The objective of this exercise is that you analyse the code provided and make the link with the course. You have to provide a short report that comments and analyse the results. You can use directly the results or adapt them to you needs.

import the numerical library

```
import numpy as np
# import signal processing library
import scipy.signal as sp
# import ploting library
import pylab as py
py.ion()
py.close('all')
```

load the ecg signal

```
x = np.genfromtxt('ecg.dat')
# sampling frequency of the signal is 500 Hz
fs = 500
# generate correponding time vector
t = np.arange(len(x))/fs
```

The signal is an ECG signal with visible PQRST complex. If you zoom on the signal plot you can see that there is a 50Hz perturbation due to the power network. The objective is to remove this component without altering the PQRST complex. Several filtering techniques are used. Comment the advantages and disadvantages.

Plot time signal and FFT. Q: Comment the figures.

Compute the FFT of the signal

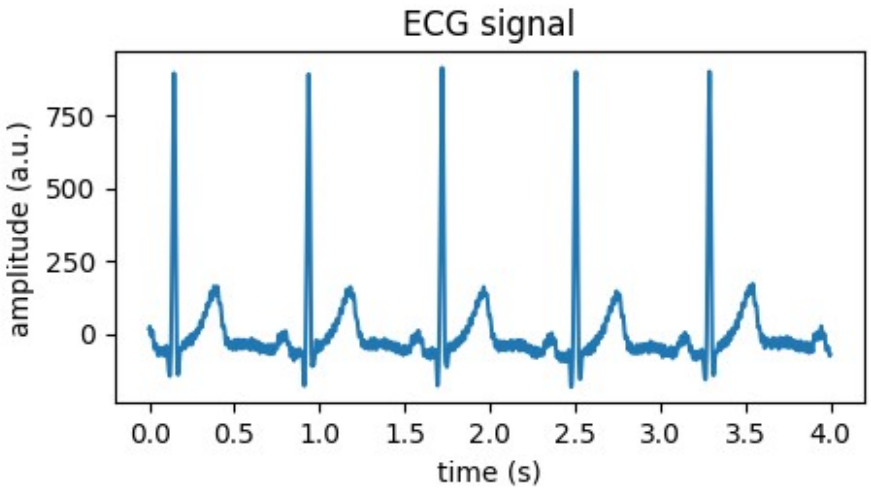
```
x_fft = np.fft.fft(x)
# Determine the frequency scale
f_fft = np.arange(len(x_fft))/len(x_fft)*fs
```

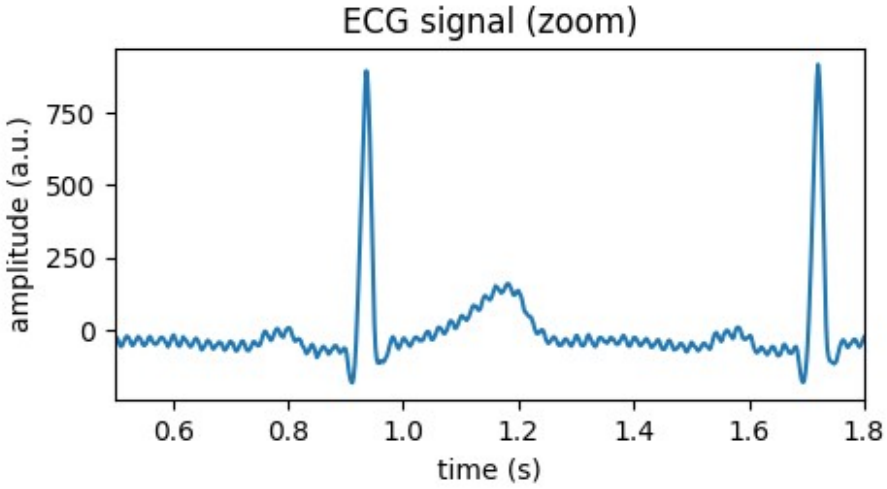
plot the signal

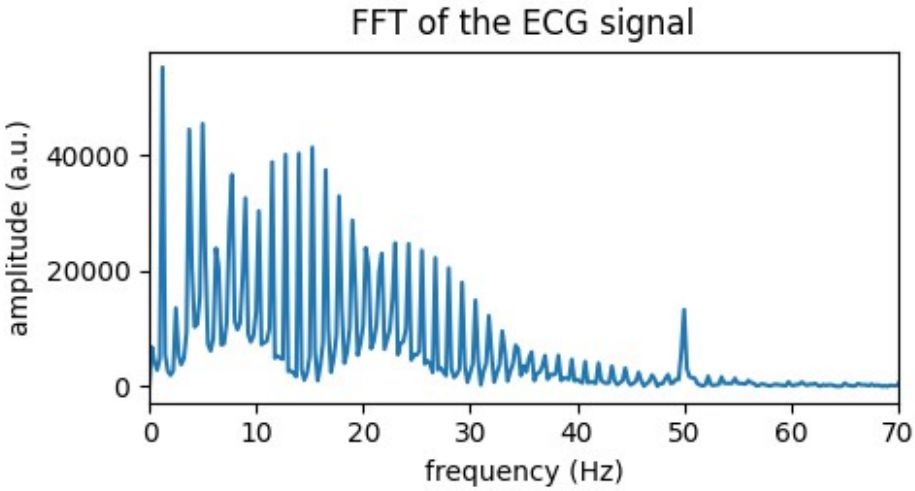
```
py.figure(1, figsize=[5,8])
py.clf()
py.subplot(3,1,1)
py.plot(t, x)
py.xlabel('time (s)')
py.ylabel('amplitude (a.u.)')
py.title('ECG signal')
py.subplot(3,1,2)
py.plot(t, x)
py.xlabel('time (s)')
py.ylabel('amplitude (a.u.)')
py.title('ECG signal (zoom)')
```

```
py.xlim(0.5, 1.8),
py.subplot(3,1,3)
py.plot(f_fft, abs(x_fft))
py.xlabel('frequency (Hz)')
py.ylabel('amplitude (a.u.)')
py.title('FFT of the ECG signal')
py.tight_layout()
py.xlim(0,70)
```

```
(0.0, 70.0)
```




**Answer**

In the first figure we see the ECG signal with the distinct PQRST phases, but then zooming in we can clearly see the interference at each 0.02 seconds (50Hz), and this interference makes it hard to distinguish the start of the PQRST (P-point). And plotting the Fourier transform, we can

effectively see in the FFT the presence of the interference from the power grid at 50 Hz. The goal is to filter out this noise without distorting the important parts of the ECG (PQRST). We can apply a band stop filter in order to remove the frequency at 50Hz. But we see that most of the energy of the signal is located before 35Hz and therefore we can understand the choice of taking 35Hz as the start frquency of the bandstop (see next point).

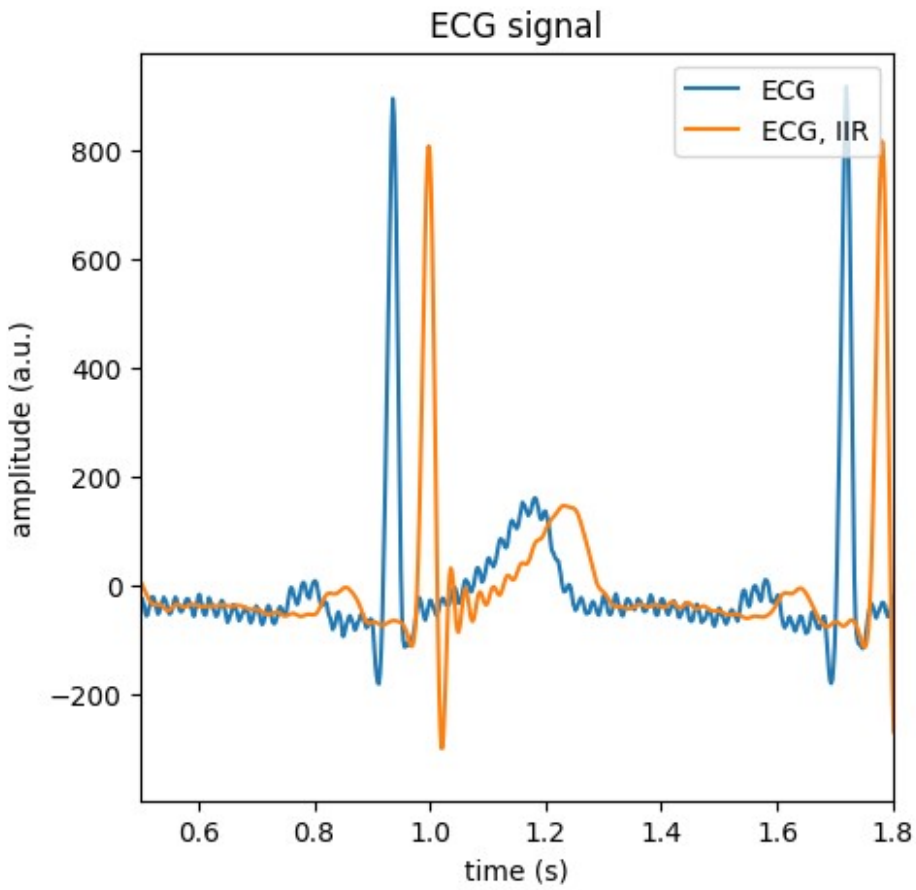
IIR filter: Define a filter with a pass-band up to 35 Hz and a stop band from 50Hz. Maximum attenuation in passband 3 dB Minimum attenuation in stopband 40 dB Q: Comment the results (distorsion of the PQRST, delay, ...) Q: Based on the FFT spectrum comment the selection of the pass and stop band frequencies.

Analogic limit of the passband frequency

```
f_pass = 42
# Analogic limit of the stopband frequency
f_stop = 50
# Convertion into Nyquist frequency
f_pass_N = f_pass/fs*2
f_stop_N = f_stop/fs*2
# Max attenutation in passband (dB)
g_pass = 3
# Min attenuation in stopband (dB)
g_stop = 40
# Determine the order and the cutoff frequency of a butterworth filter
ord, wn = sp.buttord(f_pass_N, f_stop_N, g_pass, g_stop)
# Compute the coeffcients of the filter
b, a = sp.butter(ord, wn)
# Filter the signal
x_f = sp.lfilter(b ,a, x)

py.figure(2, figsize=[5,5])
py.clf()
py.plot(t ,x, label='ECG')
py.plot(t, x_f, label='ECG, IIR')
py.xlabel('time (s)')
py.ylabel('amplitude (a.u.)')
py.title('ECG signal')
py.legend(loc='upper right')
py.xlim(0.5, 1.8)

(0.5, 1.8)
```




**Answer**

With the IIR filter, we see that there is a shift in phase delay (Time Delay) and small distortion of the ECG Signal (S point) but the frequencies at 50Hz are attenuated. A narrower stop band or the use of a different filtering technique, for example a zero-phase IIR filter, might reduce the delay and preserve the integrity of the PQRST complex. Additionnaly we can quickly comment the choice of 35Hz, we decide to adjust it but taking smaller bandstop we create an overshooted filtered signal and we distort even more the signal.

IIR filter (zero phase): Use the same filter but apply a zero phase approach. Q: Comment the results (distorsion of the PQRST, delay, ...)

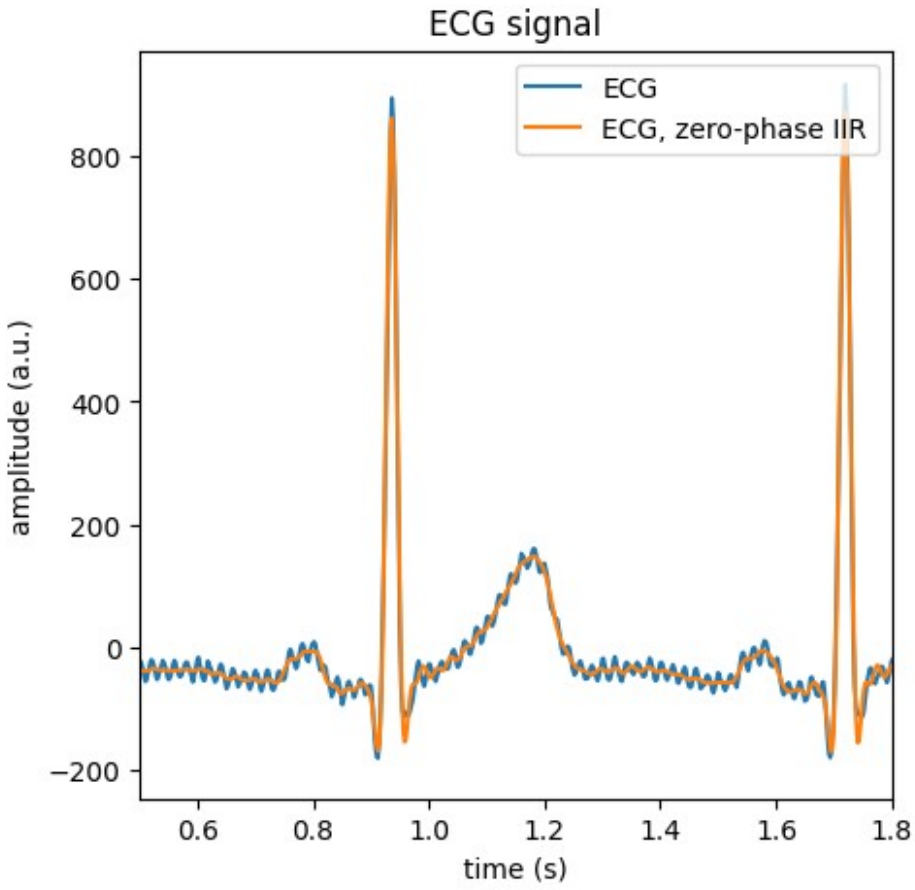
Filter the signal

```
x_f = sp.filtfilt(b ,a, x)

py.figure(3, figsize=[5,5])
py.clf()
py.plot(t ,x, label='ECG')
py.plot(t, x_f, label='ECG, zero-phase IIR')
py.xlabel('time (s)')
py.ylabel('amplitude (a.u.)')
py.title('ECG signal')
```

```
py.legend(loc='upper right')
py.xlim(0.5, 1.8)
```

```
(0.5, 1.8)
```




**Answer**

We obtain a very accurate filtered signal with the zero-phase IIR filter compared to the previous IIR filter. It successfully removes the unwanted 50Hz frequency and we see really small distortion for the ECG (always for S-point). But this can be due to the 50Hz oscillations that we removed and also with the frequencies that were deleted between 35Hz and 50Hz. Hence, we obtain a filtered signal that shows us clearly the PQRST of the ECG signal, and this time without phase delay. This makes it an ideal filtering choice for ECG signals.

Linear phase FIR filter. Define a FIR filter with the same properties. Q: Comment the results (distorsion of the PQRST, delay, ...).

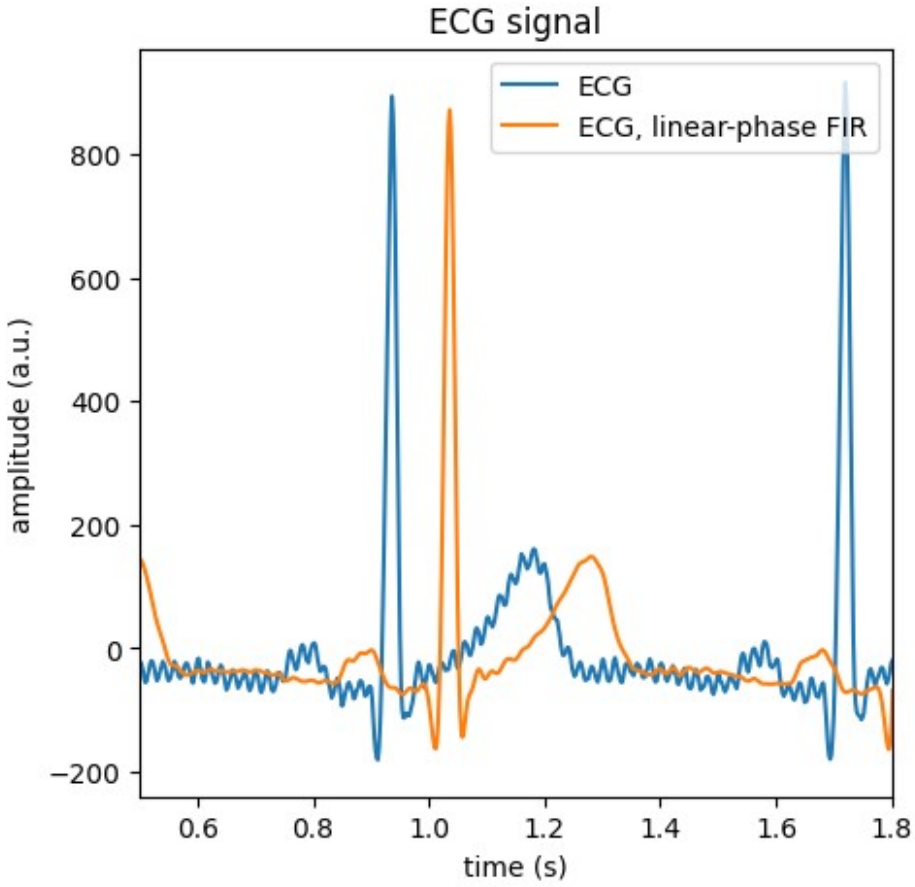
length of the filter

```
l_fir = 101
# compute the filter coefficients using least square approach
b = sp.firls(l_fir, [0, f_pass_N, f_stop_N, 1], [1, 1, 1/100, 1/100])
```

```
a = [1]
# filter the signal
x_f = sp.lfilter(b, a, x)

py.figure(4, figsize=[5,5])
py.clf()
py.plot(t ,x, label='ECG')
py.plot(t, x_f, label='ECG, linear-phase FIR')
py.xlabel('time (s)')
py.ylabel('amplitude (a.u.)')
py.title('ECG signal')
py.legend(loc='upper right')
py.xlim(0.5, 1.8)

(0.5, 1.8)
```




**Answer**

This time, the results are worse. Despite the FIR filter successfully removing the 50 Hz interference, it introduces a phase delay. As a consequence, the ECG signal is distorted and does not correspond to the correct ECG signal anymore.

The objective of this exercise is that you analyse the code provided and make the link with the course. You have to provide a short report that comments and analyse the results. You can use directly the results or adapt them to you needs.

import the numerical library

```
import numpy as np
# import signal processing library
import scipy.signal as sp
# import ploting library
import pylab as py
py.ion()
py.close('all')
```

load the ecg signal

```
x = np.genfromtxt('respiration.dat')
# sampling frequency of the signal is 500 Hz
fs = 2
# generate correponding time vector
t = np.arange(len(x))/fs
```

The signal is a measurement of the breathing obtained by inductance plethysmography. The objective is to estimate the breathing frequency.

The Hilbert transforms permits to estimate the instaneous amplitude and phase of a narrow band signal. Q: Comment the figures. Q: Why the envelope does no follow the maxima of the signal

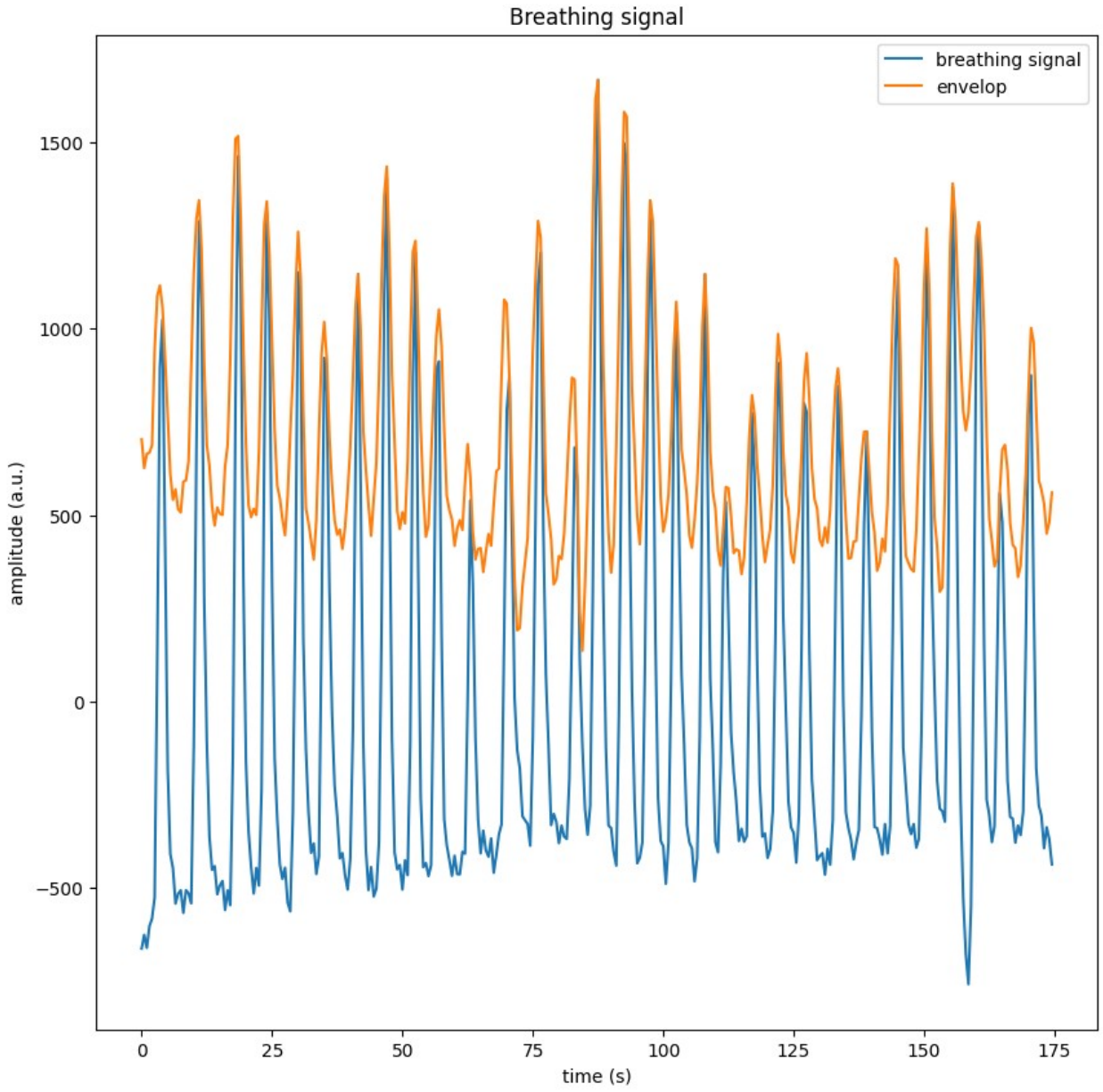
compute the analytical signal of x (Hilbert transform)

```
xa = sp.hilbert(x)
```

plot the signal

```
py.figure(1, figsize=[10,10])
py.clf()
py.plot(t, x, label='breathing signal')
py.plot(t, np.abs(xa), label='envelop')
py.xlabel('time (s)')
py.ylabel('amplitude (a.u.)')
py.legend(loc='upper right')
py.title('Breathing signal')
```

```
Text(0.5, 1.0, 'Breathing signal')
```




**Answer**

The signal is not narrow band. Consequently, the Hilbert transform does not produce a meaningful result, as it assumes the signal to be narrow band. The envelope is not following the maxima for this exact reason; the Hilbert transform does not expect the signal to have a wide range of frequencies.

The raw breathing signal does not fullfil the requirement of narrow band. The normal range of frequency for the breathing is within 0.1 to 0.25 Hz. The signal is first filtered for this interval. Q: Comment the figures Q: How is the estimation of the amplitude envelope.

Analogic limit of the passband frequency

```
f_pass = np.array([0.1, 0.25])
# Analogic limit of the stopband frequency
f_stop = np.array([0, 0.6])
# Convertion into Nyquist frequency
f_pass_N = f_pass/fs*2
f_stop_N = f_stop/fs*2
# Max attenutation in passband (dB)
g_pass = 3
# Min attenuation in stopband (dB)
g_stop = 40
# Determine the order and the cutoff frequency of a butterworth filter
ord, wn = sp.buttord(f_pass_N, f_stop_N, g_pass, g_stop)
# Compute the coeffcients of the filter
b, a = sp.butter(ord, wn, btype='band')
# Filter the signal
x_bp = sp.filtfilt(b ,a, x)
```

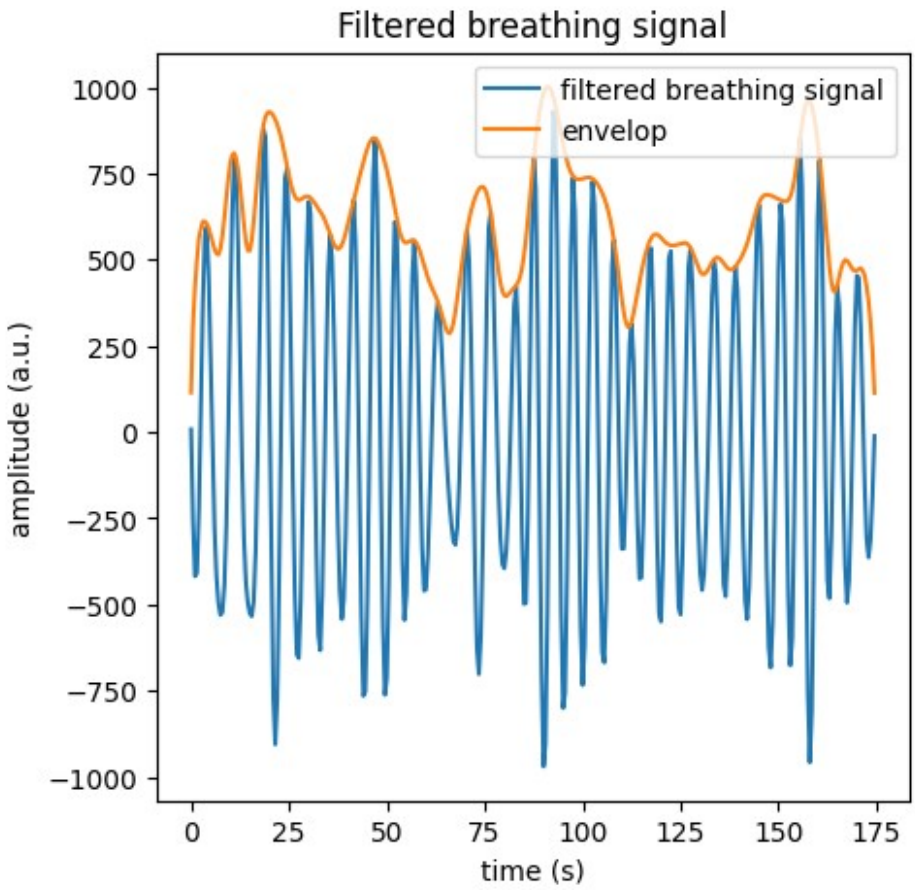
```
/usr/local/anaconda3/envs/ABSP/lib/python3.12/site-packages/scipy/
signal/_filter_design.py:3868: RuntimeWarning: divide by zero
encountered in divide
  nat = ((stopb ** 2 - passb[0] * passb[1]) /
```

Compute the Hilbert transform.

```
xa = sp.hilbert(x_bp)

py.figure(2, figsize=[5,5])
py.clf()
py.plot(t, x_bp, label='filtered breathing signal')
py.plot(t, np.abs(xa), label='envelop')
py.xlabel('time (s)')
py.ylabel('amplitude (a.u.)')
py.title('Filtered breathing signal')
py.legend(loc='upper right')
```

```
<matplotlib.legend.Legend at 0x11dd84290>
```




**Answer**

In this case, the filtered signal now fullfills the condition of being narrow band. Hence we see that the Hilbert transform is perfectly following the maxima of the filtered breathing signal, and the estimation of the amplitude envelope is accurate.

The angle of the Hilbert transform gives the instaneous phase of the signal. Q: Comment the figure. Q: What is the role of the unwrap function

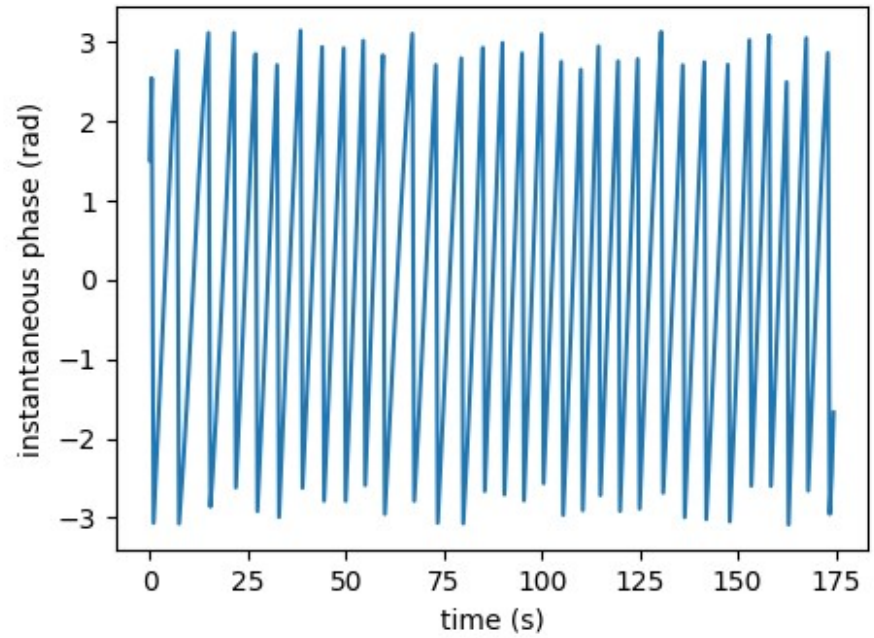
estimate the instantaneous phase from the Hilbert transform

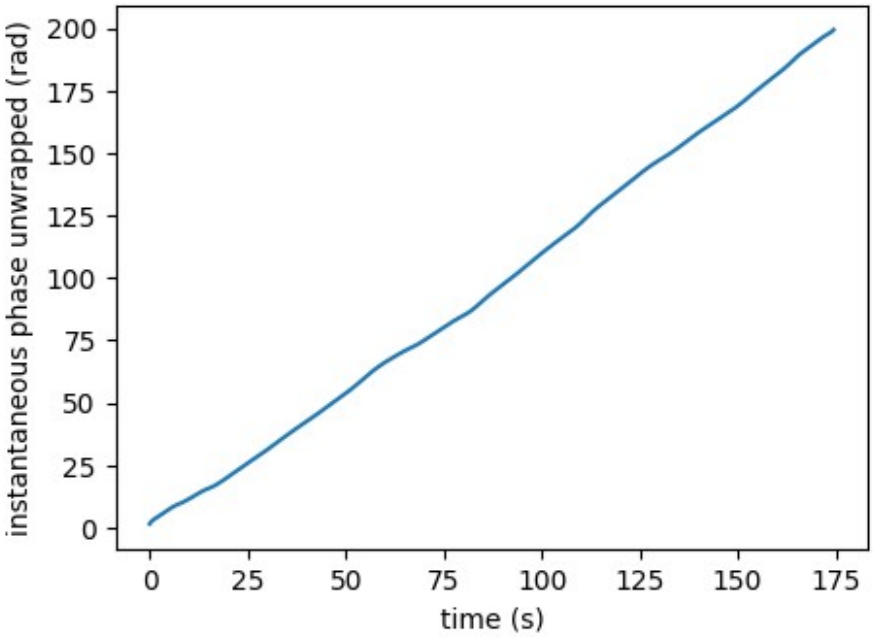
```
phi_xa = np.angle(xa)
# phase is bounded between -pi and pi -> reconstruct continuous signal
phi_xa_unw = np.unwrap(phi_xa)

py.figure(3, figsize=[5, 8])
py.clf()
py.subplot(2,1,1)
py.plot(t, phi_xa)
py.xlabel('time (s)')
py.ylabel('instantaneous phase (rad)')
py.subplot(2,1,2)
py.plot(t, phi_xa_unw)
```

```
py.xlabel('time (s)')
py.ylabel('instantaneous phase unwrapped (rad)')
```

```
Text(0, 0.5, 'instantaneous phase unwrapped (rad)')
```




**Answer**

It is difficult to see in the first figure that the phase is steadily increasing because the phase is wrapped from $-\pi$ to $\pi$. The unwrapped function is used to extend the boundary of the phase beyond $[-\pi, +\pi)$, revealing the steady phase increase.

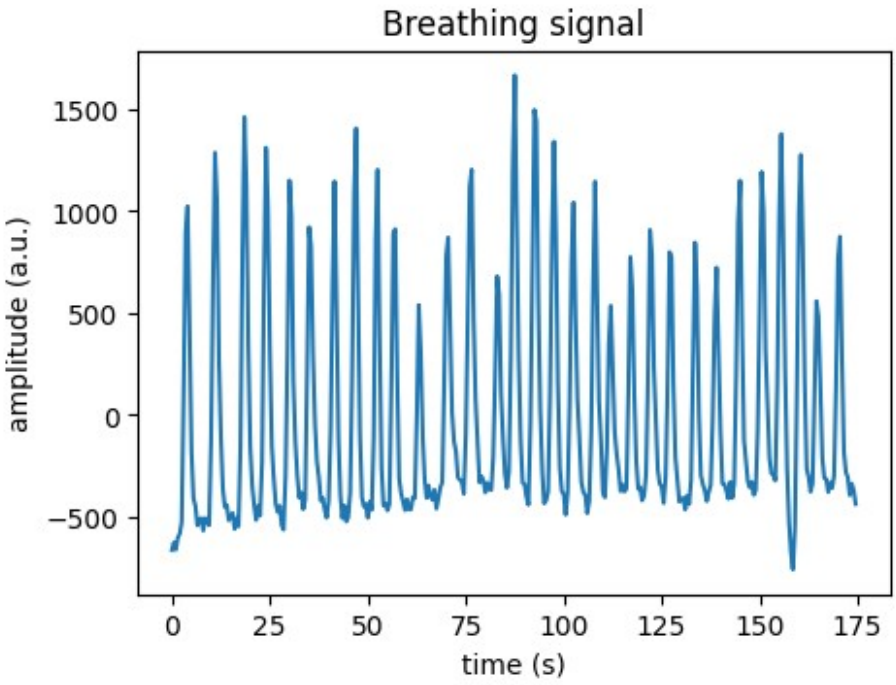
The time derivate of the instaneous phase is the instaneous frequency of the signal. Q: Comment the figure. Q: Compare the original waveform with the estimation of the breathing frequency

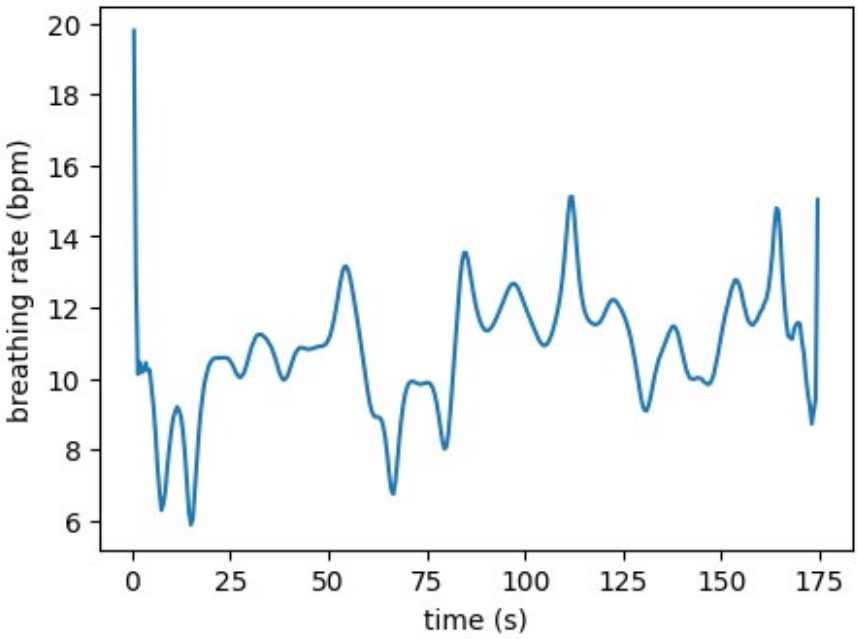
compute the derivative of the phase (angular frequency).

```
d_phi = np.diff(phi_xa_unw)
# convert angular frequency to frequency.
d_phi /= 2*np.pi
# convert digital frequency to analog frequency and in breathing per
minute
# (bpm)
d_phi *= fs*60

py.figure(4, figsize=[5,8])
py.clf()
py.subplot(2,1,1)
py.plot(t, x, label='breathing signal')
py.xlabel('time (s)')
py.ylabel('amplitude (a.u.)')
py.title('Breathing signal')
py.subplot(2,1,2)
py.plot(t[1:], d_phi)
py.xlabel('time (s)')
py.ylabel('breathing rate (bpm)')

Text(0, 0.5, 'breathing rate (bpm)')
```




**Answer**

There are a lot of fluctuations (amplified by the numerical derivation), but by averaging the instantaneous frequency over a small interval we may get a good estimation of the breathing frequency.

The objective of this exercise is that you analyse the code provided and make the link with the course. You have to provide a short report that comments and analyse the results. You can use directly the results or adapt them to you needs.

import the numerical library

```
import numpy as np
# import signal processing library
import scipy.signal as sp
# import ploting library
import pylab as py
py.ion()
py.close('all')
```

load the ecg signal

```
x = np.genfromtxt('accel.dat')
# sampling frequency of the signal is 500 Hz
fs = 40
# generate correponding time vector
t = np.arange(len(x))/fs
```

Plot time signal. Q: Comment the figure.

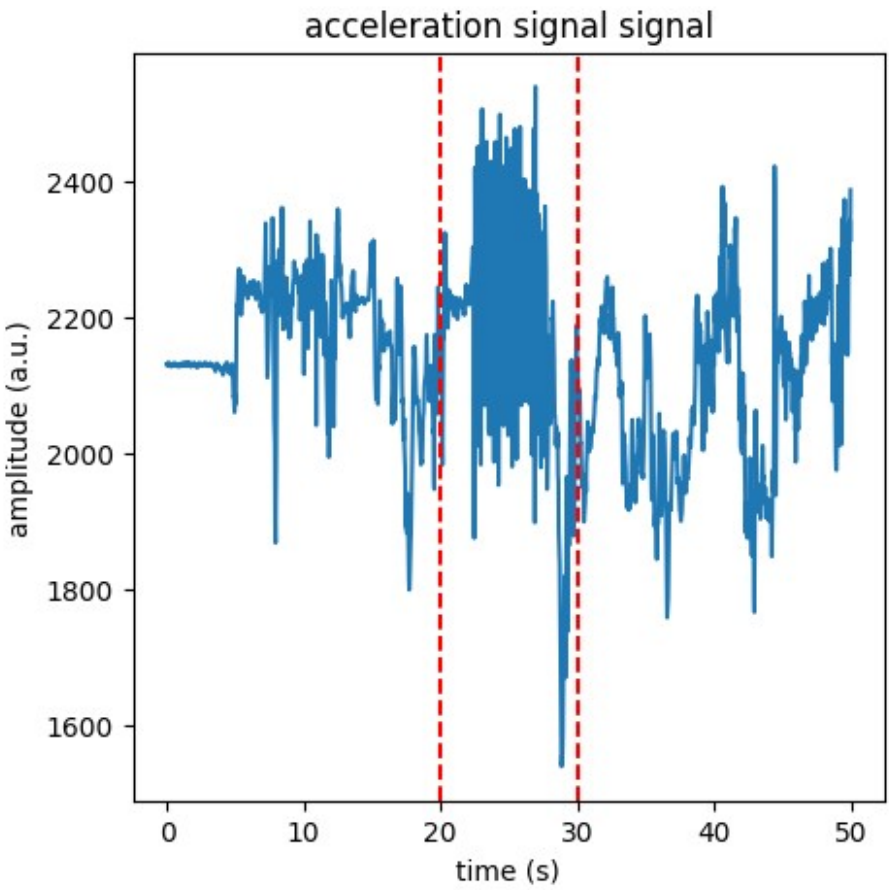
Compute the FFT of the signal

```
x_fft = np.fft.fft(x)
# Determine the frequency scale
f_fft = np.arange(len(x_fft))/len(x_fft)*fs
```

plot the signal

```
py.figure(1, figsize=[5,5])
py.clf()
py.plot(t, x)
py.xlabel('time (s)')
py.ylabel('amplitude (a.u.)')
# Put vertical lines btw 20 and 30 seconds (Handwashing),
discontinuous line
py.axvline(20, color='r', linestyle='--')
py.axvline(30, color='r', linestyle='--')
py.title('acceleration signal signal')
```

```
Text(0.5, 1.0, 'acceleration signal signal')
```




**Answer**

When plotting the signal, we can clearly observe fast movements in the 20-30 second range. This strongly suggests that the regular and repetitive movements during this period correspond to the hands rubbing against each other during hand-washing activity.

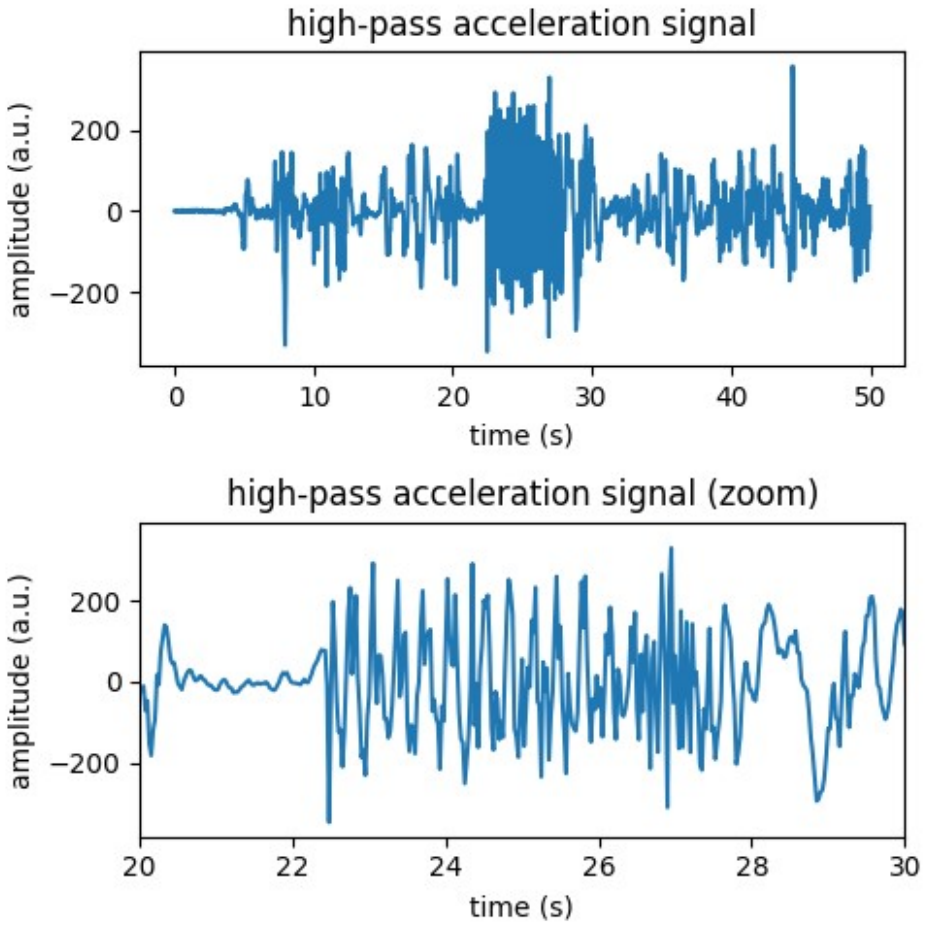
High pass the signal. Q: Comment the figure.

high-pass filter with cutoff frequency of 0.5 Hz

```
b, a = sp.butter(4, 0.5/fs*2, btype='high')
# zero-phase filtering of the signal
x_hp = sp.filtfilt(b, a, x)

py.figure(2, figsize=[5,5])
py.clf()
py.subplot(2,1,1)
py.plot(t, x_hp)
py.xlabel('time (s)')
py.ylabel('amplitude (a.u.)')
py.title('high-pass acceleration signal')
py.subplot(2,1,2)
py.plot(t, x_hp)
```

```
py.xlabel('time (s)')
py.ylabel('amplitude (a.u.)')
py.title('high-pass acceleration signal (zoom)')
py.xlim(20, 30)
# Adjust to the figure to show it properly
py.tight_layout()
```




**Answer**

By applying a high-pass filter with a cutoff frequency of 0.5 Hz, we reduce the impact of low-frequency components, resulting in a clearer trend between 20 and 30 seconds. And zooming in on the filtered signal, we can more clearly identify the hand-washing activity with a significantly change in the signal. It suggests that the frequencies of the handwashing is slightly higher in frequency than this 0.5 Hz.

Band pass the signal between 2.4 and 3.2 Hz. Q: Based on previous figure, comment the selection of the frequencies. Q: Why zero phase filter (filtfilt) is used?

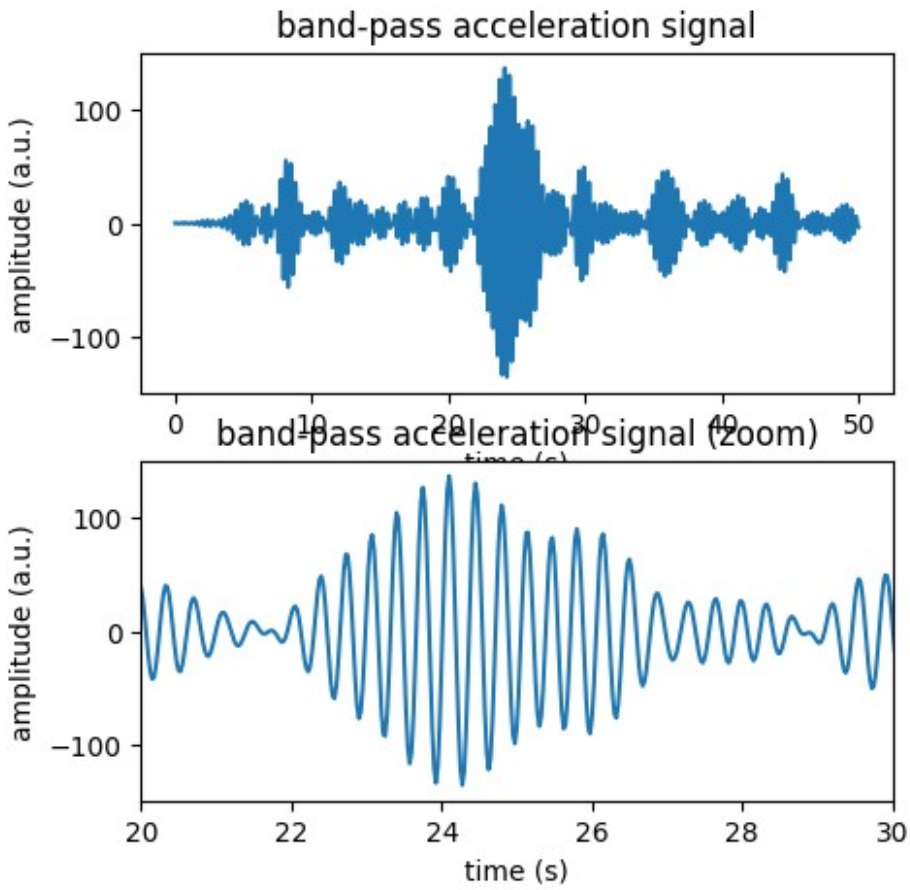
Analogic limit of the passband frequency

```
f_pass = np.array([2.4, 3.2])
# Analogic limit of the stopband frequency
f_stop = np.array([0, 5])
# Convertion into Nyquist frequency
f_pass_N = f_pass/fs*2
f_stop_N = f_stop/fs*2
# Max attenutation in passband (dB)
g_pass = 3
# Min attenuation in stopband (dB)
g_stop = 40
# Determine the order and the cutoff frequency of a butterworth filter
ord, wn = sp.buttord(f_pass_N, f_stop_N, g_pass, g_stop)
# Compute the coeffcients of the filter
b, a = sp.butter(ord, wn, btype='band')
# Filter the signal
x_bp = sp.filtfilt(b ,a, x_hp)
```

```
/usr/local/anaconda3/envs/ABSP/lib/python3.12/site-packages/scipy/
signal/_filter_design.py:3868: RuntimeWarning: divide by zero
encountered in divide
  nat = ((stopb ** 2 - passb[0] * passb[1]) /
```

```
py.figure(3, figsize=[5,5])
py.clf()
py.subplot(2,1,1)
py.plot(t, x_bp)
py.xlabel('time (s)')
py.ylabel('amplitude (a.u.)')
py.title('band-pass acceleration signal')
py.subplot(2,1,2)
py.plot(t, x_bp)
py.xlabel('time (s)')
py.ylabel('amplitude (a.u.)')
py.title('band-pass acceleration signal (zoom)')
py.xlim(20, 30)
```

```
(20.0, 30.0)
```




**Answer**

We removed the frequencies outside the range of 2.4 and 3.2 Hz using a band-pass filter, which reduced the number of frequencies in the signal and highlighted those associated with hand-washing. The oscillations are more regular and clearly show the hand-washing movements, especially between 22 and 26 seconds.

A zero-phase filter was used, as in Lab 2, to avoid introducing phase delay into the filtered signal. Using a standard filter usually adds a phase shift (as seen in Lab 2 too), which can distort the desired signal and reduce accuracy

Low-pass filter of the power of the band-pass signal. Q: Why use the power of the acceleration signal? Q: How the detection of hand washing is obtained?

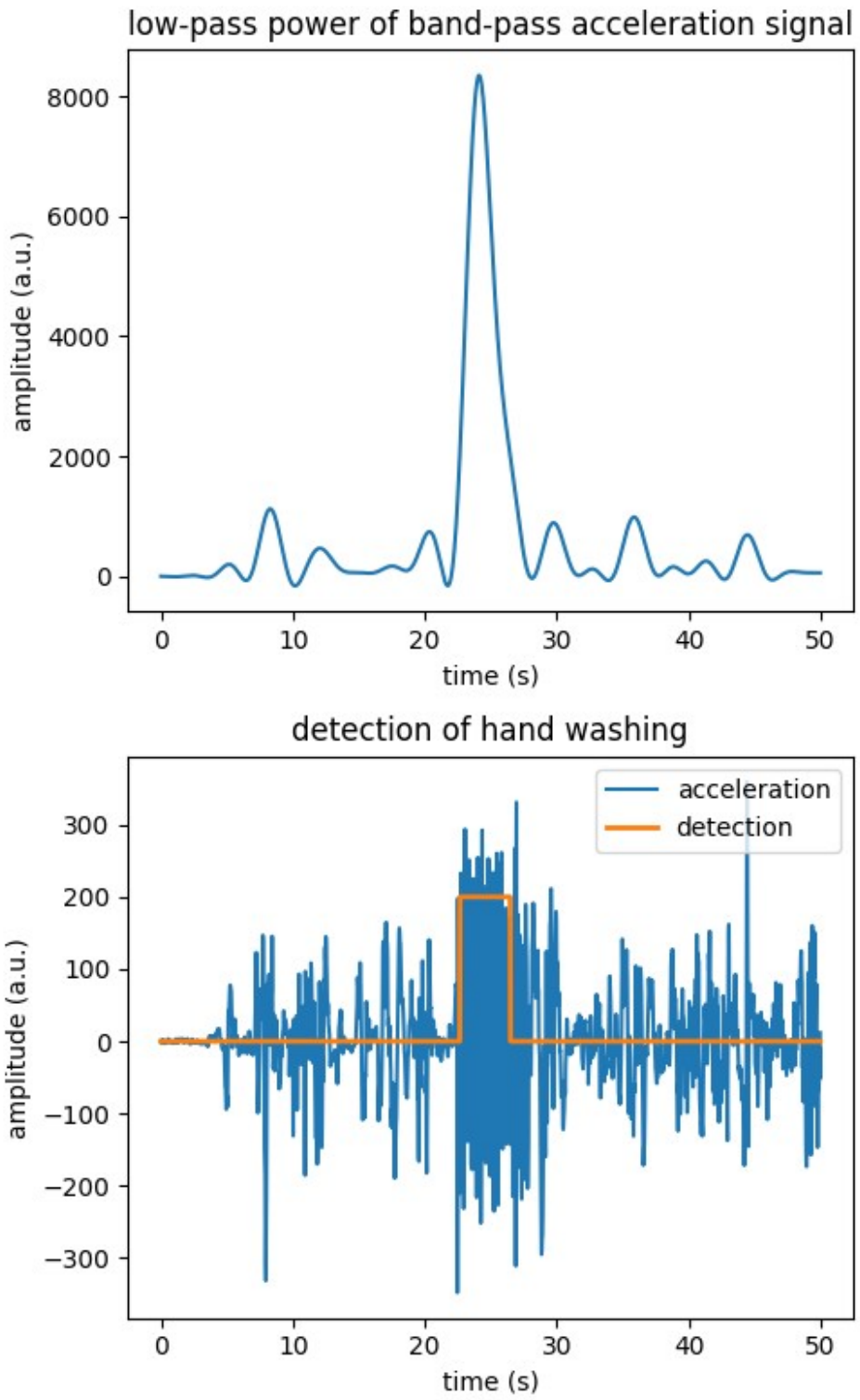
Analogic limit of the passband frequency

```
f_pass = 0.4
# Analogic limit of the stopband frequency
f_stop = 0.8
# Convertion into Nyquist frequency
f_pass_N = f_pass/fs*2
f_stop_N = f_stop/fs*2
# Max attenutation in passband (dB)
```

```
g_pass = 3
# Min attenuation in stopband (dB)
g_stop = 40
# Determine the order and the cutoff frequency of a butterworth filter
ord, wn = sp.buttord(f_pass_N, f_stop_N, g_pass, g_stop)
# Compute the coeffcients of the filter
b, a = sp.butter(ord, wn)
# Filter the signal
x_pow = sp.filtfilt(b ,a, x_bp**2)
# detection
det = x_pow > 2000

py.figure(4, figsize=[5,8])
py.clf()
py.subplot(2,1,1)
py.plot(t, x_pow)
py.xlabel('time (s)')
py.ylabel('amplitude (a.u.)')
py.title('low-pass power of band-pass acceleration signal')
py.subplot(2,1,2)
py.plot(t, x_hp, label='acceleration')
py.plot(t, det*200, linewidth=2, label='detection')
py.xlabel('time (s)')
py.ylabel('amplitude (a.u.)')
py.title('detection of hand washing')
py.tight_layout()
py.legend(loc='upper right')
```

```
<matplotlib.legend.Legend at 0x1201613a0>
```




**Answer**

The power of the acceleration signal $\left(x_{bp}^2\right)$ is used to accentuate the energy of the signal. Since the power is the squared magnitude, that is highlighting the larger peaks and reducing the

smaller ones, hence in this case it's accentuating our handwashing movement, and minimizing the other frequencies. Using the power allows us to detect the handwashing movements more clearly.

After applying the low-pass filter on the signal obtained by band-pass filter, as explained above, we reduced the impact of smaller peaks and increase the more significant ones. We see that there is a clear and abrupt change in movement detection and we can deduce that is our handwashing movement (around 22s, shown with orange line) which is coeherent with what we discuss until this point. This technique permits to increase accurate handwashing detection.