UNIVERSITY OF CAMBRIDGE

Mobile Health Practical 1

Introduction to Signal Processing

Jake Stuchbury-Wass
js2372@cam.ac.uk

Kayla-Jade Butkow
kjb85@cam.ac.uk

Goal of the Practical

- Intro to signal processing in Python
- Introduction to PPG data
- Python Notebook in Colab for data processing
  - Load and organise data
  - Data visualisation
  - Signal processing in Python
- Learn about the upcoming assignment

- You should be already familiar with the following concepts:
  - Analogue and digital signal
  - Nyquist theorem
  - Discrete Fourier transform and Fast Fourier transform
  - Spectrograms
  - Basics of filtering
- Most common tools for digital signal processing – Python, MATLAB

Intro to Signal Processing with Python

Python tools necessary for this practical

- **Data loader**
- **Data organiser**
- **Data visualiser**
- **Actual signal processing**

Data loader | Data organiser | Data visualiser | Actual signal processing

- Need to load:
  - sensor data — could be a CSV, WAV, etc.
  - metadata — typically a CSV

```python
import csv

with open('data.csv', 'r') as csvfile:
    data = csv.reader(csvfile)
    for row in data:
        print(row)
```

```python
import pandas as pd

data = pd.read_csv('data.csv')
print(data)
```

Python tools necessary for this practical

Data loader | **Data organiser** | Data visualiser | Actual signal processing

- Display the data and transform it if needed

```python
import pandas as pd
import numpy as np
import re
```

pandas: Library for data manipulation and analysis

numpy: Supports and allows complex operations on large, multi-dimensional matrices

re: Library for working with regular expressions

Data loader | Data organiser | **Data visualiser** | Actual signal processing

- Always a good idea not to work with the data blindly
- The most straightforward library — `matplotlib.pyplot`

```
import matplotlib.pyplot as plt

x = [1, 2, 3, 4, 5]
y = [2, 4, 6, 8, 10]
plt.plot(x, y)
plt.show()
```

Intro to Signal Processing with Python

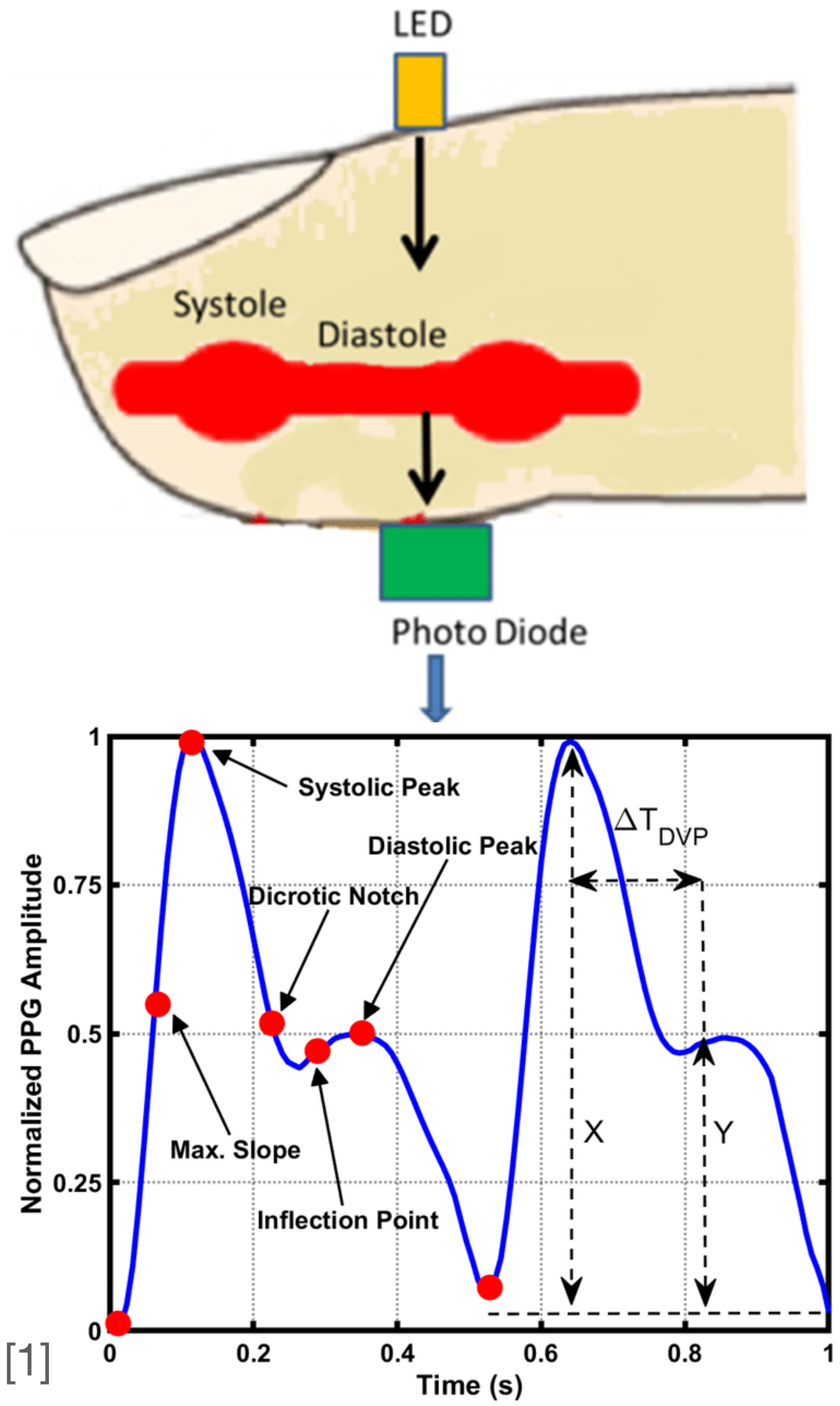
Python tools necessary for this practical

Data loader | Data organiser | Data visualiser | **Actual signal processing**

- Multiple libraries with predefined functions

```
import numpy as np
from scipy import signal

# generate a 1kHz sine wave
fs = 10e3
f = 1e3
t = np.linspace(0, 1, fs, endpoint=False)
x = np.sin(2 * np.pi * f * t)

# filter the signal using a Butterworth low-pass filter
b, a = signal.butter(4, 0.5, 'low', analog=False)
y = signal.lfilter(b, a, x)
```

Photoplethysmography (PPG)



[1]

- Optical technique for detecting changes in blood volume at the surface of the skin
- Comprises of an AC component related to heart beats and a DC component from light attenuated by skin, fingernails, tissue, bone, and static blood. It also has a slowly varying component which relates to respiration

[1] Hasanzadeh, Navid & Ahmadi, Mohammad Mahdi & Mohammadzade, Hoda. (2019). Blood Pressure Estimation Using Photoplethysmogram Signal and Its Morphological Features. IEEE Sensors Journal. PP. 1-1. 10.1109/JSEN.2019.2961411.

Colab Notebook for the Practical

- We'll be using google colab for the practical and the assignments
- Online, interactive Python notebook
- Open the notebook using the link or scan the QR code
- **https://shorturl.at/deuwR**

Importing Libraries

```
import os
```
Provides a way of using operating system (OS) dependent functionality

```
import numpy as np
```
NUMerical PYthon

```
import scipy
```
SCIentific PYthon, provides functions for stats and signal processing

```
import pandas as pd
```
Used for working with datasets

```
import matplotlib.pyplot as plt
```
Plotting library

Loading data

- We will be using PPG from the MARSH dataset [1]
- Steps:
  1. Load the data using pandas
  2. Min-max normalise the data
  3. Define the sampling rate (500Hz for the device [1])

```
filename = "/content/drive/My Drive/mobile-health-prac-1-2024/example_ppg.csv"

# load the PPG data into a np array
ppg = pd.read_csv(filename, header=None)
ppg = ppg[0].to_numpy()

# Normalise the data as we aren't interested in information related to the amplitude
ppg = (ppg-min(ppg))/(max(ppg)-min(ppg))

# sampling rate of the PPG data
fs = 500
```

[1] Mikko Pirhonen and Vehkaoja Antti, Fusion enhancement for tracking of respiratory rate through intrinsic mode functions in photoplethysmography. Biomedical Signal Processing and Control. 2020

We’ve loaded the data! How do we now start to analyse and process it?

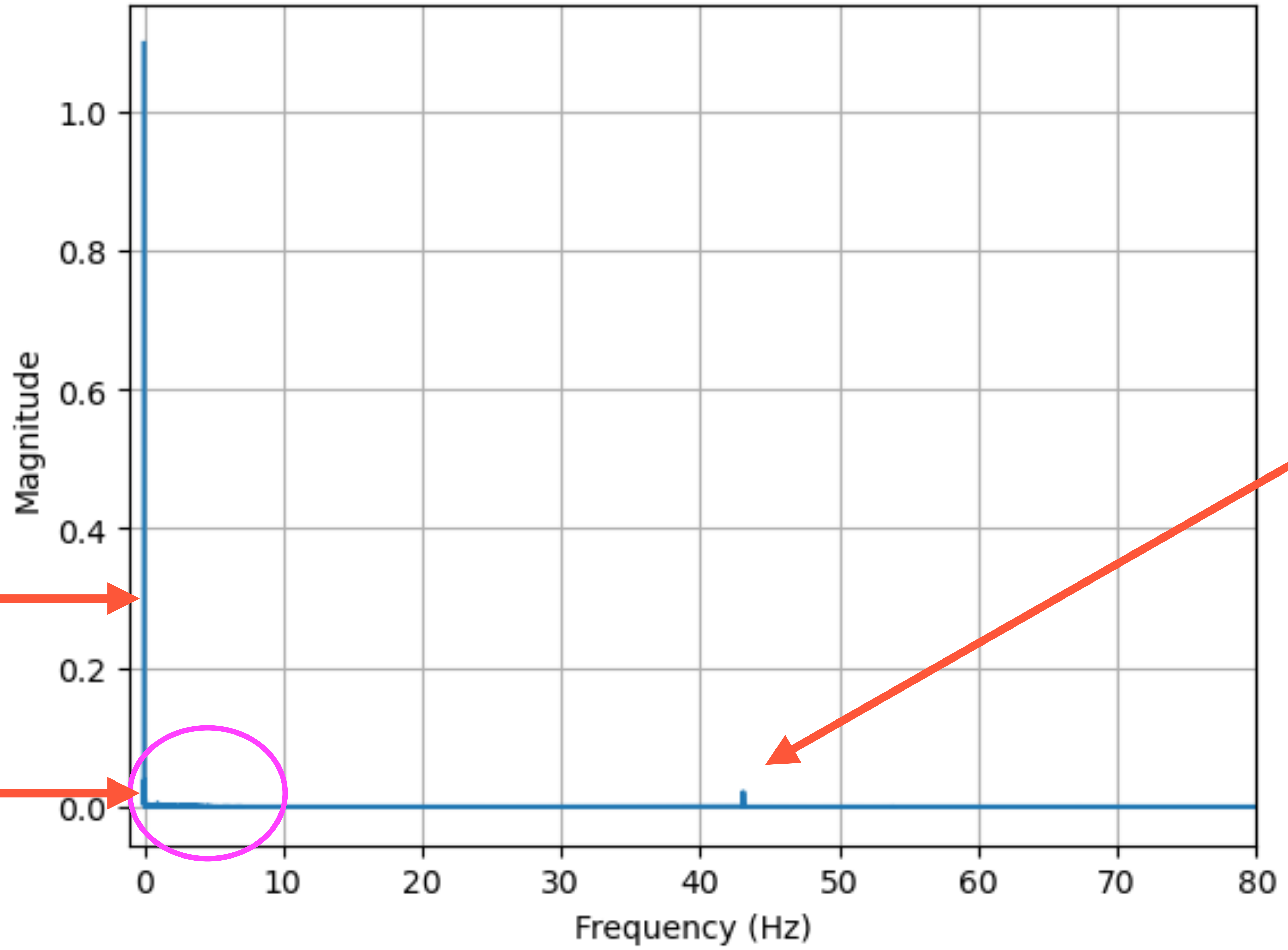
Time series visualisation

```
fig, ax = plt.subplots(4, 1, figsize=(12, 10), dpi=200)
ax[0].set_title("Whole data")
ax[0].plot(time,ppg)

ax[1].plot(time,ppg)
ax[1].set_xlim(100,120)
ax[1].set_ylim(0.4,0.8)
```

Frequency spectrum

DC offset

Non zero amplitudes



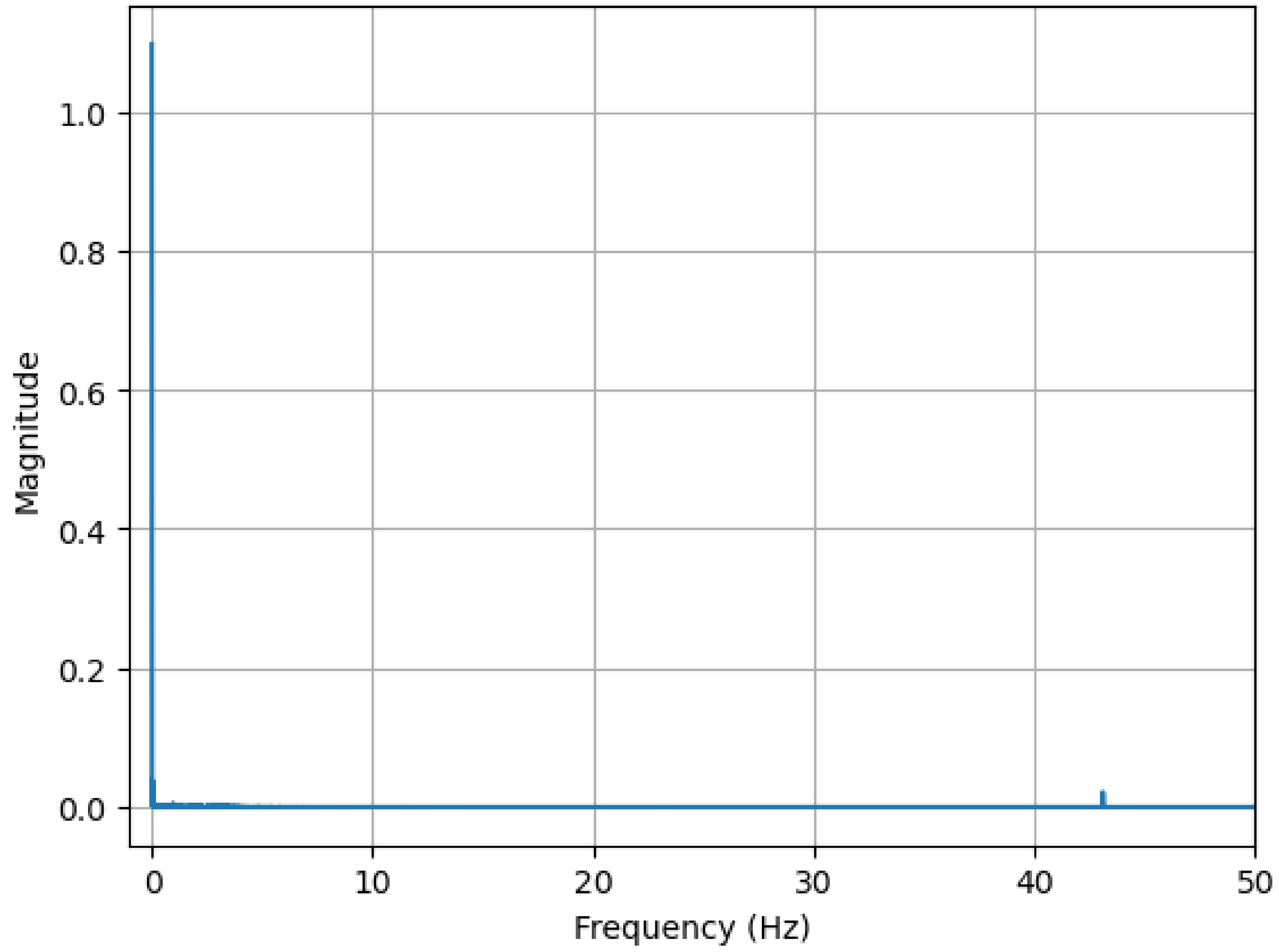
~45Hz. Could this be the heart rate?

Frequency visualisation (via FFT)

```python
def fft_plot(data, fs, xlim_l, xlim_r):
    n = len(data)
    yf = fft(data)
    xf = np.linspace(0.0, fs/(2.0), n//2)
    fig,ax= plt.subplots()
    ax.plot(xf, 2.0/n * np.abs(yf[:n//2]))
    plt.grid()
    plt.xlabel("Frequency (Hz)")
    plt.ylabel("Magnitude")
    plt.xlim(left=xlim_l, right=xlim_r)
    return plt.show()
```

Frequency visualisation (via FFT)

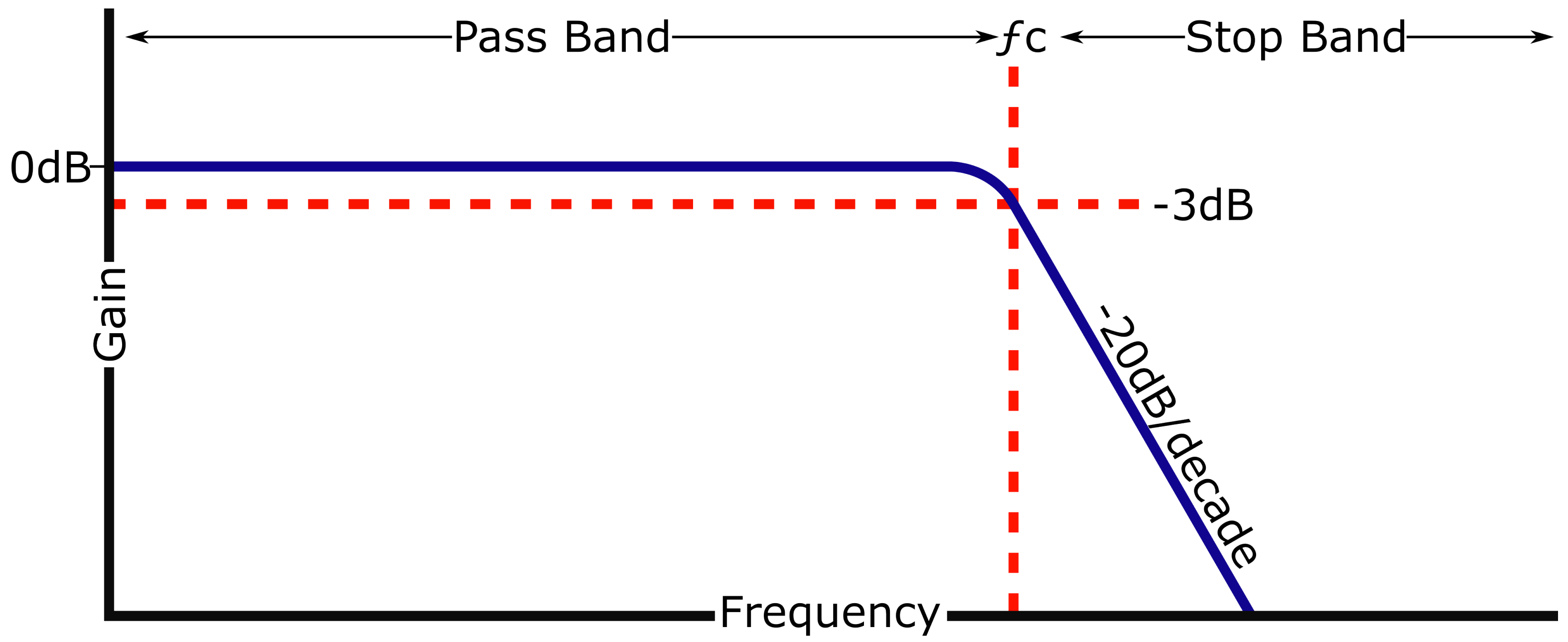
```
fft_plot(ppg,fs,xlim_l=0,xlim_r=50)
```



We've visualised the data and noticed possible features of interest. Now lets pre-process it and try extract features!

Filtering

- Let's try remove the 42Hz artefact
- **Low pass** filter
- From the FFT, we can see that the data we're interested in has a frequency of less than ~**15Hz**
  - Lets apply a **15Hz** filter and look at the result



Low Pass Filter

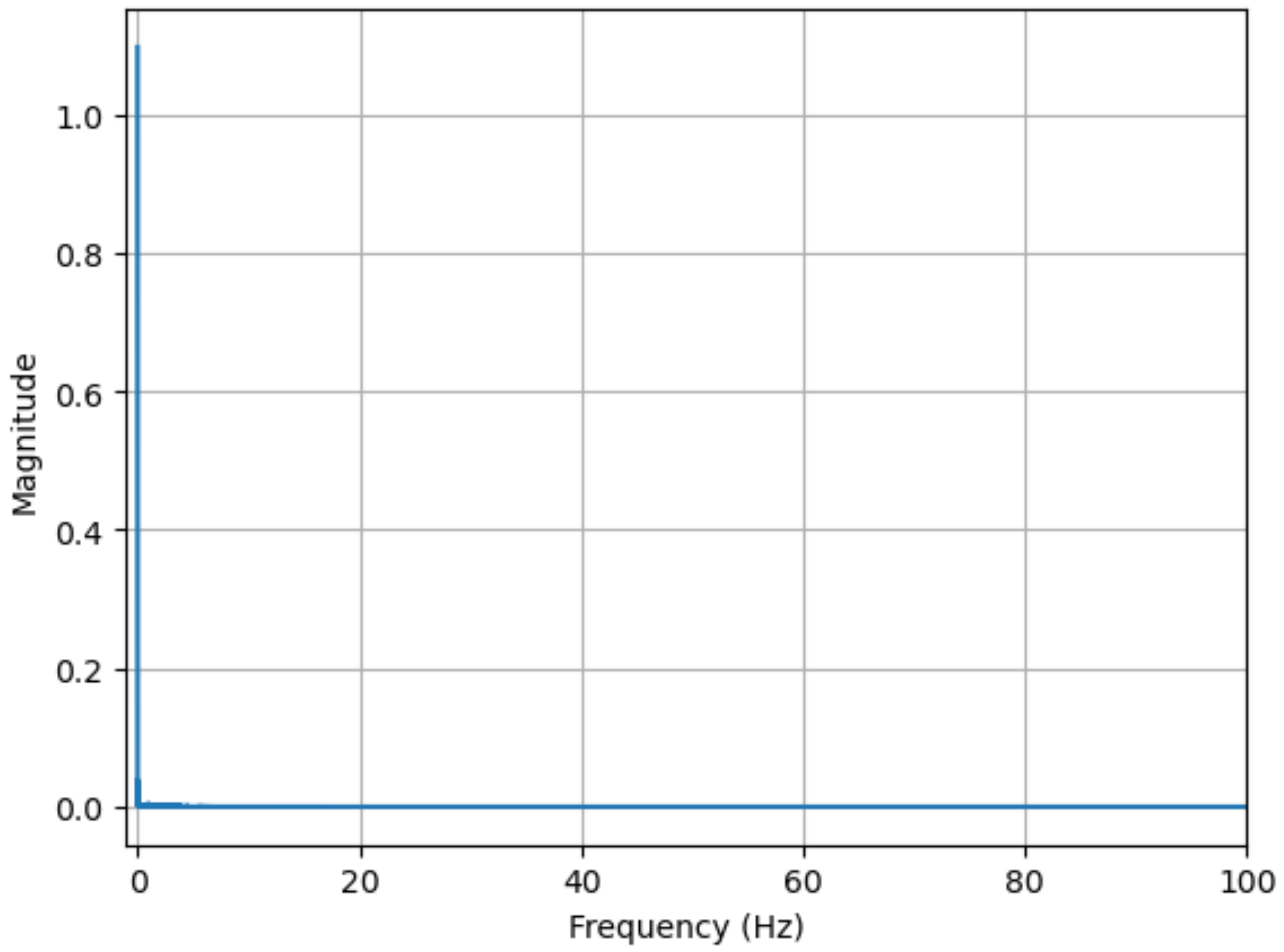
```
def butter_lowpass_filter(data, cutoff, fs, order):
    nyq = fs * 0.5
    normal_cutoff = cutoff / nyq
    sos = butter(order, normal_cutoff, btype='low', analog=False,output='sos')
    y = sosfiltfilt(sos, data)

    return y
```

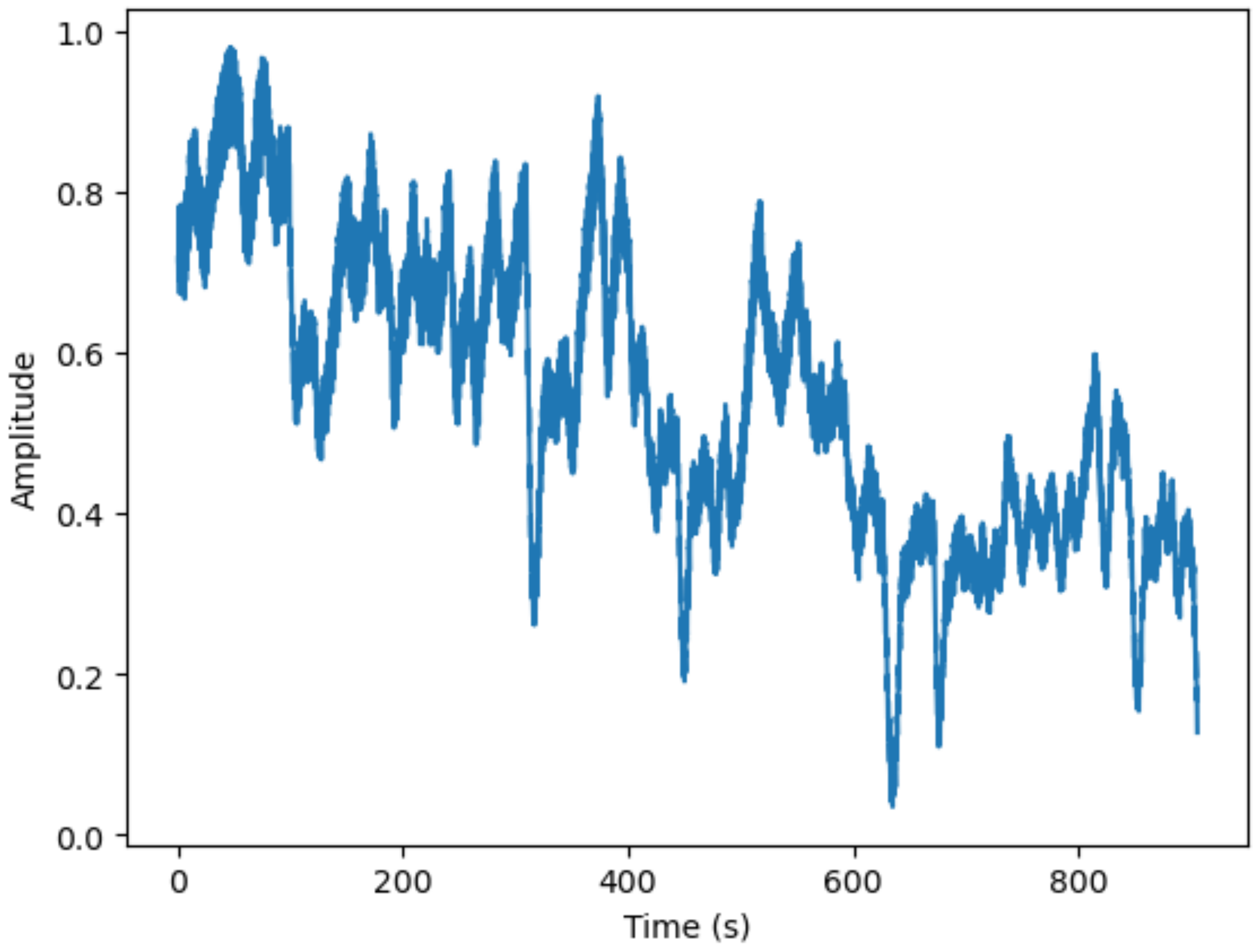
Apply Low Pass Filter

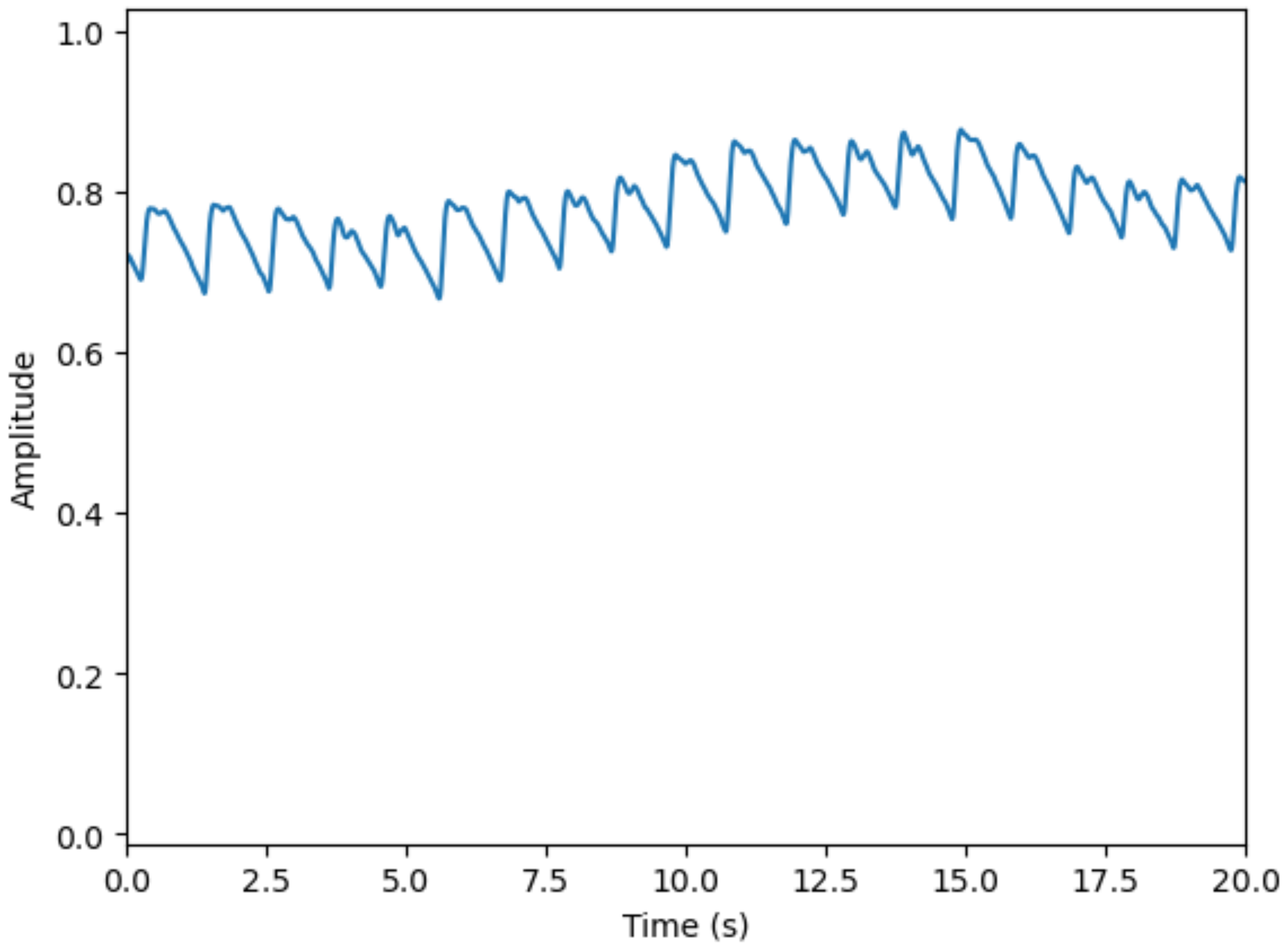
```
ppg_lp = butter_lowpass_filter(ppg,15, fs, 6)

plt.figure()
plt.plot(ppg)

plt.figure()
plt.plot(ppg_lp)

plt.figure()
plt.plot(ppg_lp)
plt.xlim(0,20)


fft_plot(ppg_lp,fs,0,100)
```

After Low Pass Filter

19 peaks in 20 seconds. What is the corresponding heart rate?

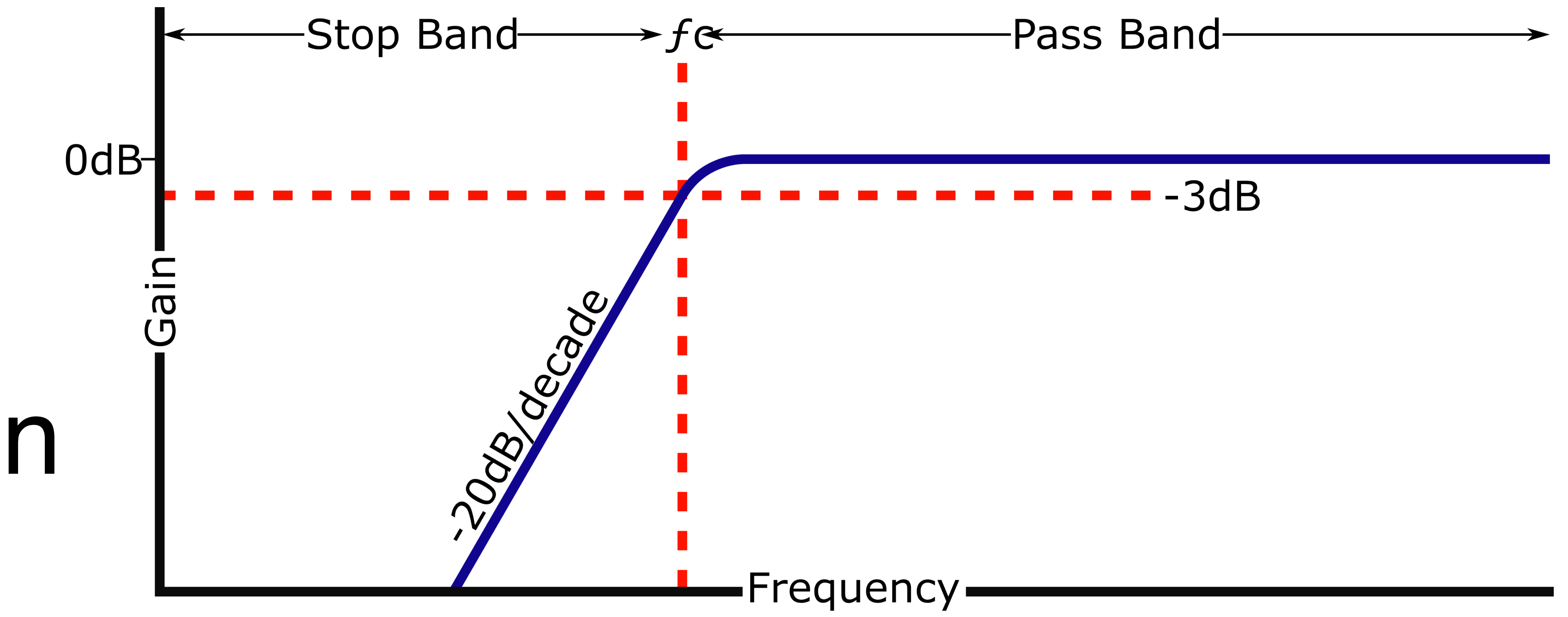






Filtering

- Remove the **DC offset**
- **High pass** filter
- From the FFT, we can see that we have **peaks** of interest in low frequency ranges
  - We need to apply the filter at a **lower cutoff** than the signals of interest
  - Try some frequencies and look at the resulting FFTs



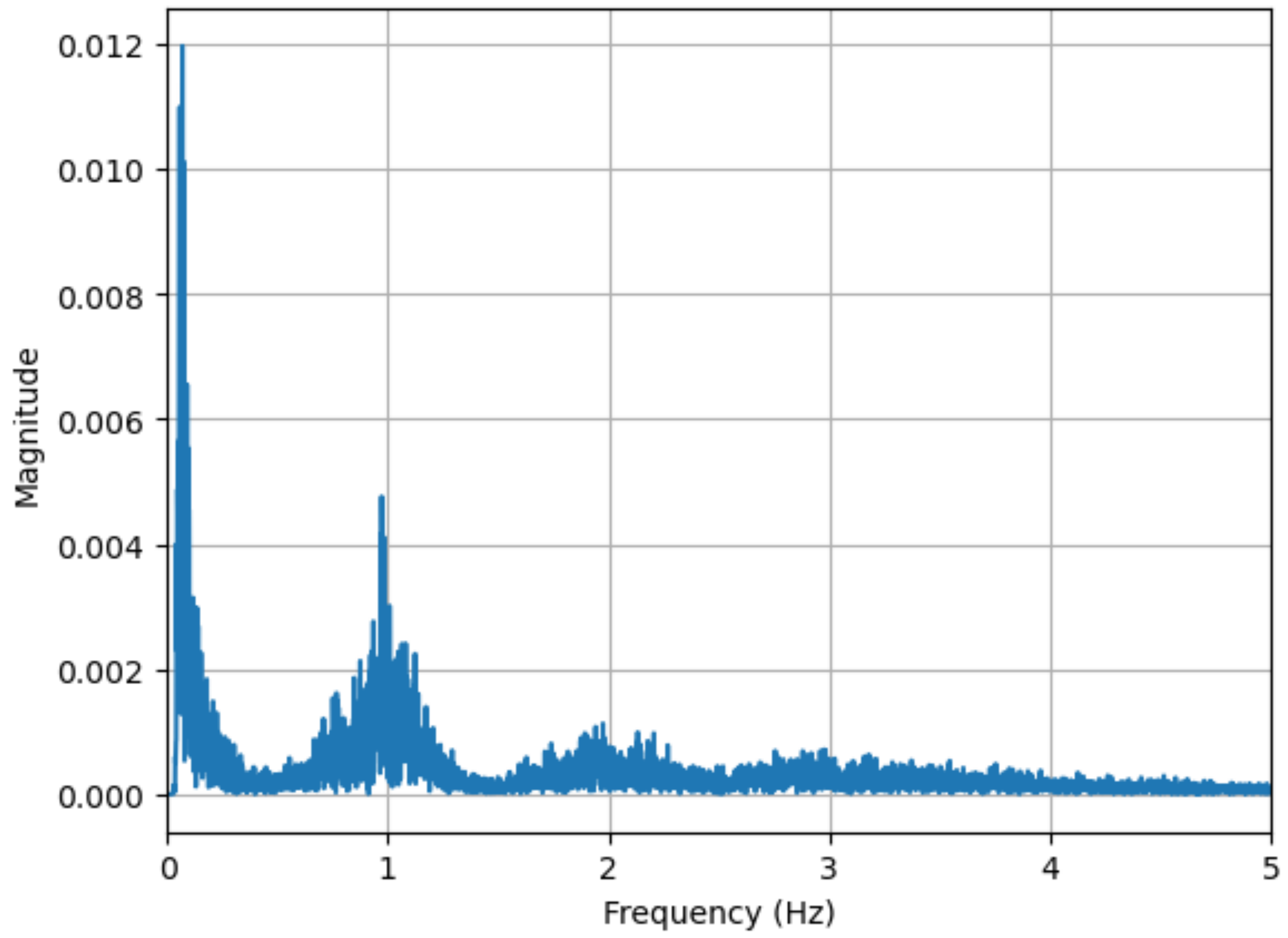
High Pass Filter

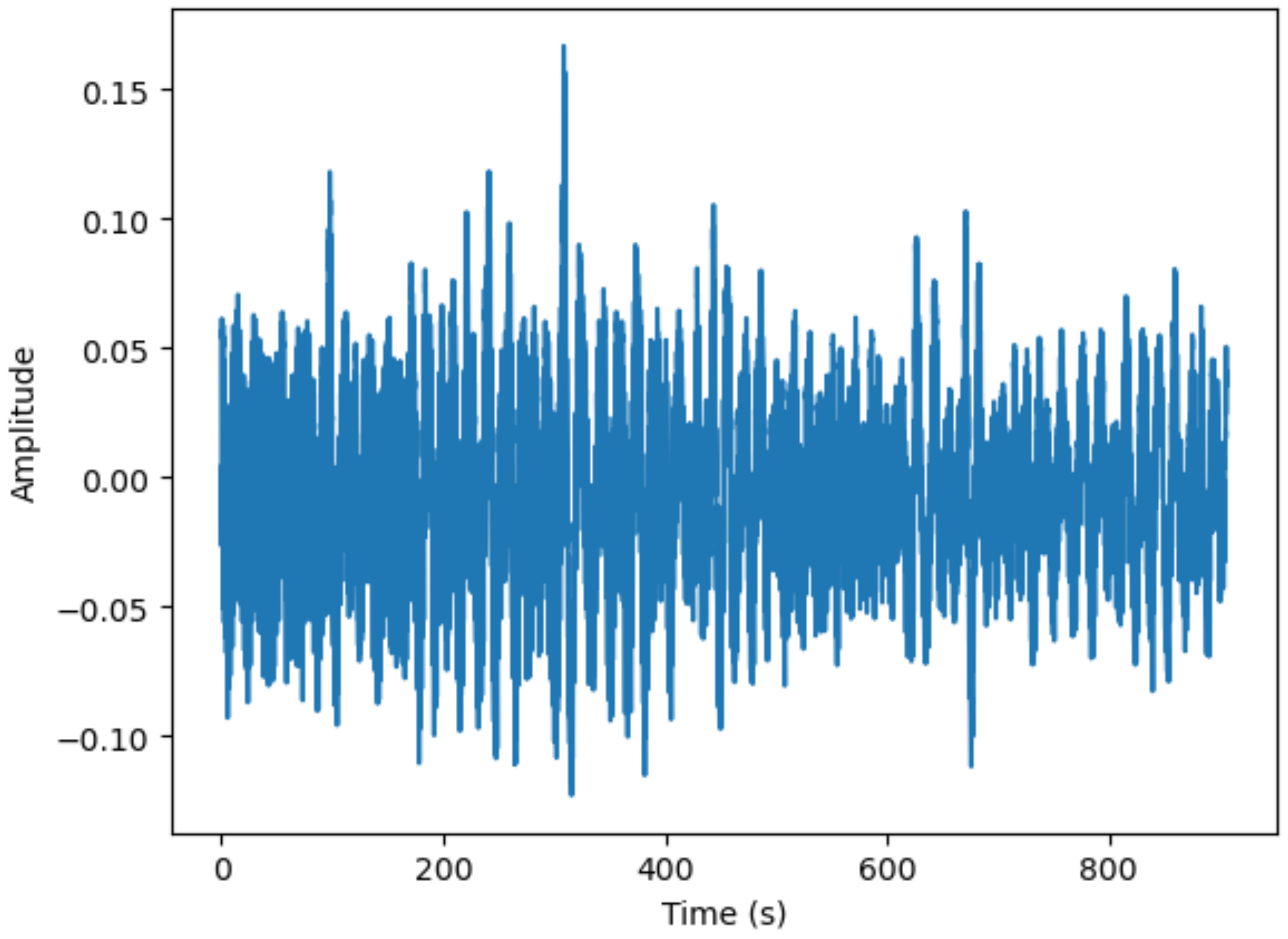
```
def butter_highpass_filter(data, cutoff, fs, order):
    nyq = fs * 0.5
    normal_cutoff = cutoff / nyq
    sos = butter(order, normal_cutoff, btype='high', analog=False,output='sos')
    y = sosfiltfilt(sos, data)

    return y
```

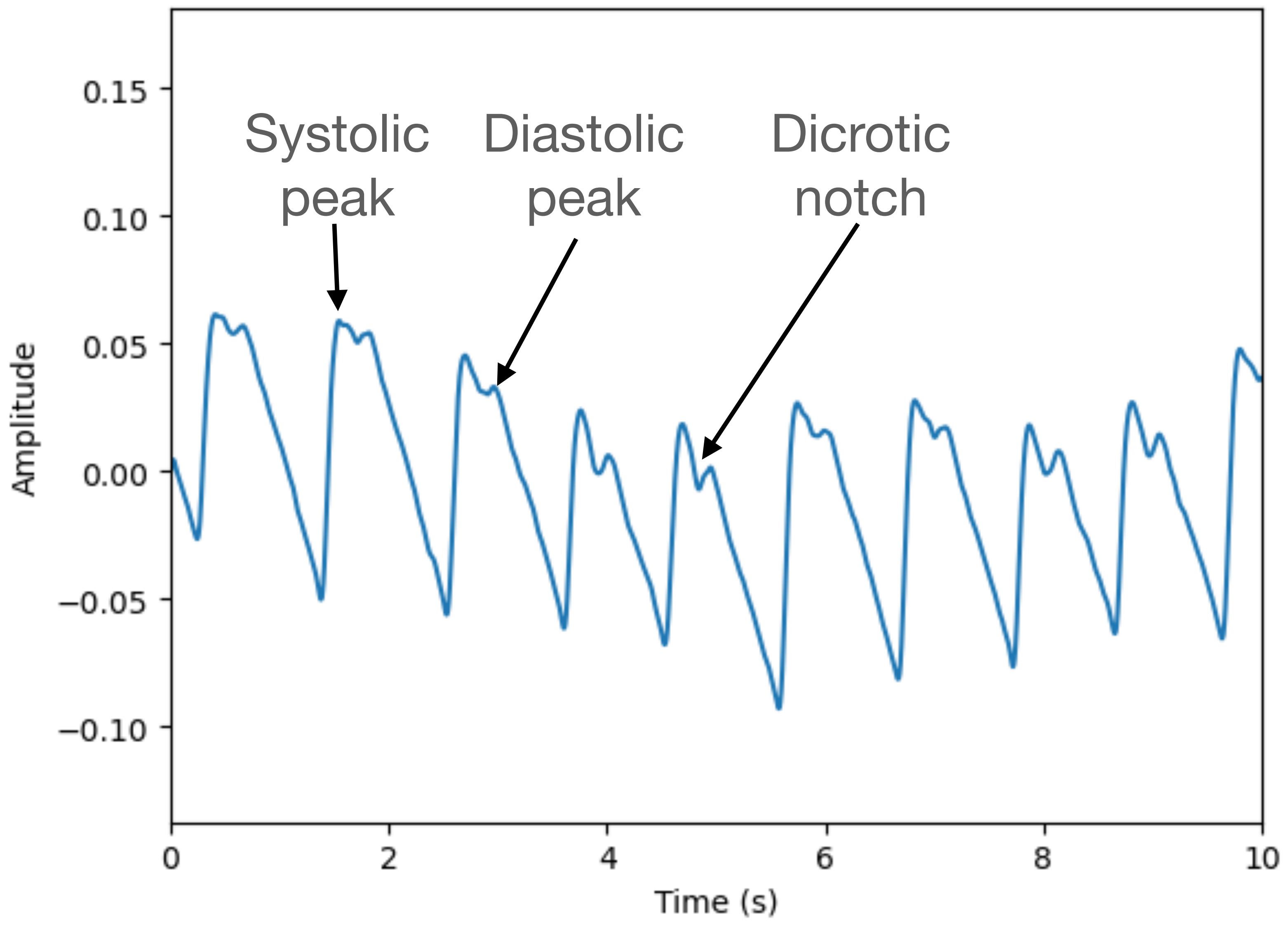
Apply High Pass Filter

```
ppg_hp = butter_highpass_filter(ppg_lp,0.05, fs, 6)

plt.figure()
plt.plot(ppg_lp)

plt.figure()
plt.plot(ppg_hp)

plt.figure()
plt.plot(ppg_hp)
plt.xlim(0,10)

fft_plot(ppg_hp,fs,0,5)
```

Signal after filtering

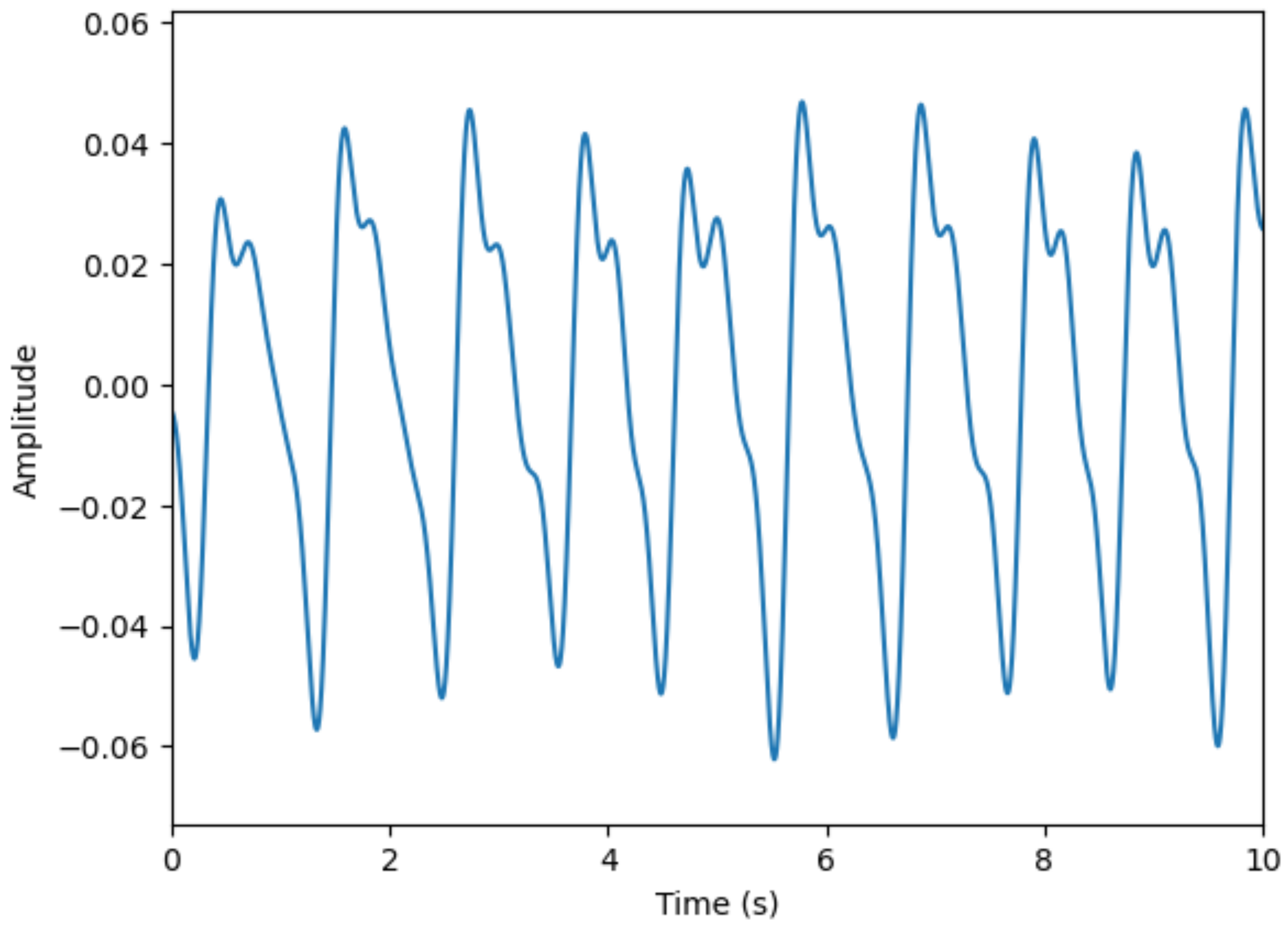




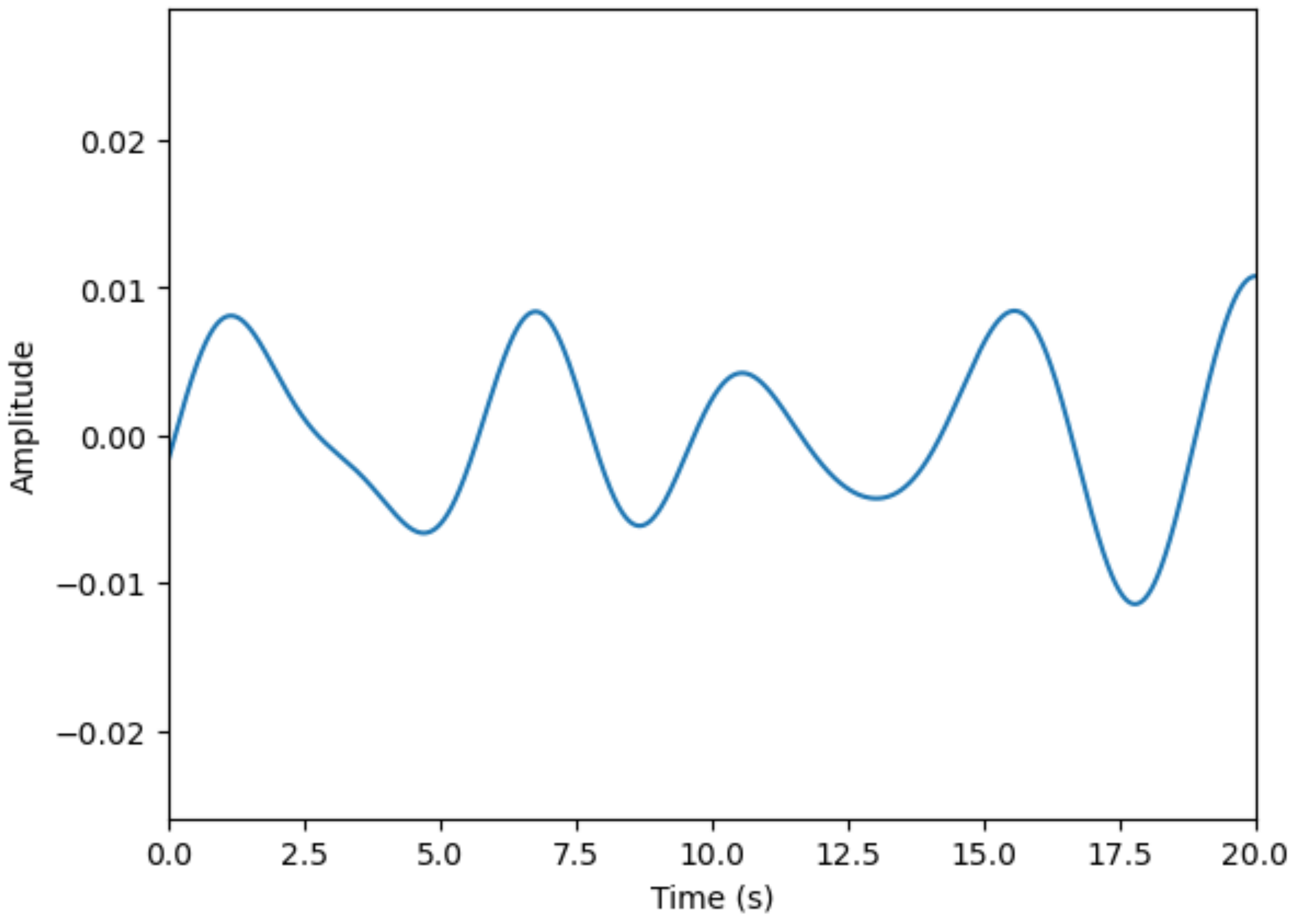


Lets try extract heart based signals and breathing signals using BPF

0.75Hz - 3.5Hz

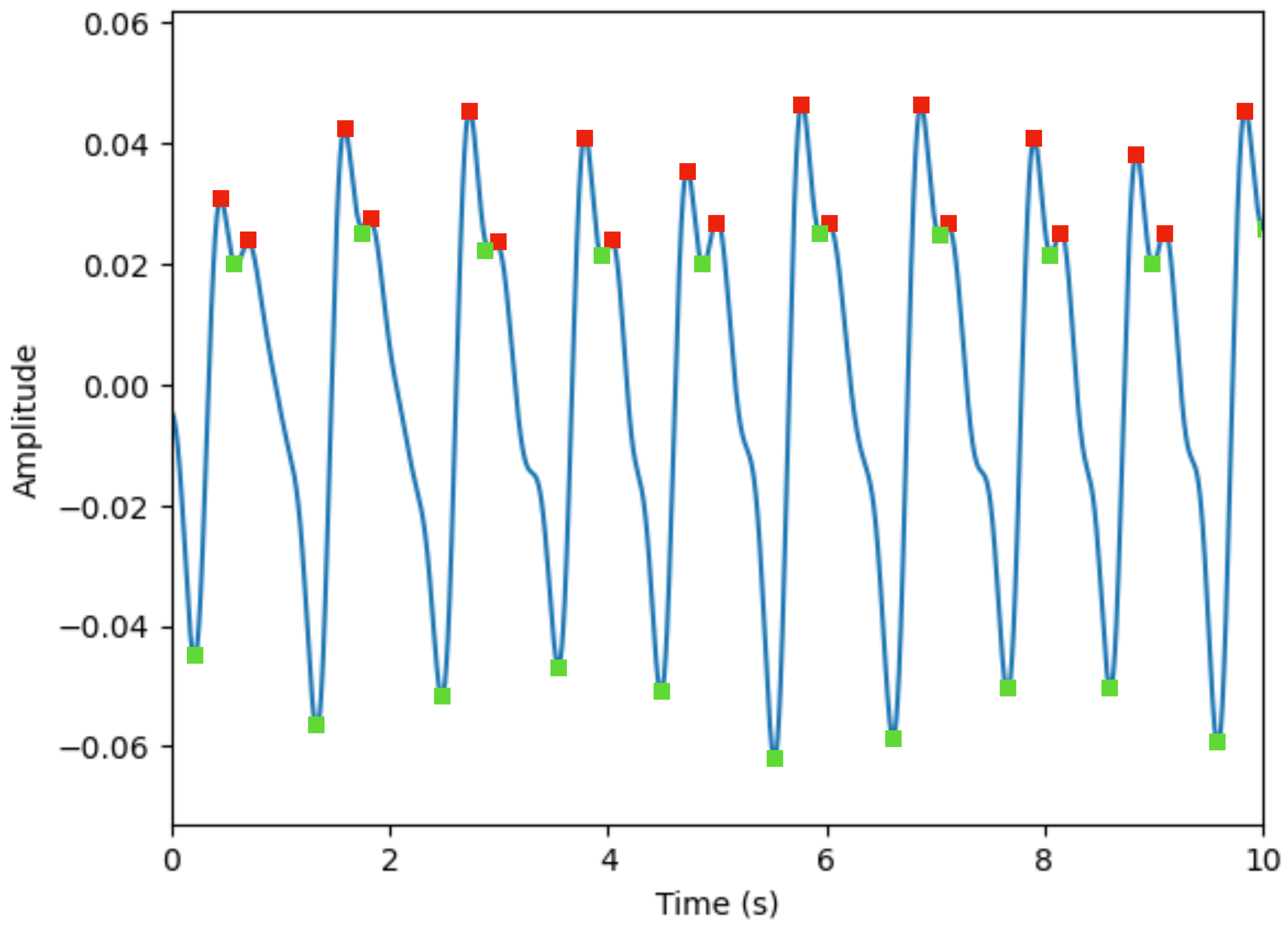


0.15Hz - 0.35Hz

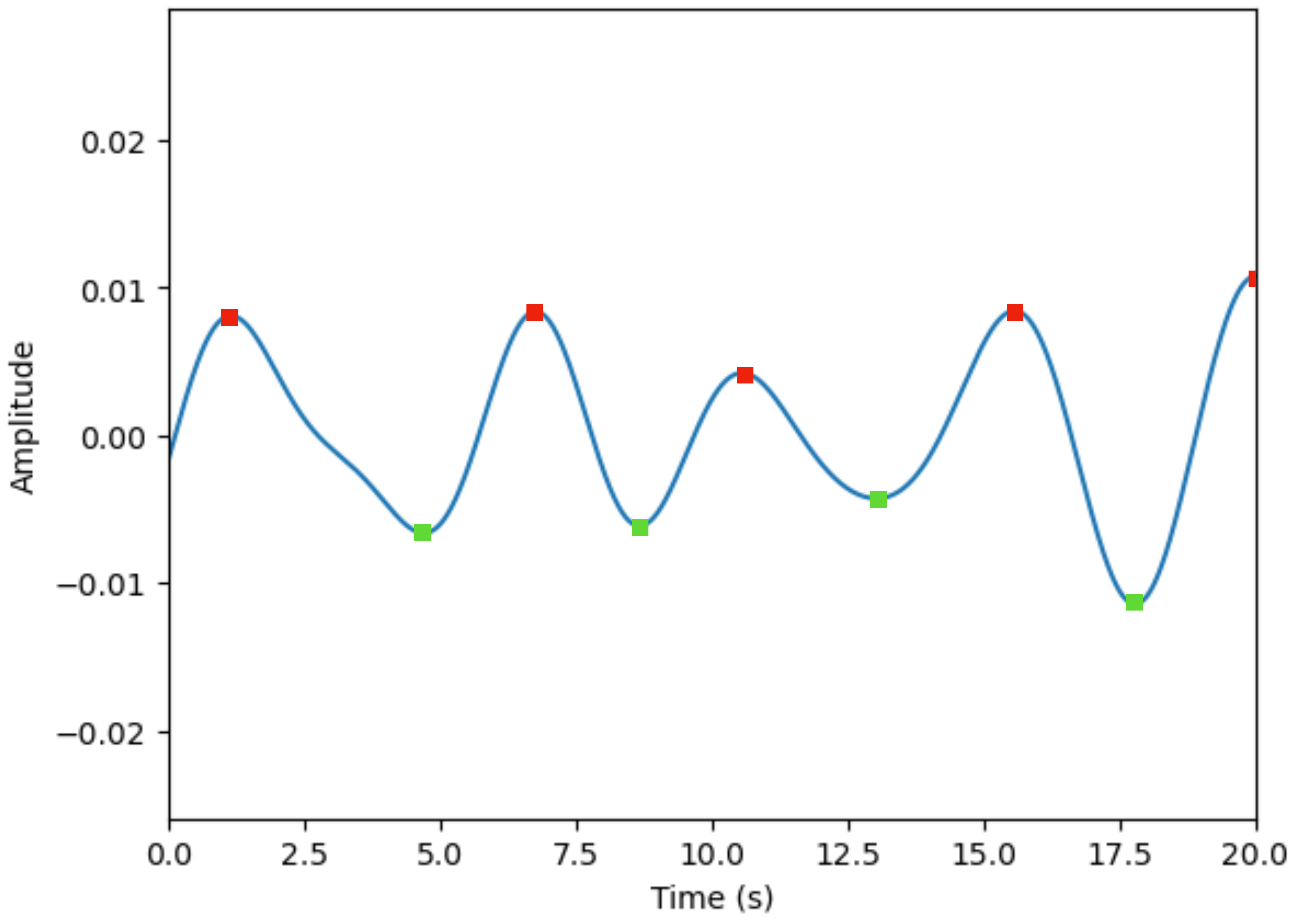


Basic feature extraction - Peak detection

0.75Hz - 3.5Hz



0.15Hz - 0.35Hz



Assignment

- Uploaded on Moodle
- Due **19 February**
- Audio dataset of **heart sounds**
- Preprocessing and filtering techniques for PPG can also be used for audio!
- Different sensing modalities have different established feature extraction techniques. You'll learn more about audio feature extraction in the assignment :)

Any questions?