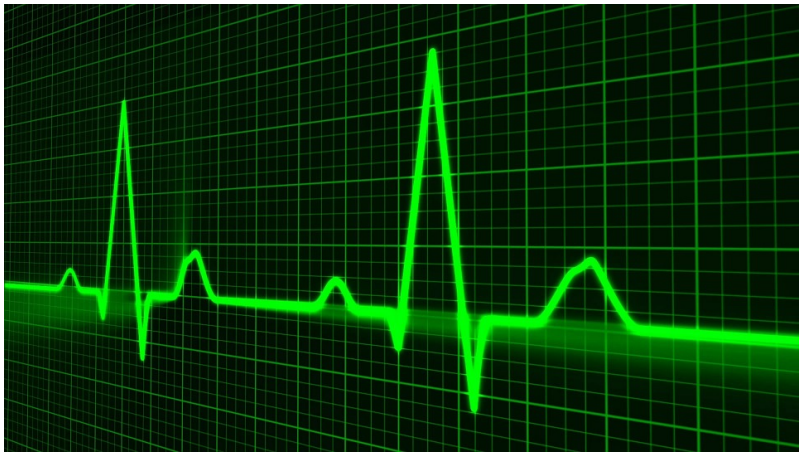
Signals and Systems
Lab 3: Filtering of 1D signals



Data Science and Engineering, 2019-2020

Contents

1 Introduction

This lab focuses on digital filters, their design process and the filtering of one-dimensional signals. We will work with real signals from the database ECG-ID. This database contains 310 ECG recordings, obtained from 90 people. Each recording contains an ECG lead I, recorded for 20 seconds, digitized at 500 Hz with 12-bit resolution over a nominal ±10 mV range.

An ECG signal is an electrical signal produced by heart activity and captured by an ECG sensor. It is characterized by 5 peaks and valleys named P, Q, R, S and T. The properties of these peaks and valleys, their speed of occurrence, heights and widths provide information on cardiologists about the conditions of the heart.

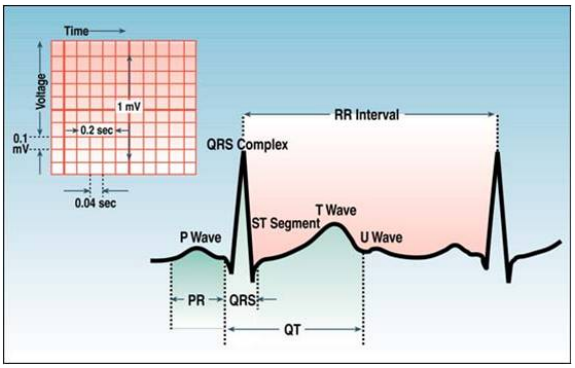


Figure 1: ECGs intervals and waves. Image from https://ecg.utah.edu/lesson/1

An ECG signal occupies the frecuency range from 0.01 to 250 Hz (diagnostic-quality ECG). Often, the signal captured by the sensor is contaminated by the interference of the power line (50 Hz in Europe or 60 Hz in the USA) and its harmonics. In addition, the signal may contain a superimposed noise due to muscle movements that may occur at approximately 40 Hz or more. Finally, there may be a baseline drift mainly caused by the movement and respiration of the patient. All this may render the captured signal unusable if no pre-processing is done.

An enhanced ECG signal can be obtaining by eliminating the interference from the power line and its main harmonics. The enhanced ECG signal obtained can serve for general ECG signal

analysis. To remove the baseline drift and to filter muscle noise, filtering alternatives that allow to pass the band from 0.25 to 40 Hz can be used[1]. Notice that the obtained signal may not be longer valid for general ECG applications, as the ECG signal has information from 0.01 to 250 Hz. The usefulness of the signal enhancements techniques will depend, of course, on the application.

In this lab, we will design and apply linear filters to ECGs from ECG-ID. As you will see, some of them are rather noisy. Start your code as follows:

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.fftpack import fft
from scipy import signal

def plotzp(zeros, poles):
    ax1 = plt.subplot(111)
    circle = plt.Circle((0, 0), 1, color='black', fill=False)
    ax1.add_artist(circle)
    plt.plot(np.real(zeros), np.imag(zeros), 'ob')
    plt.plot(np.real(poles), np.imag(poles), 'xr')
    plt.legend(['Zeros', 'Poles'], loc=2)
    plt.title('Pole / Zero Plot')
    plt.ylabel('Real')
    plt.xlabel('Imaginary')
    plt.xlim([-1.2, 1.2])
    plt.ylim([-1.2, 1.2])
    ax1.set_aspect(1.0)  # make aspect ratio square
    plt.grid()

def plotresp(freq, H):
    plt.subplot(211)
    plt.plot(freq, 20*np.log10(np.abs(H)))
    plt.ylabel('Gain (dB)')
    plt.grid()
    plt.subplot(212)
    angles = np.unwrap(np.angle(H))
    plt.plot(freq, angles)
    plt.ylabel('Angle (radians)')
    plt.xlabel('Frequency')
    plt.grid()
```

Important note: Remember that, if you are using Spyder, separating sections with `#%%` will allow you to execute each section independently. To display figures in a separated window (so that interactions can be enabled) click Tools, Preferences, Ipython Console, Graphics and under Graphics Backend select "automatic" instead of "inline".

For interactive plots with Jupyter Notebooks execute the command `%matplotlib notebook`.

Before starting the lab session, you may need to review some of the contents included in section 2.1. We will use the following functions from the `scipy` package: `signal.remez` to design optimal FIR filters and `signal.iirdesign` to design IIR filters (see section 3 for examples of usage).

[1] Li Tan *Digital Signal Processing, Fundamentals and Applications*, Elsevier, 2008. pp. 370-377.

2 Filter concepts review

2.1 Transfer function and other basic concepts

The description of a system by a finite differences equation allows expressing the output of the system as a weighted sum of the current input and previous values of the input and output:

$$y[n] = \sum_{k=0}^{Q} b_k x[n-k] - \sum_{k=1}^{P} a_k y[n-k] \tag{1}$$

By applying the z transform on both sides of eq. 1, the transfer function, $H(z)$, of the system can be obtained:

$$Y(z) = \sum_{k=0}^{Q} b_k X(z) z^{-k} - \sum_{k=1}^{P} a_k Y(z) z^{-k} \Longrightarrow H(z) = \frac{Y(z)}{X(z)} = \frac{b_0 + b_1 z^{-1} + ... + b_Q z^{-Q}}{1 + a_1 z^{-1} + ... + a_P z^{-P}} \tag{2}$$

The values of z that make zero $H(z)$ (i.e. the roots of the numerator) are called **zeros** of $H(z)$, while those values of z that make $H(z)$ infinite (roots of the denominator) are called **poles**. If the coefficients $\{b_k\}$ are real, the zeros are real or appear in complex conjugated pairs. The same is true for the poles if the denominator coefficients $\{a_k\}$ are real. Finally, the **order** of the transfer function and, by extension, of the system is defined as $\max(P, Q)$.

The transfer function, $H(z)$, is the z transform of the impulse response, $h[n]$, of the system:

$$H(z) = TZ\{h[n]\} = \sum_{n=-\infty}^{\infty} h[n] z^{-n}. \tag{3}$$

For causal systems $h[n] = 0$ for $n < 0$. Therefore, the region of convergence (ROC) of $H(z)$ is the exterior of a circle outside the out-most pole.

If $P = 0$ all the poles are at $z = 0$ and the impulse response has length equal to Q (FIR system, of *Finite Impulse Response*). On the other hand, if $P \neq 0$ the system has poles outside the origin and the impulse response has infinite length (system IIR, *Infinite Impulse Response*).

Evaluating $H(z)$ in $z = e^{j2\pi F}$ (circle of unit radius in the complex plane, also called unitary circle) we obtain the Fourier transform of the impulse response:

$$H(F) = H(z)|_{z=e^{j2\pi F}} \tag{4}$$

For a system to be stable, $h[n]$ must fulfill that $\sum_{n=-\infty}^{\infty} |h[n]| < \infty$, which is equivalent to say that $\sum_{n=-\infty}^{\infty} |h[n]||z|^{-n}|_{|z|=1} < \infty$. Therefore, the ROC of its transfer function $H(z)$ must include the unit circle $|z| = 1$, i.e. the frequency response, $H(F)$, exists.

In summary, for a real, causal and stable filter, the zeros and poles are real or appear in complex conjugated pairs, the ROC includes the unitary circle and all the poles are inside the unit circle of the z-plane, i.e., the magnitudes of all poles are smaller than 1.

2.2 Notch filters

For a system to eliminate a frequency component F_0, it is enough for its $H(z)$ to present a zero in $z = e^{j2\pi F_0}$, so

$$H(z)|_{z=e^{j2\pi F_0}} = 0 \tag{5}$$

For the coefficients $\{b_k\}$ to be real, the filter must have another zero in $z = e^{-j2\pi F_0}$.

In addition, for the system to alter as little as possible the rest of frequency components, a pole next to each zero is usually placed to compensate for the effect of the zero.

Summarizing, to cancel a certain frequency F_0, we can use a system whose $H(z)$ has two conjugated complex zeros on the unitary circle: $c = e^{\pm j\phi}$, with $\phi = 2\pi F_0$, and two conjugate complex poles of equal phase that the zeros and next[2] to the unitary circle: $p = r \cdot e^{\pm j\phi}$,

$$H(z) = k\frac{(1 - cz^{-1})(1 - c^* z^{-1})}{(1 - pz^{-1})(1 - p^* z^{-1})} \tag{6}$$

Finally, the constant k is adjusted so that $H(z)|_{z=1} = 1$, i.e. $H(F = 0) = 1$. Therefore,

$$k = \frac{1 - 2r\cos\phi + r^2}{2 - 2\cos\phi} \tag{7}$$

This type of system is known as ***notch filter***.

2.3 Digital filters design

In general, for many applications it will not be enough to cancel a single frequency, but a certain frequency band, while allowing other bands to pass. Depending on the selected band/s filters are classified in low-pass, high-pass, band-pass and stop-band filters.

Ideally, the frequency response of a filter, $H(F)$, has a constant modulus and a linear phase in the pass band, and is zero outside the passband:

$$H(F) = e^{-j2\pi Fm} \text{in the pass band, } H(F) = 0 \text{ outside the pass band} \tag{8}$$

Denoting the part to be preserved from the input as $x[n]$, the Fourier transform of the output of the ideal filter can be expressed as $Y(F) = H(F)X(F) = X(F)e^{-j2\pi Fm}$. Therefore, the output will be no more than a version, delayed m samples, of the signal of interest, $y[n] = x[n - m]$.

If the frequency response in the pass band is not ideal, the signal undergoes a transformation called **linear distortion**. The distortion can be of amplitude (if the modulus of $H(F)$ is not constant in the pass band) or of phase (if the phase of $H(F)$ is not linear in the pass band).

In practice, obtaining filters with linear phase is possible, but unfortunately it is impossible to obtain a filter with $|H(F)|$ constant in the pass band, $|H(F)| = 0$ in the attenuated band and instantaneous transition between the pass and attenuated band, since this would require an infinite number of coefficients.

[2] If r is greater than 1, when doing the inverse transformation of $H(z)$ you would get an impulse response that tends to infinity with increasing time, that is, the system would be unstable.

As a result, the filters are designed so that their response is as close as possible to the ideal response, admitting a certain **tolerance** for the values of $|H(F)|$ (amplification) in the pass and attenuated bands. In addition, the amplification will change gradually so there will be a **transition band** between both bands, as seen in figure 2.

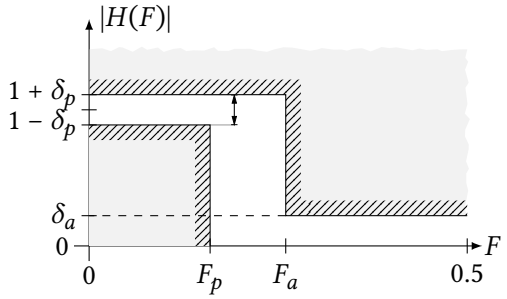


Figure 2: Especification of the $|H(F)|$ of a low-pass filter

When working with filters in the frequency domain, it is common to work with attenuation. The attenuation is measured in decibels (dB) and is logarithmically related to the inverse of $|H(F)|$ and a reference value:

$$\alpha(F) = 20 \log_{10} \frac{H_{ref}}{|H(F)|} \tag{9}$$

Taking as a reference the maximum value of $|H(F)|$, $1 + \delta_p$, the tolerance $\pm\delta_p$ translates into a maximum attenuation in the pass band (with respect to the maximum value)

$$\alpha_p = 20 \log_{10} \frac{1 + \delta_p}{1 - \delta_p} = 20 \log_{10}(1 + \delta_p) - 20 \log_{10}(1 - \delta_p) \tag{10}$$

while the tolerance δ_a corresponds to a minimum attenuation in the attenuated band (with respect to the maximum value)

$$\alpha_a = 20 \log_{10} \frac{1 + \delta_p}{\delta_a} = 20 \log_{10}(1 + \delta_p) - 20 \log_{10} \delta_a \tag{11}$$

The discrimination and selectivity of a filter are defined as:

$$K_d = \sqrt{\frac{10^{\frac{\alpha_p}{10}} - 1}{10^{\frac{\alpha_a}{10}} - 1}}; \qquad K_s = \frac{f_p}{f_a} \text{ (lowpass filter)} \tag{12}$$

.

Both K_d and K_s are always between 0 and 1. K_d close to 0 corresponds to a highly discriminant filter. K_s close to 1 corresponds to a highly selective filter. In practice, it is sought not to exceed the requirements of selectivity and discrimination essential for the application in order to not unnecessarily increase the order of the filter.

The way to approximate the response of a filter to the ideal response differs slightly for FIR or IIR filters, as described next.

2.3.1 FIR filters design

A FIR filter has an impulse response of finite length $L = Q + 1$, where Q is the order of the filter. The output signal at time n is obtained as follows:

$$y[n] = \sum_{k=0}^{Q} b_k x[n-k] \Longrightarrow H(z) = \sum_{k=0}^{Q} b_k z^{-k} \tag{13}$$

Since all the poles are in $z = 0$, an FIR filter can always be made causal and stable. In addition, a FIR filter with an impulse response with odd or even symmetry, i.e. $h[n] = \pm h[Q-n]$, will have linear phase. In that case, the slope of the phase will be $-0.5Q$, so the output will have a delay of $0.5Q$ samples (it may be a non-integer value) with repect to the input.

To design a FIR filter, different alternatives can be used, which aim to obtain a response as close as possible to the ideal response, $H(F)_{ideal} = e^{-j2\pi F\frac{Q}{2}}$ in the passband (0 outside):

- Windowing of the ideal impulse response:

$$h_{ideal}[n] = FT^{-1}\{H_{ideal}(F)\} \Longrightarrow h[n] = h_{ideal}[n] \cdot w_N[n]$$

- Sampling the ideal frequency response:

$$h[n] = DFT_N^{-1}\{H_{ideal}(F)|_{F=\frac{k}{N}} \quad k = 0, ..., N-1\}$$

- **Remez Algorithm**: Given the number of coefficients and the frequencies that delimit the pass and attenuated bands, the **Remez algorithm** computes the coefficients that minimize the maximum absolute error between the desired frequency response and the realized one in the specified bands. (Weighting factors can be introduced to prioritize the minimization of δ_p or δ_a).

  It can be shown that to satisfy certain restrictions of selectivity and discrimination, there is no design of a lower order than that which has a constant amplitude ripple behavior, which is why these filters are said to be optimal. Therefore, the Remez algorithm provides filters with constant amplitude ripple in the pass and attenuated bands.

In applications where a filter with linear phase is required, the best solution is to design an optimal FIR filter. It is the solution that best fits the specifications of selectivity and discrimination with minimum order.

2.3.2 IIR filters design

The output for an IIR filter is calculated as:

$$y[n] = y[n] = \sum_{k=0}^{Q} b_k x[n-k] - \sum_{k=1}^{P} a_k y[n-k] \Longrightarrow H(z) = \frac{Y(z)}{X(z)} = \frac{b_0 + b_1 z^{-1} + ... + b_Q z^{-Q}}{1 + a_1 z^{-1} + ... + a_P z^{-P}} \tag{14}$$

The design of IIR filters from α_p dB (maximum attenuation in the passband), α_a dB (minimum attenuation in the stopband), F_p, F_a is based on a transformation of the approximations used for

analog filters that give rise to different types of filters. The most common approaches are Butterworth, Chebishev type I, Chebishev type II (also called Chebishev inverse) and elliptical.

Chebishev type I and elliptical approaches present a constant ripple in the pass band and Chebishev type II in the attenuated band. The figure 3 visually displays the module of the frequency response of the four filter types for similar parameters of order, low pass characteristic, etc. All the filters in figure 3 have order $n = 4$ and the same cutt-off frequency f_c. The ripple in the bandpass is 3 dB (Chebyshev 1 and elliptical) and the minimum attenuation in the stopband is 20 dB (Chebyshev 2 and elliptical).

To comply with certain specifications of attenuation:

- The **Butterworth** approximation is the one that requires a higher order, but presents a phase characteristic closer to the ideal.
- The **elliptical** approach is the one that requires a smaller order, but is the one that presents a less desirable phase.
- The **inverse Chebishev** approximation provides filters with less phase distortion than the elliptical approach at the cost of slightly increasing the order with respect to the elliptic one. The approximation of **Chebishev** requires the same order as the inverse Chebishev but its phase characteristics are considerably worse.

The maximum value of $|H(F)|$ is usually normalized to 1 (if a larger maximum value is needed, the numerator coefficients can be multiplied by a scale factor).

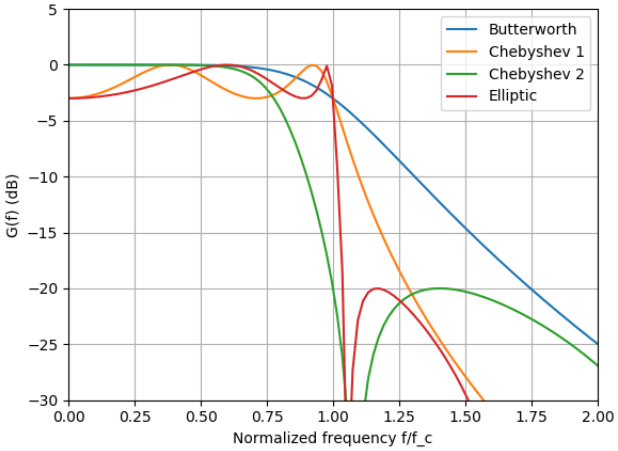


Figure 3: $G(f) = 20 \log_{10} |H(f)|$ comparison for different type of filters, all with order $n = 4$

3 Design and filtering tools in Python

Filter design tools in Python:

The Python signal processing toolbox scipy.signal currently contains a limited set of filter design tools.
For FIR filters:

- `scipy.signal.firwin`(numtaps, cutoff, width=None, window='hamming', pass_zero=True, scale=True, nyq=None, fs=None)
- `scipy.signal.firwin2`(numtaps, freq, gain, nfreqs=None, window='hamming', nyq=None, antisymmetric=False, fs=None)
- `scipy.signal.remez`(numtaps, bands, desired, weight=None, Hz=None, type='bandpass', maxiter=25, grid_density=16, fs=None)

For IIR filters:

- `scipy.signal.iirdesign`(wp, ws, gpass, gstop, analog=False, ftype='ellip', output='ba')
- `scipy.signal.iirfilter`(N, Wn, rp=None, rs=None, btype='band', analog=False, ftype='butter', output='ba')

To calculate the filter coefficients of FIR filters, we will use the `remez` function in the `scipy.signal` package. For example, to design a low pass filter with 10 coefficients, desired response equal to 1 between 0 and 10 Hz, desired response equal to 0 between 20 and $f_s/2$, and sampling frequency 500 Hz, we do:

```
taps = signal.remez(10, [0, 10, 20, 250], [1, 0], fs=500).
```

To design IIR filters, we will use the `iirdesign` function in the `scipy.signal` package. The usage is slightly different from the `remez` function. An example of usage is:

```
b, a = signal.iirdesign(wp = 10/250, ws= 20/250, gstop= 45, gpass=3, ftype='butter')
```

arguments wp and ws indicate the limit of the passband and the stopband respectively. Both wp and ws values must be normalized with respect to 0.5 times the sampling frequency. The next arguments are gstop, the minimum attenuation (in dB) of the stopband, and gpass, the maximum attenuation (in dB) of the passband. Finally, the argument ftype indicates the type of approximation we want to use for the filter design, namely 'butter', cheby1', 'cheby2', 'ellip', or 'bessel'.

Filtering tools in Python

To obtain zeros, poles, and frequency response from a numerator, denominator representation of a linear filter:

- `scipy.signal.tf2zpk`(b, a)
- `scipy.signal.freqz`(b, a=1, worN=512, whole=False, plot=None)

To filter data along one-dimension with an IIR or FIR filter:

- `scipy.signal.lfilter`(b, a, x, axis=-1, zi=None)

Functions for obtaining zeros, poles, frequency response and filtering when working with second order sections forms (sos):

- `scipy.signal.sos2zpk`(sos)
- `scipy.signal.sosfreqz`(sos, worN=None, whole=False)
- `scipy.signal.sosfilt`(sos, x, axis=-1, zi=None)

Final note: Sometimes, when filtering with an IIR filter, unexpected results may occur due to numerical rounding errors (particularly for high order filters). If this is the case, you may avoid this by designing the filter in a second order section form, using the function `signal.iirdesign` with the parameter `output='sos'`.

4 Lab work part 1: Filters design

1. Open the Atenea quiz (one per group). Write the name and surname of the two people working in the same quiz.

Preliminary analysis of ECGs

2. Consider the lowest heart rate as 30 beats per minute. What is the frequency of the heart beat in Hz?
3. Consider the highest heart rate as 200 beats per minute. What is this frequency in Hz?

Load the signals[3] by doing `s=np.load('ECGs_noisy.npy')`. Analyze signal number 1, i.e. `x=s[1]`, both in time and frequency. As the frequency content may vary over time, we will do the frequency analysis for the segment between $L : 2L$ samples with $L = 2500$.

```
plt.close('all')
s = np.load('ECGs_noisy.npy')
fs = 500 #sampling frequency
id = 1
x = s[id] #signal to work with
#DFT computation for a signal segment of length L:
N = 2**16
L = 2500
X = abs(fft(x[L:2*L]*hamming(L), N))
f = np.arange(N)/N*fs
#Time and frequency plots
plt.figure(1)
plt.plot(x)
plt.figure(2)
plt.plot(f, X)
plt.figure(3)
plt.plot(f, 20*np.log10(X))
```

4. Observing the signal, both in the time and frequency domain, what artifacts do you identify?
5. Propose techniques to remove the baseline drift
6. Propose techniques to remove the high-frequency noise

[3]We previously downloaded some of the binary files from the database and saved them in a numpy array, so you could focus on the signal processing part. However, if you are interesting you can download more ECGs from https://physionet.org/physiobank/database/ecgiddb/ or as a .mat file from https://physionet.org/cgi-bin/atm/ATM.

7. Although our goal is to enhance through filtering the ECG signals (so anyone can work with them, including non-Fourier experts), could you estimate the heart beat rate from the DFT?

Notch filter design

The goal of this section is to design a notch filter with a 4Hz bandwidth to eliminate the interference from the power supply network. For the module of the poles, use the design formula: $r \approx 1 - \frac{BW}{f_s}\pi$ (approx. valid for $0.9 \leq r < 1$), with f_s is the sampling frequency and BW the bandwidth of the filter.

Compute the transfer function of the filter. Visualize the zero-plot diagram and the frequency response of the filter to validate your design.

```
plt.close('all')
f0 = #to be completed
r = #to be completed
angle_pole = #to be completed
k = (1-2*r*np.cos(angle_pole)+r**2)/(2-2*np.cos(angle_pole))
b = #to be completed
a = #to be completed

z, p, k = signal.tf2zpk(b,a)
plt.figure()
plotzp(z, p)

w, H = signal.freqz(b, a)
freq = w/(2*np.pi)*fs
plt.figure()
plotresp(freq, H)
```

8. Indicate the phase of the zeros of the filter
9. Indicate the module of the poles
10. Measure the gain of the filter at frequency f_x
11. Measure the phase introduced by the filter at frequency f_x

Store the numerator and denominator of $H(z)$ for later use: `notch_num, notch_den = b, a`

A low pass filter

The goal of this section is to design a low pass filter with the following specifications: sampling frequency 500 Hz, passband from 0 to 40 Hz, stopband from 60 to 250 Hz, passband ripple 0.5 dB, and stopband attenuation 45 dB (this value may change in class).

A FIR filter: Use the function `signal.remez` from the scipy package to compute the coefficients of a FIR filter with previous specifications. See section 3 if you need to. Then, plot the zeros-poles diagram and the frequency response to check your design.

12. What is the minimum length of the impulse response of the filter (i.e. number of coefficients) to satisfy the specifications?
13. How many poles (outside the origin) does the filter have?
14. Discuss the stability of the filter
15. Can you observe any particularity regarding the coefficients of the filter?
16. What do you observe regarding the phase of the filter?
17. What is the delay in samples of the filter?

An IIR filter: Design now a IIR filter, Butterworth, with the same specifications of the FIR filter. Use the function `signal.iirdesign` from the scipy package to compute the coefficients of the filter. See section 3 if you need to. Then, plot the zeros-poles diagram and the frequency response to check your design.

18. What is the minimum number of coefficients for the filter to satisfy the specifications?
19. How many poles does the designed filter have?
20. What do you observe regarding the phase of the filter?

Store the numerator and denominator of the transfer function, $H(z)$, for later use: `lp_num, lp_den = b, a`

A high pass filter

Finally, design a high pass IIR filter, type Butterworth, with the following specifications: sampling frequency 500 Hz, stopband from 0 to 0.15 Hz, passband from 0.5 to 250 Hz, passband ripple 0.5 dB, stopband attenuation 40 dB.

21. What is the $H(z)$ of the filter?

Store the numerator and denominator of $H(z)$ for later use: `hp_num, hp_den = b, a`

5 Lab work part 2: Filtering of noisy ECGs

Open the Atenea quiz (one per group). You may use Jupyter notebook for this part if you prefer so. You will have an Atenea quiz open for a week to introduce your answers.

1. Write the name and surname of the two people working in the same quiz.

ECG enhancement

Filter signal number 1 successively with each one of the 3 filters designed in the first part of the lab: notch, low pass (IIR), and high pass filter. Use the function `y = signal.lfilter(num, den, x)`. Observe how the signal changes (both in time and frequency) at each step.

2. Upload a short document with your results and comments. Include figures. Is the final result different if we conmute the order of the filters?

Heart rate detection

Imagine you have received the following assignment: to develop a software product to automatically estimate the heart beat rate for a high number of ECG signals. This may seem a simple task and as you only need to count the number of R-waves. The signals are rather noisy, but you already designed your filters in the first part of the lab. Use this function:

```
def crossings(x, threshold):
    pos = x > threshold
    crossing_at = (pos[:-1] & ~pos[1:]).nonzero()[0]
    dif = np.diff(crossing_at)
    dif = dif[dif>100] #We erase outliers: 300 beats/minute
    dif = dif[dif<1500] #We erase outliers: 20 beats/minute
    peak_separation = np.mean(dif)
    plt.axhline(y=threshold)
    return peak_separation, crossing_at
```

Take several ECGs of the folder. Filter each signal with the filters designed in the first part of the lab. Estimate the heart rate. As the heart rate may change over time, for each signal do the estimation for the segment between $L : 2L$ samples with $L = 2500$. (Important: filter the whole signal, then select the segment $L : 2L$. This way, the initial transient in the filter output will not affect the estimation).

3. Even though you did a good job of pre-filtering the signals, P or T waves may difficult the estimation in some cases. Indicate one of the signals for which you experience difficulties to carry out the automatic estimation.

You did some research, and found out that typical frequency components of QRS complex range[4] from around 5Hz and 25 Hz. So for this particular application (heart rate estimation), design a band pass filter with the following specifications: Butterworth, passband between 10 and 20 Hz, stop band from 0 to 3Hz and from 30 to 250 Hz, minimum attenuation in the stopband 40 dB, and maximum attenuation in the bandpass 0.5 dB.

4. Filter the original signals with this new filter (only with this one). Compare the filtered signals and your new heart rate estimations with your previous ones. Explain your observations and results.

5. Introduce the estimation for the set of signals indicated in the quiz.

[4] B. Subramanian, *ECG signal classification and parameter estimation using multiwavelet transform*, Biomedical Research (2017) Volume 28, Issue 7