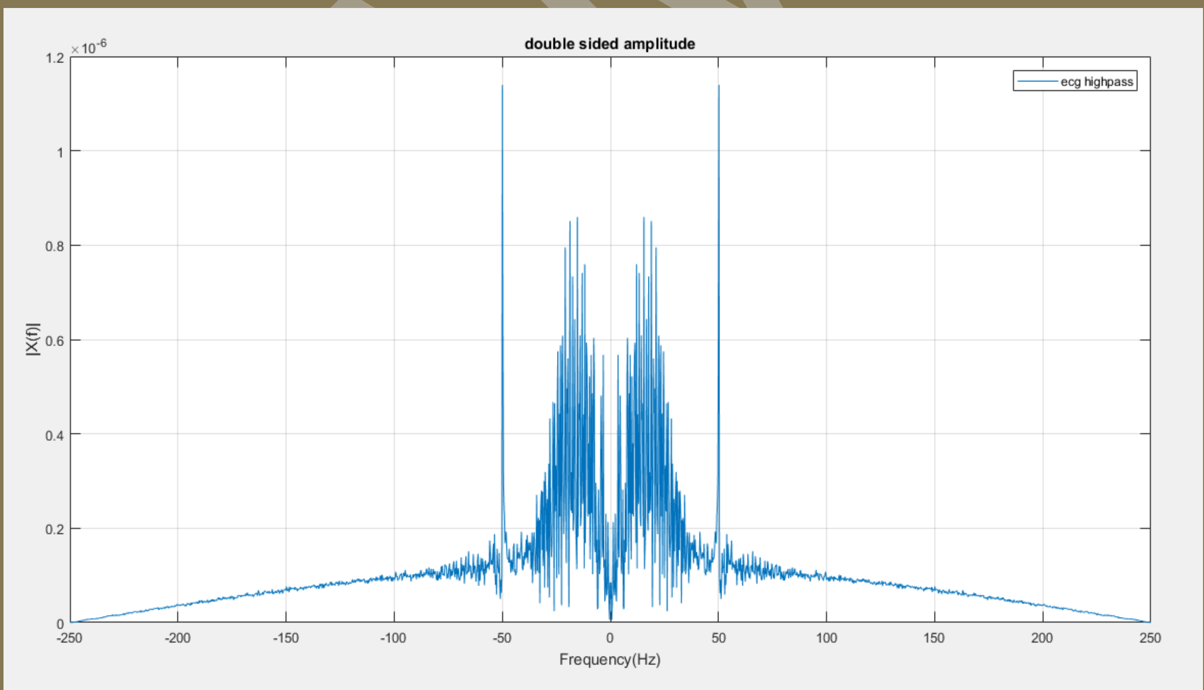
Part 3:

With the usage of the highpassed_filter we are about to omit some noises coming though low frequencies (around 0)with a filter acting like a |H |=2*pi*|f| if you check the diagram of it you'll understand the reason of naming(highpassed).

When we are done the signal would look like:

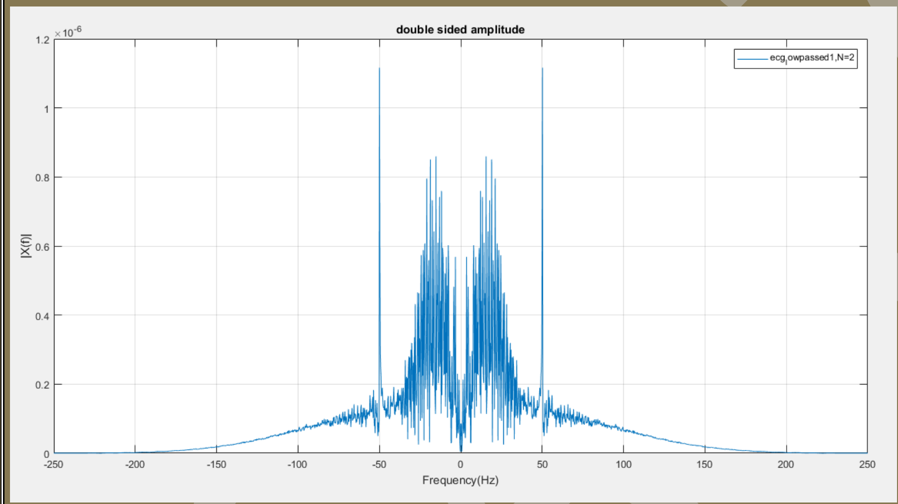


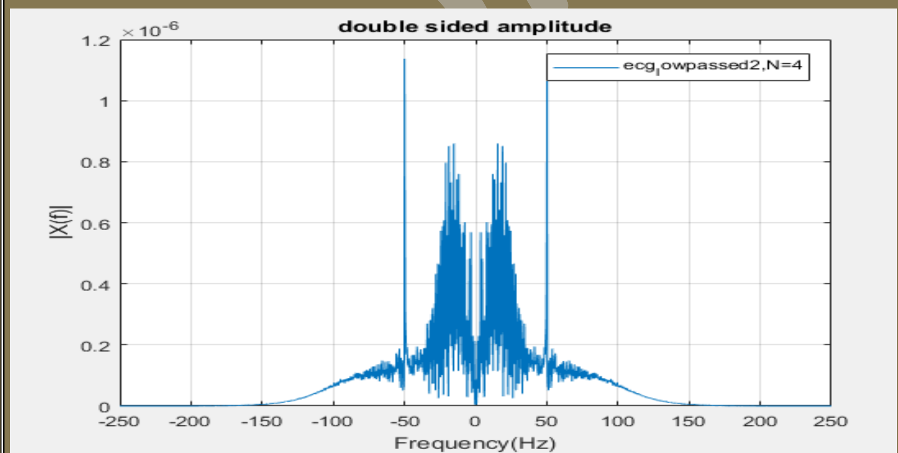
Fourier transform and result around 0 frequency is observable.



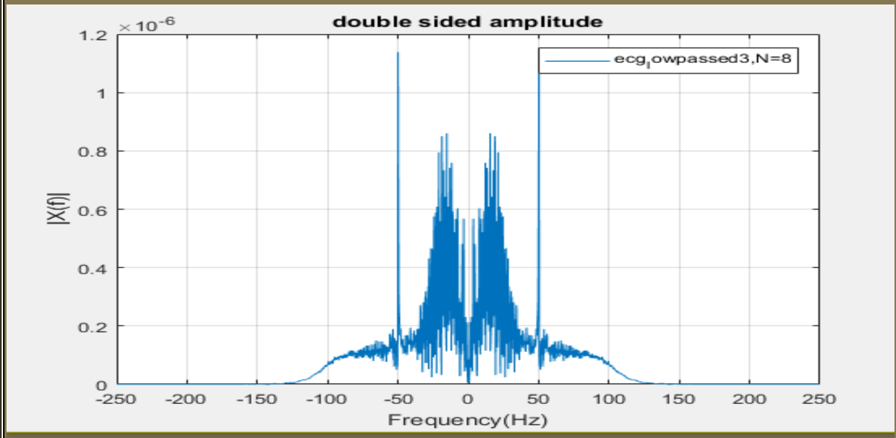
Part 4:

```
fc=100;
wc=fc/(fs/2);
fc2=70;
wc2=fc2/(fs/2);
n1=2;
n2=4;
n3=8;
n4=8;

[b1 a1]= butter(n1,wc,'low');
[b2 a2]= butter(n2,wc,'low');
[b3 a3]= butter(n3,wc,'low');

[b4 a4]= butter(n4,wc2,'low');

ecg_lowpassed1=filter(b1,a1,ecg_highpassed);
ecg_lowpassed2=filter(b2,a2,ecg_highpassed);
ecg_lowpassed3=filter(b3,a3,ecg_highpassed);

ecg_lowpassed=filter(b4,a4,ecg_highpassed);


Fourierfunc(2,ecg_lowpassed1,fs,'ecg_lowpassed1');

Fourierfunc(2,ecg_lowpassed2,fs,'ecg_lowpassed2');

Fourierfunc(2,ecg_lowpassed3,fs,'ecg_lowpassed3');
```

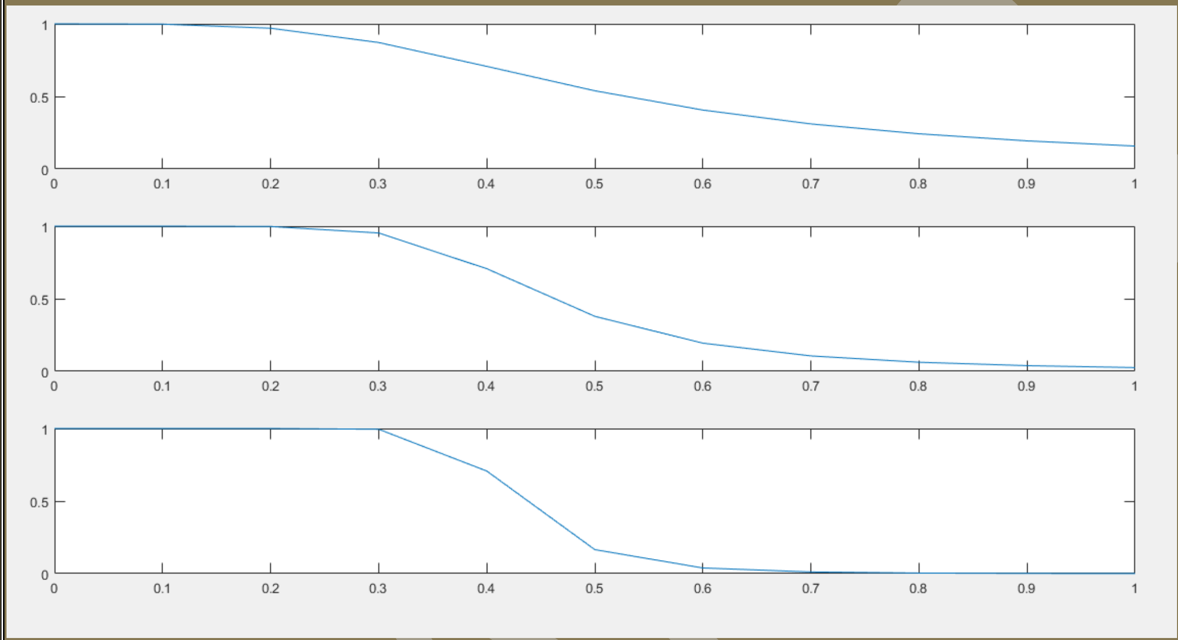




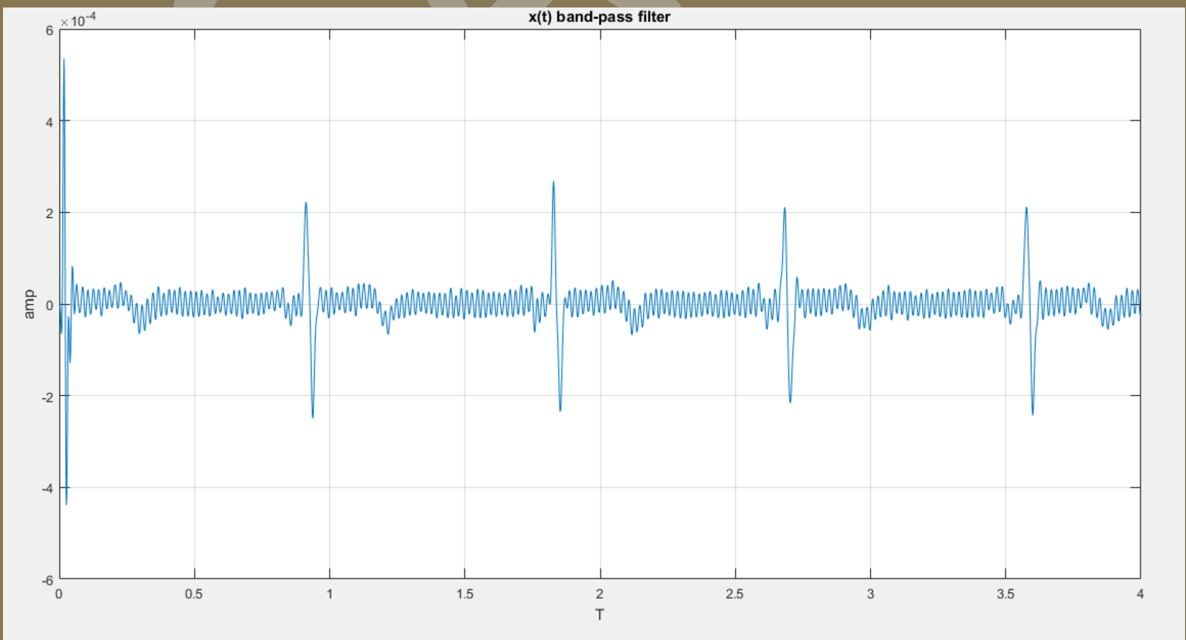
About the level of filter (N) there would be an article within the file;



As the N increase it would be better and sharper.



And there it is the signal after the filter has taken action:



Last part:

After we gained such information we can have heartbeat/s=10^3 * 3600/451 =7980;