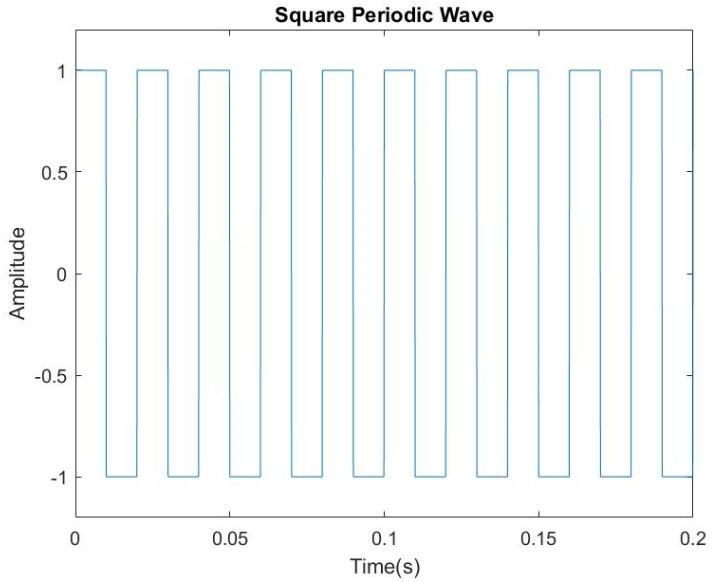
Signal Processing for Neuroscience - Session 1

LIU Jianghao

1 FFT with Matlab

The function fft returns the discrete Fourier transform of vector x, computed with a fast Fourier transform algorithm. Now, we create firstly a square wave as original wave.

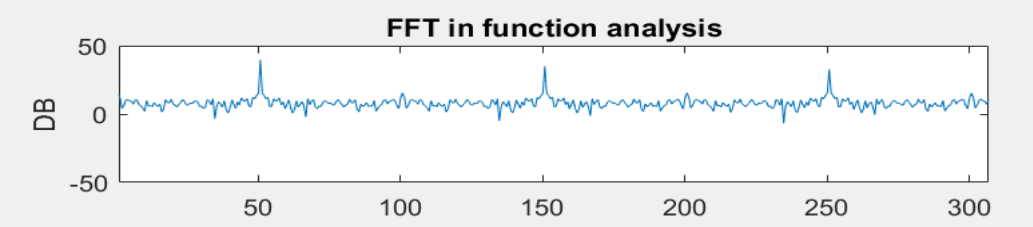


The square wave is composed of a series of simple harmonic wave. These waves have differents frequencies. To capture these high frequencies in the wave composition, we should choose a high sampling frequency, here, fs=10000.

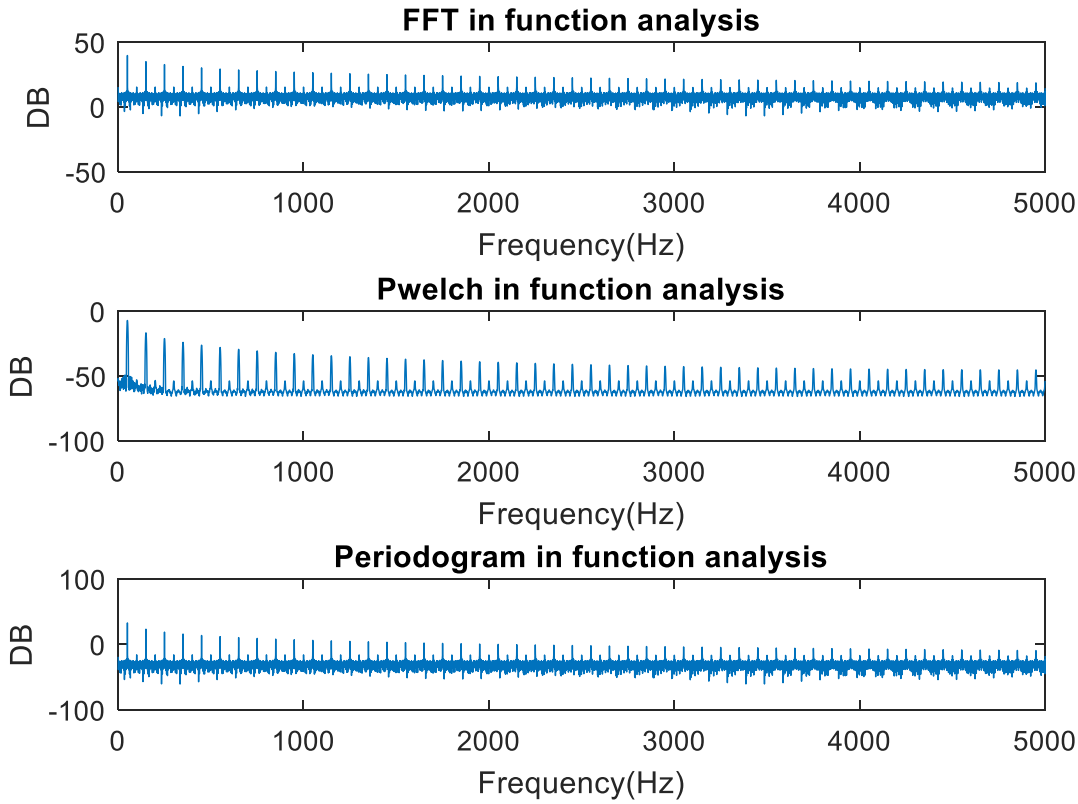
The computer only use limited number of simpling points but the Fourier method requires unlimited points.

To compare different methodes, we use fft,pwelch and periodogram function to plot the Fourier spectrum of the square wave. The pwelch method adds some window function to analyse the wave.

I took a half part of frequency spectrum and the other part is symmetrical and i find that these three results are similars.

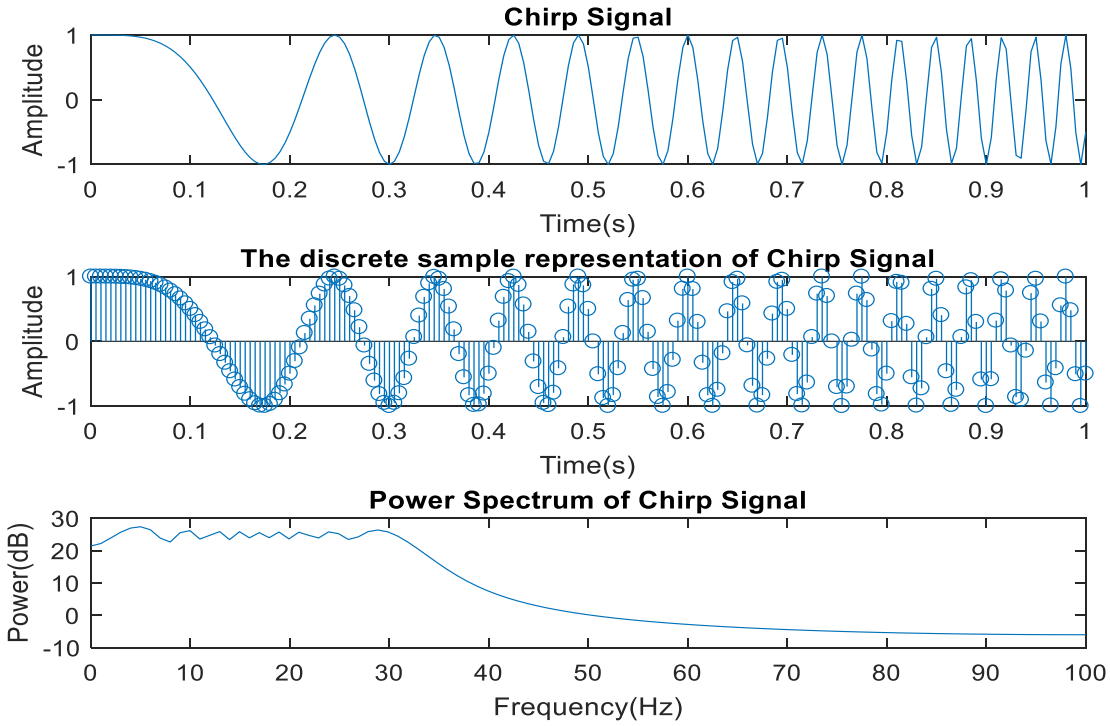


In zooming in it, the peak appears at the w=50hz which is also the frequency of the square wave. The peaks in 150, 250 etc sont the third, fifth composition of Fourier series. Their magnitude decrease with the increase of the frequency.



2 Chirp Signals

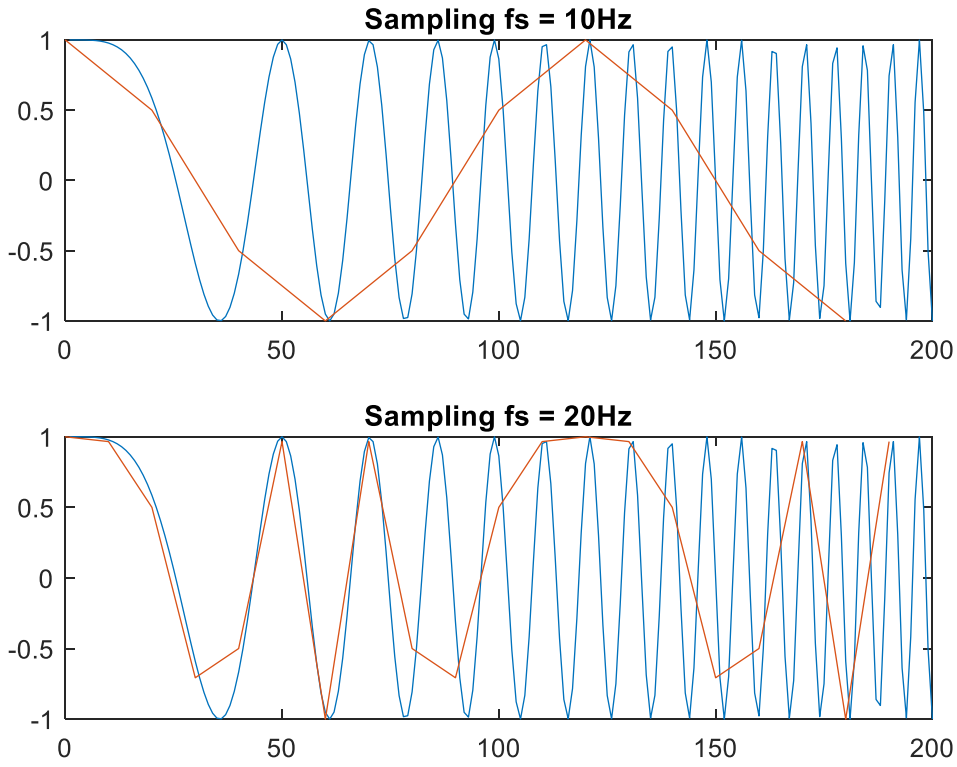
A chirp is a signal in which the frequency increases (up-chirp) or decrease(down-chirp) with time. The fs=200Hz. The frequency is from 0 to Fs/6 (33.3Hz).



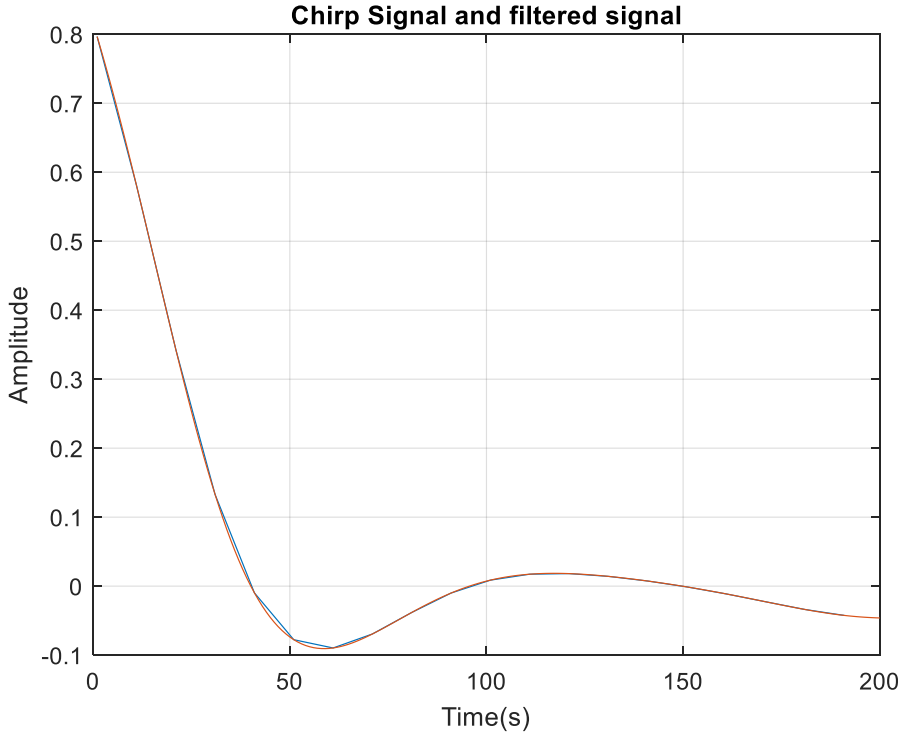
We use a stem function to plot a discrete sample representation. Finally, we use fft function to plot the Fourier power spectrum. We found that the main frequency of this chirp signal is between 0 to 30Hz which correspond to the former parameter. There is a slow attenuation between the 30 to 50 Hz.

3 Downsampling And Aliasing

Nyquist-Shannon sampling theorem: Sampling frequency should be 2 times higher than the signal frequency. The frequency component above the Nyquist frequency (fs/2) will be reconstructed into a signal below the Nyquist frequency. The distortion caused by the overlap of this spectrum is called aliasing, that is, the high frequency signal is mixed into a low frequency signal. To avoid the aliasing, theoretically, we should set a lowpass filter to remove all the frequency above the Nyquist frequency. However, no filter is perfect in real physic exercise, the aliasing will appear between 80% of fs/2 and fs/2. So, we should use a sampling frequency at least 2.5 times higher than signal frequency. The scalp EEG signal is up to 85Hz and minimal sampling rate (up to 1kHz) should be above **5*f0.** For the ERP, only low frequencies are considered (sampling fs=125Hz).



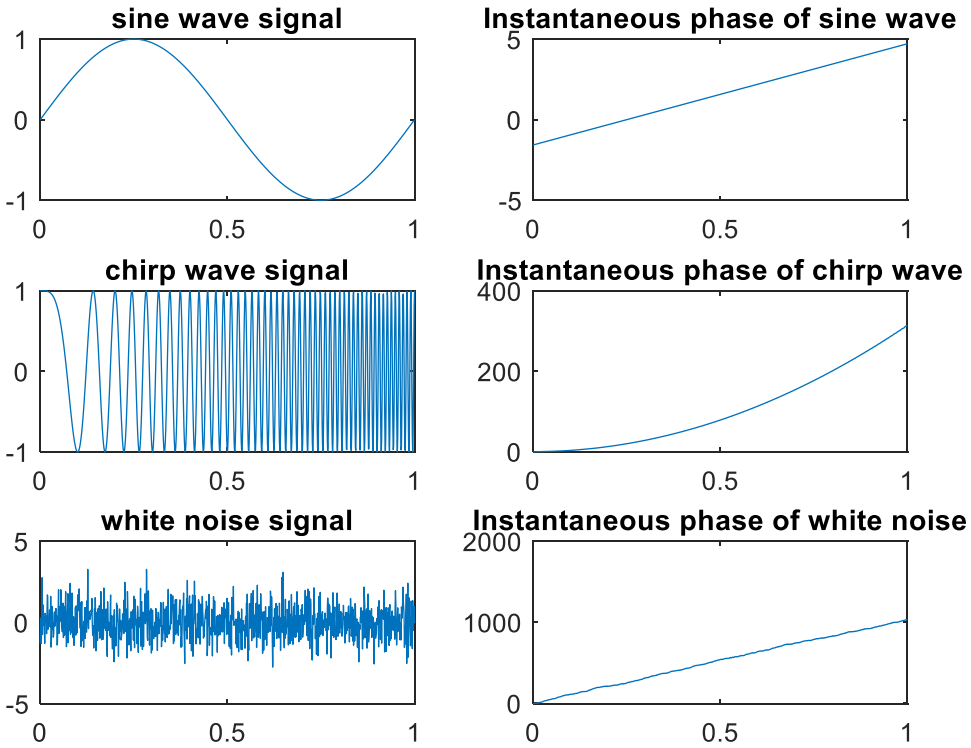
In this exercise, we downsample the chirp signals whose frequency is 0 to 50Hz. The fs in last exercise was 200 and here, we set it to 10 and 20. We found that with a low sampling frequency, it can cause an aliasing and we can't reconstruct the original signal. However, with a higher fs, the result is better.



Then, the Butterworth filters allows to put a cut-off percentage in frequency domain of the signal. With a design of 'low pass band'(anti-aliasing), we can remove all the frequencies higher than the cut-off point. The filtered signal is presented on red light in the figure above. We use the same fs=20Hz, the sampling signal presented in blue curve fits very well the original signal. So, we conclude that the lower frequencies are, the lower fs is needed to avoid aliasing.

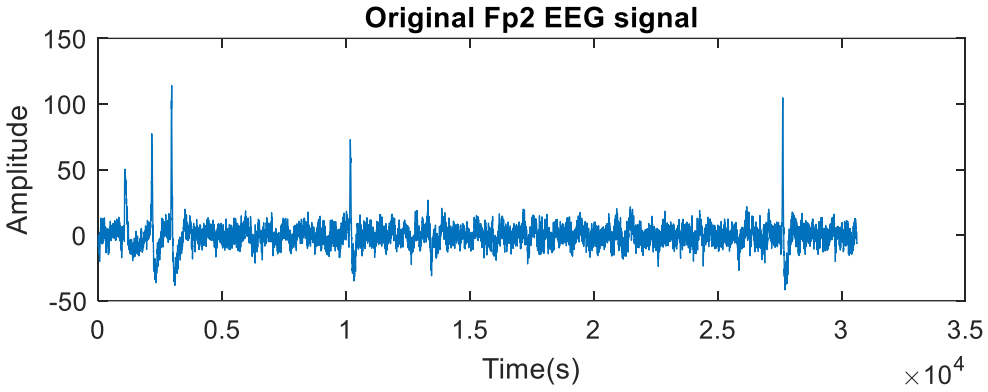
4 Fourier phase and magnitude

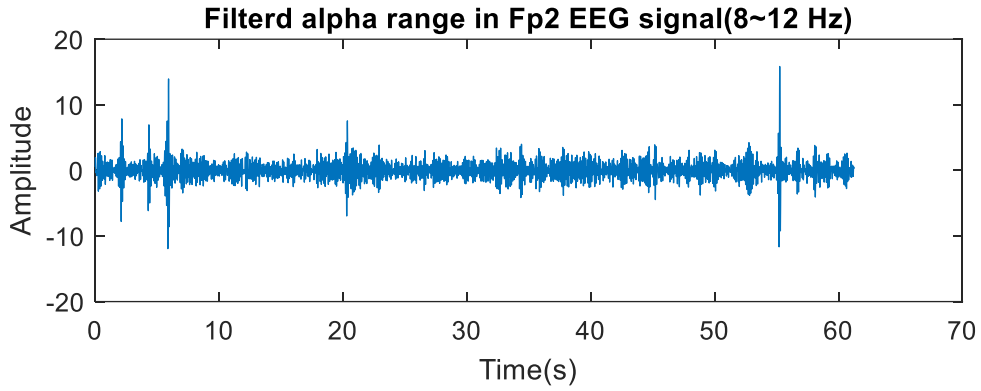
The output of the FFT is a complex vector containing information about the frequency, especially, we can find two symmetrical signals.
Here, we use the Hilbert transform to observe the evolution of the phase along time.

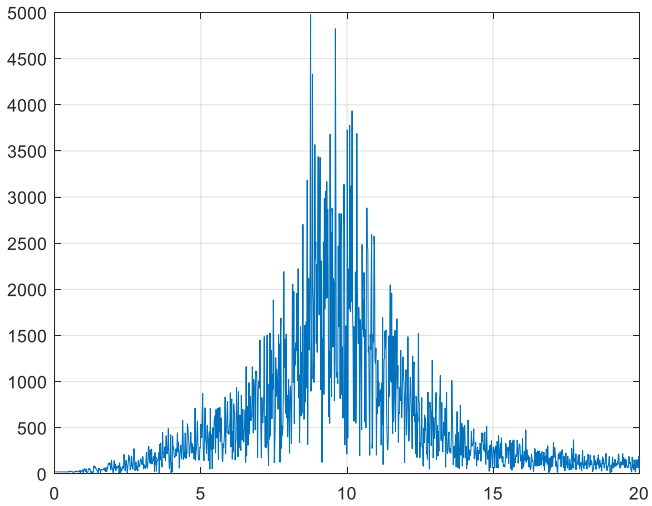


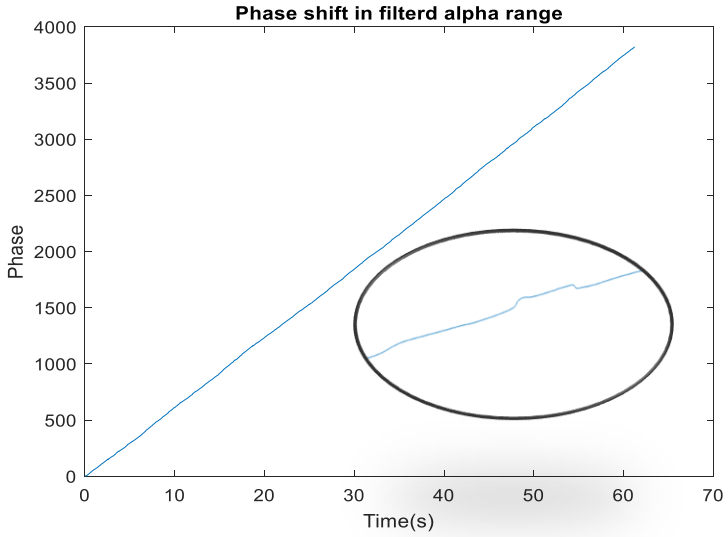
In the phase diagram, we observe that the phase increase.

We get the original Fp2 signal and it's presented in the figure below. Then, we use a Butterworth low-pass filter to get the alpha range which is between 8 and 12 Hz.





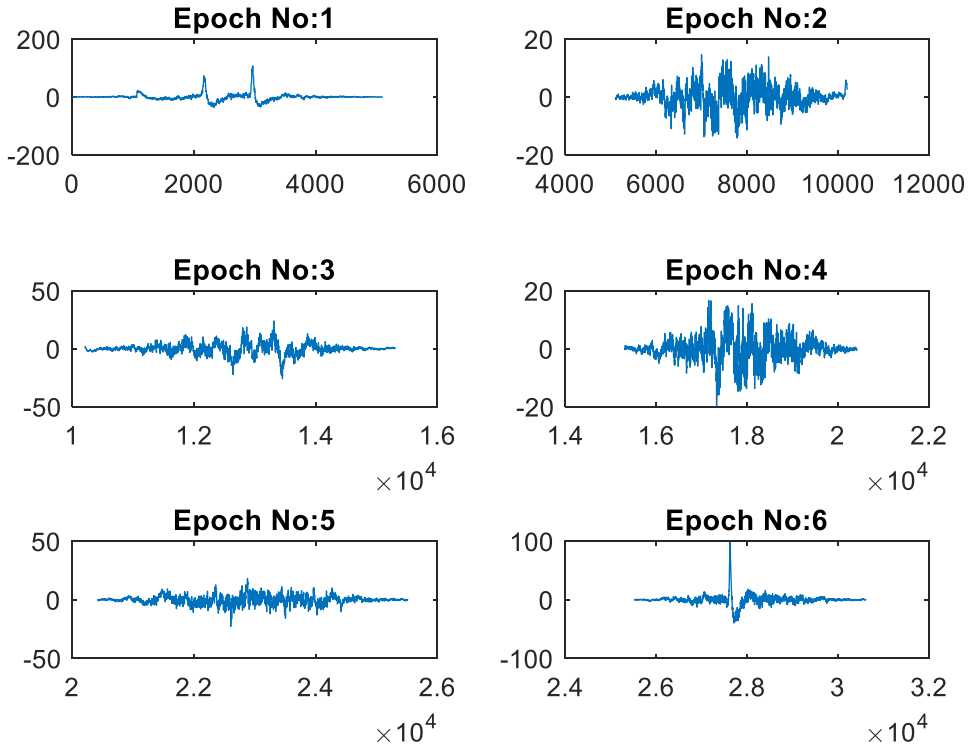




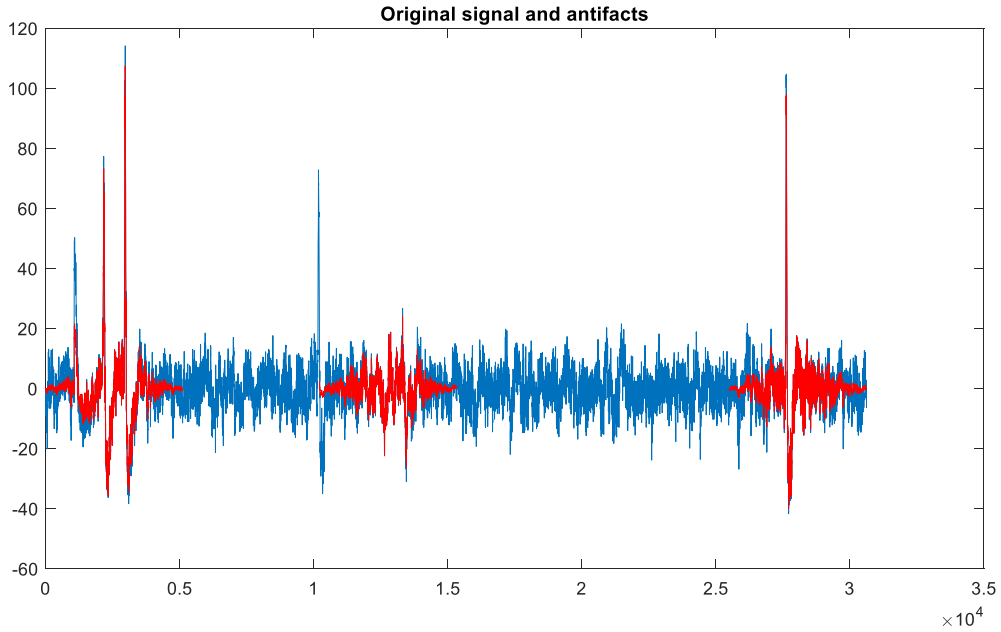
After extracting the alpha rhythm, we verify the result by using a fft function and in the spectrum, we can see that the main frequency is concentrated in 10 Hz.
The phase increase almost linearly along time but there are few variations while zooming.

5 Pre-processing an EEG recording

The EEG signal is a long signal which continues several seconds or minutes. The brain activity is unpredictable due to its non-stationarity. So, we use some smaller windows where the signal is nearly stationary. We analyze them one by one.



We divided the Fp2 signal into 6 epochs by multiplying a Hamming window for each epoch. Hamming window can avoid the high frequencies which could appear at both ends of the window.
The epoch 1 and 6 seem to be an eye blink artifact. Also, the epoch 3 seems to be an EMG. These artefacts are presented in red domain below.



6 Conclusion

This session allows us to learn the basic programming skill in Matlab. I think the difficulty is that I don't have knowledge in the signal processing before so that it's hard for me to understand the meaning of each function and choose their parameters properly. I have try my best to analyze the result in each exercise with the help of various resources.

ANNEX – Matlab Codes

1 FFT WITH MATLAB

```
clc
clear all;

%%
%generate a square function with a period of 2*pi*f
fs=10000;                              %Sampling frequency
t=0:1/fs:1.5;
y=square(2*pi*50*t);
plot(t,y),axis([0 0.2 -1.2 1.2])
xlabel('Time(s)');
ylabel('Amplitude');
title('Square Periodic Wave')

%%
% fft analysis
NFFT=length(y);
Y_FFT=fft(y,NFFT);

window = [];                           %the window function and noverlap as a default
value
noverlap = [];
Y_PWE = pwelch(y,window,noverlap,NFFT,fs);
Y_PERI = periodogram(y,window,NFFT);

magnitudeY_FFT=abs(Y_FFT);             %Magnitude
magnitudeY_PWE=abs(Y_PWE);
magnitudeY_PERI=abs(Y_PERI);

dBmagnitudeY_FFT=10*log10(magnitudeY_FFT);
dBmagnitudeY_PWE=10*log10(magnitudeY_PWE);
dBmagnitudeY_PERI=10*log10(magnitudeY_PERI);
%%
F=(1:(NFFT+1)/2)*fs/NFFT;

figure
subplot(3,1,1)
plot(F,dBmagnitudeY_FFT(1:(NFFT+1)/2));
xlim([0 5000])
xlabel('Frequency(Hz)');
ylabel('DB');
title('FFT in function analysis')
subplot(3,1,2)
plot(F,dBmagnitudeY_PWE);
xlabel('Frequency(Hz)');
ylabel('DB');
title('Pwelch in function analysis')
xlim([0 5000])
subplot(3,1,3)
plot(F,dBmagnitudeY_PERI);
xlim([0 5000])
xlabel('Frequency(Hz)');
ylabel('DB');
title('Periodogram in function analysis')
```

2 Chirp signals

```
Fs = 200;                  %Sampling frequency
t = 0:1/Fs:1;              %Time vector of 1 second
x=chirp(t,0,1,Fs/6);

%%
%Fast fourier transform
NFFT=length(x);
F=((0:1/NFFT:1-1/NFFT)*Fs);%1/NFFT is the frequency resolution
X_FFT=fft(x,NFFT);
Ampli_Xfft=20*log10(abs(X_FFT));

%%
%Draw the graph
figure
subplot(3,1,1)
plot (t,x)
title('Chirp Signal');
xlabel('Time(s)');
ylabel('Amplitude');

subplot(3,1,2)
stem(t,x)
title('The discrete sample representation of Chirp Signal');
xlabel('Time(s)');
ylabel('Amplitude');

subplot(3,1,3)
plot(F,Ampli_Xfft)
axis([0 100 -10 30])
title('Power Spectrum of Chirp Signal');
xlabel('Frequency(Hz)');
ylabel('Power(dB)');
```

3 Downsampling and aliasing

```
%%
Fs = 200;%Sampling frequency
t = 0:1/Fs:1-1/Fs;%Time vector of 1 second
x=chirp(t,0,1,Fs/6);

%10HZ
dx=x(1:20:end);% sample every 20 points
dt=t(1:20:end)*Fs;


%real 20HZ
fs = 20;                   %Sampling frequency
t2 =0:1/fs:1-1/fs;         %Time vector of 1 second
x2=chirp(t2,0,1,Fs/6);
%%
figure
subplot(2,1,1)
plot(x);
hold on
```

```
plot(dt,dx);
title('Sampling fs = 10Hz')

subplot(2,1,2)
plot(x);
hold on
plot(t2*Fs,x2)
title('Sampling fs = 20Hz')

%% Filter the data
x=chirp(t,0,1,Fs/6);
[b,a]=butter(3,4/200,'low');
y=filtfilt(b,a,x);

                        %resample real 20HZ
t2=1:10:length(y);      %Time vector of 1 second
y2=y(1:10:end);

figure
plot(t2,y2)
hold on
plot(y)
grid on
title('Chirp Signal and filtered signal');
xlabel('Time(s)');
ylabel('Amplitude');
```

4 Phase and magnitude

```
%% unwrap(angle(hilbert(x)) phase
% fs = 1kHz
fs = 1000;
t = 0:1/fs:1;

%sampling
sinusN = sin(2*pi*t);
chirpN = chirp(t,0,1,100);
whiteN = wgn(length(t), 1, 0);

SP=unwrap(angle(hilbert(sinusN)));
CP=unwrap(angle(hilbert(chirpN)));
WP=unwrap(angle(hilbert(whiteN)));

%%
figure
subplot(3,2,1)
plot(t,sinusN)
title('sine wave signal')
subplot(3,2,2)
plot(t,SP)
title('Instantaneous phase of sine wave')

subplot(3,2,3)
plot(t,chirpN)
title('chirp wave signal')
subplot(3,2,4)
```

```
plot(t,CP)
title('Instantaneous phase of chirp wave')

subplot(3,2,5)
plot(t,whiteN)
title('white noise signal')
subplot(3,2,6)
plot(t,WP)
title('Instantaneous phase of white noise')


%% load eeg signal
load Fp2_eeg.mat
fs = 500;   % sampled at 500Hz
t = 1/fs:1/fs:(length(Fp2_eeg)/fs);

%% alpha filter 8-12 Hz
f_low = 8;   % low cutoff freq 8Hz
f_high = 12 ;% high cutoff freq 12Hz

[b,a] = butter(1,[f_low f_high]/(fs/2),'bandpass');
Fp2_fitred = filtfilt(b,a,Fp2_eeg);
figure
subplot(2,1,1)
plot(Fp2_eeg)
xlabel('Time(s)')
ylabel('Amplitude')
title('Original Fp2 EEG signal')

subplot(2,1,2)
plot(t,Fp2_fitred)
xlabel('Time(s)')
ylabel('Amplitude')
title('Filterd alpha range in Fp2 EEG signal(8~12 Hz)')

%% Verify the frequency of filtred signal
Nfft=length(Fp2_fitred);
V_alpha=fft(Fp2_fitred);
F=(1:Nfft)*fs/Nfft;
figure
plot(F,abs(V_alpha))
xlim([0 20])
grid on

%% phase variation
figure
Fp2_phase=unwrap(angle(hilbert(Fp2_fitred)));
plot(t,Fp2_phase)
xlabel('Time(s)')
ylabel('Phase')
title('Phase shift in filterd alpha range')
```

5 Pre-processing and EEG recording

```
clc
clear all
load s_EEG.mat
Fp2_eeg=s_EEG.data(2,:);

%% split and extract the signal for each epoch
for i=1:6
   L=round(length(Fp2_eeg)/6);
   w = hamming(L);
   epoch(i,:)= Fp2_eeg((L*(i-1)+1):(L*i)).*(w');
end

%% plot every epoch window
figure
for i=1:6
subplot(3,2,i)
x=((L*(i-1)+1):(L*i));
plot(x,epoch(i,:))
var=strcat('Epoch No:',int2str(i));
title(var)
end

%%
figure
plot(Fp2_eeg)
hold on
i=1;
x=((L*(i-1)+1):(L*i));
plot(x,epoch(i,:),'r')
hold on
i=3;
x=((L*(i-1)+1):(L*i));
plot(x,epoch(i,:),'r')
hold on
i=6;
x=((L*(i-1)+1):(L*i));
plot(x,epoch(i,:),'r')
```