ISS Projekt 2023/24

Honza Pavlus, Honza Brukner a Honza Černocký, ÚPGM FIT VUT

6.11. 2023

1. Úvod

V projektu budete pracovat se biomedicínskými signály a to konkrétně se signálem elektrokardiogramu EKG. Vyzkoušíte si filtraci tohoto druhu signálu, abyste dostali krásné EKG křivky, které můžete vidět ve filmech. Dále si zkusíte vybudovat jednoduchý, ale účinný detektor QRS a ti, kteří se vrhnou i na bonusový úkol, si zkusí odhalit srdeční patologii. K dispozici dostanete každý 3 nahrávky jednokanálového EKG signálu, jeden zdravý a dva s různými patologiemi.

Projekt je nejlépe možno řešit v Python-u a to přímo v dodaném Python notebooku, který si můžete zkopírovat do vlastního Google Colabu. Projekt je také možno řešit v Matlab-u, Octave, Julii, jazyce C, Java nebo v libovolném jiném programovacím či skriptovacím jazyce. Je možné použít libovolné knihovny. Projekt se nezaměřuje na "krásu programování", není tedy nutné mít vše úhledně zabalené do okomentovaných funkcí (samozřejmě se ale okomentovaný kód lépe opravuje a to hlavně v případě podivných výsledků), ošetřené všechny chybové stavy, atd. Důležitý je výsledek.

Vaši práci odevzdáváte vyexportovanou do dvou souborů: (1) do PDF souboru login.pdf, (2) do Python notebooku login.ipynb. PDF musí obsahovat výsledky prokazatelně vytvořené Vaším kódem. V případě řešení projektu v jiném jazyce nebo prostředí než v dodaném Python notebooku, je prvním souborem protokol v PDF, druhý soubor je archiv s Vaším kódem. Ten musí být spustitelný na standardní fakultní distribuci Windows nebo Linuxu.

3. Vstup

Pro řešení projektu má každý student/ka k disposici osobní soubor se zdravým signálem (sinusovým rytmem): *login.wav*, kde login je váš xlogin popřípadě VUT číslo (pro studenty FSI). Dále jsou k disposici ještě další dva signály: *FIS.wav* a *KES.wav*. První signál obsahuje fibrilaci a druhý komorovou extrasystolu. Tyhle dva soubory jsou pro všechny společné a využijete je při řešení bonusového úkolu.

In [115]:

```
#Načtení Vašeho signálu - xlogin99 nahraďte Vaším loginem
import soundfile as sf

!wget https://www.fit.vutbr.cz/study/courses/ISS/public/proj2023-24/xmovse00.wav
!wget https://www.fit.vutbr.cz/study/courses/ISS/public/proj2023-24/FIB.wav
!wget https://www.fit.vutbr.cz/study/courses/ISS/public/proj2023-24/KES.wav

x, fs = sf.read("xmovse00.wav")
```

```
--2023-12-17 15:42:05--  https://www.fit.vutbr.cz/study/courses/ISS/public/proj2023-24/xm
ovse00.wav
Resolving www.fit.vutbr.cz (www.fit.vutbr.cz)... 147.229.9.23, 2001:67c:1220:809::93e5:91
7
Connecting to www.fit.vutbr.cz (www.fit.vutbr.cz)|147.229.9.23|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 10044 (9.8K) [audio/x-wav]
Saving to: 'xmovse00.wav.3'

xmovse00.wav.3      100%[===================>]   9.81K  --.-KB/s    in 0s

2023-12-17 15:42:07 (140 MB/s) - 'xmovse00.wav.3' saved [10044/10044]

--2023-12-17 15:42:07--  https://www.fit.vutbr.cz/study/courses/ISS/public/proj2023-24/FI
B.wav
Resolving www.fit.vutbr.cz (www.fit.vutbr.cz)... 147.229.9.23, 2001:67c:1220:809::93e5:91
```

```
7
Connecting to www.fit.vutbr.cz (www.fit.vutbr.cz)|147.229.9.23|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 10044 (9.8K) [audio/x-wav]
Saving to: 'FIB.wav.3'

FIB.wav.3           100%[===================>]   9.81K  --.-KB/s    in 0s

2023-12-17 15:42:07 (152 MB/s) - 'FIB.wav.3' saved [10044/10044]

--2023-12-17 15:42:07--  https://www.fit.vutbr.cz/study/courses/ISS/public/proj2023-24/KE
S.wav
Resolving www.fit.vutbr.cz (www.fit.vutbr.cz)... 147.229.9.23, 2001:67c:1220:809::93e5:91
7
Connecting to www.fit.vutbr.cz (www.fit.vutbr.cz)|147.229.9.23|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 10044 (9.8K) [audio/x-wav]
Saving to: 'KES.wav.3'

KES.wav.3           100%[===================>]   9.81K  --.-KB/s    in 0s

2023-12-17 15:42:07 (110 MB/s) - 'KES.wav.3' saved [10044/10044]
```

4. Úkoly

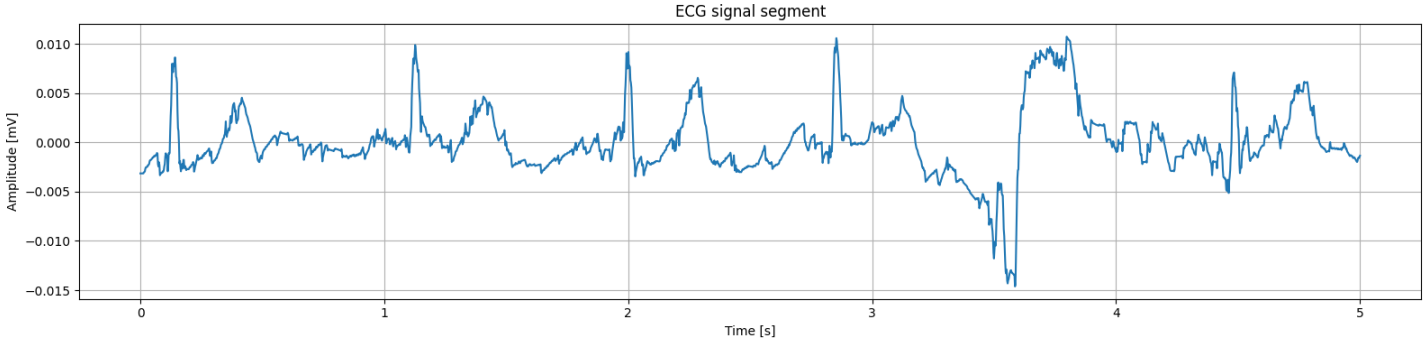
4.1. [2.5b] Nahrání a zobrazení EKG signálu

Nezapomeňte na popisy os u jednotlivých grafů.

a) [1b] Nahrajte EKG signál login.wav, vyberte 5-sekundový úsek a zobrazte ho v časové doméně. Pro nahrání signálu použijte knihovny numpy a soundfile.

In [116]:

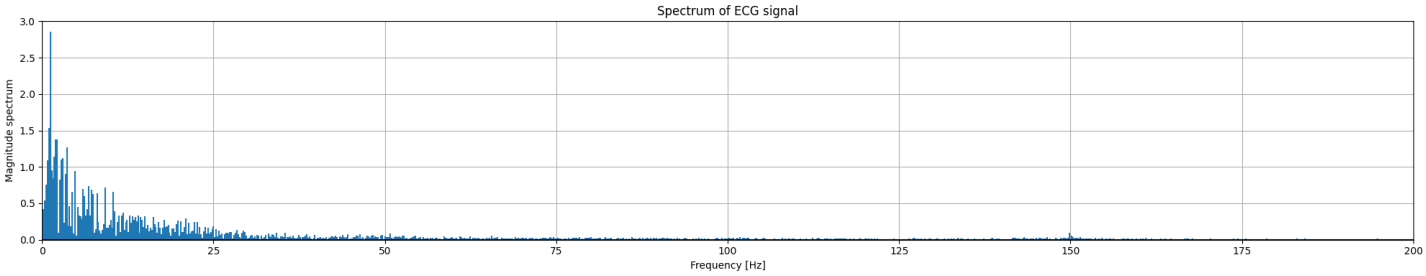
```
import numpy as np
import matplotlib.pyplot as plt

start_index = int(0 * fs)
end_index = int(5 * fs)
segment = x[start_index:end_index]

plt.figure(figsize=(16, 4))
plt.plot(np.linspace(0,5, len(segment)), segment)
plt.title('ECG signal segment')
plt.xlabel('Time [s]')
plt.ylabel('Amplitude [mV]')
plt.grid(True)
plt.tight_layout()
plt.show()
```



b) [1b] Spočítejte spektrum z 5 sekundového úseku nahraného signálu a zobrazte jej.

In [117]:

```
from scipy.fft import fft

start_index = int(0 * fs)
end_index = int(5 * fs)
segment = x[start_index:end_index]

fft_result = np.abs(fft(segment))
freq_values = np.fft.fftfreq(len(segment), 1/fs)

plt.figure(figsize=(20, 4))
plt.stem(freq_values, fft_result, basefmt='black', markerfmt=' ')
plt.title('Spectrum of ECG signal')
plt.xlabel('Frequency [Hz]')
plt.ylabel('Magnitude spectrum')
plt.xlim(0, 200)
plt.ylim(0, 3)
plt.grid(True)
plt.tight_layout()
plt.show()
```



c) [0.5b] Ve spektru vidíte rušení na 50Hz nebo 60Hz a jejich harmonických frekvencích. Vysvětlete, čím je způsobeno.

This interference is caused by electrical grid noise due to the supplied 50 Hz or 60 Hz alternating current. In the laboratory where the ECG measurement was performed, electrical equipment generates noise at this frequency, which is captured by the measuring machine.

4.2. [3b] Převzorkujte nahraný signál

a) [2b] Převzorkujte signál na vzorkovací frekvenci 100 Hz, nezapomeňte na filtr pro antialiasing. Můžete například odstranit část spektra od $\frac{Fs}{2}$ nebo použít filtr dolní propusti.

In [118]:

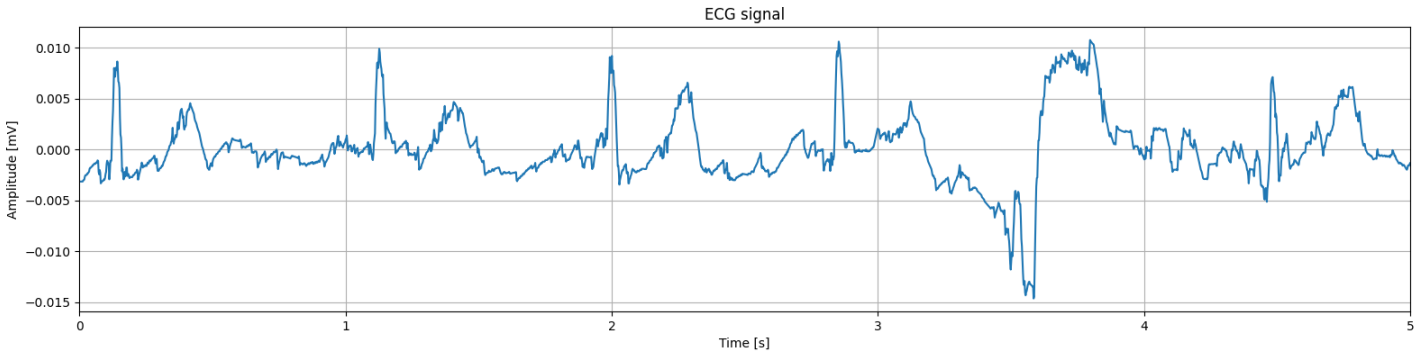
```
from scipy.signal import lfilter, resample, butter, spectrogram

cutoff_frequency = 50
nyquist_frequency = 0.5 * fs
normalized_cutoff = cutoff_frequency / nyquist_frequency

b, a = butter(4, normalized_cutoff, btype='low')

x_filtered = lfilter(b, a, x)

x_resampled_filtered = resample(x_filtered, int(len(x_filtered) * 100/fs))
```

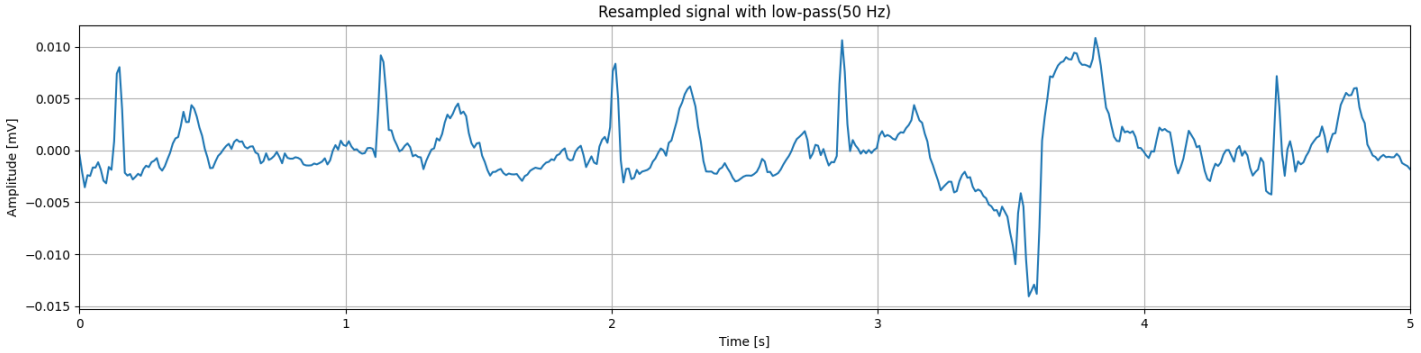
b) [1b] Zobrazte 5 sekundový úsek původního a převzorkovaného signálu v časové doméně a zobrazte i jejich spektra.

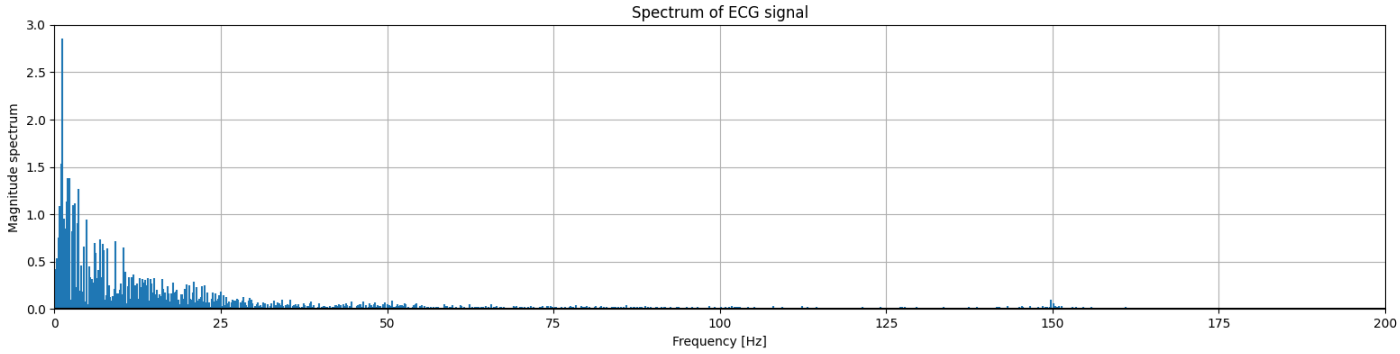
In [119]:

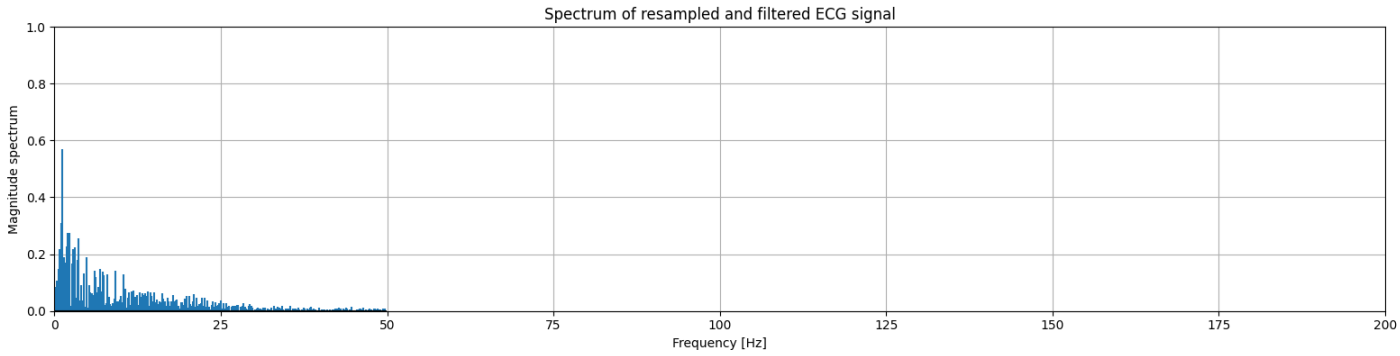
```
start_index = int(0 * fs)
end_index = int(5 * fs)
segment = x[start_index:end_index]

start_index_resampled_filtered = int(0 * 100)
```

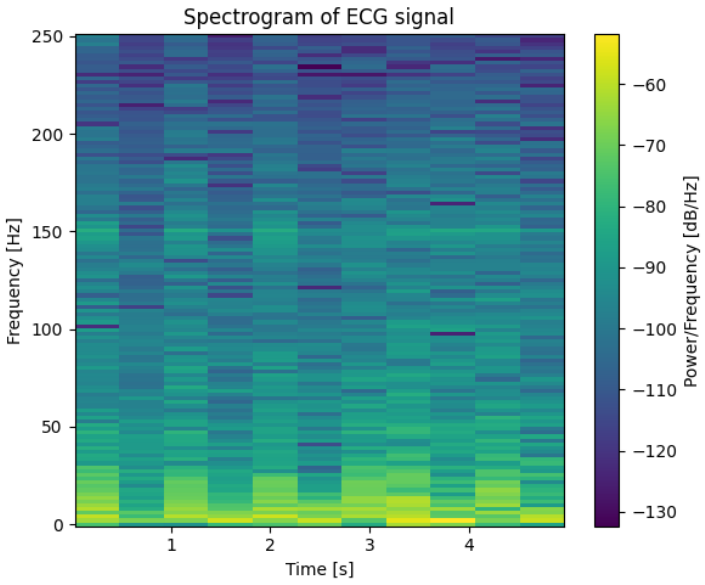
```
end_index_resampled_filtered = int(5 * 100)
segment_resampled_filtered = x_resampled_filtered[start_index_resampled_filtered:end_inde
x_resampled_filtered]

fft_result_orig = np.abs(fft(segment))
freq_values_orig = np.fft.fftfreq(len(segment), 1/fs)

fft_result_filt = np.abs(fft(segment_resampled_filtered))
freq_values_filt = np.fft.fftfreq(len(segment_resampled_filtered), 1/100)

frequencies_orig, times_orig, Sxx_orig = spectrogram(segment, fs)
frequencies_filt, times_filt, Sxx_filt = spectrogram(segment_resampled_filtered, 100)

plt.figure(figsize=(16, 16))
plt.subplot(4, 1, 1)
plt.plot(np.linspace(0,5, len(segment)), segment)
plt.title('ECG signal')
plt.xlabel('Time [s]')
plt.ylabel('Amplitude [mV]')
plt.xlim(0, 5)
plt.grid(True)

plt.subplot(4, 1, 2)
plt.plot(np.linspace(0,5, len(segment_resampled_filtered)), segment_resampled_filtered)
plt.title(f'Resampled signal with low-pass({cutoff_frequency} Hz)')
plt.xlabel('Time [s]')
plt.ylabel('Amplitude [mV]')
plt.xlim(0, 5)
plt.grid(True)

plt.subplot(4, 1, 3)
plt.stem(freq_values_orig, fft_result_orig, basefmt='black', markerfmt=' ')
plt.title('Spectrum of ECG signal')
plt.xlabel('Frequency [Hz]')
plt.ylabel('Magnitude spectrum')
plt.xlim(0, 200)
plt.ylim(0, 3)
plt.grid(True)

plt.subplot(4, 1, 4)
plt.stem(freq_values_filt, fft_result_filt, basefmt='black', markerfmt=' ')
plt.title('Spectrum of resampled and filtered ECG signal')
plt.xlabel('Frequency [Hz]')
plt.ylabel('Magnitude spectrum')
plt.xlim(0, 200)
plt.ylim(0, 1)
plt.grid(True)
plt.tight_layout()
plt.show()

fig, axs = plt.subplots(1, 2, figsize=(12, 5))

pcm1 = axs[0].pcolormesh(times_orig, frequencies_orig, 10 * np.log10(Sxx_orig))
axs[0].set_title('Spectrogram of ECG signal')
axs[0].set_ylabel('Frequency [Hz]')
axs[0].set_xlabel('Time [s]')
fig.colorbar(pcm1).set_label('Power/Frequency [dB/Hz]')

pcm2 = axs[1].pcolormesh(times_filt, frequencies_filt, 10 * np.log10(Sxx_filt))
axs[1].set_title('Spectrogram of resampled and filtered  ECG signal')
axs[1].set_ylabel('Frequency [Hz]')
axs[1].set_xlabel('Time [s]')
fig.colorbar(pcm2).set_label('Power/Frequency [dB/Hz]')

plt.tight_layout()
plt.show()
#
#
#
#
#
#
```

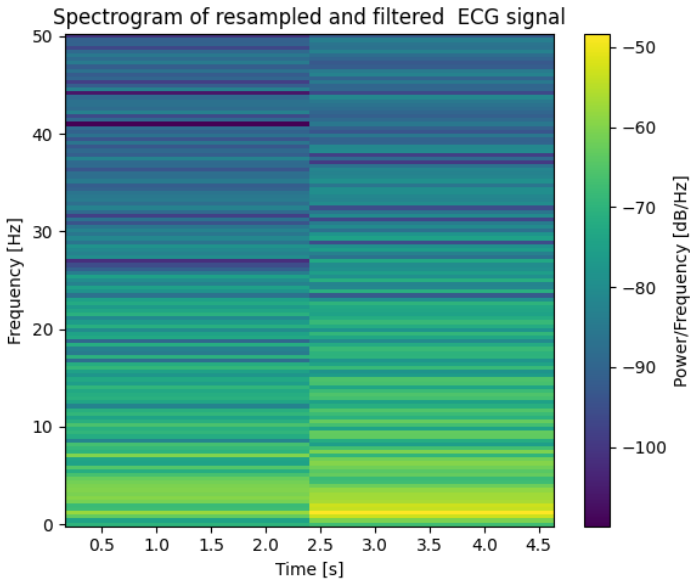










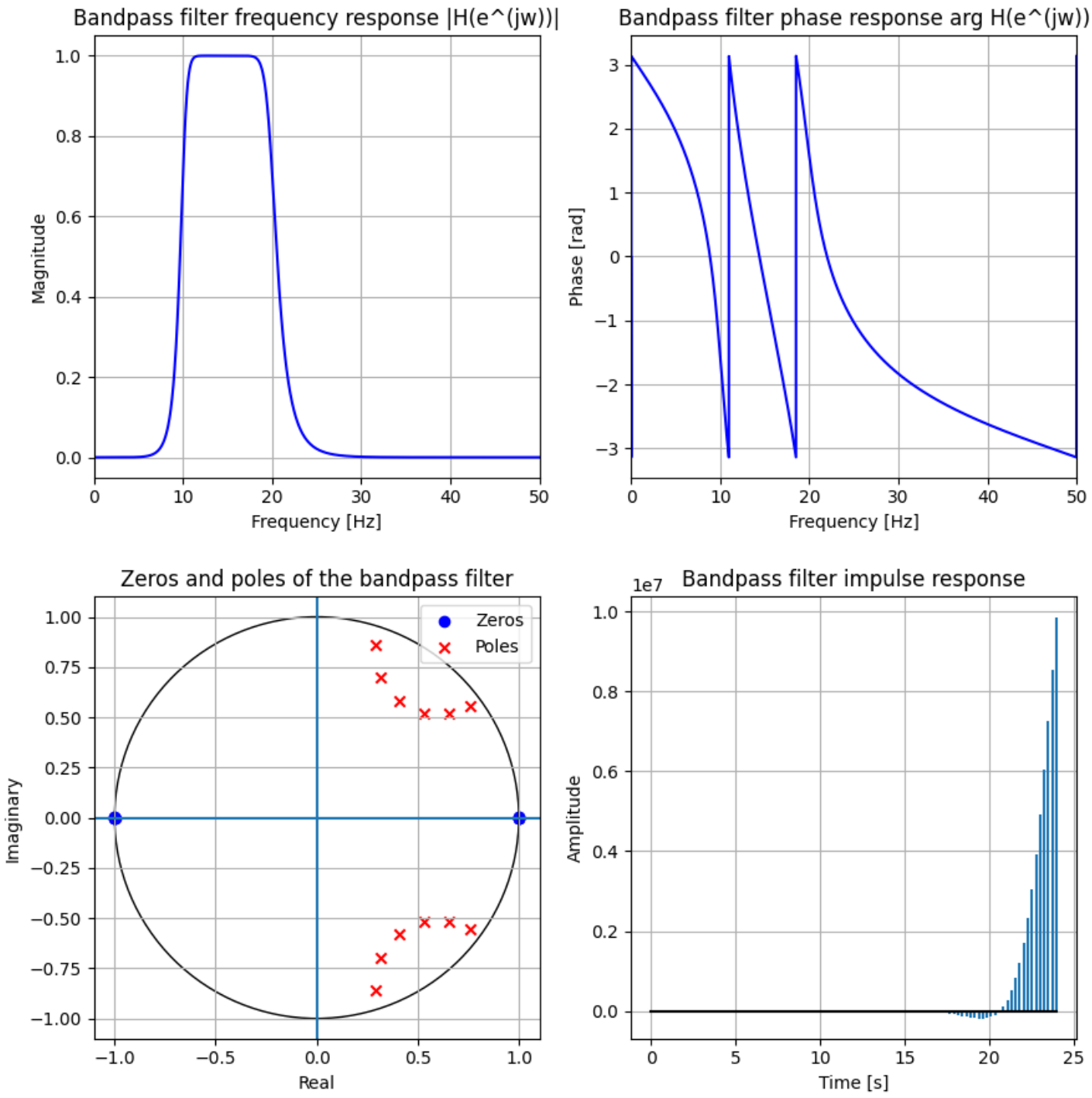


4.3. [4b] Vyfiltrujte nahraný signál pásmovou propustí 10Hz-20Hz

a) [2b] Vytvořte filtr pásmové propusti, možnosti jsou dvě: buďto filtrovat pomocí klasického návrhu filtrů, kde získáte koeficienty `a` a `b` (pomocí např. scipy.butter) a zobrazíte charakteristiku filtru + nuly a póly. Nebo se můžete vydat cestou filtrování ve frekvenční doméně, frekvenční charakteristiku vykreslete pomocí spektrální masky.

In [126]:

```
from scipy.signal import filtfilt, freqz, tf2zpk, impulse

new_fs = 100
x_resampled = resample(x, int(len(x) * new_fs/fs))

nyquist = 0.5 * new_fs
low = 10 / nyquist
high = 20 / nyquist
b, a = butter(6, [low, high], btype='band')

w, h = freqz(b, a, worN=8000)

plt.figure(figsize=(9, 9))
plt.subplot(2, 2, 1)
plt.plot(0.5 * new_fs * w / np.pi, np.abs(h), 'b')
plt.title('Bandpass filter frequency response |H(e^(jw))|')
plt.xlabel('Frequency [Hz]')
plt.ylabel('Magnitude')
plt.xlim(0, 50)
plt.grid(True)

plt.subplot(2, 2, 2)
plt.plot(0.5 * new_fs * w / np.pi, np.angle(h), 'b')
plt.title('Bandpass filter phase response arg H(e^(jw))')
plt.xlabel('Frequency [Hz]')
plt.ylabel('Phase [rad]')
plt.xlim(0, 50)
plt.grid(True)

zeros, poles, _ = tf2zpk(b, a)

plt.subplot(2, 2, 3)
plt.scatter(np.real(zeros), np.imag(zeros), marker='o', color='b', label='Zeros')
plt.scatter(np.real(poles), np.imag(poles), marker='x', color='r', label='Poles')
unit_circle = plt.Circle((0, 0), 1, fill=False)
plt.gca().add_patch(unit_circle)
plt.title('Zeros and poles of the bandpass filter')
plt.xlabel('Real')
plt.ylabel('Imaginary')
plt.axhline(0)
plt.axvline(0)
plt.grid(True)
plt.legend()

t_imp, h_imp = impulse((b, a))

plt.subplot(2, 2, 4)
plt.stem(t_imp, h_imp, basefmt='black', markerfmt=' ')
plt.title('Bandpass filter impulse response')
plt.xlabel('Time [s]')
plt.ylabel('Amplitude')
plt.grid(True)
plt.tight_layout()
plt.show()
#
#
#
#
#
#
#
#
#
```

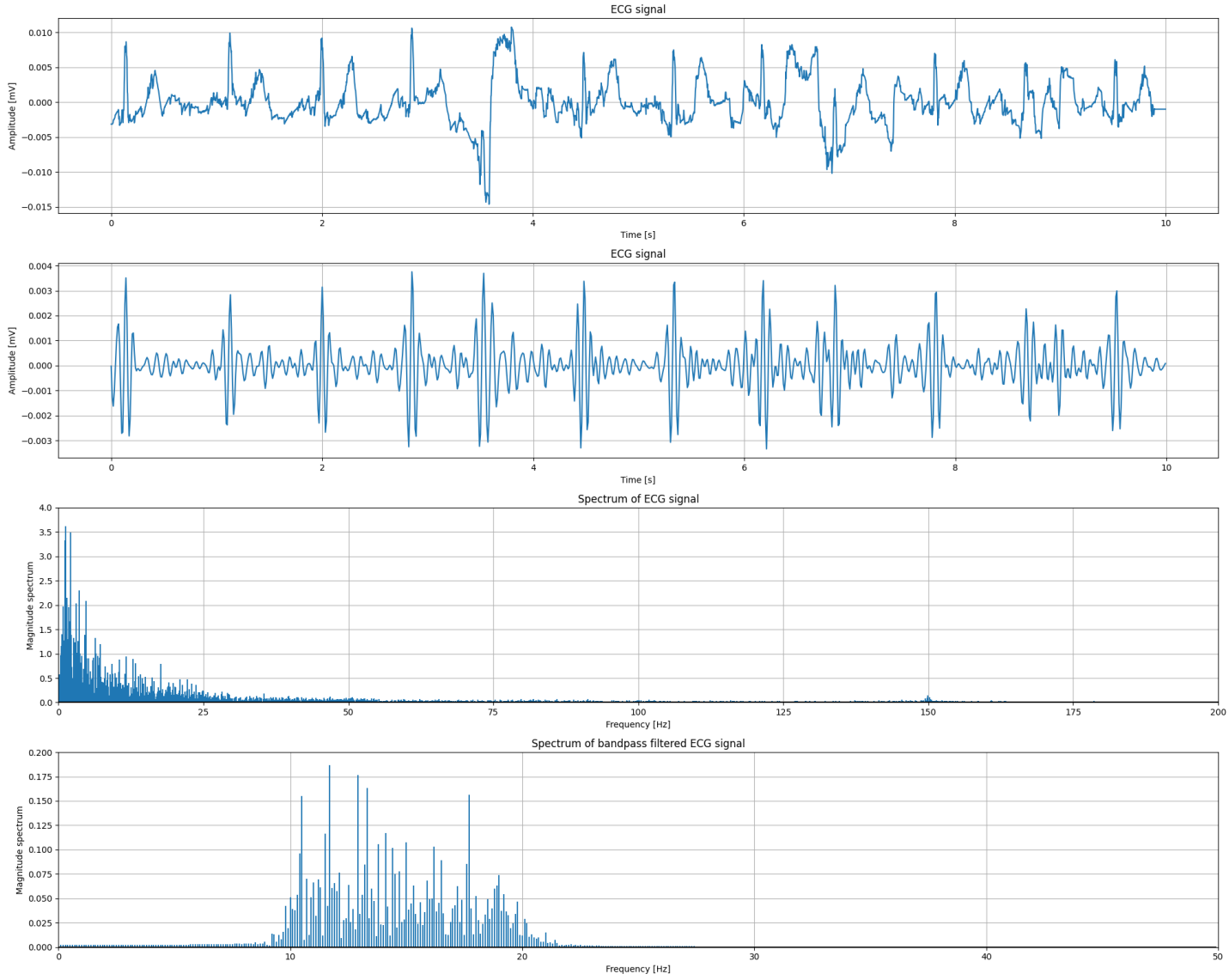


b) [1b] Použijte navržený filtr na nahraný signál. Pokud máte navržený klasický filtr, proveďte filtrování z obou stran, abyste se vyhnuli fázovému posunu, to za vás zajistí například funkce scipy.signal.filtfilt. Vykreslete původní a vyfiltrovaný signál v časové doméně a spočítejte a zobrazte jejich spektra.

In [121]:

```
x_filtered = filtfilt(b, a, x_resampled)

fft_result_orig = np.abs(fft(x))
freq_values_orig = np.fft.fftfreq(len(x), 1/fs)

fft_result_filt = np.abs(fft(x_filtered))
freq_values_filt = np.fft.fftfreq(len(x_filtered), 1/100)

plt.figure(figsize=(20, 16))
plt.subplot(4, 1, 1)
plt.plot(np.arange(0, len(x)/fs, 1/fs), x)
plt.title('ECG signal')
plt.xlabel('Time [s]')
plt.ylabel('Amplitude [mV]')
plt.grid(True)

plt.subplot(4, 1, 2)
```

```
plt.plot(np.arange(0, len(x_filtered)/new_fs, 1/new_fs), x_filtered)
plt.title('ECG signal')
plt.xlabel('Time [s]')
plt.ylabel('Amplitude [mV]')
plt.grid(True)

plt.subplot(4, 1, 3)
plt.stem(freq_values_orig, fft_result_orig, basefmt='black', markerfmt=' ')
plt.title('Spectrum of ECG signal')
plt.xlabel('Frequency [Hz]')
plt.ylabel('Magnitude spectrum')
plt.xlim(0, 200)
plt.ylim(0, 4)
plt.grid(True)

plt.subplot(4, 1, 4)
plt.stem(freq_values_filt, fft_result_filt, basefmt='black', markerfmt=' ')
plt.title('Spectrum of bandpass filtered ECG signal')
plt.xlabel('Frequency [Hz]')
plt.ylabel('Magnitude spectrum')
plt.xlim(0, 50)
plt.ylim(0, 0.2)
plt.grid(True)

plt.tight_layout()
plt.show()
```



c) [1b] Okomentujte rozdíl mezi filtrovaným a nefiltrovaným signálem a jejich spektry. Pokud bychom použili filtrování pouze z jedné strany (obyčejnou konvoluci), jaké je teoreticky největší posunutí ve vzorcích, které se může objevit a proč?

In [127]:

```
#
#
#
#
```

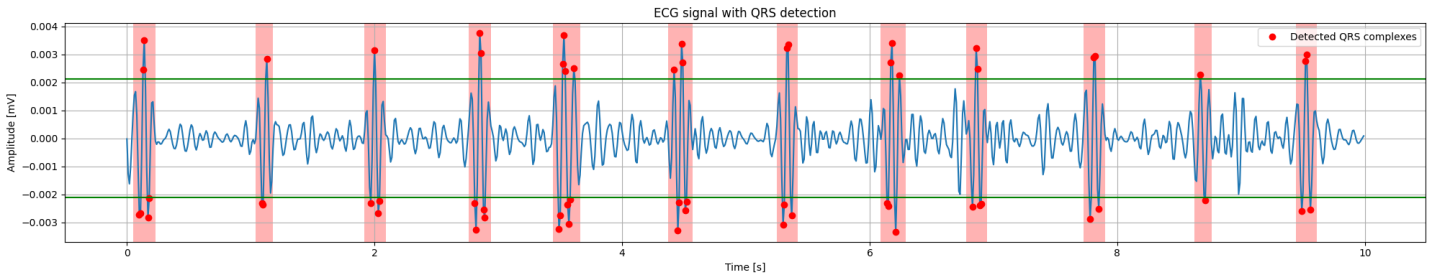
Original signal has high-frequency noise, which is cut off in a filtered signal by low-pass filter, and low-frequency baseline variations (for example due to breathing), which are removed by a high-pass filter. Original ECG signal has wide spectrum containing all frequencies, while filtered signal has limited spectrum (from 10 to 20 Hz) because the filter cuts off frequencies higher than 20 Hz and lower than 10 Hz. If we used one-sided filtering, the largest possible shift in the samples would be equal to the impulse response length of the filter minus 1. When using two-sided convolution, there is no theoretical shift in samples.

4.4. [3b] Vytvořte detektor QRS v časové doméně. Detekované QRS komplexy uložte do vhodné struktury a zároveň zobrazte graf v časové ose se zvýrazněnými QRS detekcemi.

a) [1b] Detekujte QRS v převzorkovaném vyfiltrovaném signálu pomocí tresholdu (prahu). Pro tuto detekci musíte nejdříve získat vzorek jednoho QRS ze signálu, spočítat si maximální amplitudu a jako treshold vzít vámi určené procento této hodnoty. Dávejte pozor na možnost otočeného QRS v signálu. Do vykresleného signálu s detekcemi vykreslete i čáru udávající použitý treshold.

In [122]:

```
end = int(new_fs * 0.5)
qrs_sample = x_filtered[0:end]
qrs_amplitude = np.max(np.abs(qrs_sample))

threshold = 0.6 * qrs_amplitude
qrs_indices = np.where(np.abs(x_filtered) > threshold)[0]

plt.figure(figsize=(20, 4))
plt.plot(np.arange(len(x_filtered))/new_fs, x_filtered)
plt.plot(qrs_indices / new_fs, x_filtered[qrs_indices], 'ro', label='Detected QRS complex
es')
plt.axhline(y=threshold, color='green')
plt.axhline(y=-threshold, color='green')
for index in qrs_indices:
    plt.axvspan(index/new_fs - 0.05, index/new_fs + 0.05, facecolor='#ffb3b3', alpha=1.0
)
plt.title('ECG signal with QRS detection')
plt.xlabel('Time [s]')
plt.ylabel('Amplitude [mV]')
plt.grid(True)
plt.legend()
plt.tight_layout()
plt.show()
```

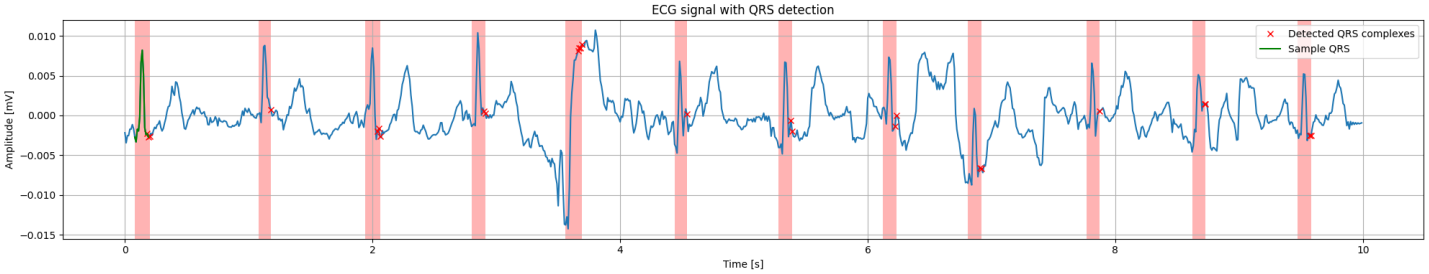


b) [2b] Detekujte QRS v signálu pomocí autokorelace v převzorkovaném nefiltrovaném signálu. Pro tuto detekci musíte nejdříve získat vzorek jednoho QRS ze signálu. Dále budete autokorelovat signál právě s tímto výstřižkem. QRS se budou nacházet na místech, kde vám budou vycházet vysoké hodnoty korelace. Do vykresleného signálu s detekcemi zaznačte i vámi zvolený výstřižek.

In [123]:

```
start = int(new_fs * 0.08)
end = int(new_fs * 0.20)
```

```
sample_qrs = x_resampled[start:end]

autocorrelation = np.correlate(x_resampled, sample_qrs, mode='full')
threshold = 0.6

normalized_autocorrelation = autocorrelation / np.max(autocorrelation)
peaks = np.where(normalized_autocorrelation > threshold)[0]

plt.figure(figsize=(20, 4))
plt.plot(np.arange(len(x_resampled)) / new_fs, x_resampled)
plt.plot((np.arange(0, len(x_resampled)) / new_fs)[peaks], x_resampled[peaks], 'rx', lab
el='Detected QRS complexes')
for peak in peaks:
    plt.axvspan(peak / new_fs - end / (2 * new_fs), peak / new_fs, facecolor='#ffb3b3',
alpha=1.0)
plt.plot((np.arange(0, len(x_resampled)) / new_fs)[start:end], sample_qrs, label='Sample
QRS', color='green')
plt.title('ECG signal with QRS detection')
plt.xlabel('Time [s]')
plt.ylabel('Amplitude [mV]')
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```



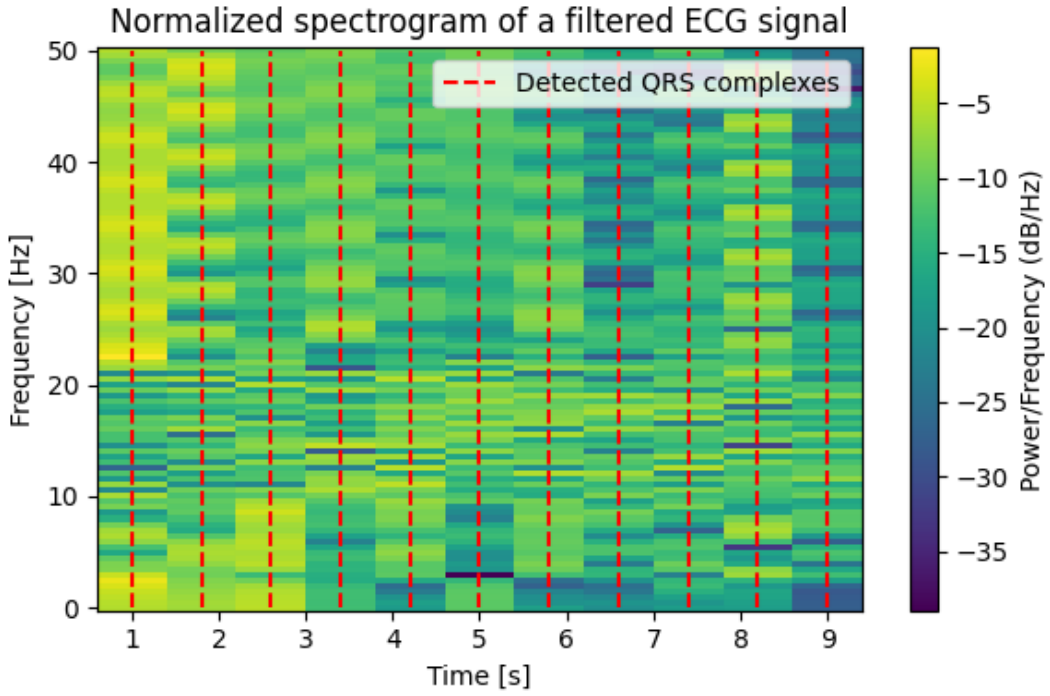
4.5. [3.5b] Vytvořte detektor QRS v frekvenční doméně a detekované QRS zakreslete jako v předchozí úloze 4.4

a) [2b] Detekujte QRS pomocí použití spektrogramu. Spočítejte a zobrazte spektrogram nahraného převzorkovaného filtrovaného signálu. Použijte parametry, `hop_size=120ms` a `window_len=200ms`, popřípadě si zkuste s těmito parametry pohrát. Spektrogram dále normalizujte v čase. Spočítejte sumy energie spektra pro jednotlivé časové biny. Dále vytvořte práh podle hodnoty energie spektra u prvního vámi zvoleného QRS komplexu. Tento práh použijte pro detekci zbylých QRS komplexů v signálu.

In [124]:

```
hop_size = 120
window_len = 200

frequencies, times, Sxx = spectrogram(x_filtered, new_fs, nperseg=window_len, noverlap=h
op_size, scaling='spectrum')
Sxx_normalized = Sxx / np.sum(Sxx, axis=1, keepdims=True)
energy_sum = np.sum(Sxx, axis=0)
threshold = energy_sum[0]
qrs_indices = np.where(energy_sum >= threshold)[0]

plt.figure(figsize=(6, 4))
plt.pcolormesh(times, frequencies, 10 * np.log10(Sxx_normalized), shading='auto')
plt.ylabel('Frequency [Hz]')
plt.xlabel('Time [s]')
plt.title('Normalized spectrogram of a filtered ECG signal')
plt.colorbar(label='Power/Frequency (dB/Hz)')
plt.vlines(times[qrs_indices], frequencies[0], frequencies[-1], colors='r', linestyles='
dashed', label='Detected QRS complexes')
plt.legend()
plt.tight_layout()
plt.show()
#
#
```

```
#
#
#
#
#
#
#
```



b) [1b] Detekujte QRS pomocí použití obálek a Hilbertovy transformace.

Hilbertova transformace je spočítaná podle následujícího vzorce

$$
\begin{aligned}
&x_a \\
&= F^{-1}(F(x)2U) \\
&= x + iy,
\end{aligned}
$$

kde F je Fourierova transformace a F^{-1} je její zpětná varianta. U je Heavisideova funkce neboli funkce jednotkového skoku, která je definována: $U(x)$:

$$
\begin{cases}
0.5 & x = 0 \\
1 & 0 < x < \frac{N}{2} \text{ pro } N \text{ liché} \\
0.5 & x = \frac{N}{2} \text{ pro } N \text{ liché} \\
1 & 0 < x \leq \frac{N}{2} \text{ pro } N \text{ sudé} \\
0 & jinak
\end{cases}
$$

kde N je počet koeficientů Fourierovy transformace - pokud není určeno jinak, je to počet vzorků signálu.

Jinými slovy obálku spočítate tak, že:

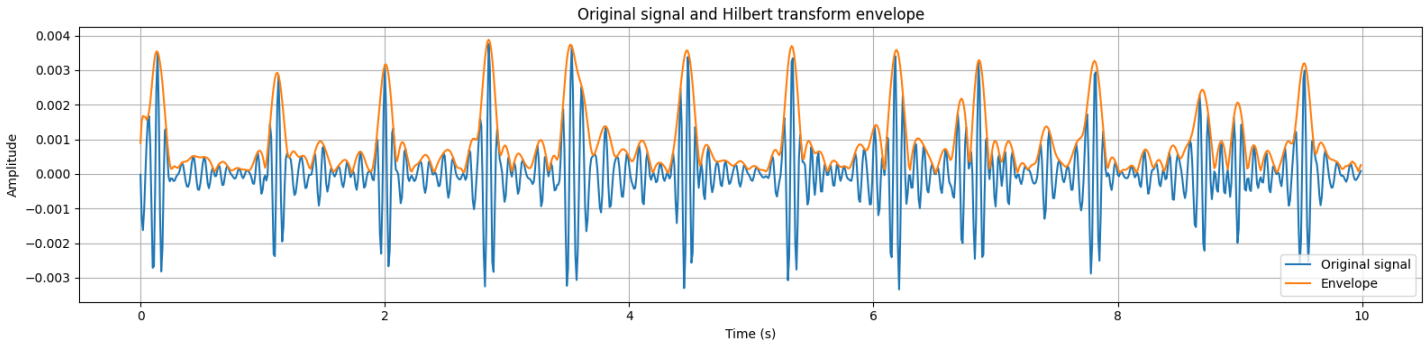
- **Spočítáte FFT F na filtrovaném a převzorkovaném signálu**
- **Vynulujete pravou symetrickou část spektra**
- **Levou část spektra vynasobíte 2 kromě prvního a prostředního binu (při sudém počtu frekvenčních binů).**
- **Provedete zpětnou FFT F^{-1}**

Abyste získali obálku signálu, je třeba vzít absolutní hodnotu signálu získaného Hilbertovou transformací.

Obálku a signál vykreslete do jednoho grafu přes sebe, obálka by měla obalovat daný signál.

In [125]:

```
def find_heaviside(x, N):
    result = np.zeros_like(x, dtype=float)
    mask1 = (x == 0)
    mask2 = (0 < x) & (x < N/2) & (N % 2 == 1)
    mask3 = (x == N/2) & (N % 2 == 1)
    mask4 = (0 < x) & (x <= N/2) & (N % 2 == 0)

    result[mask1] = 0.5
    result[mask2] = 1
    result[mask3] = 0.5
    result[mask4] = 1
    return result

fft_result = np.fft.fft(x_filtered)
N = len(fft_result)
for i in range(N):
    fft_result[i] *= 2 * find_heaviside(i, N)
hilbert_transform = np.fft.ifft(fft_result)
hilbert_signal = np.abs(hilbert_transform)

plt.figure(figsize=(16, 4))
plt.plot(np.arange(len(x_filtered))/new_fs, x_filtered, label='Original signal')
plt.plot(np.arange(len(x_filtered))/new_fs, np.abs(hilbert_signal), label='Envelope')
plt.title('Original signal and Hilbert transform envelope')
plt.xlabel('Time (s)')
plt.ylabel('Amplitude')
plt.legend()
plt.grid('True')
plt.tight_layout()
plt.show()
```



c) [0.5b] Při kterých metodách detekcí QRS nám vadí otočený (flipnutý) signál, při kterých ne a proč?

4.6 [2b] Detekce R-R intervalu

a) Detekujte R-R intervaly pomocí detekovaných QRS z jednotlivých metod, které jste použili dříve. Vykreslete hodnoty R-R intervalu do stejného grafu jako EKG signál a detekované QRS. Vykreslení proveďte nad EKG signál, kde osa x bude i nadále časová a každý R-R interval bude zakreslen na x pozici detekovaného QRS. Osa y pro R-R interval bude určovat hodnotu samotného R-R intervalu.

In [ ]:

```
# Zde napište váš kód
```

4.7 Bonus

a) Načtěte si signál obsahující fibrilaci FIS.wav. Proveďte na něm filtrace a převzorkování. Poté zkuste použít nějaký QRS detektor. Z detekovaných QRS detekujte R-R intervaly. Porovnejte R-R intervaly pro fibrilaci a klasický signál bez patologie (sinusovy rytmus). Měli byste vidět prudké změny v R-R intervalech a jejich nepravidelnost. Zároveň se vám může stát, že vám některé metody detekce QRS nepodají tak kvalitní výkon jako při sinusovém rytmu.

In [ ]:

```
# Zde napište váš kód
```

b) Načtěte si signál obsahující komorovou extrasystolu KES.wav. Proveďte na něm filtrace a převzorkování. Spočítejte a zobrazte spektrogram úseku tohoto signálu. Porovnejte spektrogramy vašeho signálu a signálu KES.wav. Měli byste vidět rozšířenou aktivitu na nízkých frekvencích. Dále zobrazte a porovnejte tyto signály v časové doméně. Obsažené komorové extrasystoly by se měly projevit jako zvláštní široké QRS.

In [ ]:

```
# Zde napište váš kód
```