Module 2 Lab Report

Group Members: Mamoun Alaoui Slimani, Léa Slive, Yasaman Noorikhah, Antonio Del Priore Antunes
Date of Submission: Thursday, September 25, 2025

Please find the answer of each question written as a markdown cell after the question.

Part 1: Ecg enhance

The objective of this exercise is that you analyse the code provided and make the link with the curse. 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

In [11]:
```
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

In [12]:
```
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.
Answer:

- Filtering techniques such as IIR and FIR filters are necessary to effectively remove the 50 Hz powerline interference.
- IIR filters are more computationally efficient because they require fewer coefficients, but they may introduce phase distortion and delay, which can affect the timing of the PQRST complex. They are also not guaranteed to be stable unless the pole radius is under 1.

- FIR filters are always stable and can achieve linear phase, preserving the waveform shape and timing (no distortion). However, they may require a higher order (more coefficients) to reach the same filtering effect, increasing computational cost.
- Zero-phase filtering (e.g., filtfilt) can be used with IIR or FIR filters to minimize phase distortion and delay, preserving the clinical features of the ECG. However, they can only be used post-analysis and not real-time.
- The choice of filter depends on the balance between computational efficiency and the need to preserve signal morphology for accurate analysis.

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

- The time signal plot (cell 6) shows the ECG waveform with clear PQRST complexes. The signal is periodic and the main features are visible, but there is a noticeable oscillation due to 50 Hz powerline interference.
- The zoomed-in plot highlights the PQRST complex, making it easier to see the effect of noise and the morphology of the ECG.
- The FFT plot reveals the frequency content of the signal. Most of the energy is concentrated below 35 Hz, corresponding to the physiological ECG components. There is a distinct peak at 50 Hz, confirming the presence of powerline interference. This justifies the need for filtering above 35 Hz and especially at 50 Hz to clean the signal while preserving the PQRST features.

Compute the FFT of the signal

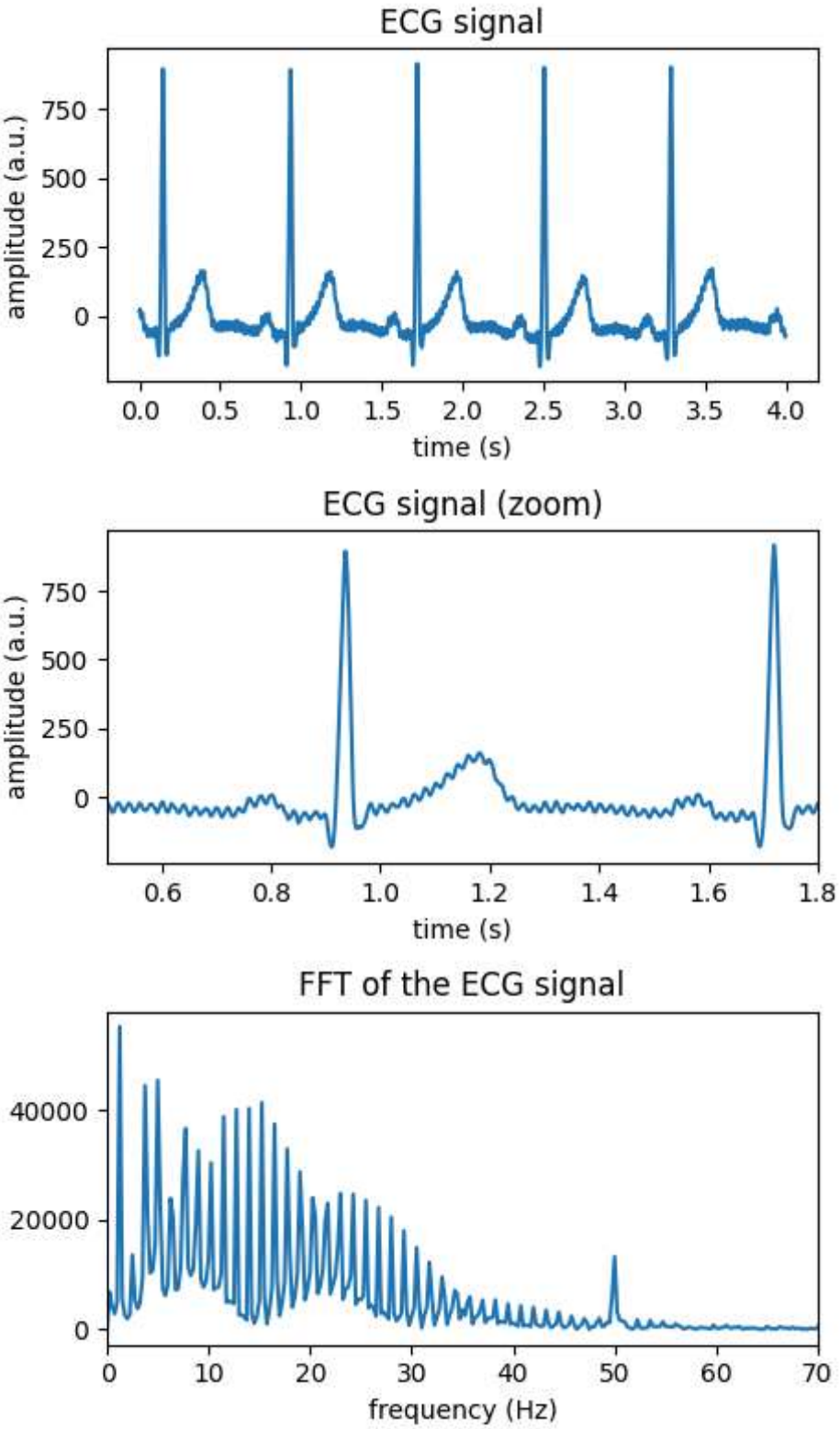
In [13]:
```
x_fft = np.fft.fft(x)
# Determine the frequency scale
f_fft = np.arange(len(x_fft))/len(x_fft)*fs
```

plot the signal

In [14]:
```
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)
```

Out[14]: (0.0, 70.0)



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 (distortion of the PQRST, delay, ...)

Answer:

- The IIR filter effectively attenuates frequencies above 50 Hz, reducing high-frequency noise and powerline interference.
- The PQRST complex is generally preserved, but some minor distortion may occur, especially near the filter cutoff frequencies. The sharpness of the QRS complex can be slightly reduced if the transition band is not steep enough.
- IIR filters introduce phase delay, which can shift the ECG waveform in time. This delay is most noticeable at frequencies near the cutoff and can affect the alignment of the PQRST features. For clinical analysis, zero-phase filtering (e.g., using `filtfilt`) is recommended to minimize delay.

Q: Based on the FFT spectrum comment the selection of the pass and stop band frequencies.
Answer:

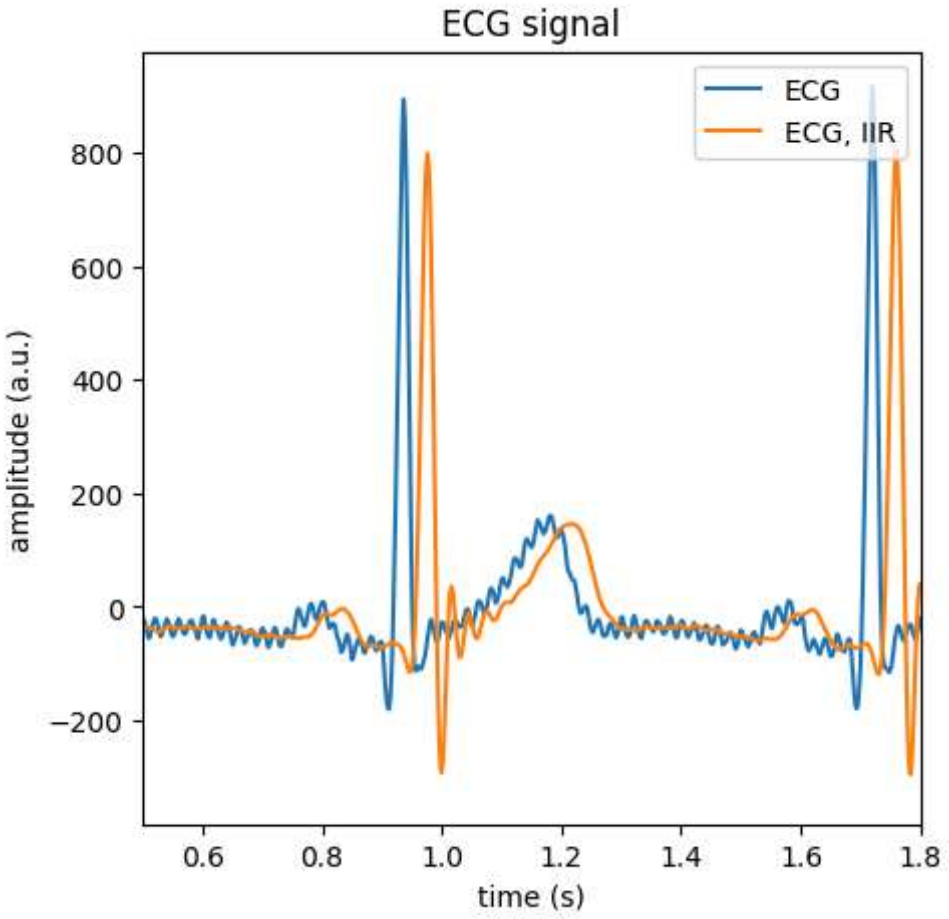
- The FFT spectrum shows that most ECG signal energy is below 35 Hz, which justifies the pass-band limit. Frequencies above 50 Hz are mainly noise (e.g., powerline interference), so setting the stop band at 50 Hz is appropriate.
- The choice of pass and stop bands should balance signal preservation and noise removal. If the stop band is set too low, important signal components may be lost; if too high, noise may remain. The transition band should be as narrow as possible for optimal filtering.

Analogic limit of the passband frequency

In [15]:
```
f_pass = 35
# 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)
```

In [16]:
```
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)
```

Out[16]:
```
(0.5, 1.8)
```



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

- Applying a zero-phase IIR filter (using `filtfilt`) removes the phase delay introduced by standard IIR filtering while still effectively removing the 50Hz powerline interference.
- The PQRST complex is preserved without time shift or phase distortion, and the waveform morphology remains intact.
- Zero-phase filtering is ideal for clinical ECG analysis because it avoids distortion and delay, ensuring accurate timing of cardiac events.
- The main limitation is that zero-phase filtering is non-causal and cannot be used in real-time applications, but it is excellent for post-processing and analysis.

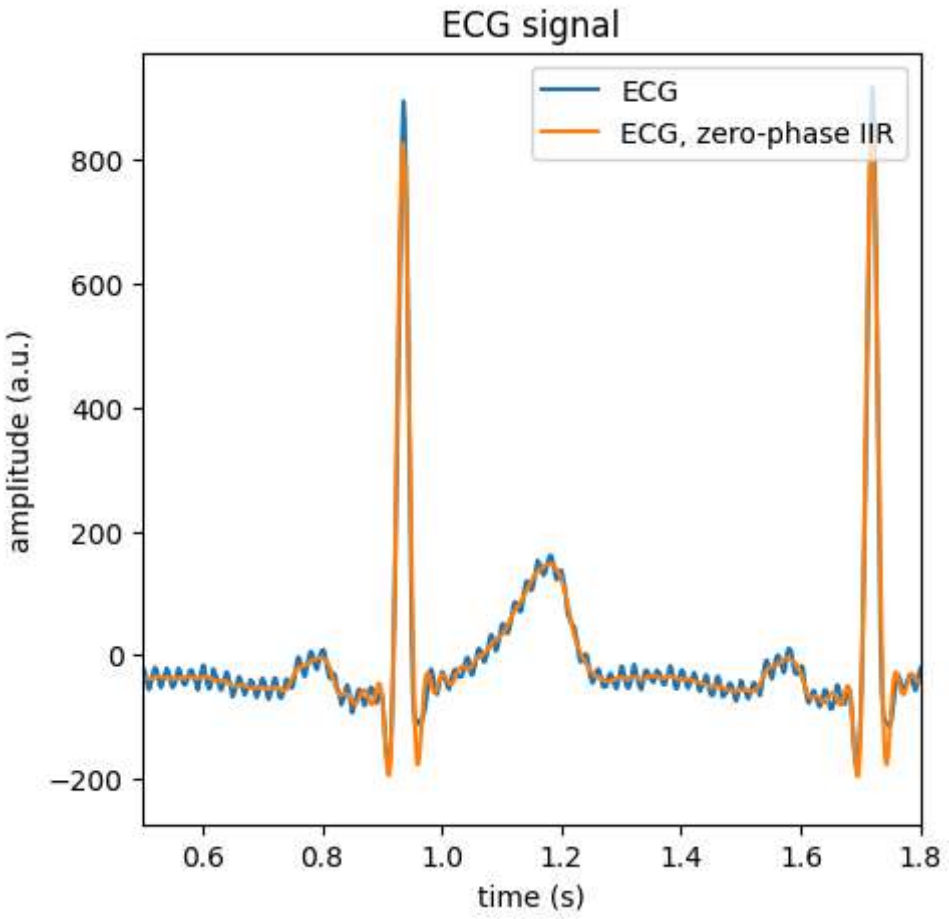
Filter the signal

In [17]:
```
x_f = sp.filtfilt(b ,a, x)
```

In [18]:
```
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)
```

Out[18]: (0.5, 1.8)



Linear phase FIR filter.

Define a FIR filter with the same properties.

Q: Comment the results (distortion of the PQRST, delay, ...).

Answer:

- The linear phase FIR filter preserves the shape and timing of the PQRST complex, avoiding phase distortion.
- FIR filters may introduce a constant group delay, shifting the signal in time, but this delay does not distort the waveform.
- The filtered ECG signal is clean, with the 50 Hz interference removed and the clinical features maintained.
- FIR filters require more coefficients (higher order) than IIR filters for similar performance, which can increase computational cost.
- For offline analysis, the delay can be compensated, and the linear phase property is highly beneficial for accurate ECG interpretation.

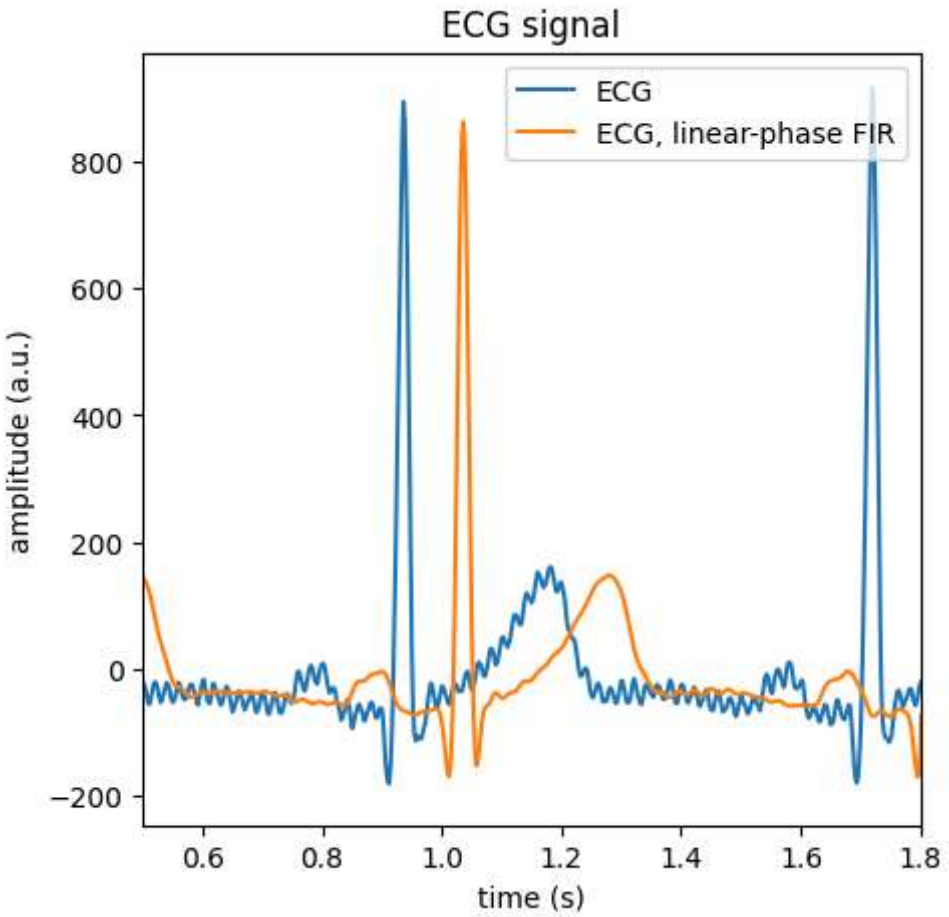
length of the filter

In [19]:
```
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)
```

In [20]:
```
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)
```

Out[20]: (0.5, 1.8)



Part 2: Breathing estimation

The objective of this exercise is that you analyse the code provided and make the link with the curse. 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

In [1]:
```
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

In [2]:
```
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.
Answer: The figure shows the raw breathing signal, which contains an oscillatory pattern from inhalation/exhalation cycles, and its Hilbert envelope showing the instantaneous amplitude of this oscillation. Since the raw signal is not narrowband (contains noise, drifts, and harmonics) it looks irregular, and the envelope does not match every peak perfectly.
Q: Why the envelope does not follow the maxima of the signal
Answer: In this plot, the envelope does trach the peaks of the signal but does not drop to the minima. This is because it represents the instantaneous amplitude, which is always non-negative since it measures overall magnitude of the oscillation at each moment, not actual waveforms values at each point.

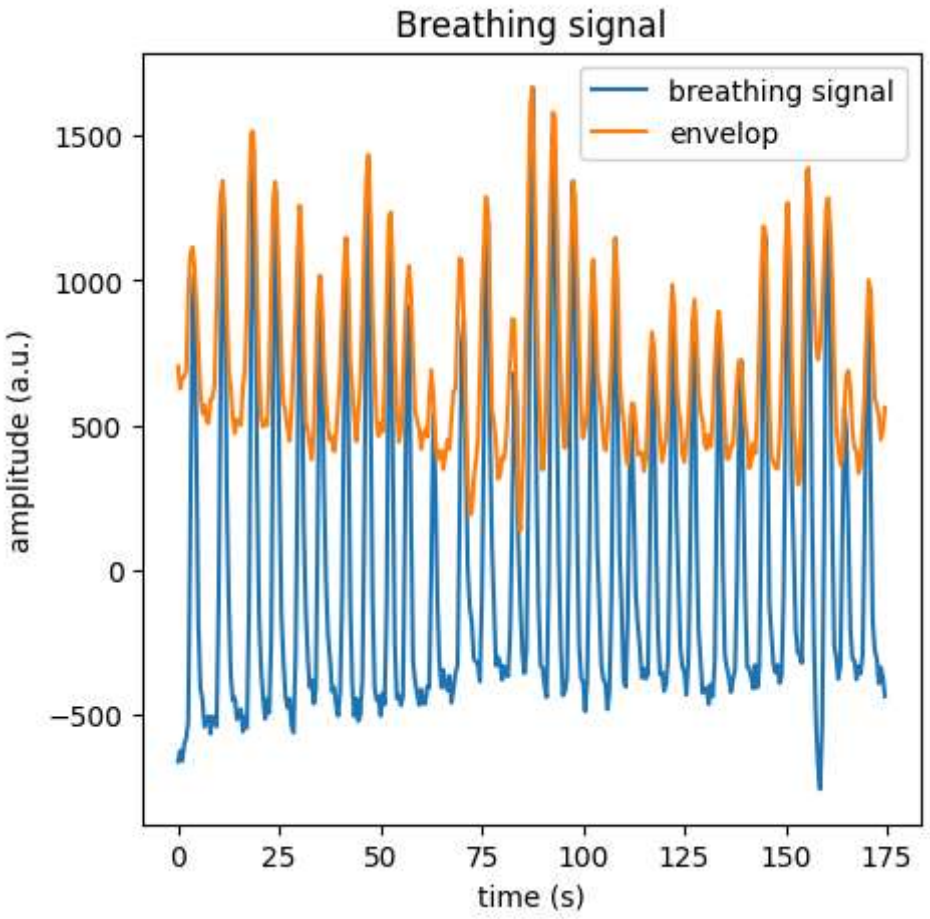
compute the analytical signal of x (Hilbert transform)

In [3]:
```
xa = sp.hilbert(x)
```

plot the signal

In [4]:
```
py.figure(1, figsize=[5,5])
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')
```

Out[4]:
```
Text(0.5, 1.0, 'Breathing signal')
```



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
Answer: After bandpass filtering (removing the high-frequency noise and the low-frequency drift) the breathing signal is smoother and satisfies the narrowband condition. The Hilbert envelope now follows the cycles more consistently.
Q: How is the estimation of the amplitude envelope.
Answer: The amplitude envelope estimation is more reliable and accurate than the raw signal and the envelope shows the true breathing cycles more clearly without being distorted by noise or unrelated frequency components.

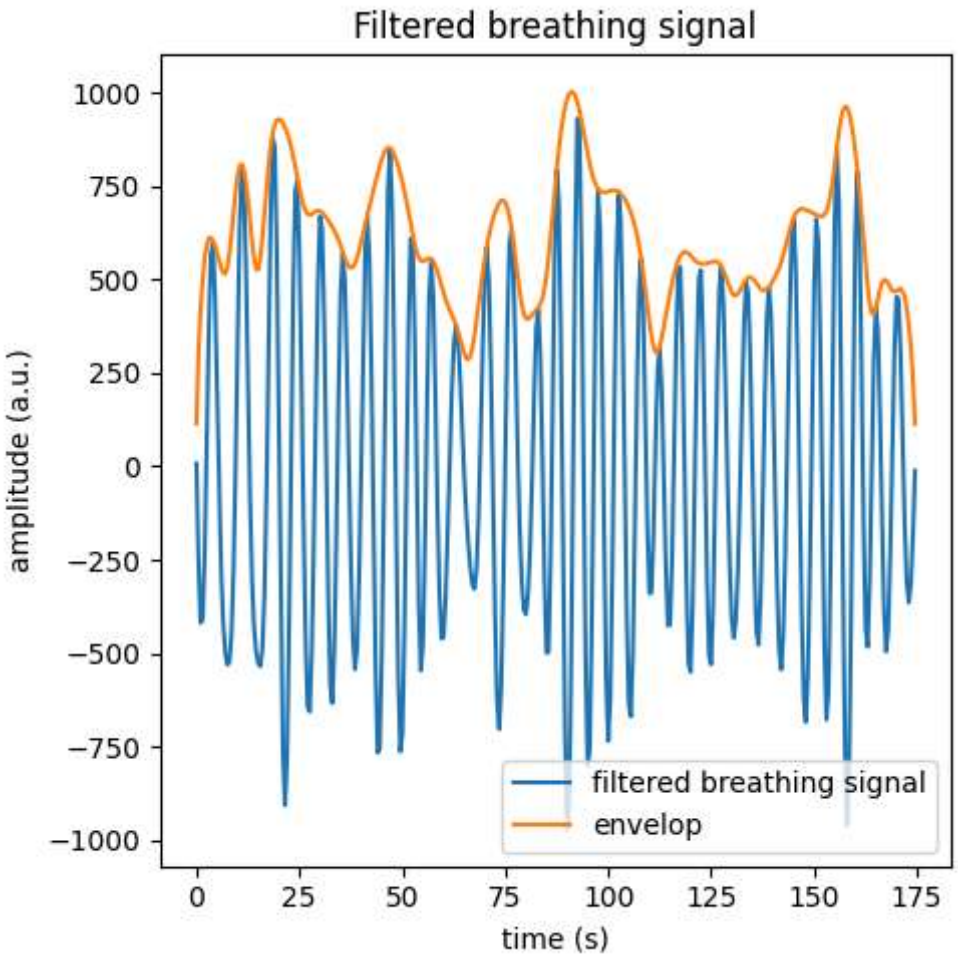
Analogic limit of the passband frequency

In [ ]:
```
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)
```

Compute the Hilbert transform.

In [6]:
```
xa = sp.hilbert(x_bp)
```

In [15]:
```
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='lower right')
py.tight_layout()
py.show()
```



The angle of the Hilbert transform gives the instaneous phase of the signal.

Q: Comment the figure.

Answer: The wrapped phase oscillates between $-\pi$ and π and jumps abruptly at the end of every cycle. The unwrapped phase removes this jump and produces a smooth and steadily increasing curve. The slope of this curve shows how quickly the breathing cycles are occuring.

Q: What is the role of the unwrap function

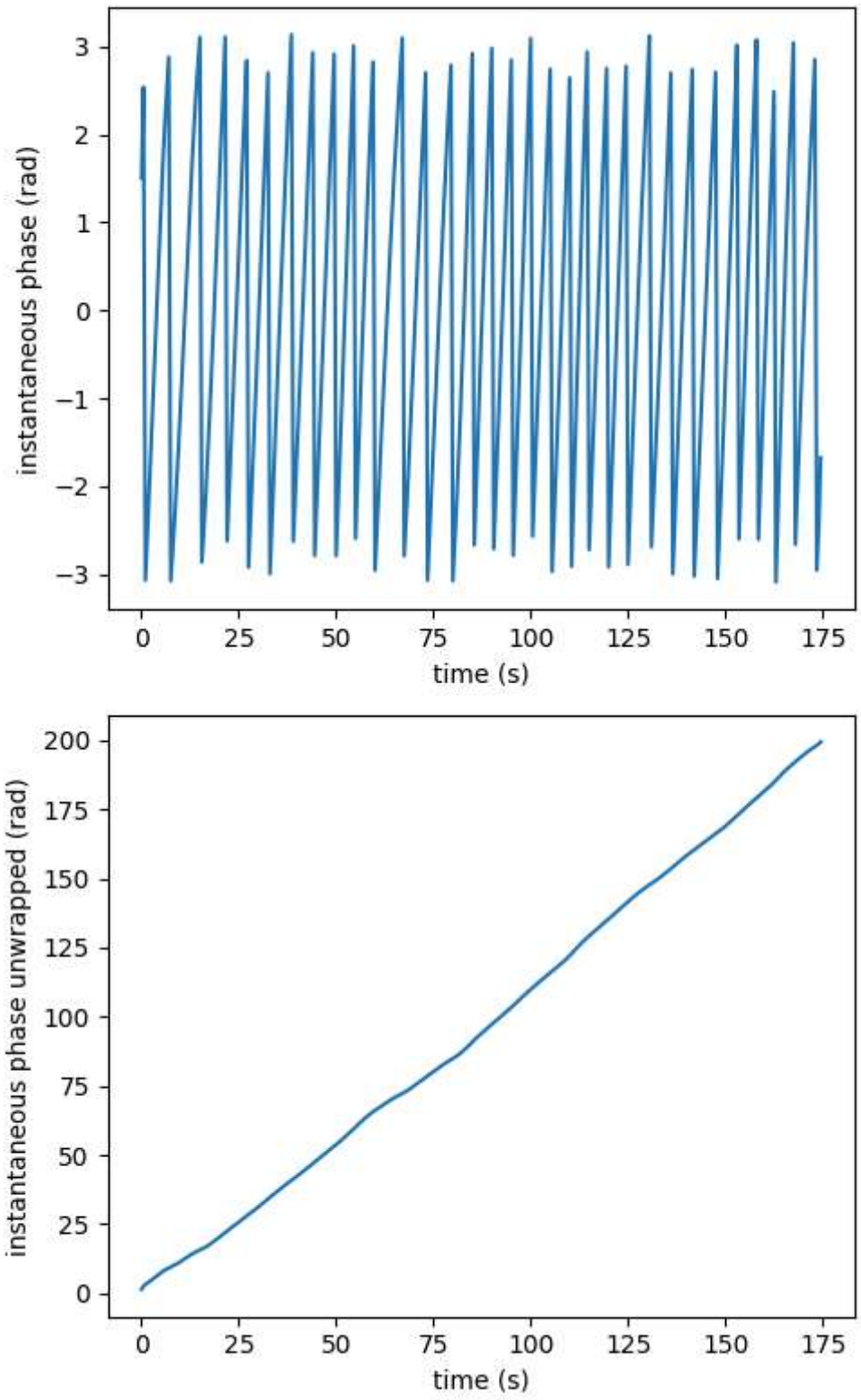
Answer: It removes the $\pm 2\pi$ jumps between the cycles in the phase representation, giving a continuous phase curve. This is important for calculating the instantaneous frequency, because without unwrapping the derivative would be dominated by the spikes from the jumps and meaningless.

estimate the instantaneous phase from the Hilbert transform

In [8]:
```
phi_xa = np.angle(xa)
# phase is bounded between -pi and pi -> reconstruct continuous signal
phi_xa_unw = np.unwrap(phi_xa)
```

In [16]:
```
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)')
py.tight_layout()
py.show()
```



The time derivate of the instantaneous phase is the instantaneous frequency of the signal.

Q: Comment the figure.

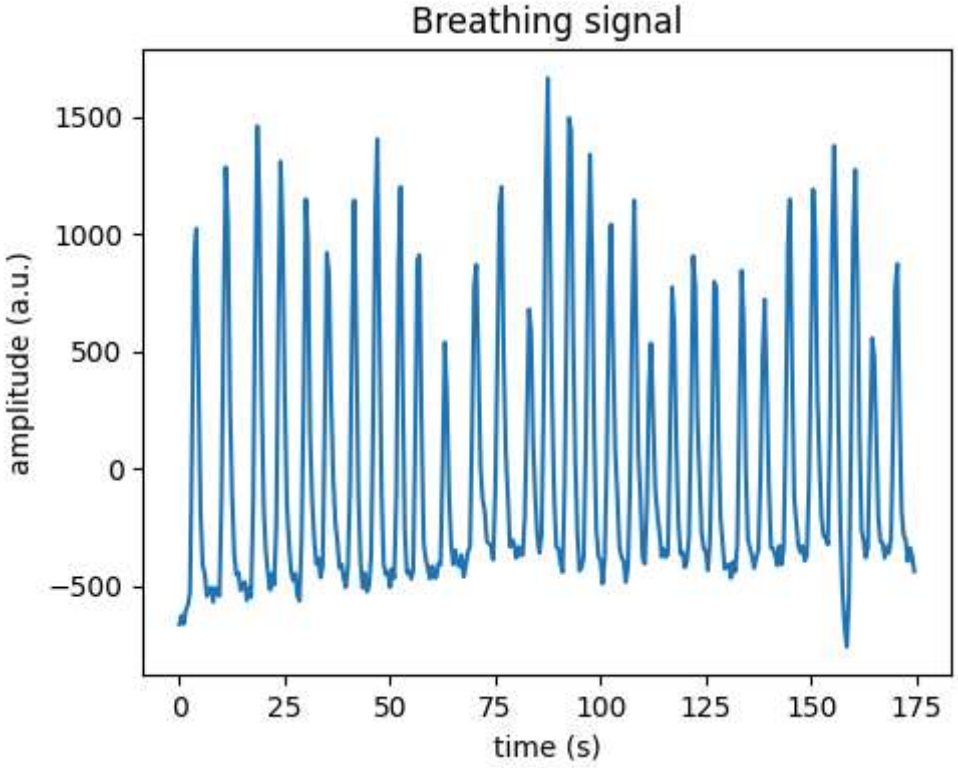
Answer: The first plot shows the raw breathing signal with its oscillatory pattern, while

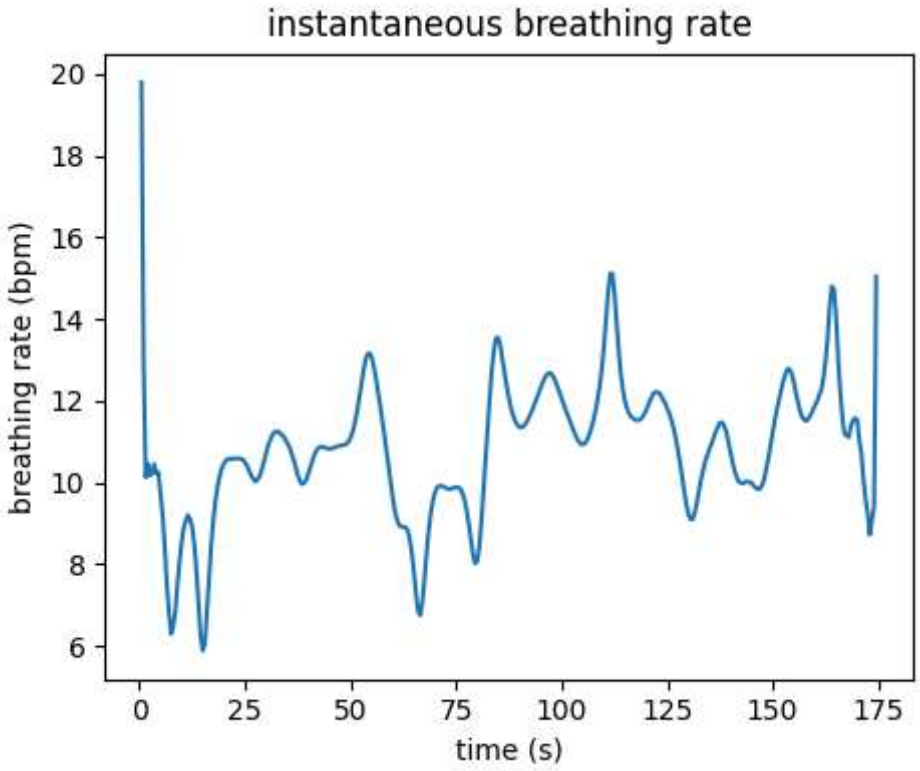
the second one shows the instantaneous breathing rate (in bpm) calculated from the derivative of the unwrapped phase. The breathing rate fluctuates around a stable value, showing how respiration rate evolves over time.
Q: Compare the original waveform with the estimation of the breathing frequency
Answer: The waveform shows the actual inhalation/exhalation cycles, while the frequency estimation condenses this into the breathing rate at each moment. It gives a direct measure of breathing dynamics over time, which is easier to interpret and analyze than counting cycles in the original waveform.

compute the derivative of the phase (angular frequency).

In [10]:
```
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
```

In [18]:
```
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)')
py.title('instantaneous breathing rate')
py.tight_layout()
py.show()
```


Breathing signal
amplitude (a.u.)
1500
1000
500
0
−500
0
25
50
75
100
125
150
175
time (s)


instantaneous breathing rate
breathing rate (bpm)
20
18
16
14
12
10
8
6
0
25
50
75
100
125
150
175
time (s)

Part 3: Hand washing detection

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

In [14]:
```
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

In [15]:
```
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
```

The signal is an acceleration signal measured at the wrist.
The signal records the acceleration during the hand washing protocol in an hospital. The goal is to detect the hand washing sequence.
The hand washing protocol produced rhythmical movements.
The protocol is known to take place between 20 and 30 seconds.

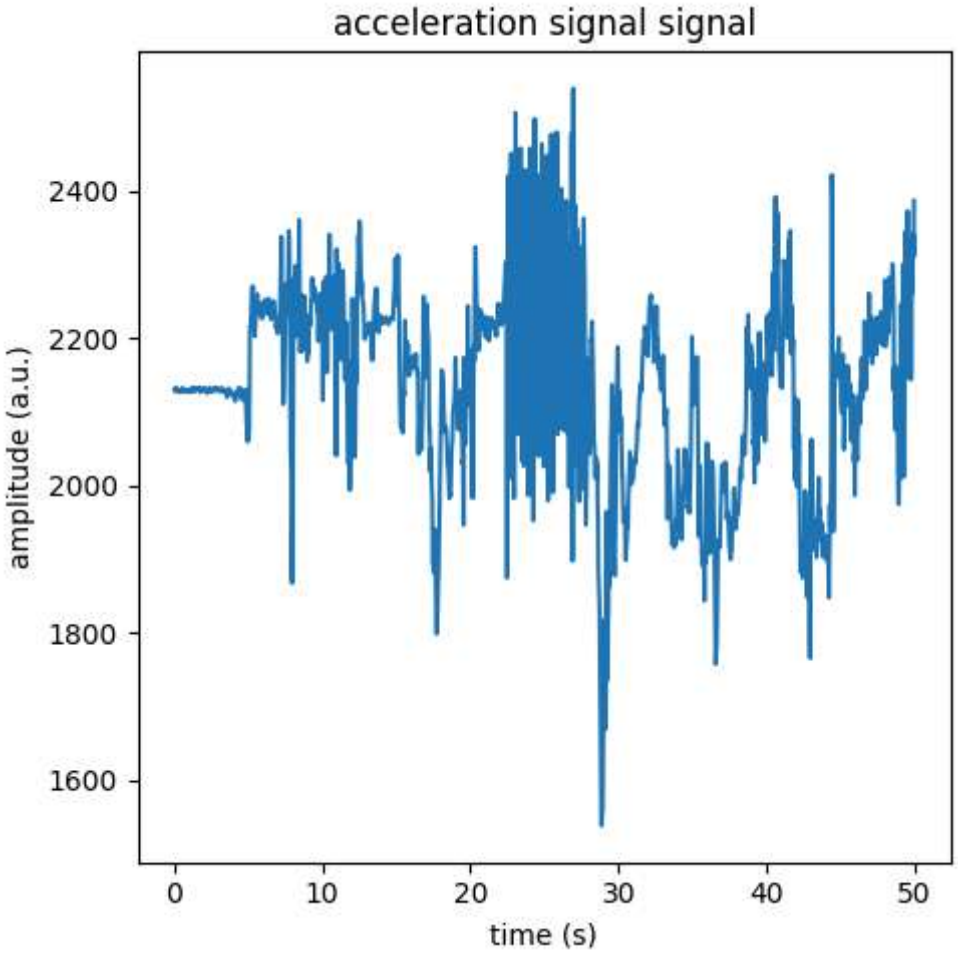
Plot time signal.
Q: Comment the figure.
Answer: Between 20 to 30 seconds, there is an increase in amplitude and oscillations, which corresponds to the hand washing protocol. This pattern reflects the repetitive hand movements typical of the handwashing protocol. The peak around 28-30 seconds likely corresponds to a faster or stronger motion during the protocol. Outside of the 20-30 second interval, the signal has smaller irregular amplitudes, likely representing normal wrist movement as well as noise. This clear change in signal characteristics helps identify the handwashing activity.

Compute the FFT of the signal

In [16]:
```
x_fft = np.fft.fft(x)
# Determine the frequency scale
f_fft = np.arange(len(x_fft))/len(x_fft)*fs
```

plot the signal

In [27]:
```
py.figure(1, figsize=[5,5])
py.clf()
py.plot(t, x)
py.xlabel('time (s)')
py.ylabel('amplitude (a.u.)')
py.title('acceleration signal signal')
py.tight_layout()
py.show()
```



High pass the signal.

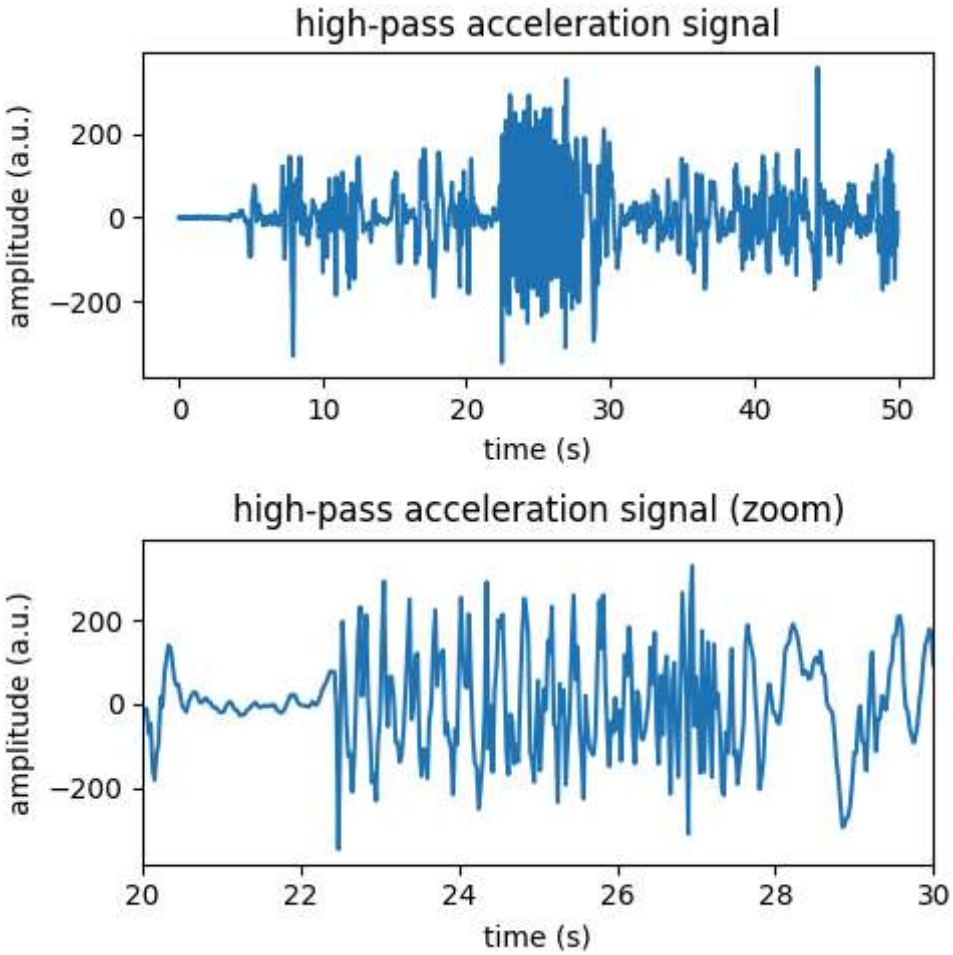
Q: Comment the figure.

Answer: After applying the high-pass filter, the baseline drift and slow fluctuations are removed, resulting in a linearized signal baseline. This makes it easier to observe the true features and rapid changes in acceleration that correspond to hand movements. The high-pass filtering enhances the visibility of the relevant activity, allowing us to better distinguish the hand washing protocol from background noise and unrelated motion. In fact, in the zoomed signal around the most active interval, we cans see an oscillatory acceleration. This corresponds to the strokes of the hands as the person washes them back en fourth according to the protocol.

high-pass filter with cutoff frequency of 0.5 Hz

In [18]:
```
b, a = sp.butter(4, 0.5/fs*2, btype='high')
# zero-phase filtering of the signal
x_hp = sp.filtfilt(b, a, x)
```

In [19]:
```
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)
py.tight_layout()
```



After applying a high-pass filter, the signal removes slow, baseline fluctuations and highlights rapid changes in acceleration. This makes the rhythmic handwashing movements more visible, especially in the 20–30 second interval. The filter helps isolate the relevant activity by suppressing low-frequency noise and drift, improving the clarity of the handwashing sequence in the signal.

Band pass the signal between 2.4 and 3.2 Hz.

Q: Based on previous figure, comment the selection of the frequencies.

Answer: The selected band-pass frequencies (2.4–3.2 Hz) correspond to the dominant frequency range of the rhythmic handwashing movements observed in the acceleration signal between 20 and 30 seconds. This range isolates the periodic activity associated with handwashing, filtering out both slower baseline fluctuations and higher-frequency noise, thus focusing on the relevant motion.

Q: Why zero phase filter (filtfilt) is used?

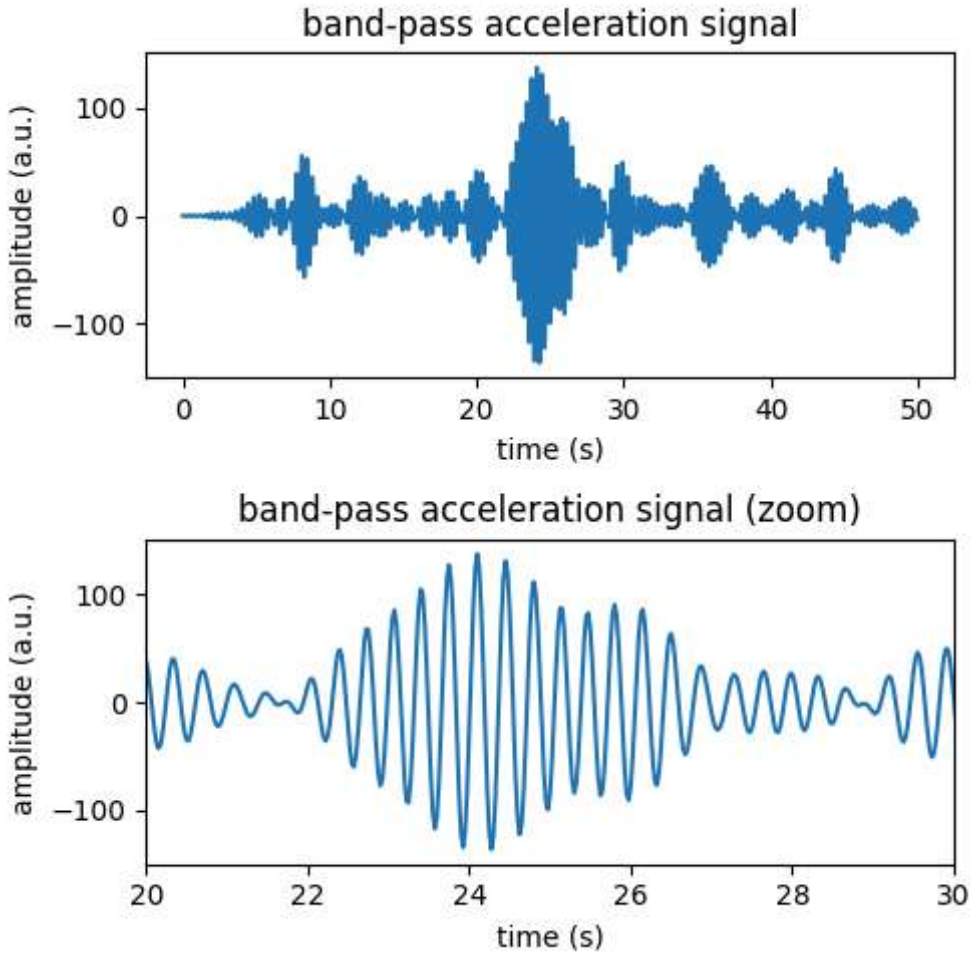
Answer: The zero-phase filter (`filtfilt`) is used to avoid phase distortion in the filtered signal. It applies the filter forward and backward, ensuring that the timing and shape of the detected handwashing events remain accurate and are not shifted or distorted by the filtering process.

Analogic limit of the passband frequency

In [ ]:
```
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)
```

In [ ]:
```
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.tight_layout()
py.xlim(20, 30)
py.show()
```

Out[ ]:
```
(20.0, 30.0)
```



Low-pass filter of the power of the band-pass signal.
Q: Why use the power of the acceleration signal?

Answer: The power of the acceleration signal (i.e., squaring the band-pass filtered signal) emphasizes periods of strong rhythmic activity. This helps to distinguish handwashing movements from background noise or less intense motions, making detection more robust. Additionally, squaring the acceleration signal removes the sign dependency. All of the accelerations become positive, therefore it is possible to see what sections have the highest magnitude.

Q: How is the detection of hand washing obtained?

Answer: Detection is achieved by low-pass filtering the power of the band-pass signal to smooth out rapid fluctuations, then applying a threshold. When the smoothed power exceeds this threshold, it indicates the presence of handwashing activity.

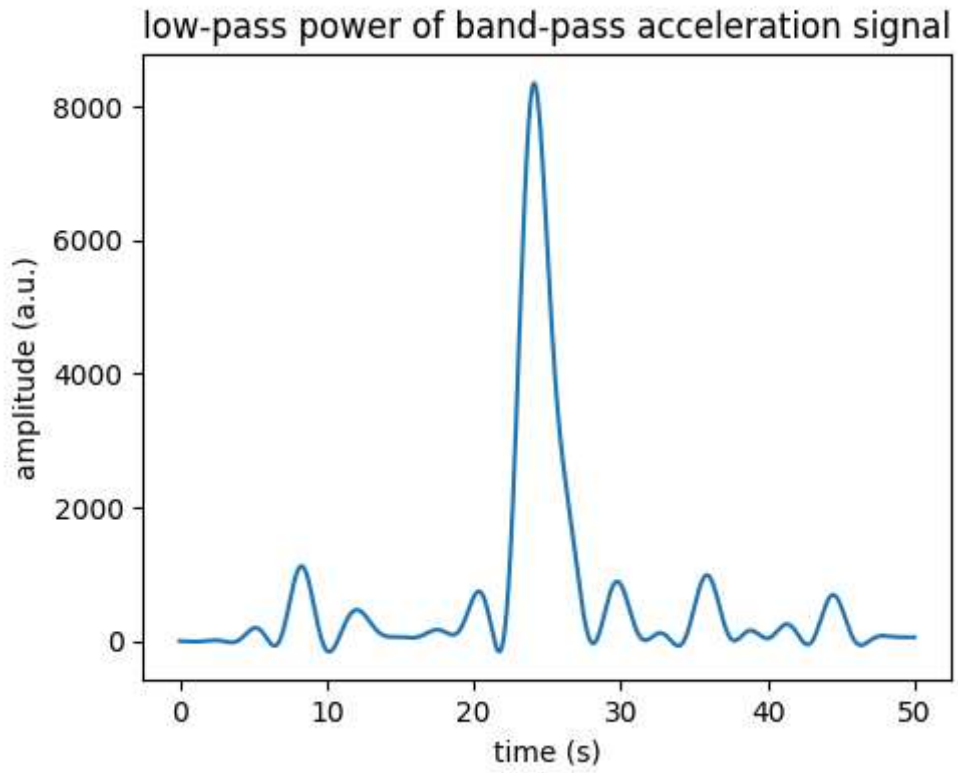
Analogic limit of the passband frequency

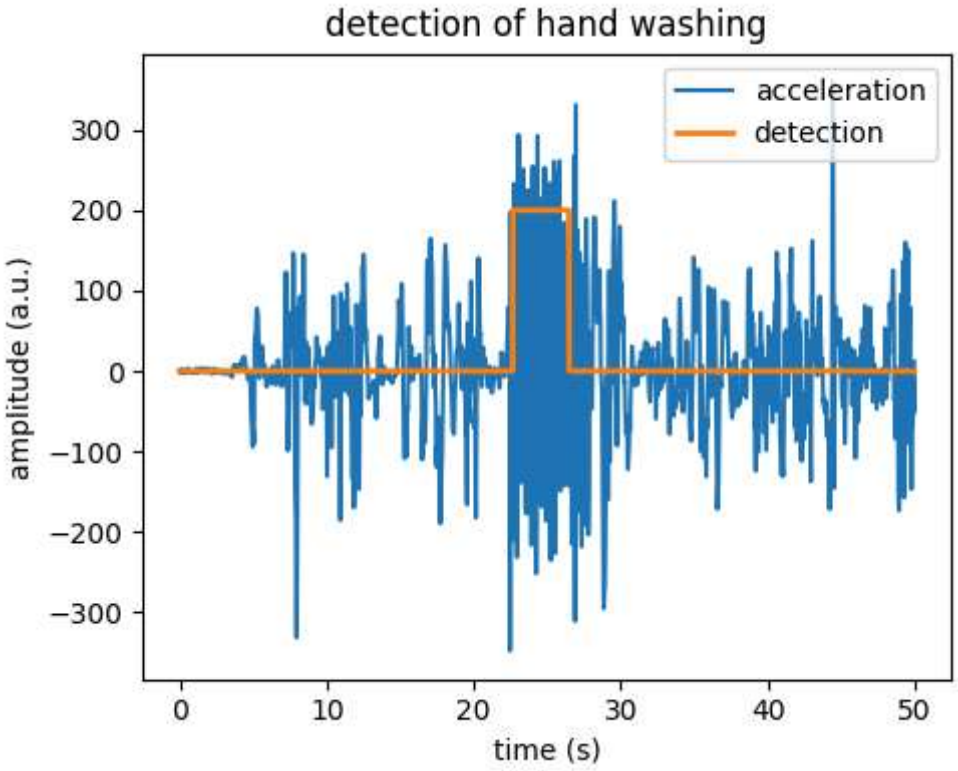
In [22]:
```
f_pass = 0.4
# Analogic limit of the stopband frequency
f_stop = 0.8
# Convertion into Nyquist frequency
f_pass_N = f_pass/fs*2
f_stop_N = f_stop/fs*2
# Max attenutation in passband (dB)
```

```
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
```

In [23]:

```
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')
py.show()
```


low-pass power of band-pass acceleration signal
amplitude (a.u.)
8000
6000
4000
2000
0
0
10
20
30
40
50
time (s)


detection of hand washing
acceleration
detection
amplitude (a.u.)
300
200
100
0
−100
−200
−300
0
10
20
30
40
50
time (s)