Students:

Vincent Roduit
Caspar Henking
Fabio Palmisano
Bastien Marconato

Experiment1

November 21, 2024

[1]:
```
import numpy as np
from numpy import genfromtxt

# Load signal
data = genfromtxt('ECG.csv', delimiter=',')

# Perform SVD
U1, S1, V1 = np.linalg.svd(data[:, 0:5], full_matrices=False)
print(S1)

# Perform SVD
U2, S2, V2 = np.linalg.svd(data[:, 6:11], full_matrices=False)
print(S2)
```

```
[4.39635131e-01 3.11031279e-01 6.62584161e-08 6.26867811e-08
 4.86262541e-08]
[1.20061665 0.76592543 0.34630379 0.21222007 0.13759651]
```

1 Answers

a) Explain the difference between singular values of the six first columns, and of the last 6 columns.

Looking at the values we get from the first SVD, one notices that the first two components are much larger than the other four, meaning that the data is essentially varying along two principal components. It shows that there are dependance between signals.

On the other hand, the values of the second SVD do not feature such a high contrast, therefore we cannot reduce the number of dimensions in this case. This show less meaningful dependance between the signals.

b) Computate of the effective rank with a threshold of 0.98.

[2]:
```
def effective_rank(singular_values, threshold=0.98):
    cumulative_energy = np.cumsum(singular_values**2) / np.
 ↪sum(singular_values**2)
    rank = np.searchsorted(cumulative_energy, threshold) + 1
    return rank
```

```python
# Compute effective rank for S1 and S2
rank_S1 = effective_rank(S1, 0.98)
rank_S2 = effective_rank(S2, 0.98)

print(f"Effective rank with threshold of 0.98 for S1: {rank_S1}")
print(f"Effective rank with threshold of 0.98 for S2: {rank_S2}")
```

```
Effective rank with threshold of 0.98 for S1: 2
Effective rank with threshold of 0.98 for S2: 4
```

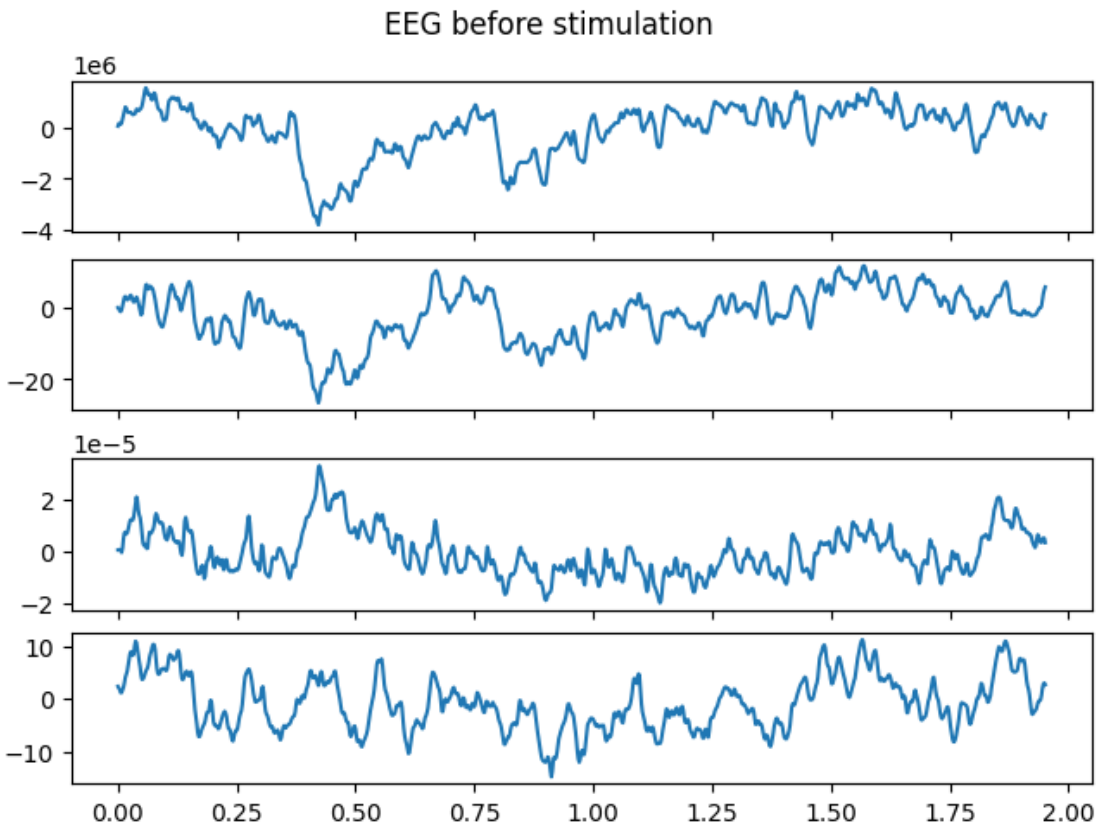
Experiment2

November 21, 2024

[1]:
```
import scipy
import scipy.io
import numpy as np
import matplotlib.pyplot as plt
```

[2]:
```
# Load data
EEG_before = scipy.io.loadmat('EEG_av.mat')
EEG_during = scipy.io.loadmat('EEG_pe.mat')
EEG_after = scipy.io.loadmat('EEG_ap.mat')

EEG_before = EEG_before['EEG_before']
EEG_during = EEG_during['EEG_during']
EEG_after = EEG_after['EEG_after']

# Set sampling frequency variable
fs = 512
```

[3]:
```
# Plot 4 EEG leads before stimulation
fig, axes = plt.subplots(
    4, 1, sharex='all', constrained_layout=True)
plt.suptitle('EEG before stimulation')
time = np.arange(0, len(EEG_before)/fs, 1/fs)
for i in np.arange(4):
    axes[i].plot(time, EEG_before[:, i])
```



[4]:
```
# Perform SVD
Ub, Sb, Vb = np.linalg.svd(EEG_before)

# Print singular values divided by max value
Before = 'Singular values before stimulation: {:.5f}, {:.5f}, {:.5f}, {:.5f}'
print(Before.format(Sb[0]/max(Sb), Sb[1]/max(Sb), Sb[2]/max(Sb), Sb[3]/max(Sb)))
```

```
Singular values before stimulation: 1.00000, 0.00001, 0.00000, 0.00000
```

0.0.1 Question 2.1 a)

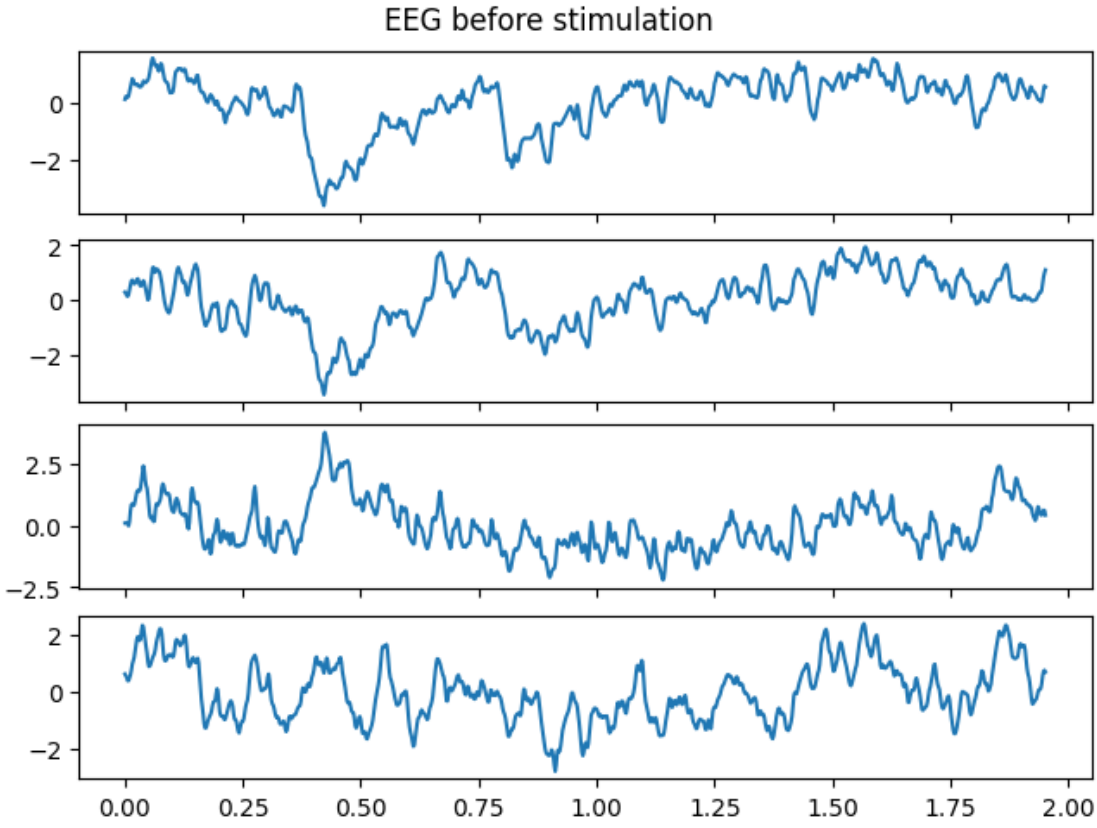
Answer

From these singular values, this means that all four signals are mainly varying along the same component, i.e. they are correlated.

0.0.2 Question 2.1 b)

[5]:
```
# Implement here the pre-processing of the signals
EEG_before = scipy.stats.zscore(EEG_before)
```

[6]:
```
# Plot 4 EEG leads before stimulation after pre-processing
fig, axes = plt.subplots(
    4, 1, sharex='all', constrained_layout=True)
plt.suptitle('EEG before stimulation')
time = np.arange(0, len(EEG_before)/fs, 1/fs)
for i in np.arange(4):
    axes[i].plot(time, EEG_before[:, i])
```



[7]:
```
# Perform here the svd
Ub, Sb, Vb = np.linalg.svd(EEG_before)

# Print singular values divided by max value
Before = 'Singular values before stimulation: {:.5f}, {:.5f}, {:.5f}, {:.5f}'
print(Before.format(Sb[0]/max(Sb), Sb[1]/max(Sb), Sb[2]/max(Sb), Sb[3]/max(Sb)))
```

```
Singular values before stimulation: 1.00000, 0.82741, 0.27099, 0.23259
```

0.0.3 Question 2.1 c)

Answer

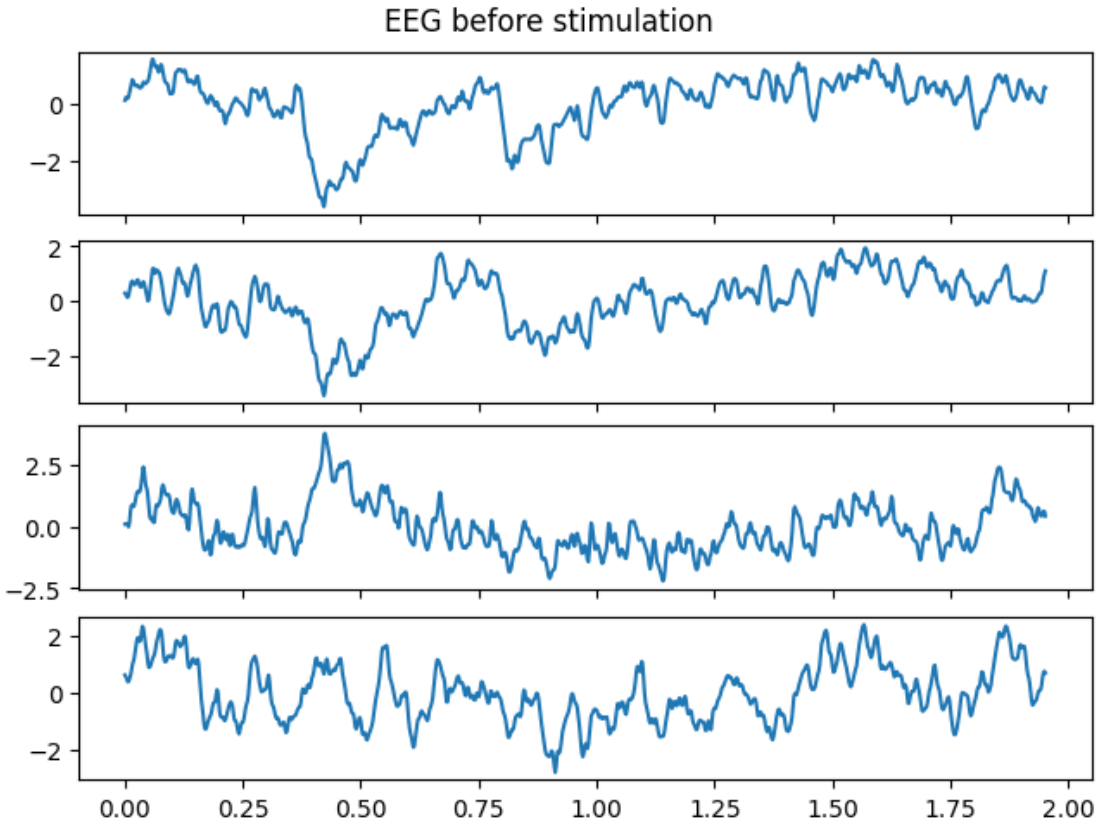
Here, one notices a higher variation along all components, as they are all different from zero.

This means that the signals are independant from each other now that they are normalized. This noticeable difference probably is due to the removal of the mean and the division by the standard deviation, which reduces the dependency of the signals with each other.

0.0.4 Question 2.2

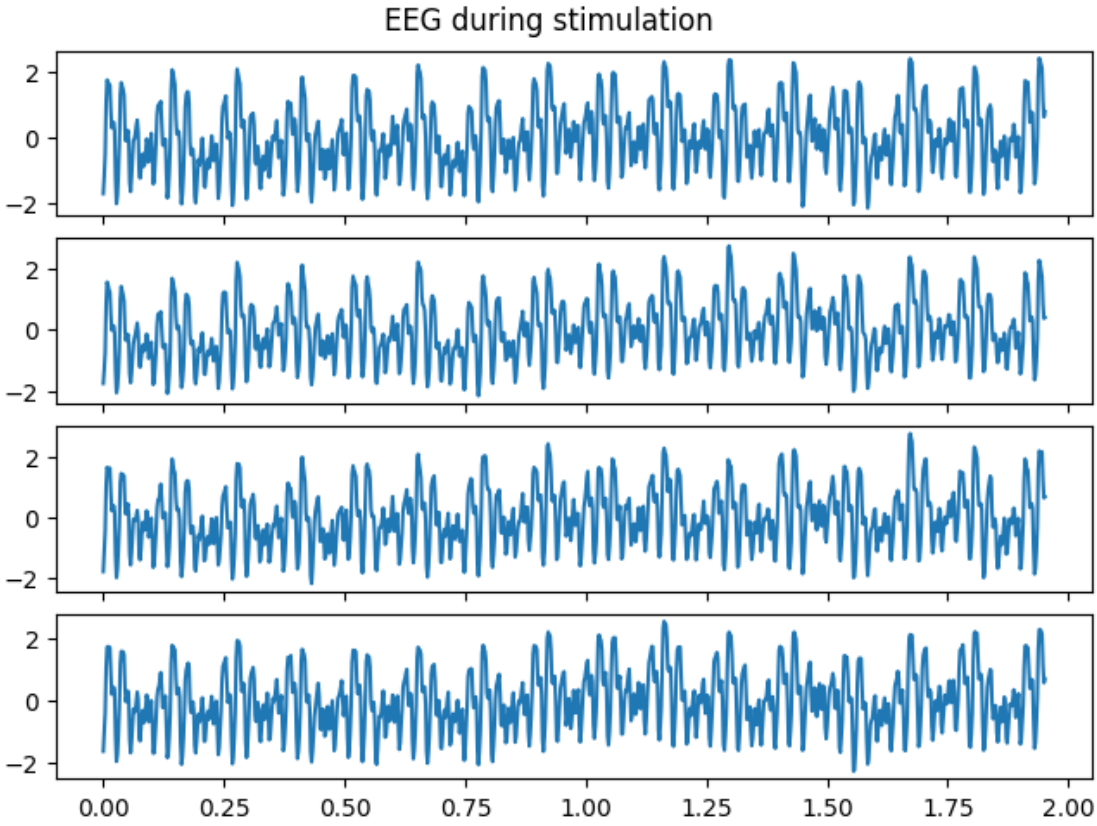
[8]:
```
# Implement here the pre-processing of the signals during and after stimulation
EEG_during = scipy.stats.zscore(EEG_during)
EEG_after = scipy.stats.zscore(EEG_after)
```

[9]:
```
# Plot 4 EEG leads before stimulation
fig, axes = plt.subplots(
    4, 1, sharex='all', constrained_layout=True)
plt.suptitle('EEG before stimulation')
time = np.arange(0, len(EEG_before)/fs, 1/fs)
for i in np.arange(4):
    axes[i].plot(time, EEG_before[:, i])
```

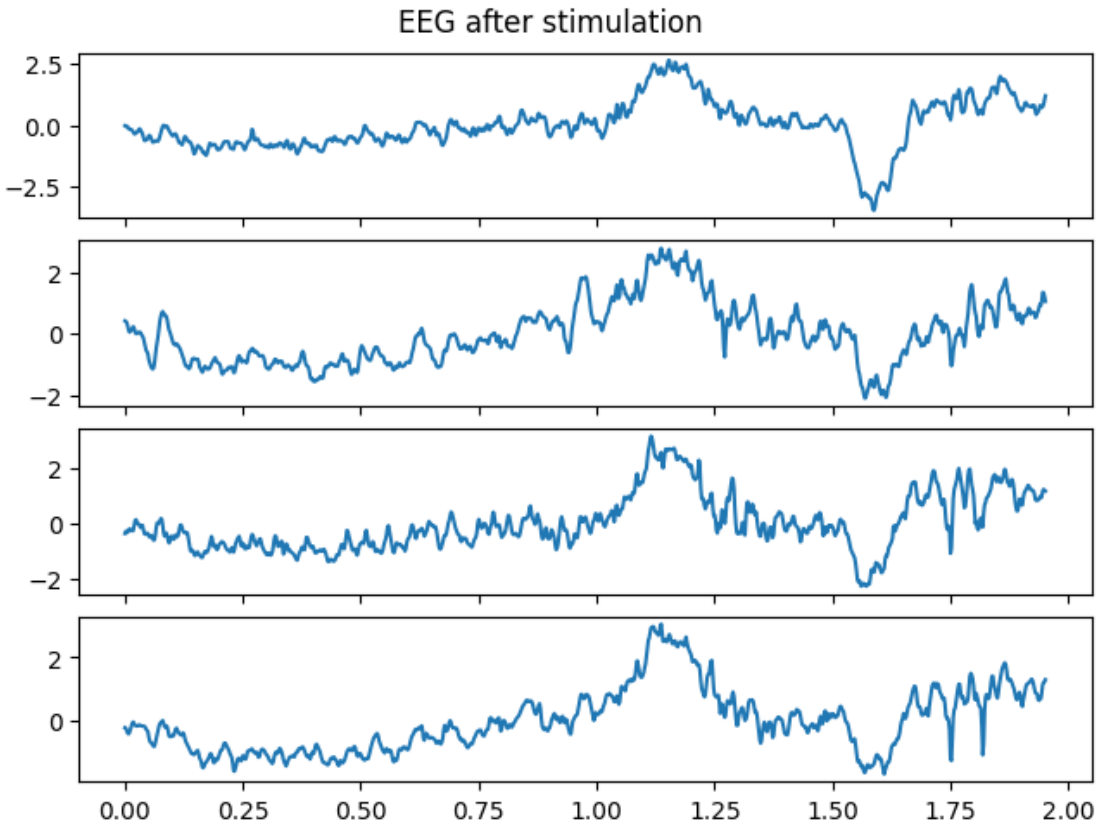


[10]:
```
# Plot 4 EEG leads during stimulation
fig, axes = plt.subplots(
    4, 1, sharex='all', constrained_layout=True)
```

```
plt.suptitle('EEG during stimulation')
time = np.arange(0, len(EEG_during)/fs, 1/fs)
for i in np.arange(4):
    axes[i].plot(time, EEG_during[:, i])
```



[11]:
```
# Plot 4 EEG leads after stimulation
fig, axes = plt.subplots(
    4, 1, sharex='all', constrained_layout=True)
plt.suptitle('EEG after stimulation')
time = np.arange(0, len(EEG_after)/fs, 1/fs)
for i in np.arange(4):
    axes[i].plot(time, EEG_after[:, i])
```



[12]:
```
Ub, Sb, Vb = np.linalg.svd(EEG_before)
Ud, Sd, Vd = np.linalg.svd(EEG_during)
Ua, Sa, Va = np.linalg.svd(EEG_after)
# Print singular values
Before = 'Singular values before: {:.5f}, {:.5f}, {:.5f}, {:.5f}'
print(Before.format(Sb[0]/max(Sb), Sb[1]/max(Sb), Sb[2]/max(Sb), Sb[3]/max(Sb)))
During = 'Singular values during: {:.5f}, {:.5f}, {:.5f}, {:.5f}'
print(During.format(Sd[0]/max(Sd), Sd[1]/max(Sd), Sd[2]/max(Sd), Sd[3]/max(Sd)))
After = 'Singular values after: {:.5f}, {:.5f}, {:.5f}, {:.5f}'
print(After.format(Sa[0]/max(Sa), Sa[1]/max(Sa), Sa[2]/max(Sa), Sa[3]/max(Sa)))
```

```
Singular values before: 1.00000, 0.82741, 0.27099, 0.23259
Singular values during: 1.00000, 0.10128, 0.07567, 0.06634
Singular values after: 1.00000, 0.21414, 0.17477, 0.09851
```

0.0.5 Question 2.2

a)

The singular values during the simulation are all very low, apart from the highest one, meaning that the signals all vary along a single component and they are all strongly correlated.

b)

One notices that the singular values of the signals after simulation have significantly lower values than the ones before, but are still higher than during the simulation. This shows that there is a leftover effect on the signals from the stimulation.

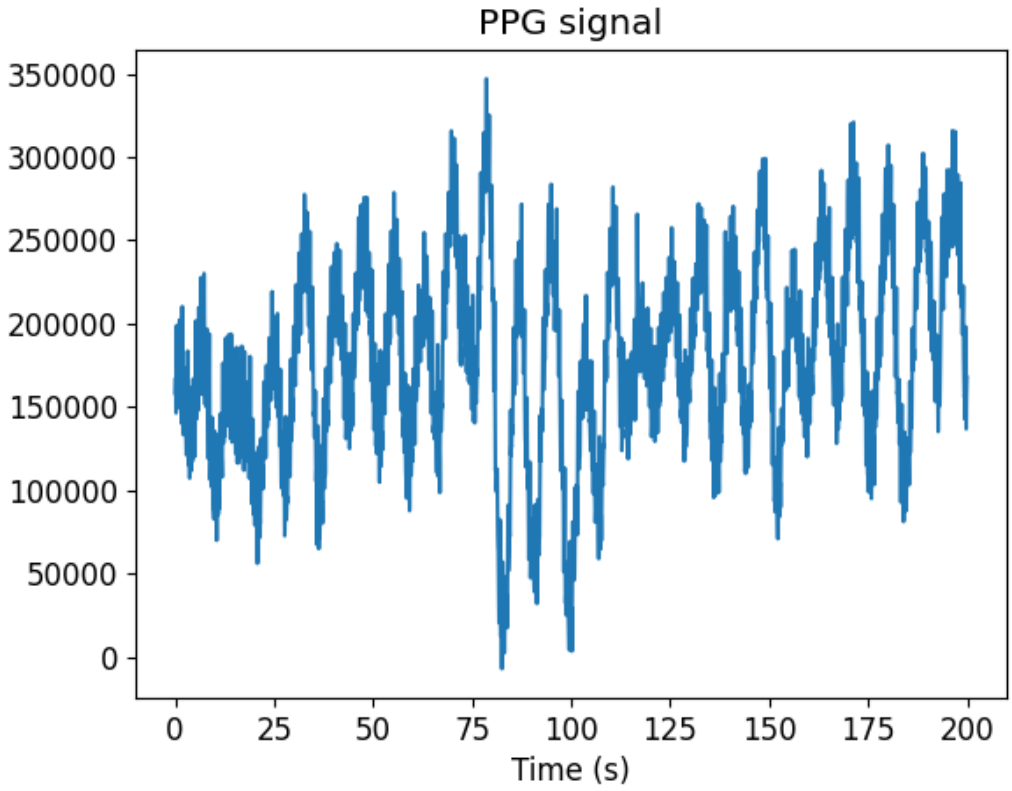
Experiment3

November 21, 2024

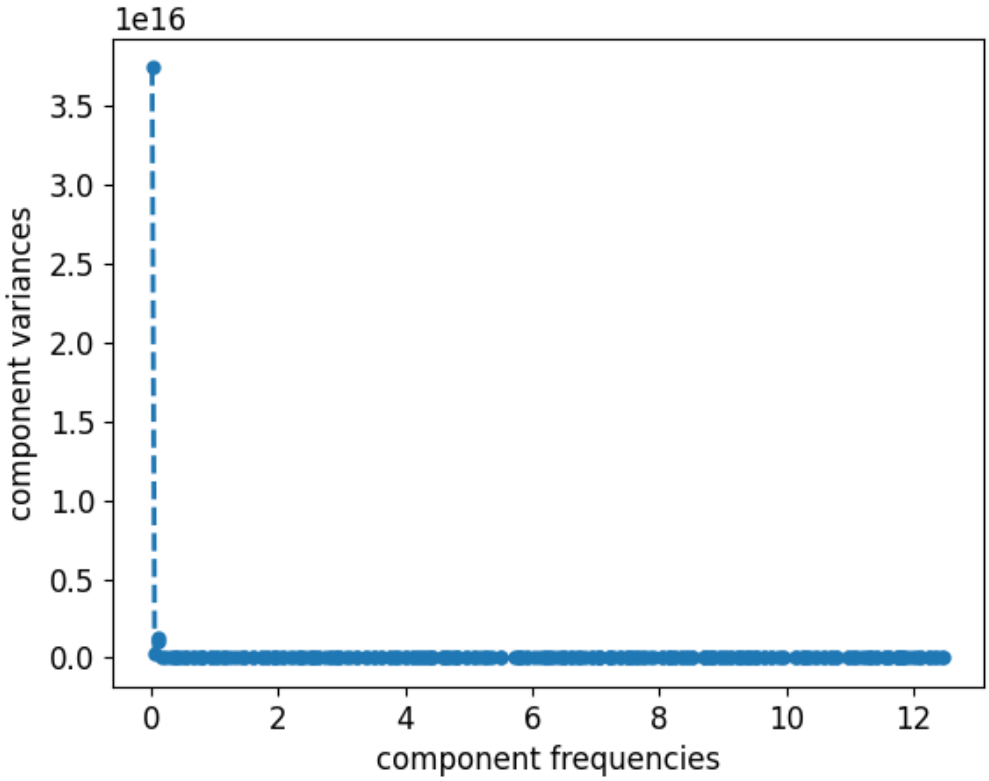
[53]:
```
%matplotlib widget

import pandas as pd
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
from SSA_Decomposition import ssa_decomposition

font = {'size': 12}
matplotlib.rc('font', **font)
```

[54]:
```
# The PPG signal (sampled at 25Hz)
data_csv = pd.read_csv('ppg.csv')
fs = 25
# Extract ppg and acceleromter signals between 200s and 400s
ppg = data_csv['ppg'][200*fs:400*fs].to_numpy()
acc_x = data_csv['acc_x'][200*fs:400*fs].to_numpy()
acc_y = data_csv['acc_y'][200*fs:400*fs].to_numpy()
acc_z = data_csv['acc_z'][200*fs:400*fs].to_numpy()
acc_norm = np.sqrt(acc_x*acc_x+acc_y*acc_y+acc_z*acc_z)

time = np.arange(0, len(ppg)/fs, 1/fs)
```

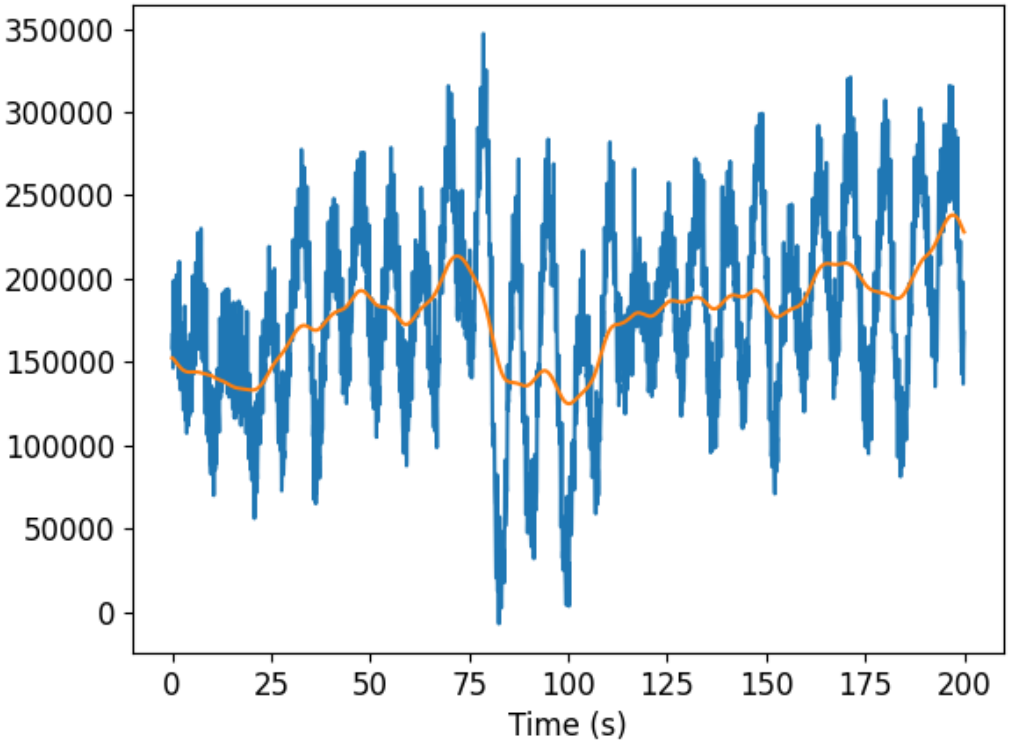
[55]:
```
# Plot data
fig = plt.figure()
plt.plot(time, ppg)
plt.title('PPG signal')
plt.ylabel('PPG')
plt.xlabel('Time (s)')
plt.show()
```



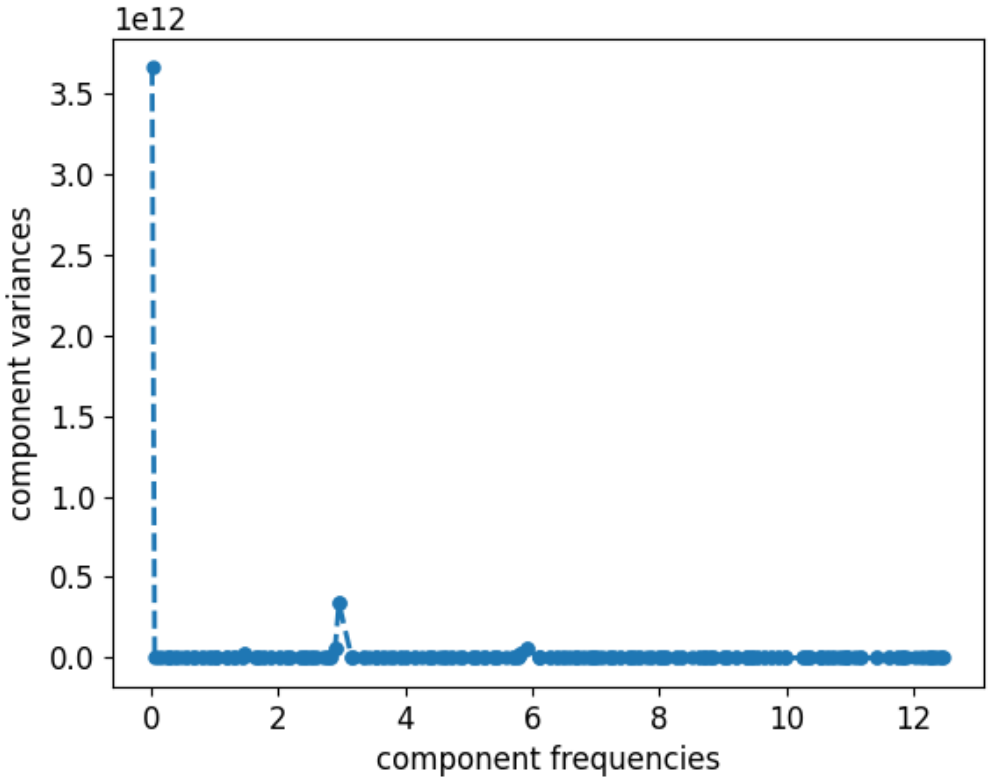
[56]:
```
# First SSA decomposition to extract long term drift
# Find the length L in samples, so that SVD first component corresponds to
# long term drift signal.
L = 10*fs
Y_1, fc_1, sig_1 = ssa_decomposition(ppg, L, fs, 2)
```



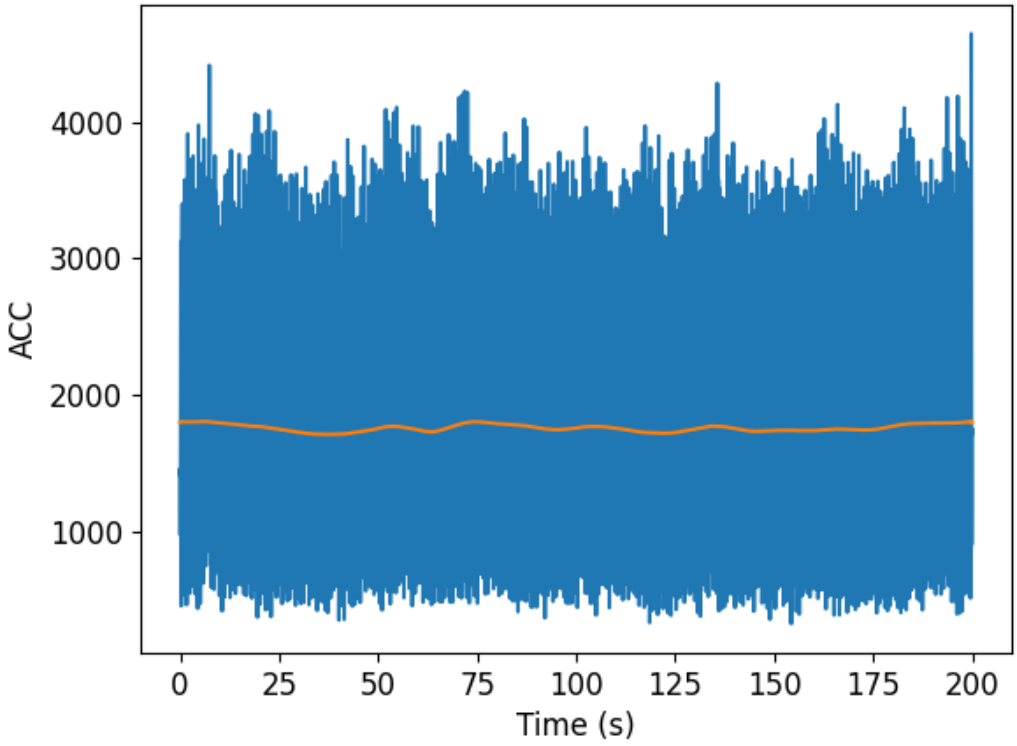
[57]:
```
# Plot signal and its baseline
fig = plt.figure()
plt.plot(time, ppg)
plt.plot(time, Y_1[:, 0].flatten(), label='baseline')
plt.ylabel('PPG')
plt.xlabel('Time (s)')
plt.show()
```



[58]:
```
# First SSA decomposition to extract long term drift from accelerometer
Y_acc1, fc_acc1, sig_acc1 = ssa_decomposition(acc_norm, L, fs, 2)
```



[59]:
```
fig = plt.figure()
plt.plot(time, acc_norm)
plt.plot(time, Y_acc1[:, 0].flatten(), label='baseline')
plt.ylabel('ACC')
plt.xlabel('Time (s)')
plt.show()
```

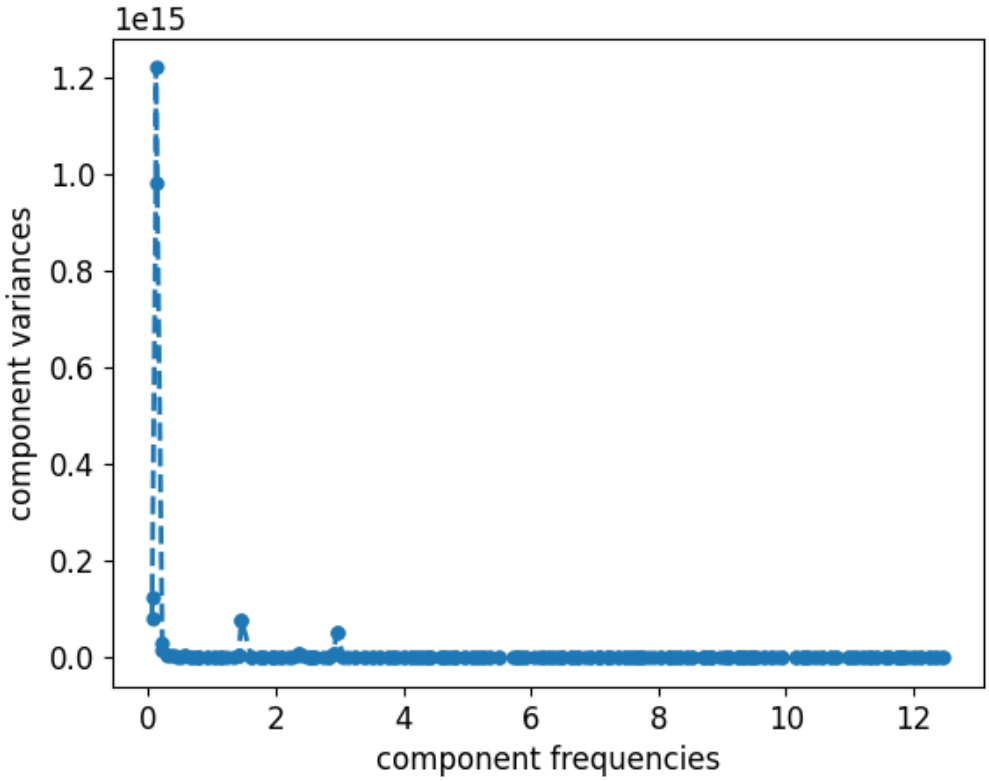


0.0.1 Question 3.1

One finds that $L = 10f_s$ is a good window length and highlights well the long-term drifting component for both PPG and accelerometer signals. Moreover one can see that the orange line follows more or less the original signal.

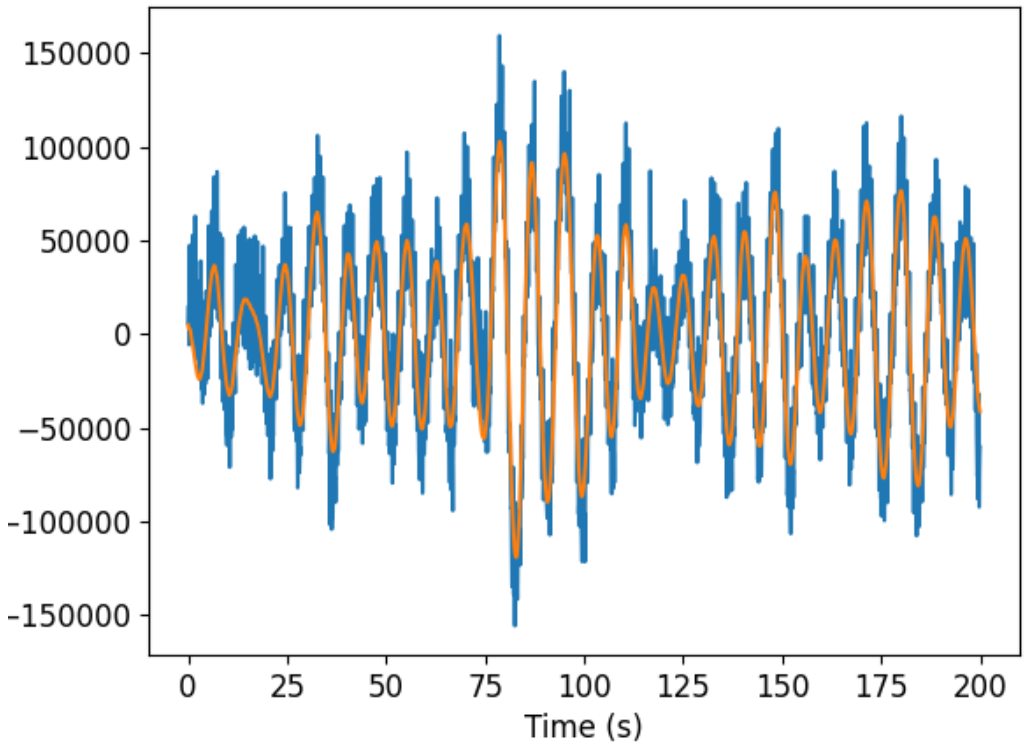
0.0.2 Question 3.2

[60]:
```
# Remove First component
filt_x = ppg - Y_1[:, 0]
# Apply SSA on signal without baseline
# Set L to identify and remove respiration component
Y_2, fc_2, sig_2 = ssa_decomposition(filt_x, L, fs, 2)
```



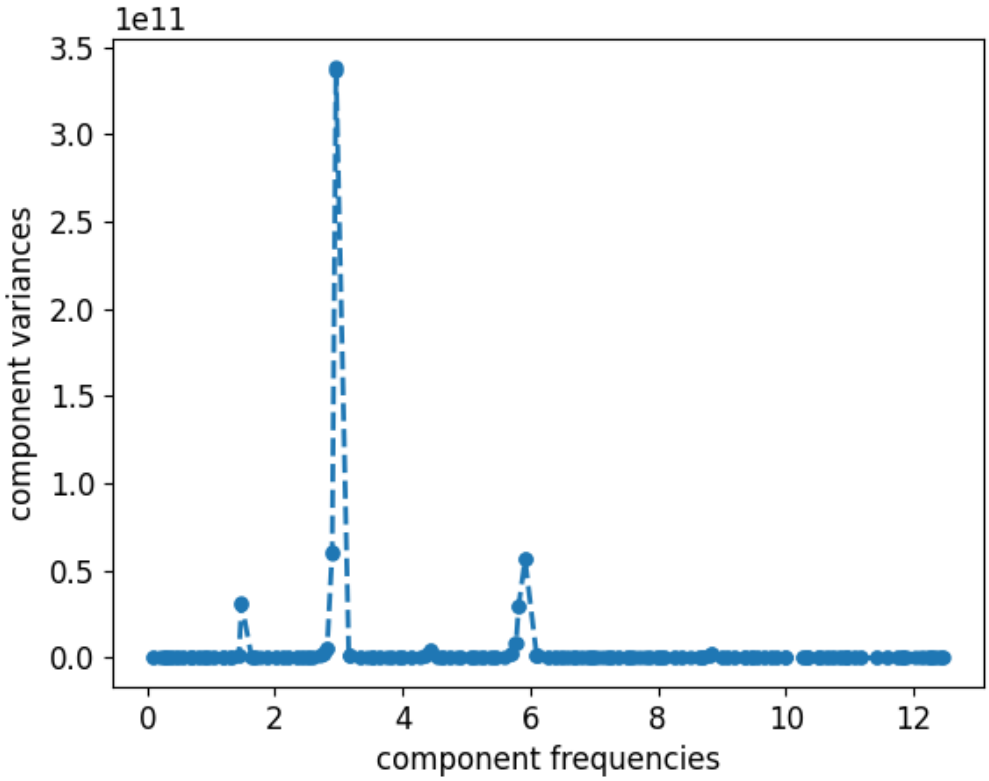
0.0.3 Question 3.2 a)

[66]:
```
# Find components characterizing respiration
resp_components = [0, 1, 3]
fig = plt.figure()
plt.plot(time, filt_x)
plt.plot(time, np.sum(Y_2[:, resp_components], axis=1), label='Respiration')
plt.ylabel('PPG')
plt.xlabel('Time (s)')
plt.show()
```



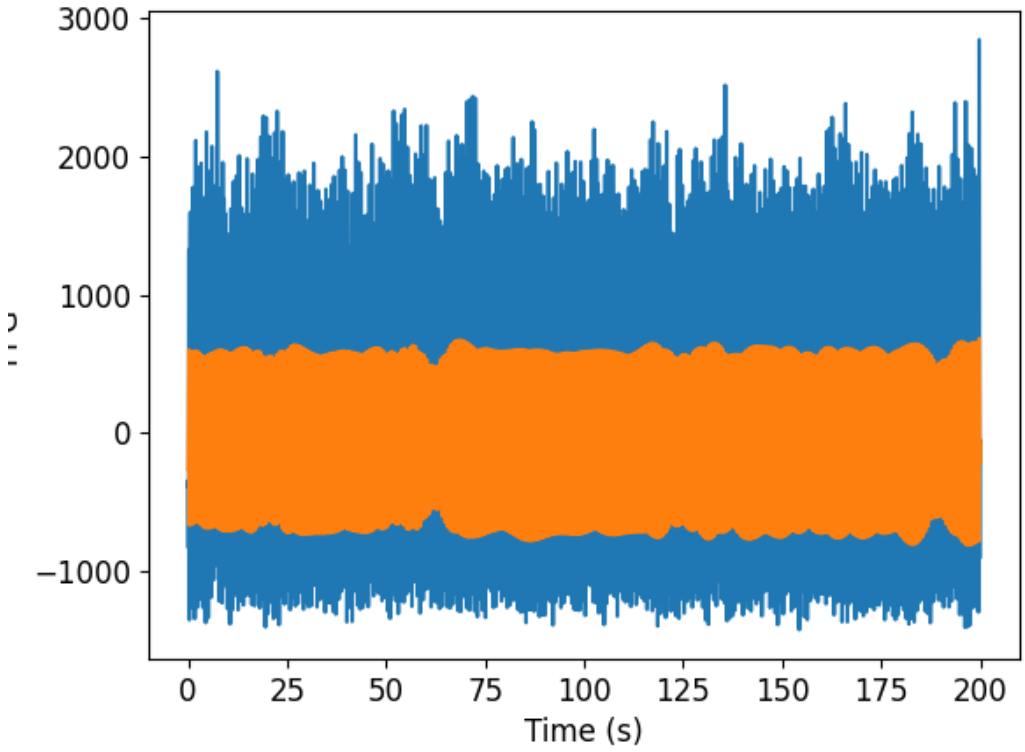
0.0.4 Question 3.2 b)

[62]:
```
# Remove First component
filt_acc = acc_norm - Y_acc1[:, 0]
# Apply SSA on accelerometer norm without baseline
Y_acc2, fc_acc2, sig_acc2 = ssa_decomposition(filt_acc, L, fs, 2)
```



One reads on the graph above a dominant frequency of 3 steps per second (which corresponds to the 180 steps per minute), representing the cadence.

[63]:
```
cadence_components = [1,3,6]
fig = plt.figure()
plt.plot(time, filt_acc)
plt.plot(time, np.sum(Y_acc2[:, cadence_components], axis=1), label='Cadence')
plt.ylabel('PPG')
plt.xlabel('Time (s)')
plt.show()
```

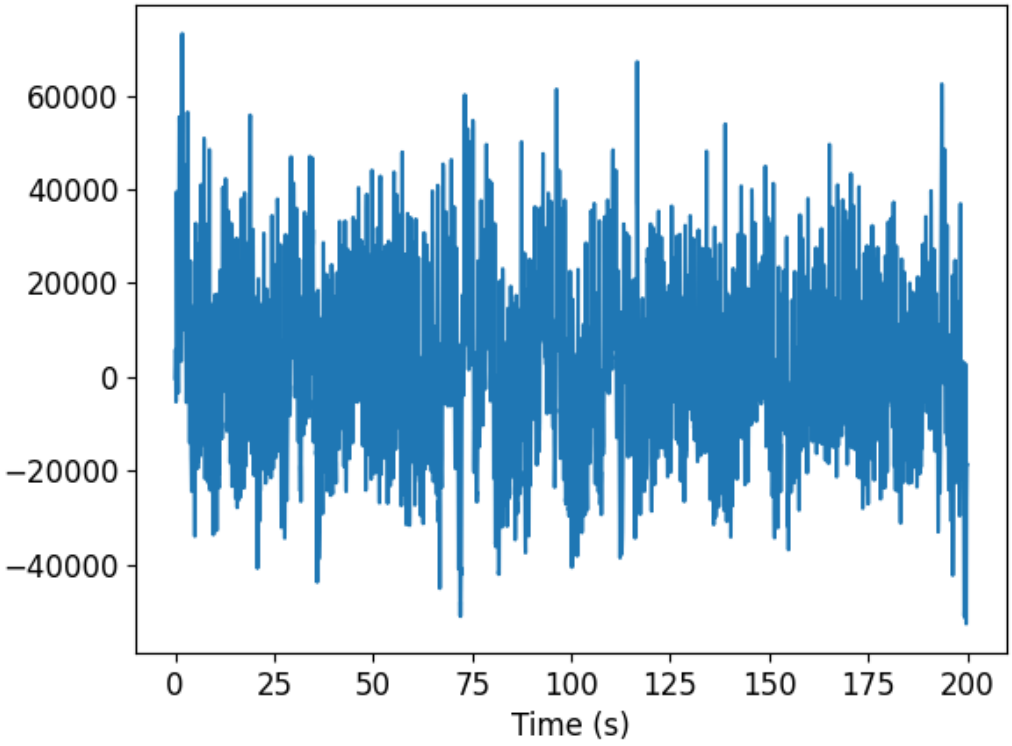


0.0.5 Question 3.2 c)

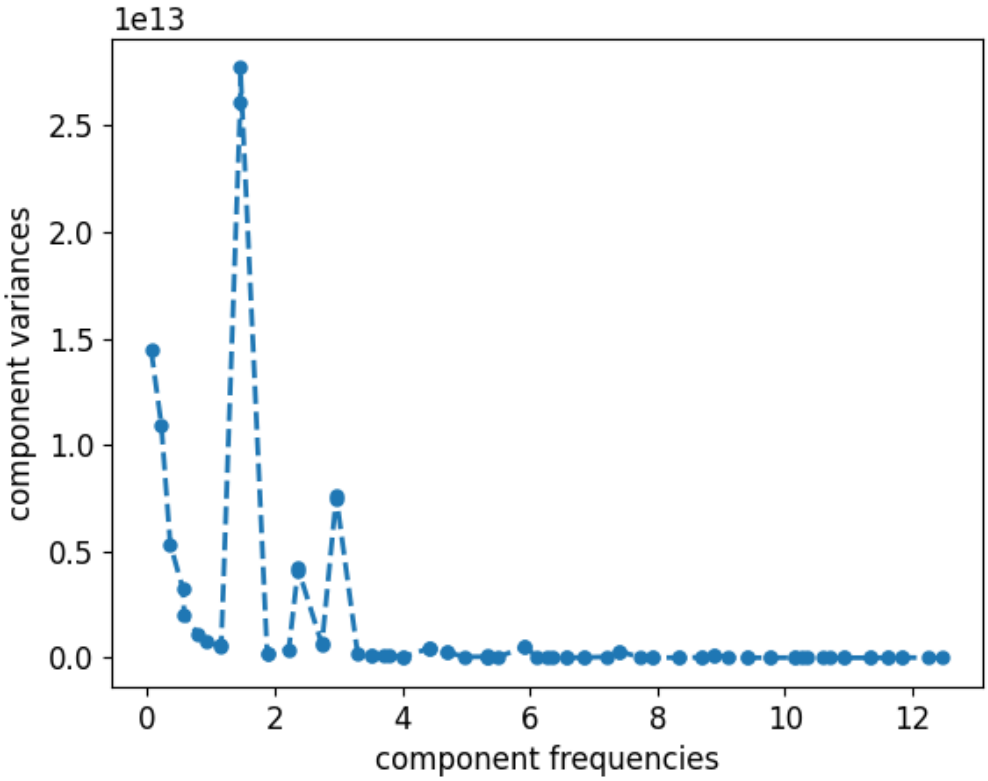
- The 1.5 Hz component comes from the arm movement, which corresponds to half of the step frequency (c.f. Lab 4).
- The frequency at 6Hz corresponds to the first harmonic of the step frequency (3Hz)

0.0.6 Question 3.3

[64]:
```
# Remove respiration component(s)
filt_x2 = filt_x - np.sum(Y_2[:, resp_components], 1)
fig = plt.figure()
plt.plot(time, filt_x2)
plt.ylabel('PPG')
plt.xlabel('Time (s)')
plt.show()
```



[65]:
```
# Apply SSA on signal without baseline, nor respiration
# Set L to identify running cadence and heart rate
Y_3, fc_3, sig_3 = ssa_decomposition(filt_x2, 80, fs, 2)
```



From the above graph, one can observe a strong component at ~1.5Hz, coresponding to 90 bpm which is coherent with the activity done by the subject.