Basics of signal processing (4)

Prof. Febo CINCOTTI, febo.cincotti@uniroma1.it

Dept. of Computer, Control and Management Engineering

(DIAG, Via Ariosto)

Material for this section of the course

- Matlab notebooks available here:
  - https://drive.matlab.com/sharing/d5ad1819-5e50-442a-81fc-6017505d91f3
    - NEng_1920_03_Spectr.mlx (cont'd)
    - NEng_1920_04_Filt.mlx

- Not a textbook, but readings for those who want to have some context:
  - Steven W. Smith
    The Scientist and Engineer's Guide to Digital Signal Processing
    https://www.dspguide.com/pdfbook.htm
- See also:
  - John L Semmlow
    Biosignal and medical image processing (3rd Ed.)
    CRC Press
    Chapter 3-4

Neuroengineering - Spectral analysis

Fourier Analysis, i.e. decomposition of signals into sum of sinewaves. Since each sinewave carries power at exactly one frequency, the decomposition can be used to analyze the signal in the frequency domain. Specifically Discrete Fourier Transform can be used to transform the (time-limited and sampled) time-domain representation of a signal into its (bandwidth limited an sampled) representation in the frequency domain.

Using the DFT to analyze the spectral content of a (stochastic) signal may limit the ability to interpret the results. Specific techniques are commonly in use:

- Zeropadding and windowing are techniques aimed at compensating the effect on the spectrum of a limited number of samples in the time domain (low resolution, sidelobes)
- Power Spetral Density (PSD) can be estimated techniques such as the *averaged periodogram* which limit the variability of the power estimate at the expense of a loss of spectral resolution.

See also:

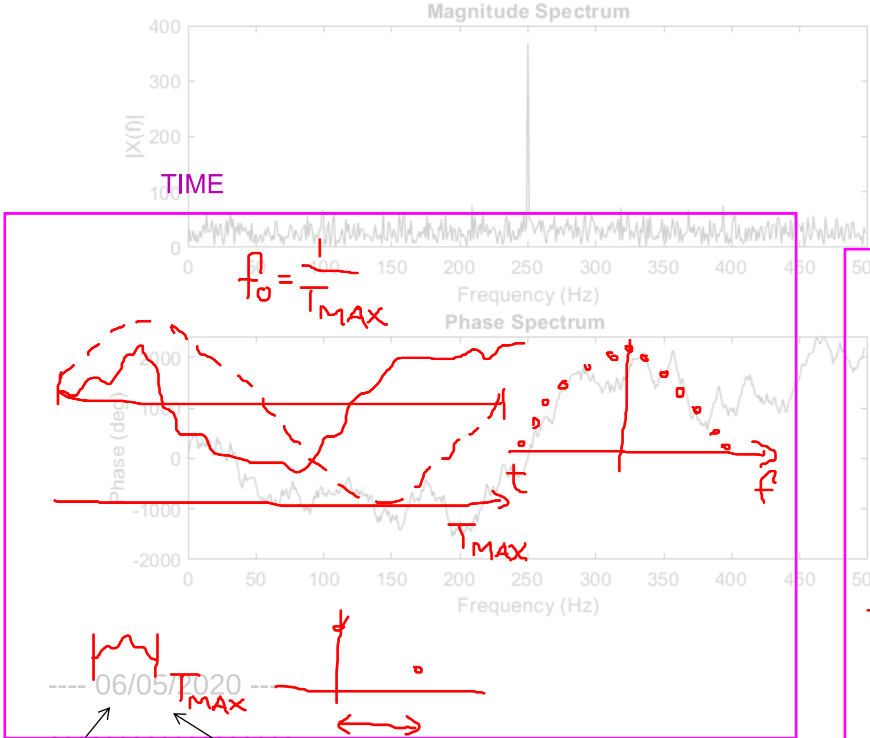
- Semmlow, Biosignal and medical image processing, Chapter 3
- (https://www.dspguide.com/pdfbook.htm, Chapters 6 and 31)

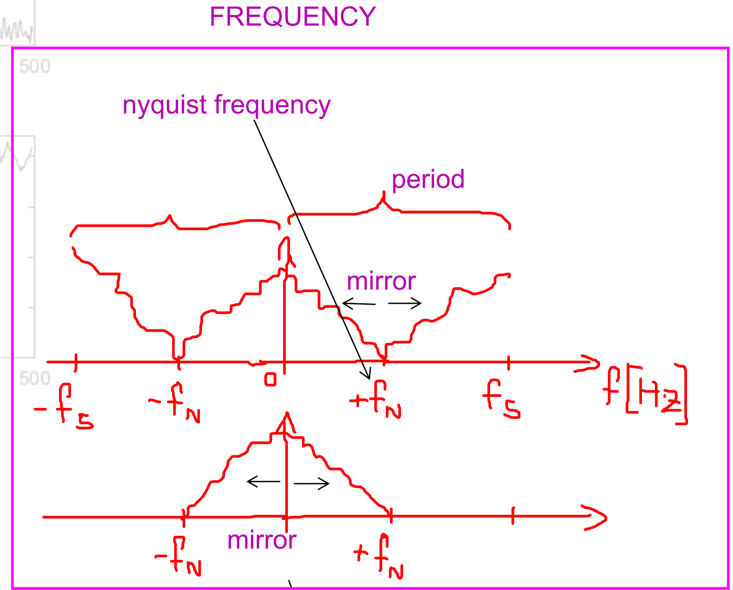
A sinewave is a function of time, with parameters: frequency f, amplitude A, and initial phase ϕ

```
sinewave = @(t, f,A,phi) A * cos(2*pi*f*t + phi);
t = (0:.005:1)';  % seconds

f = 10;  % Hz
A = 0.5;
phi = 0.4 * pi;  % radians

figure(1)
clf
subplot 211
plot(t, sinewave(t, f,A,phi))
xlim([0 1])
ylim([-2 2])
grid on
xlabel("time [s]")
```

$C = Ae^{j\phi} \Leftrightarrow A = |C|, \phi = \angle C$

```
C = A*exp(1j*phi);  % Complex amplitude
subplot 212
plot(real(C), imag(C), '.', 'MarkerSize', 20)

axis square equal
xlim([-2 2])
ylim([-2 2])
grid on; grid minor
title("Complex coefficient, f = " + f + " Hz")
```





if the signal is shorter when you make the transformation, in the frequency domain we have that the distance between 2 adjacent samples in the frequency domain is much longer

we can consider only the posistive part of the plane

Zero padding

Duality in time vs. frequency domain

the question is what is the maximum frequency that appears in the spectrum?

Higher sampling rate in t ⇔ Broader spectrum in f

f_samp = 1 kHz <=> T_s = 0.001 s || f_min = 0 f_max = 1000 Hz (even if useful info only up to 500 Hz)

Longer signal in t ⇔ Higher resolution in f

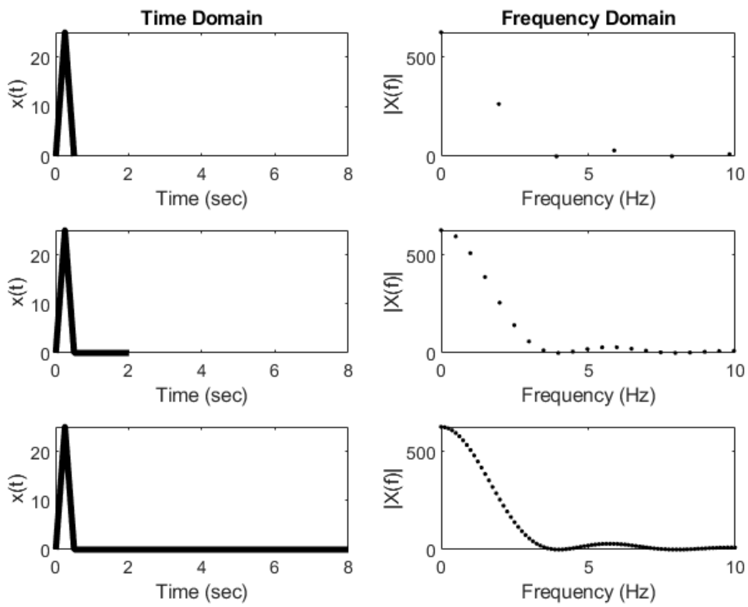
x[t] from t=0 to T=T_Max || Delta_f = 1/T_max

if the signal is short the resolution is poor

Zeropadding (i.e. adding a row of zeros at the end of the signal) allows to increase (artificially) the spectral resolution.

```
% Example 3.4 Generate a 1.0 second wave symmetrical triangle wave.  Make fs = 100 Hz s
% N = 100 points.  Calculate and plot the magnitude spectrum.
% Zero pad so the period is extended to 2 and 8 sec. and recalculate and plot the
% magnitude spectrum.  Since fs = 100 Hz the signal should be padded to 200 and 600 Poi
%
% clear all, close all;
fs = 100;                          % Sample frequencies
N1 = [0 150 750];                  % Padding added to the original 50 point signal
x =[(0:25) (24:-1:0)];             % Generate basic test signal, 50 pts long

figure(7); clf
for k = 1:3
    x1 = [x zeros(1,N1(k))];        % Zero pad signal
    N = length(x1);                 % Data length
    t = (1:N)/fs;
    f = (0:N-1)*fs/N;                % Frequency vector
    subplot(3,2,k*2-1);
```

```
    plot(t,x1,'k', "LineWidth",3);                          % Plot test signal
    xlabel('Time (sec)','FontSize',14);
    ylabel('x(t)','FontSize',14);
    if k == 1
        title('Time Domain','FontSize',14);
    end
    %     xlim([0 t(end)]);
    xlim([0 (length(x)+max(N1)) /fs]);
    subplot(3,2,k*2);
    X1 = abs(fft(x1));                            % Calculate the magnitude spectrum
    plot(f, X1,'.k');                            % Plot magnitude spectrum
    xlabel('Frequency (Hz)','FontSize',14);
    ylabel('|X(f)|','FontSize',14);
    xlim([0 10]);
    % axis([0 10 0 max(X1)*1.2]);
    if k == 1
        title('Frequency Domain','FontSize',14);
    end
end
```

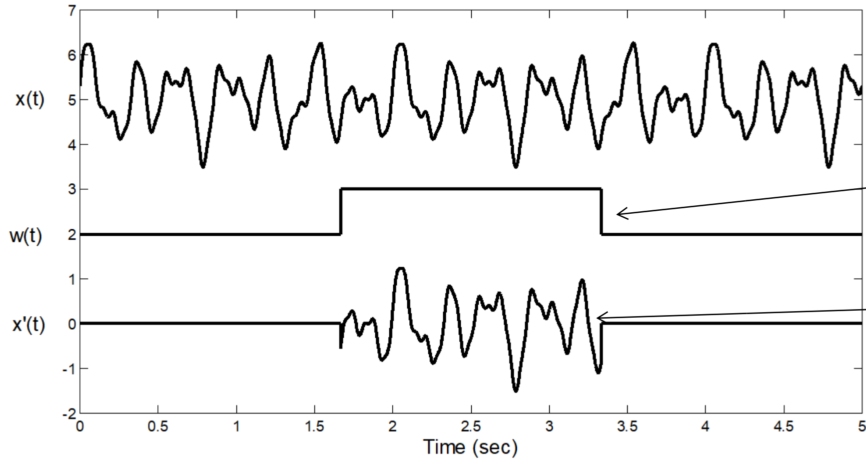


spectrum that has more samples

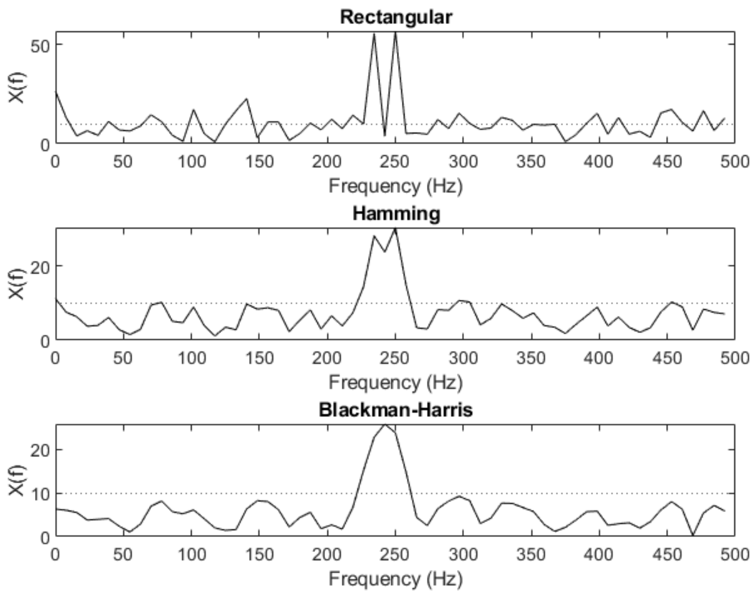
Windowing

Taking a finite span of a signal is mathematically equivalent to multiplying by a rect() function.

Edge effects can be reduced using a tapered windowing function.

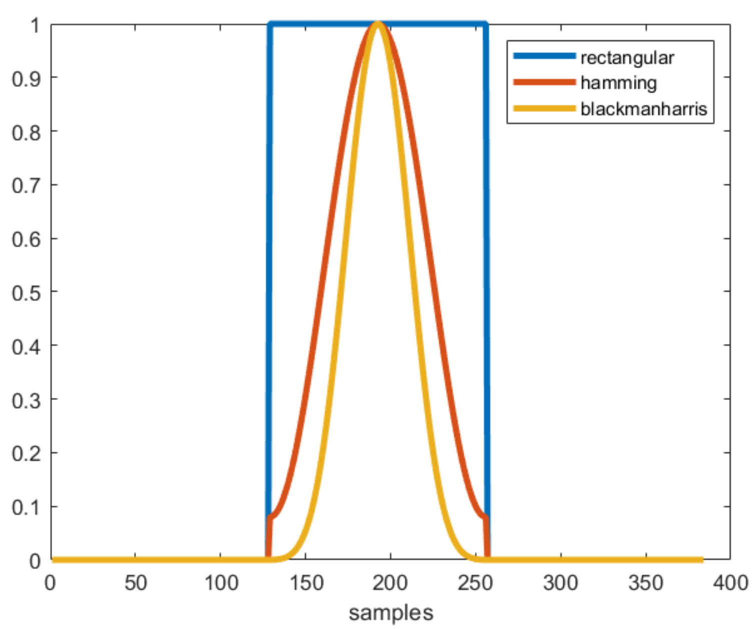



```
% Example 3.5 Application of several window functions
%
% clear all; close all;
fs = 1000;                 % Sampling freq assumed by sig_noise
N = 128;
x = sig_noise([235 250],-3,N);    % Generate data
f = (0:N-1)*fs/N;         % Frequency vector
%
X_mag = abs(fft(x));  % Mag. spect: rect. (no) window

figure(8); clf;
subplot(3,1,1);
plot(f(1:N/2),X_mag(1:N/2),'k');  % Plot magnitude
% semilogy(f(1:N/2),X_mag(1:N/2),'k');  % Plot magnitude
yline(10, ':');
xlabel('Frequency (Hz)','FontSize',14); % Lables
ylabel('X(f)','FontSize',14); % Lables
title('Rectangular','FontSize',14); % Title
%
x1 = x .* hamming(N)';  % Apply Hamming window (Eq. 2.26)
X_mag = abs(fft(x1));  % Mag. spect: Hamming window
subplot(3,1,2);
plot(f(1:N/2),X_mag(1:N/2),'k');  % Plot magnitude
% semilogy(f(1:N/2),X_mag(1:N/2),'k');  % Plot magnitude
yline(10, ':');
xlabel('Frequency (Hz)','FontSize',14); % Lables
ylabel('X(f)','FontSize',14); % Lables
title('Hamming','FontSize',14); % Title
%
x1 = x .* blackmanharris(N)';     % Apply Blackman-Harris (Eq. 2.27)
X_mag = abs(fft(x1));  % Mag. spect: Blackman-Harris window
subplot(3,1,3);
plot(f(1:N/2),X_mag(1:N/2),'k'); % Plot magnitude
% semilogy(f(1:N/2),X_mag(1:N/2),'k'); % Plot magnitude
yline(10, ':');
xlabel('Frequency (Hz)','FontSize',14); % Lables
ylabel('X(f)','FontSize',14); % Lables
title('Blackman-Harris','FontSize',14); % Title
```



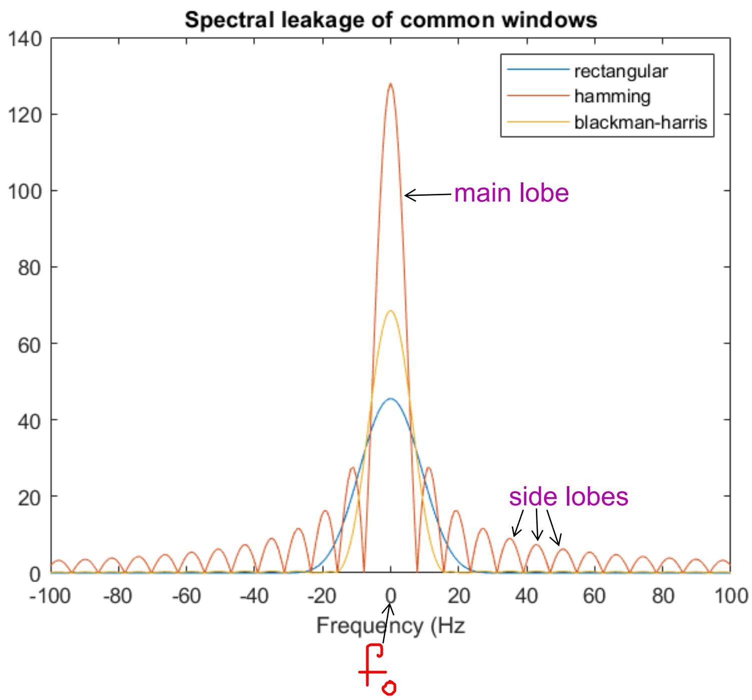
Edge effects determine the spectral *leakage*

```
figure(9); clf
plot([zeros(N,1); rectwin(N); zeros(N,1)], "LineWidth", 3)
hold on
plot([zeros(N,1); hamming(N); zeros(N,1)], "LineWidth", 3)
plot([zeros(N,1); blackmanharris(N); zeros(N,1)], "LineWidth", 3)
hold off
xlabel("samples")
legend(["rectangular" "hamming" "blackmanharris"])
```




```
allwindows = [rectwin(N) hamming(N)  blackmanharris(N)];
```

```
zeropadded_len = N*10;
allwindows(zeropadded_len, :) = 0;  % zero pad
f_windows = fs/(zeropadded_len) * [0:zeropadded_len-1];
f_windows(f_windows>=fs/2) =  f_windows(f_windows>=fs/2) - fs;  % fold frequencies abov
figure(10)
clf
Win_mag = abs(fft(allwindows));
% plot(f_windows(1:zeropadded_len/2),Win_mag(1:zeropadded_len/2, :)); % Plot magnitude
plot(fftshift(f_windows),fftshift(Win_mag)); % Plot magnitude
xlim([-100 100])
xlabel("Frequency (Hz")
title("Spectral leakage of common windows")
legend(["rectangular", "hamming", "blackman-harris"])
```



Power Spectrum

Power spectral density describes the power of the signal in each frequency band of the spectrum.

DISCRETE FOURIER TRANSFORM

$PSD(f) = |DFT(x(t))|^2$

if you could have several estimates of the spectrum, you will go closer and closer to the ideal spectrum, the spectrum of the process which generated the several signals

Spectral averaging - Welch's method

The PSD of a stationary stochastic process (i.e. non-deterministic signal) is an estimate subject to error (variability)

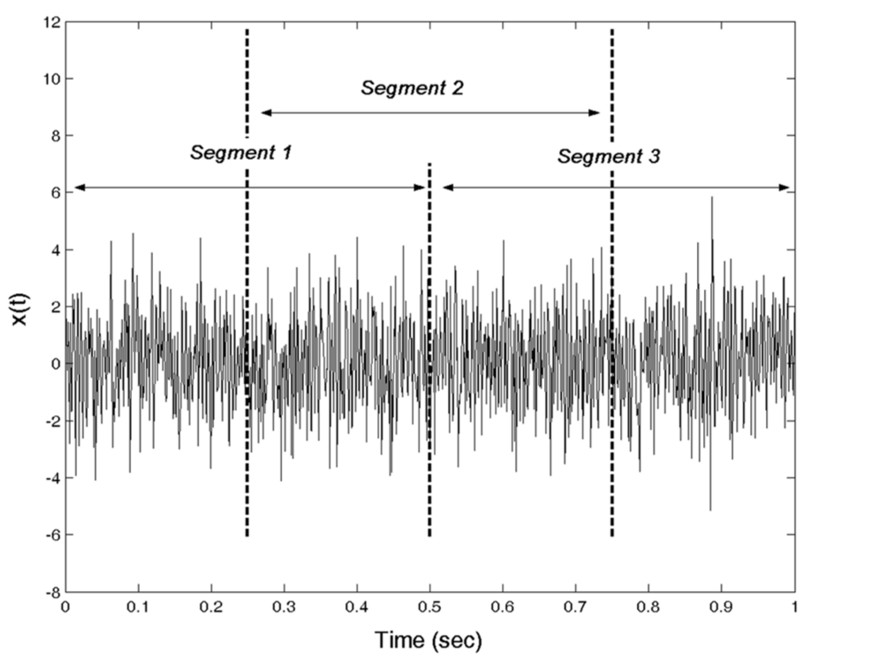
A long signal (many time samples) yield a very high spectral resolution (possibly way beyond the required resolution)

The *averaged periodogram* method trades-off spectral resolution with lower estimate variability. A windowing function can be applied (Welch's method).

0. starting from a long signal
1. Take a short segment of the signal of length nsamples
2. Apply a windowing function
3. Compute the periodogram, i.e. $|DFT(win(t) \cdot x(t))|^2$, and store the resulting PSD
4. Shift nsamples to the right (or a fraction of this amount, if overlapping segments are desired)
5. Repeat steps 1-4 until the end of the signal is reached (num_segments)
6. Average all PSDs

The spectral resolution is determined by the length of the segment (1/nsamples)

The variability of the PSD is reduced by a factor $\sqrt{num\_segments}$

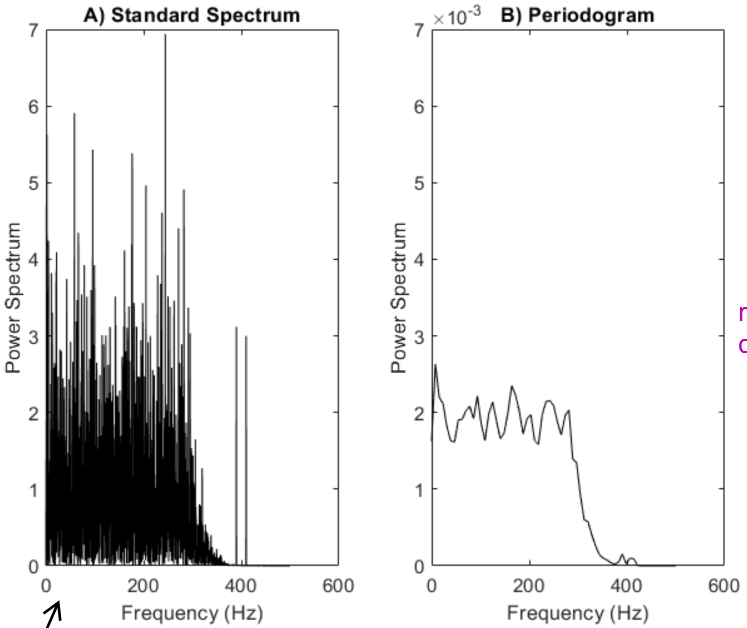



```
% Example 3.7  Investigation of the influence of averaging to improve
%   broadband spectral characteristics in the Power Spectrum.
%  Loads file broadband1 that contains broadband and narrowband signals
% and noise.
%  Calculates the standard Power Spectrum and one obtained using
%   segment averaging.  Use 126 sample segement length with a 99% overlap
%  Assumes the data is variable x in the .mat file and was
%   acquired at a sampling frequency of 1.0 kHz
%
close all; clear all;
```

```
Warning: Error occurred while executing the listener callback for event ObjectBeingDestroyed defined
 for class matlab.ui.Figure:
Error using matlab.ui.container.Menu/horzcat
```

```
load broadband1;                                    % Load data (variable x)
fs = 1000;                                          % Sampling frequency
nfft = 128;                                         % Segment size for averaging
PS = abs((fft(x)).^2)/length(x);                    % Calculate un-averaged PS
half_length = fix(length(PS)/2);   % Find data length /2
f = (0:half_length-1)* fs/(2*half_length);    % Frequency vector for plotting

figure(11); clf;
ax1 = subplot(1,2,1);
plot(f,PS(1:half_length),'k');                      % Plot un-averaged Power Spectrum
xlabel('Frequency (Hz)','FontSize',14); ylabel('Power Spectrum','FontSize',14);
title('A) Standard Spectrum','FontSize',12);
%
[PS_avg,f] = pwelch(x,nfft,nfft-1,nfft,fs);  % Use 99% overlap
ax2 = subplot(1,2,2);
plot(f,PS_avg,'k');                             % Plot averaged Power Spectrum
xlabel('Frequency (Hz)','FontSize',14); ylabel('Power Spectrum','FontSize',14);
ax2.YLim = 7/3*ax2.YLim;
title('B) Periodogram','FontSize',12);
```

estimating on a long signal



resolution is lower, because maybe we have that the distance between frequency sample is much larger

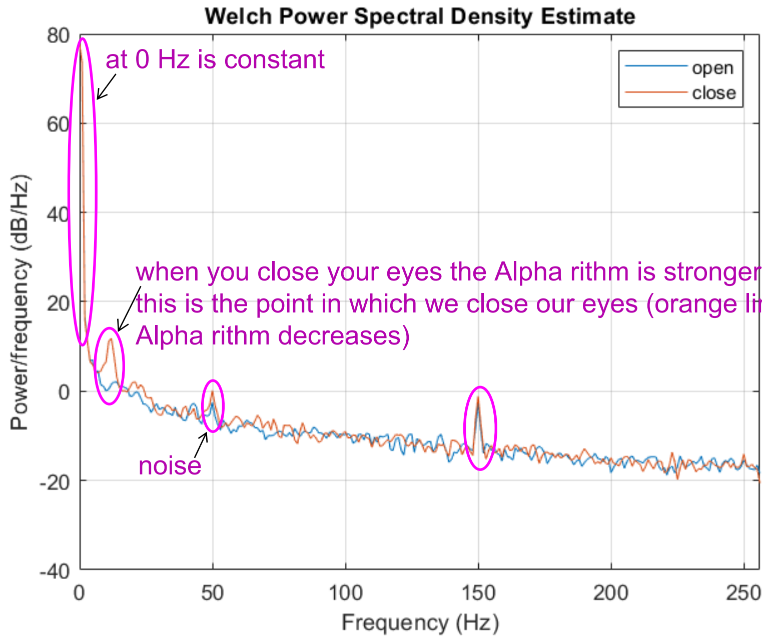
very high resolution but also very noisy

Spectrogram

When the signal is not stationary, PSD should not be used to describe the signal's spectrum. If short segments of the signal can be considered stationary (the signal is pseudo-stationary), the evolution of the spectrum over time can be displayed using a spectrogram.

STFT (short-time fourier transform) is the simplest method to achieve this goal

```
clear
load eo_trial
load ec_trial
fs = 512;

fft_lenght = 1 * fs;  % determines the spectral resolution
overlap_length = 0.5 * fs;
window = hamming(fft_lenght);  % determines the spectral leakage

figure(1)
clf
hold on    eyes open    eyes close
pwelch([eo_trial ec_trial],window,overlap_length,fft_lenght,fs);

hold off
legend(["open", "close"])
```

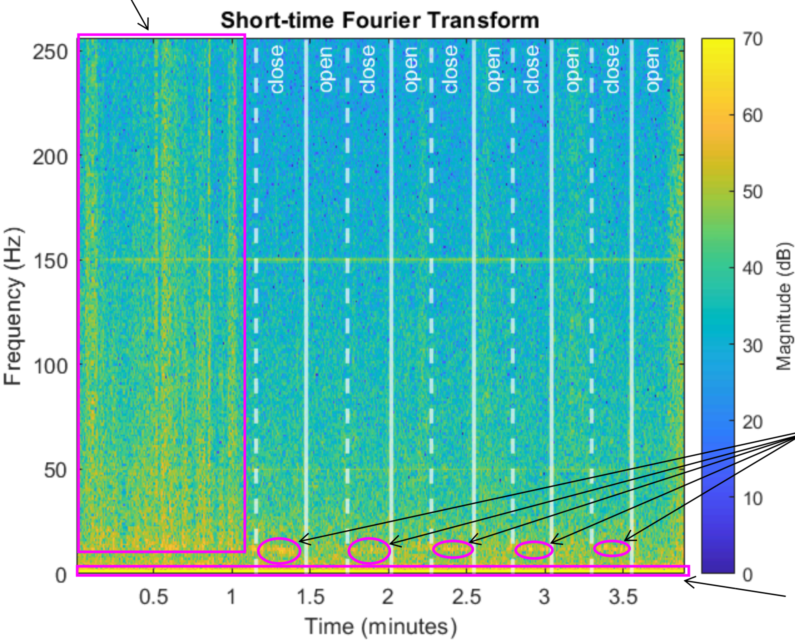


spectrum of those 2 conditions (10 s eyes closed, and 10 s eyes opened)

when you close your eyes the Alpha rithm is stronger then when you open them, so this is the point in which we close our eyes (orange line) and open (blue line->the Alpha rithm decreases)

```
%%
load eeg.mat eeg eo_start_samp ec_start_samp


figure(2)
clf
stft(eeg,fs,'Window',hamming(fs),'OverlapLength',fs/2,'FFTLength',fs);
ylim([0 fs/2])
caxis([0 70])
for smp = eo_start_samp(:)'
    xline(smp/fs/60, 'w-', "open", "LineWidth", 2);
end
for smp = ec_start_samp(:)'
    xline(smp/fs/60, 'w--', "close", "LineWidth", 2);
end
```

there are also vertical yellow lines probably because it is a EMG so
muscolar artifacts, so the subject contracted his muscles



whenever you close your eyes the alpha rithm becomes
stronger (the yellow concentrated zone -> 60/70dB)

low frequency noise (yellow line on the axis of time) or
more probably is the linkage of the constant value.
There is a linkage because we are considering windows
and the yellow line is an effect of this linkage

Internal functions NO

```
function plot_complex_coefficient(C, f)
plot(real(C), imag(C), '.', 'MarkerSize', 20)
axis square equal
xlim([-2 2])
ylim([-2 2])
grid on; grid minor
text(-2,2, " f = " + f + " Hz", "VerticalAlignment", "top", "HorizontalAlignment", "lef
end% function


function [waveform_noise, time, waveform, snr_out] = sig_noise(freqsin, snr, N)
% [waveform_noise, time, waveform, snr_out] = sig_noise(freqsin, snr, npts);
%   Function to generate test data Generates sinusoids in noise.
% Inputs
%    freqsin            is a vector specifing the frequency of sinusoid(s)
%                    assuming a sample frequency of 1 KHz
%      One sinusoid of amplitude 1 is generated for each entry
%    snr                is a vector the SNR values in db of the associated sinusoid
%                       if snr is a scalor it is used for all frequencies
%    npts       number of points in the array
% Outputs
%    waveform_noise     is the output vector containing sinusoids and noise
%    waveform   is the output containing only sinusoids (no noise)
%    time        is the time vector useful in ploting the waveform
%      i.e., plot(time,waveform)
%
%
fs = 1000;    % Assume a sampling freq of 1 kHz
Ts = 1/fs;
time = (0:(N-1))*Ts;
noise = randn(1,N);   % Generate noise and calculate RMS value
rms_noise = sqrt(mean(noise.^2));
if length(snr) < length(freqsin) && length(snr) == 1    % Check SNR vector length
    snr(2:length(freqsin)) = snr(1);    %If a scalor, use this value for all freq.
elseif length(snr) < length(freqsin)
    disp('Error: not enough SNR values')
    waveform_noise = rms_noise;
    return
end
%
for i = 1:length(freqsin)
   freq_scale = freqsin(i) * 2 * pi/fs;
   x = (1:N) * freq_scale;
   snr_n = 10^(snr(i)/20);          % Convert from dB
   A = snr_n * rms_noise * 1.414;   % Determine gain for appropriate SNR
   if i == 1
     component = sin(x) * A;
      waveform = component;
      rms_sig(i) = sqrt(mean(waveform.^2));
   else
      component = sin(x) * A;
      rms_sig(i) = sqrt(mean(component.^2));
      waveform = waveform + component;
```

```
    end
    snr_out(i) = 20 * log10(rms_sig(i)/rms_noise);  % Confirm SNR
end
waveform_noise = waveform + noise;
end% function
```

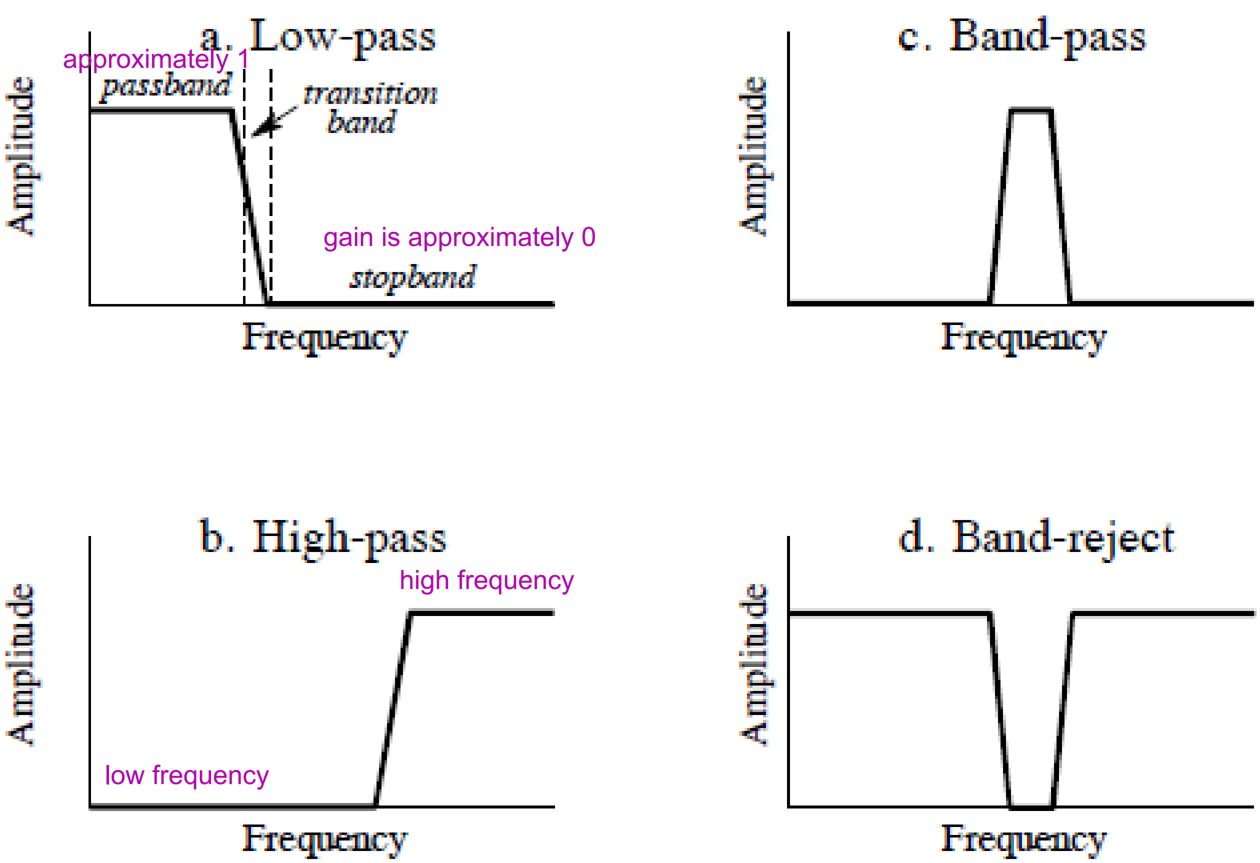
Neuroengineering - Filters

The purpose a filter is to allow some spectral component of a signal to pass (almost) unaltered, while (almost) blocking other spectral components.

See also:

- Semmlow, Biosignal and medical image processing, Chapter 4
- (https://www.dspguide.com/pdfbook.htm, Chapter 14)

The figure belowshows the four basic frequency responses.



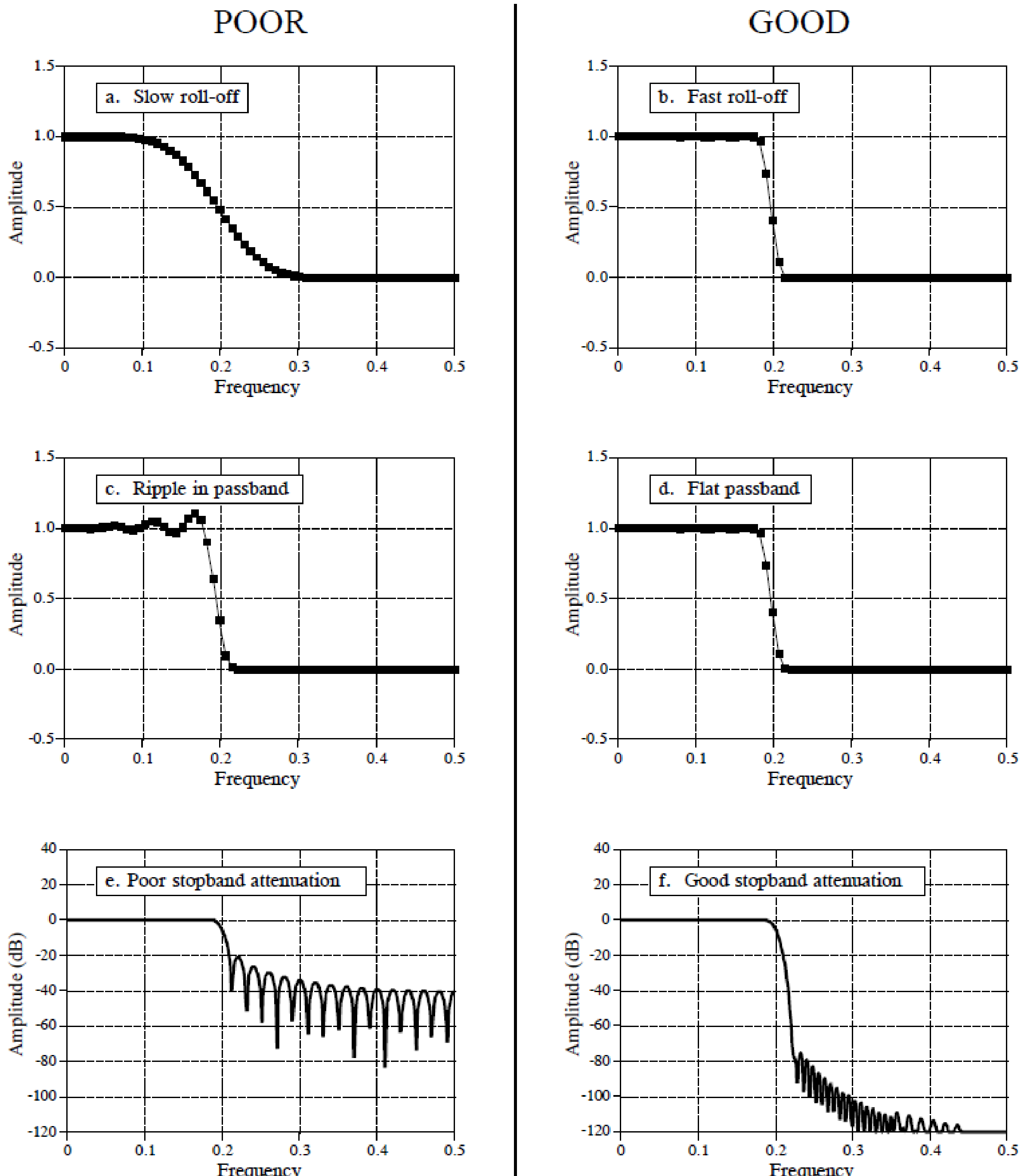
slope of the frequency responce in the transition band

The **passband** refers to those frequencies that are passed, while the **stopband** contains those frequencies that are blocked. The **transition band** is between. A fast **roll-off** means that the transition band is very narrow. The

frequency in which there is a change

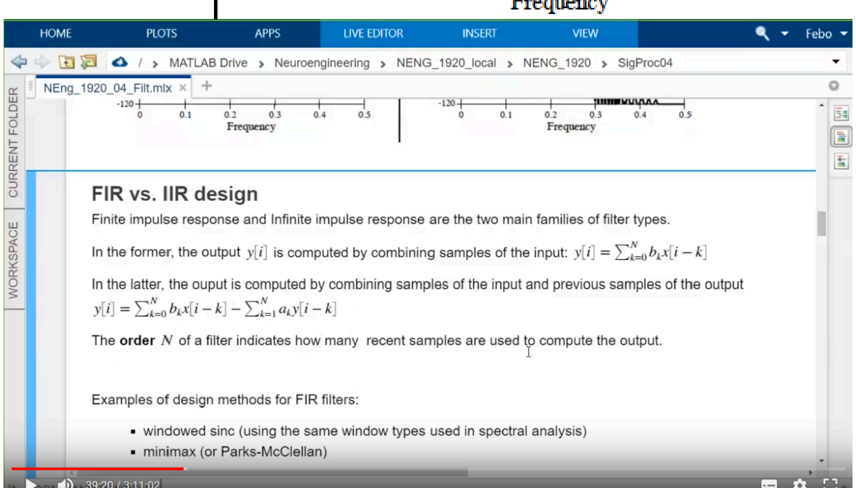
division between the passband and transition band is called the **cutoff frequency**. In analog filter design, the cutoff frequency is usually defined to be where the amplitude is reduced to 0.707 times the imput (i.e., $-3dB$). Digital filters often specify different attenuation at the cutoff frequency, depending on the synthesis process.

The figure below shows three parameters that measure how well a filter performs in the frequency domain. To separate closely spaced frequencies, the filter must have a **fast roll-off**, (top row). For the passband frequencies to move through the filter unaltered, there must be **no passband ripple**, (middle row). Lastly, to adequately block the stopband frequencies, it is necessary to have **good stopband attenuation**, (bottom row).

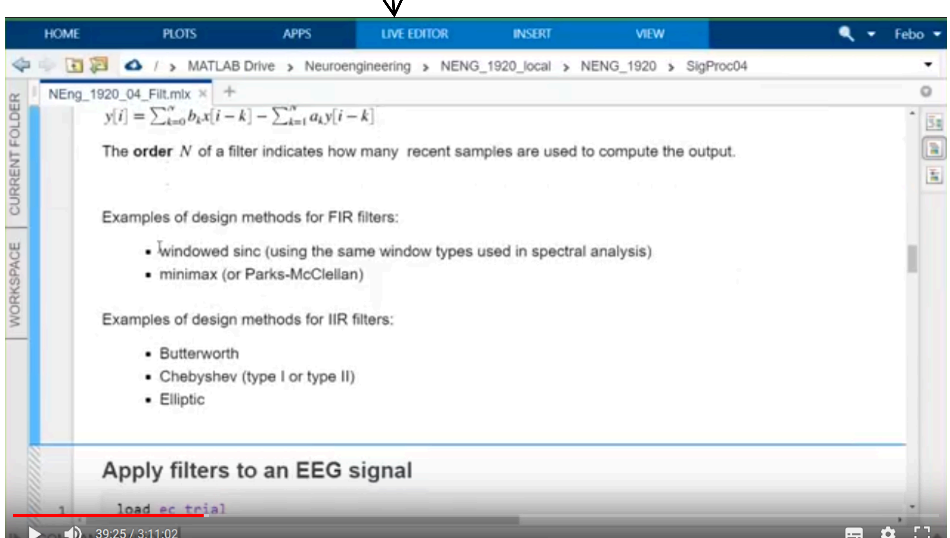


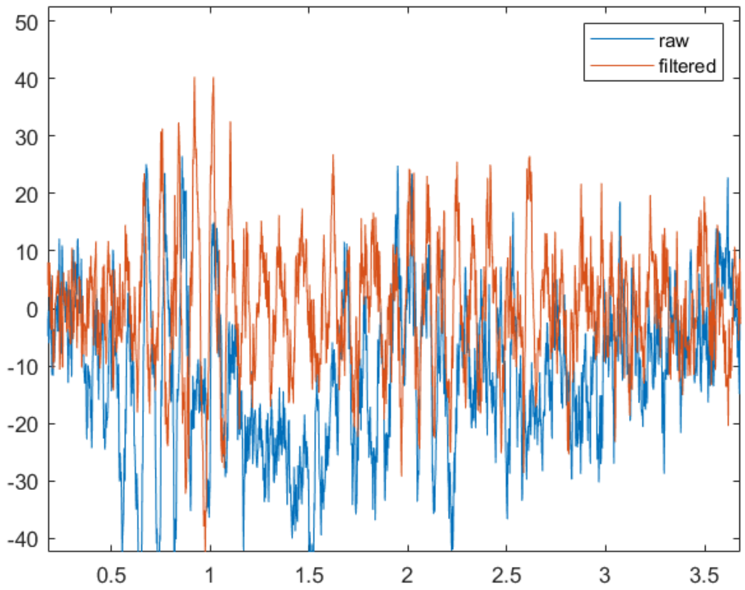
FIR vs. IIR design ...

this filter not distor
the signal

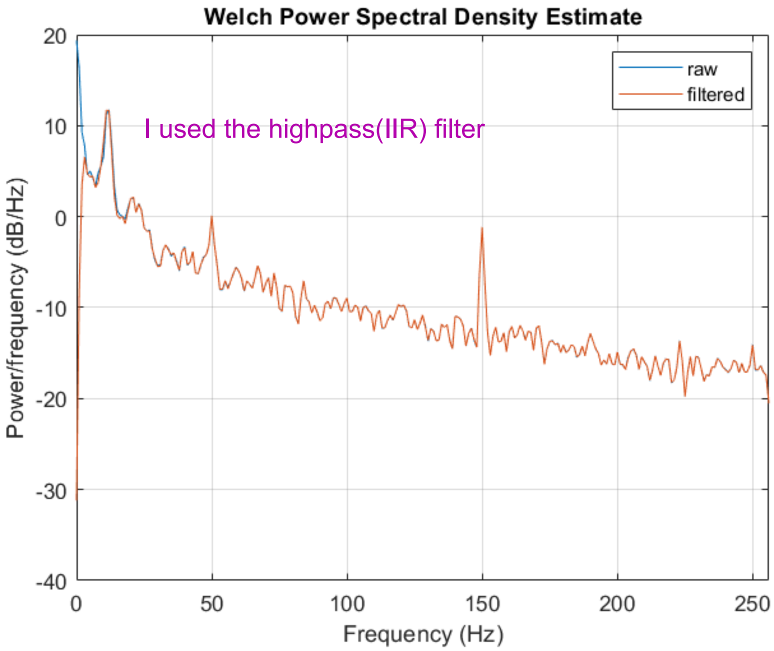


```
load ec_trial
fs = 512;
num_samp = length(ec_trial);
time = (0:num_samp-1) / fs;
ec_trial = ec_trial - mean(ec_trial);  % fix overall baseline

RESPONSE = "highpass"; here there is also other options

ORDER = 256;
FC_LOW = 1;  % Hz
FC_HIGH = 30;  % Hz             different filters applied at the same signal
F_NOTCH = 50;  % Hz
switch RESPONSE
    case "lowpass"
        filt = designfilt('lowpassiir', 'FilterOrder', ORDER, 'HalfPowerFrequency', FC_
                                                                  FC_HIGH, 'SampleRate', fs);
    case "highpass"
        filt = designfilt('highpassiir', 'FilterOrder', ORDER, 'HalfPowerFrequency', FC
                                                                  FC_LOW, 'SampleRate', fs);
    case "bandpassiir"
        filt = designfilt('bandpassiir', 'FilterOrder', ORDER, 'HalfPowerFrequency1', F
                                                                  FC_LOW, 'HalfPowerFrequency2', ...);
    case "bandpassfir"
        filt = designfilt('bandpassfir', 'FilterOrder', ORDER, 'CutoffFrequency1', FC_L
                                                                  FC_LOW, 'CutoffFrequency2', ...);
    case "notch"
        filt = designfilt('bandstopiir','FilterOrder',ORDER, 'HalfPowerFrequency1',F_NC
end                                                               F_NOTCH-.5, 'HalfPowerFrequency2', ...);


% fvtool(filt)

ec_trial_filtered = filter(filt, ec_trial);

                              filter that you have just syntetized

figure(2)
clf
plot(time, [ec_trial, ec_trial_filtered])
legend ("raw", "filtered")
```

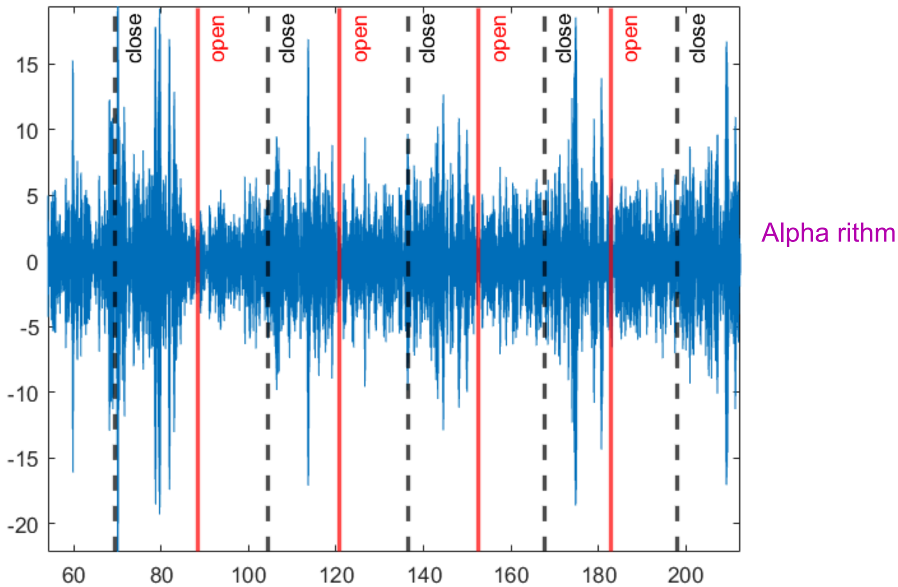





```
fft_lenght = 1 * fs;  % determines the spectral resolution
overlap_length = 0.5 * fs;
window = hamming(fft_lenght);  % determines the spectral leakage

figure(2)
clf
hold on
pwelch([ec_trial ec_trial_filtered],window,overlap_length,fft_lenght,fs);

hold off
legend(["raw", "filtered"])
```




```
load eeg.mat eeg ec_start_samp eo_start_samp
fs = 512;
num_samp_eeg = length(eeg);
time_eeg = (0:num_samp_eeg-1) / fs;
ec_trial = eeg - mean(eeg);  % fix overall baseline

alpha_filt = designfilt('bandpassfir', 'FilterOrder', 256, 'CutoffFrequency1', 8, 'Cuto
alpha = filtfilt(alpha_filt, eeg);  % filtfilt introduces no delay

figure(3)
clf
plot(time_eeg, alpha)
for smp = eo_start_samp(:)'
    xline(smp/fs, 'r-', "open", "LineWidth", 2);
end
for smp = ec_start_samp(:)'
    xline(smp/fs, 'k--', "close", "LineWidth", 2);
end
```


close
open
close
open
close
open
close
open
close
15
10
5
0
-5
-10
-15
-20
60
80
100
120
140
160
180
200
Alpha rithm