How to use ButterworthBandpass

井民全, Jing, mqjing@gmail.com

Back to the Main Page

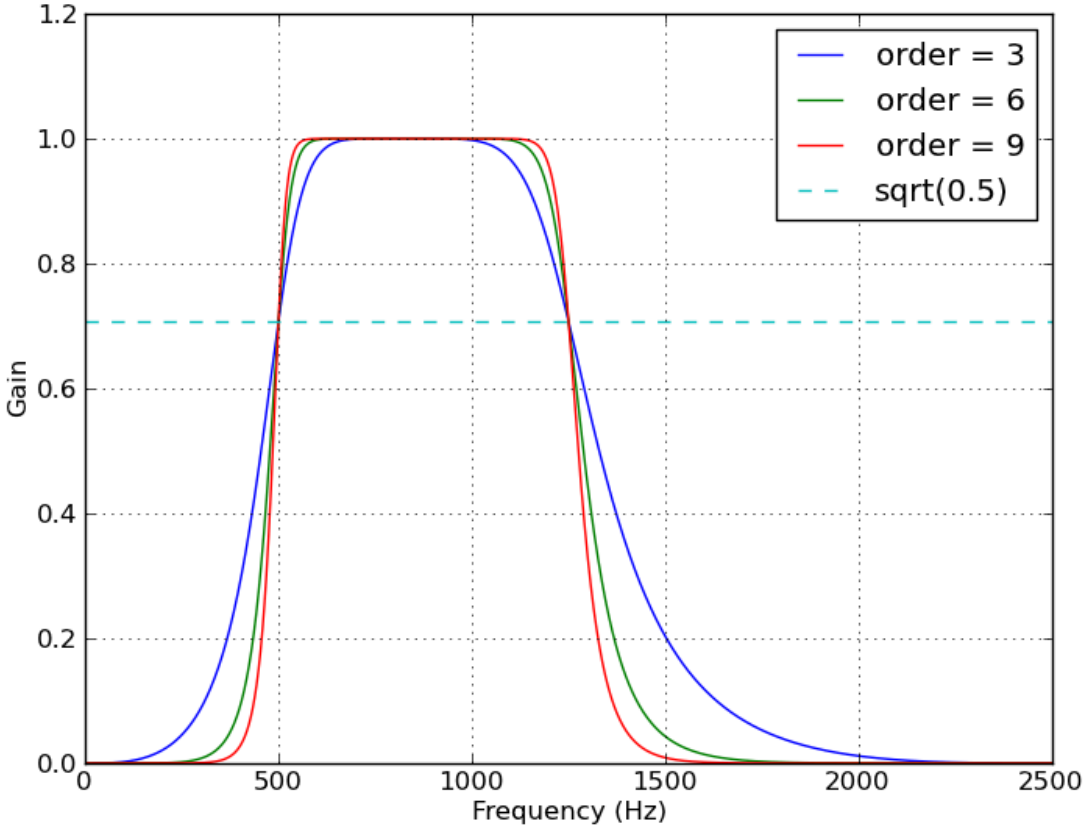
Purpose

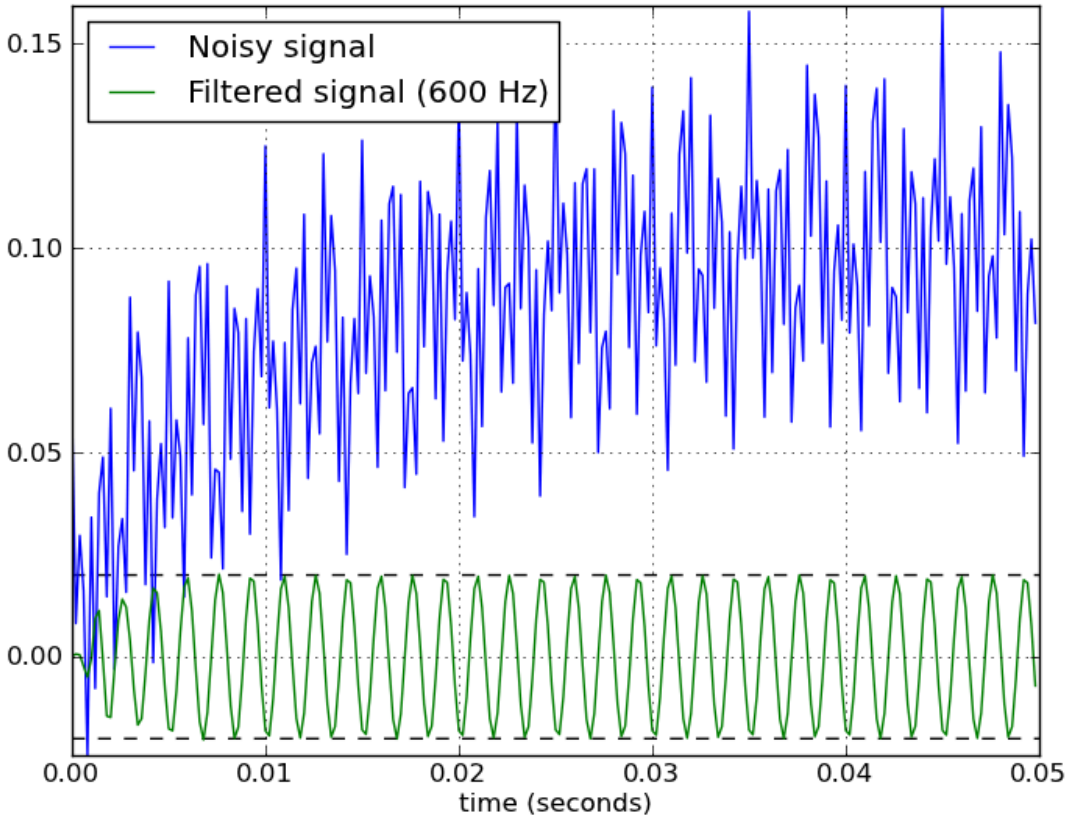
This document descrpts how to use ButterworthBandpass filter to remove the baseline wander problem on general ECG signals.

基本 ButterworthBandpass filter

```
from scipy.signal import butter, lfilter

# ============== 方便函數 ==============
def butter_bandpass(lowcut, highcut, fs, order=5):
    nyq = 0.5 * fs
    low = lowcut / nyq
    high = highcut / nyq
    b, a = butter(order, [low, high], btype='band')
    return b, a


def butter_bandpass_filter(data, lowcut, highcut, fs, order=5):
    b, a = butter_bandpass(lowcut, highcut, fs, order=order)
    y = lfilter(b, a, data)
    return y


if __name__ == "__main__":
    import numpy as np
    import matplotlib.pyplot as plt
    from scipy.signal import freqz

    # Sample rate and desired cutoff frequencies (in Hz).
    fs = 5000.0
```

```
    lowcut = 500.0
    highcut = 1250.0

    # 顯示不同 order 下的 frequence response
    plt.figure(1)
    plt.clf()
    for order in [3, 6, 9]:
        b, a = butter_bandpass(lowcut, highcut, fs, order=order)
        w, h = freqz(b, a, worN=2000)
        plt.plot((fs * 0.5 / np.pi) * w, abs(h), label="order = %d" % order)

    plt.plot([0, 0.5 * fs], [np.sqrt(0.5), np.sqrt(0.5)],
             '--', label='sqrt(0.5)')
    plt.xlabel('Frequency (Hz)')
    plt.ylabel('Gain')
    plt.grid(True)
    plt.legend(loc='best')

    # 建立有雜訊的訊號並且畫出來
    T = 0.05
    nsamples = T * fs
    t = np.linspace(0, T, nsamples, endpoint=False)
    a = 0.02
    f0 = 600.0
    x = 0.1 * np.sin(2 * np.pi * 1.2 * np.sqrt(t))
    x += 0.01 * np.cos(2 * np.pi * 312 * t + 0.1)
    x += a * np.cos(2 * np.pi * f0 * t + .11)
    x += 0.03 * np.cos(2 * np.pi * 2000 * t)
    plt.figure(2)
    plt.clf()
    plt.plot(t, x, label='Noisy signal')

    # ===== 關鍵片段 ====
    # 把資料丟進 bandpass filter 處理
    y = butter_bandpass_filter(x, lowcut, highcut, fs, order=6)
    plt.plot(t, y, label='Filtered signal (%g Hz)' % f0)
    plt.xlabel('time (seconds)')
    plt.hlines([-a, a], 0, T, linestyles='--')
    plt.grid(True)
    plt.axis('tight')
    plt.legend(loc='upper left')

    plt.show()
```


order = 3
order = 6
order = 9
sqrt(0.5)
Gain
Frequency (Hz)
0.0
0.2
0.4
0.6
0.8
1.0
1.2
0
500
1000
1500
2000
2500



ECG baseline Wander Removing

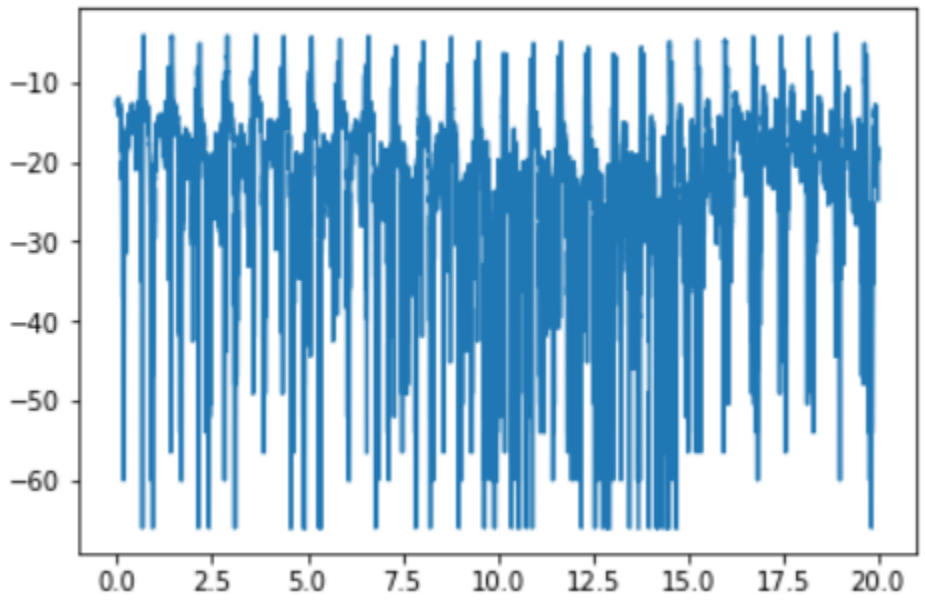
讀取 ECG 數據

```
from IPython.display import display
import numpy
import os
import shutil

import wfdb
import matplotlib.pylab as plt

# local
record = wfdb.rdrecord('./s0010_re', sampto= 10 * 1000)
```

```
#wfdb.plot_wfdb(record=record) # show all channel
#display(record.__dict__)      # print all information

signals, fields = wfdb.rdsamp('./s0010_re', channels=[0], sampfrom=0, sampto=20000)
print(signals[0])
numpy.savetxt("foo.csv", signals, delimiter=",")

# graph
t = numpy.linspace(0, 20000, 20000)
plt.plot(t, signals)
#display(signals)
#display(fields)
plt.show()
```

顯示 ECG 波形的 power spectrum 以得知 ECG 頻率範圍

```
p = 20*np.log10(np.abs(np.fft.rfft(signals)))       # 取得訊號的 power spectrum

T = 20    # duration 20 sec
nsamples = T * fs
t = np.linspace(0, T, nsamples, endpoint=False)
plt.plot(t, p)
plt.show()
```



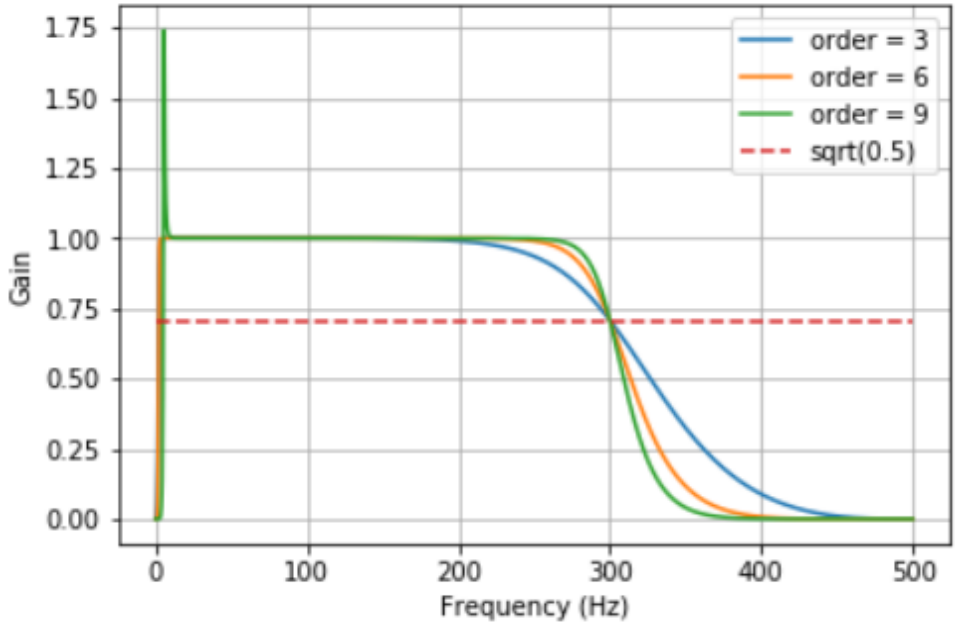
移除 baseline wander effect

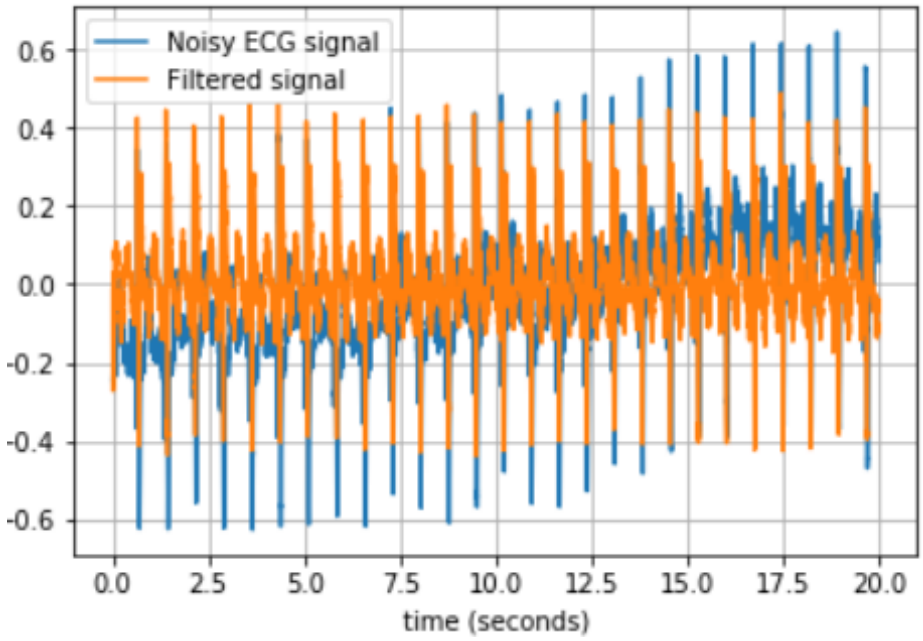
```
from scipy.signal import butter, lfilter


def butter_bandpass(lowcut, highcut, fs, order=5):
    nyq = 0.5 * fs
    low = lowcut / nyq
    high = highcut / nyq
    b, a = butter(order, [low, high], btype='band')
    return b, a


def butter_bandpass_filter(data, lowcut, highcut, fs, order=5):
    b, a = butter_bandpass(lowcut, highcut, fs, order=order)
    y = lfilter(b, a, data)
    return y


if __name__ == "__main__":
    import numpy as np
    import matplotlib.pyplot as plt
    from scipy.signal import freqz

    # Sample rate and desired cutoff frequencies (in Hz).
    fs = 1000.0
    lowcut = 2
    highcut = 300


     # Plot the frequency response for a few different orders.
    plt.figure(1)
    plt.clf()
    for order in [3, 6, 9]:
        b, a = butter_bandpass(lowcut, highcut, fs, order=order)
        w, h = freqz(b, a, worN=2000)
        plt.plot((fs * 0.5 / np.pi) * w, abs(h), label="order = %d" % order)

    plt.plot([0, 0.5 * fs], [np.sqrt(0.5), np.sqrt(0.5)],
             '--', label='sqrt(0.5)')
    plt.xlabel('Frequency (Hz)')
    plt.ylabel('Gain')
    plt.grid(True)
```

```
plt.legend(loc='best')

# Filter a noisy signal.
T = 20    # duration 20 sec
nsamples = T * fs
t = np.linspace(0, T, nsamples, endpoint=False)

plt.figure(2)
plt.clf()

plt.plot(t, signals, label='Noisy ECG signal')

signals1d= signals.reshape(-1)
```

y = butter_bandpass_filter(signals1d, lowcut, highcut, fs, order=6)

```
plt.plot(t, y, label='Filtered signal')
plt.xlabel('time (seconds)')
plt.grid(True)
plt.axis('tight')
plt.legend(loc='upper left')



plt.show()
```





全部合在一起

```
#讀取 ECG 數據
from IPython.display import display
import numpy
import os
import shutil

import wfdb
import matplotlib.pylab as plt

# local
record = wfdb.rdrecord('./s0010_re', sampto= 10 * 1000)
#wfdb.plot_wfdb(record=record) # show all channel
#display(record.__dict__)      # print all information

signals, fields = wfdb.rdsamp('./s0010_re', channels=[0], sampfrom=0, sampto=20000)
print(signals[0])
numpy.savetxt("foo.csv", signals, delimiter=",")

# graph
t = numpy.linspace(0, 20000, 20000)
plt.plot(t, signals)
#display(signals)
```

```
#display(fields)
plt.show()

# 移除 baseline wander 效應
from scipy.signal import butter, lfilter


def butter_bandpass(lowcut, highcut, fs, order=5):
    nyq = 0.5 * fs
    low = lowcut / nyq
    high = highcut / nyq
    b, a = butter(order, [low, high], btype='band')
    return b, a


def butter_bandpass_filter(data, lowcut, highcut, fs, order=5):
    b, a = butter_bandpass(lowcut, highcut, fs, order=order)
    y = lfilter(b, a, data)
    return y


if __name__ == "__main__":
    import numpy as np
    import matplotlib.pyplot as plt
    from scipy.signal import freqz

    # Sample rate and desired cutoff frequencies (in Hz).
    fs = 1000.0
    lowcut = 2
    highcut = 300


     # Plot the frequency response for a few different orders.
    plt.figure(1)
    plt.clf()
    for order in [3, 6, 9]:
        b, a = butter_bandpass(lowcut, highcut, fs, order=order)
        w, h = freqz(b, a, worN=2000)
        plt.plot((fs * 0.5 / np.pi) * w, abs(h), label="order = %d" % order)

    plt.plot([0, 0.5 * fs], [np.sqrt(0.5), np.sqrt(0.5)],
             '--', label='sqrt(0.5)')
    plt.xlabel('Frequency (Hz)')
    plt.ylabel('Gain')
    plt.grid(True)
    plt.legend(loc='best')
```

```
# Filter a noisy signal.
T = 20    # duration 20 sec
nsamples = T * fs
t = np.linspace(0, T, nsamples, endpoint=False)

plt.figure(2)
plt.clf()

plt.plot(t, signals, label='Noisy ECG signal')

signals1d= signals.reshape(-1)
 y = butter_bandpass_filter(signals1d, lowcut, highcut, fs, order=6)
plt.plot(t, y, label='Filtered signal')
plt.xlabel('time (seconds)')
plt.grid(True)
plt.axis('tight')
plt.legend(loc='upper left')



plt.show()
```

Reference

- http://scipy.github.io/old-wiki/pages/Cookbook/ButterworthBandpass

Generate noise wave

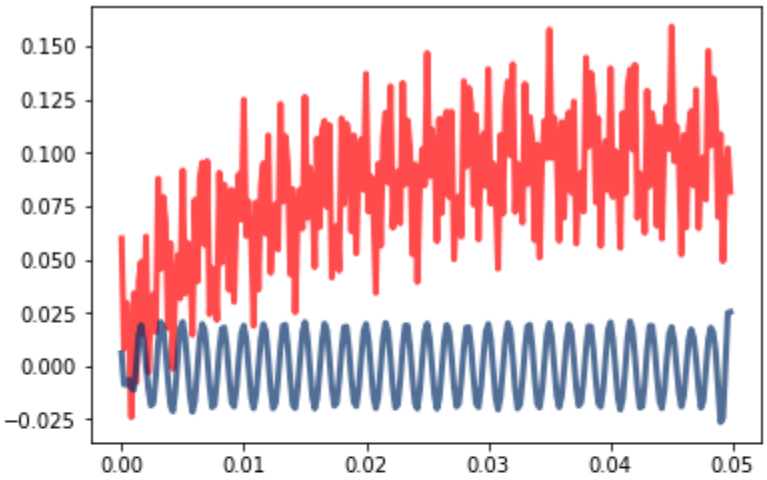
```
import numpy as np
import matplotlib.pyplot as plt

# base line remove experiments
# bandpass filter method to remove baseline
```

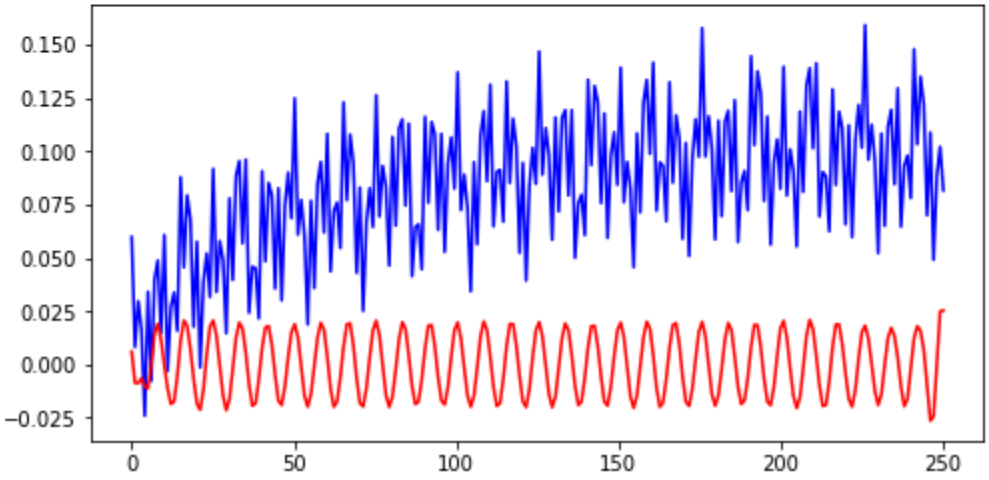
```
import sys
sys.path.insert(1, r'./functions')  # add to pythonpath
from filters import butter_bandpass_filter


from myplot import drawLine, drawMarker

def GenSign():
    # 建立有雜訊的訊號並且畫出來
    fs = 500
    T = 0.05
    nsamples = T * fs
    t = np.linspace(0, T, nsamples, endpoint=False)
    a = 0.02
    f0 = 600.0
    x = 0.1 * np.sin(2 * np.pi * 1.2 * np.sqrt(t))
    x += 0.01 * np.cos(2 * np.pi * 312 * t + 0.1)
    x += a * np.cos(2 * np.pi * f0 * t + .11)
    x += 0.03 * np.cos(2 * np.pi * 2000 * t)
    return x
```

ThinkDSP Version

```
# ThinkDSP Version: remove baseline
import sys
sys.path.insert(1, r'./functions/ThinkDSP/code')  # add to pythonpath
import thinkdsp

fs = 5000
lowcut = 500.0
highcut = 1250.0

my_data = GenSign()
wavObj = thinkdsp.Wave(ys = my_data, framerate = fs)
#wavObj.make_audio()
wavObj.plot(color='red')
segment = wavObj.segment()   # all data
#segment = wavObj.segment(0, 4) # only 4 sec
#segment.plot()
```

```
spectrum = segment.make_spectrum()
#spectrum.plot()


spectrum.low_pass(highcut) # (500)
spectrum.high_pass(lowcut) # (3)

# spectrum -> wave
filtered = spectrum.make_wave()
filtered.plot()
#print(filtered.ys)
print('ok')

plt2=drawLine(2, my_data, 'b')
plt2=drawLine(2, filtered.ys, 'r')
plt2.show()
```

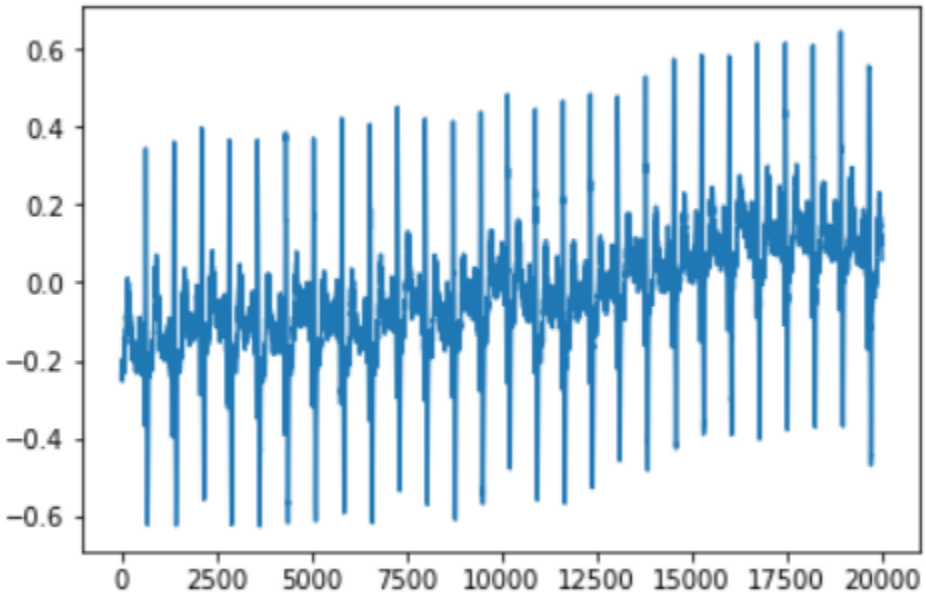


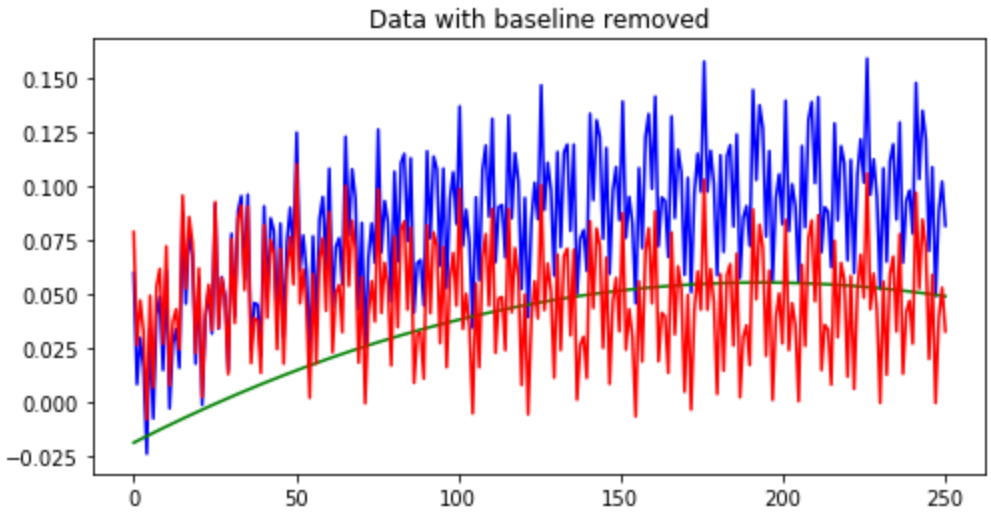


PeakUtil

```
# base line remove experiment 2
import peakutils
import matplotlib.pyplot as plt

fs = 5000
lowcut = 500.0
highcut = 1250.0
```

```
my_data = GenSign()

drawLine(1, my_data, 'b')

N = len(my_data)
t = np.linspace(0, N, N)  #產生一條 sampling vector, 範圍從 0 ~ N, 總共 N 個 samples

base = peakutils.baseline(my_data, 2)


plt.figure(1, figsize=(10,6))
plt.plot(t, base, 'g')
plt.plot(t, my_data-base, 'r')
plt.title("Data with baseline removed")


plt.show()
```


0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
0
2500
5000
7500
10000
12500
15000
17500
20000


Data with baseline removed
0.150
0.125
0.100
0.075
0.050
0.025
0.000
−0.025
0
50
100
150
200
250