北京化工大学

Beijing University of Chemical Technology

Dynamic Time Warping

生产实习报告

计科 1906 李腾飞

目录

创建日期：2022 年 8 月 28 日
更新日期：2022 年 9 月 1 日

第 1 章　论文概述

基于经验模式分解和希尔伯特-黄变换的瞬时三维脑电信号分析应用于麻醉深度。

可执行文件下载:https://letmefly.xyz/Links/Redirect/id.html?21（必须和case7.txt放在相同目录下使用）

1.1　论文中的名词中英文对照

表 1.1:　英文名词缩写-中文名词对照表

英文缩写	中文
DoA	Depth of anaesthesia（麻醉深度）
EEG	electroencephalography（脑电图学）
EMD	empirical mode decomposition（经验模态分解）
HHT	Hilbert-Huang transform（希尔伯特-黄变换）
HT	Hilbert transform（希尔伯特变换）
IMF	intrinsic mode functions（固有模态函数）
SampEn	sample entropy（样本熵）
BIS	bispectral index（脑电双频指数）
ECG	electrocardiography（心电描记术）
FFT	Fast Fourier Transform（快速傅里叶变换）
AUC ratio of $\alpha+\beta$ waves	area ratio of $\alpha+\beta$ waves (8-32 Hz) under FFT curve（快速傅立叶变换曲线下 α+β 波 (8-32 赫兹) 的面积比）
EMG	electromyography（肌电图描记法）
EOG	electrooculography（眼动电图描记法）
ESUs	electrosurgical units（电外科装置）
IFFT	Inverse fast Fourier transform（快速傅里叶逆变换）
ApEn	approximate entropy（近似熵）

1.2　预备知识

1.2.1　时域和频域

时域是客观世界中唯一实际存在域；频域是一个数学构造，也被一些学者称为上帝视角。

1.2.2 时域

以时间轴为坐标表示动态信号的关系

1.2.3 频域

正弦波是频域中唯一存在的波形

1.2.4 转换

动态信号从时间域变换到频率域主要通过傅立叶级数和傅立叶变换实现。周期信号靠傅立叶级数，非周期信号靠傅立叶变换。时域越宽，频域越短。

1.3 论文摘要

麻醉的程度 (DoA) 是评估全身麻醉剂对患者中枢神经系统抑制程度的重要指标。通过监测病人脑电信号（electroencephalography(EEG)）来判断病人是否处于全麻状态有助于调节 DoA 并减轻手术的风险。

1.4 介绍

EMD 可以从原始 EEG 信号中滤除噪音相关的频率，并可以结合挑选出来的固有模态函数得到滤波后的 EEG 信号。之后在滤波后的 EEG 信号的基础上用 HHT 得到瞬时频率和瞬时振幅。

之后就可以用“瞬时频率、瞬时振幅和 EEG 信号中的时间元素”重组并构建可以实时显示脑电信号振幅和频率的实时 3D 表示图。

第 2 章 论文复现方法简述

2.1 step1 将原有 EEG 信号分解为数个 IMF

$$x(t) = \sum_{i=1}^{n} c_i(t) + r_n(t)$$

$x(t)$ 是时域下的原始信号，$c_i(t)$ 是第 i 个 IMF（固有模态函数），$r_n(t)$ 是残余信号。

2.2 step2 将 IMF 转换到频域

使用 FFT 将 IMF 从时域转换到频域。

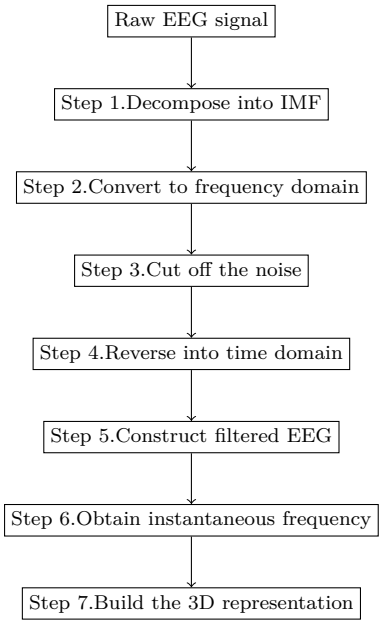

图 2.1: 方法流程图

2.3 step3 去除每个 IMF 中的噪声

人脑产生的脑电信号的正常频率在 0.5 赫兹 Hz 到 32Hz 之间，它们分别包含从低频到高频的 δ，θ，α 波和 β 波。

2.4 step4 将 IMF 转回到时域

使用快速傅立叶逆变换 IFFT

2.5 step5 构造滤波后的脑电信号的 IMF

IFFT 带来的边缘效应会使前 5 秒和后 5 秒的振幅异常地高，因此去掉前 5 秒和后 5 秒。

丢掉过小的 IMF。

2.6 step6 通过 HHT 获得实时频率

δ (0.5–4 Hz), θ (4–8 Hz), α (8–16 Hz), β (16–32 Hz)

Other: Noise

2.7 step7 构建三维表示

x 轴：频率，y 轴：时间，z 轴：振幅

第 3 章　具体实现

1. Codes/main.py：主程序，主要负责调用
2. Codes/BaseClass.py：数据类，包括数据、起始截止时间、采样频率等
3. Codes/BaseFunction.py：一些基本的函数功能
4. Codes/LetEMD.py：实现了 EEG 的 EMD 分解
5. Codes/Visualize.py：将数据可视化
6. Codes/Txt2numpy.py：读取数据
7. Codes/ShowOriginalEEGdata.py：显示原始 EEG
8. Codes/Data2IMFs.py：将数据分成很多 IMF、将 IMF 转换到频域
9. Codes/IMFs2FrequencyDomain.py：将 IMF 转换到频域
10. Codes/CutoffNoice.py：将信号中不符合频率范围的部分删除
11. Codes/IMF2TimeDomain.py：将 IMF 转回时域
12. Codes/ConstructEEG.py：将 IMF 构建回 EEG（并去除前 10s 的信号）
13. Codes/HHT.py：通过 HHT 获得实时频率
14. Codes/ShowRealtime3D.py：显示为实时三维图

3.1　主程序 main.py

main.py 是总函数，主要负责函数调用。
主要包括“导入所需库”、“取消 QT 警告”和“调用其他写好的函数”

3.1.1　导入所需库

这一小节没有太多需要讲述的。在这一小节中，所导入的**全部**是我自己实现的功能，每个库实现一个小功能，具体实现的时候很多地方使用到了现成库。

```
from Txt2numpy import txt2numpy
from ShowOriginalEEGdata import showOriginalEEGdata
from Data2IMFs import data2IMFs
from IMFs2FrequencyDomain import IMFs2FrequencyDomain
from CutoffNoise import cutoffNoise
from IMF2TimeDomain import IMF2TimeDomain
```

```
from ConstructEEG import ConstructEEG
from HHT import HHT
from ShowRealtime3D import showRealtime3D
```

3.1.2 取消 QT 警告

因为我电脑上安装了 QT（并将 QT 安装配置过程写成了笔记），因此在使用 matplotlib 画图的时候，控制台就会输出 QT 警告。

因此，这里参考https://www.likecs.com/ask-924744.html，通过临时改变一些环境变量，达到了消除警告的目的。

```
import os
# 取消QT警告
os.environ["QT_DEVICE_PIXEL_RATIO"] = "0"
os.environ["QT_AUTO_SCREEN_SCALE_FACTOR"] = "1"
os.environ["QT_SCREEN_SCALE_FACTORS"] = "1"
os.environ["QT_SCALE_FACTOR"] = "1"
```

3.1.3 调用其他写好的函数

主函数的最后一个任务就是调用其他写好的函数，包括：

```
EEG = txt2numpy() # 从文本文件读入数据
showOriginalEEGdata(EEG) # 显示原本EEG文件
IMFs = data2IMFs(EEG) # step1.将EEG数据转换为IMF
IMFs = IMFs2FrequencyDomain(IMFs=IMFs) # step2.将IMF转换到频域
IMFs = cutoffNoise(IMFs=IMFs) # step3.去除每个IMF中的噪声
IMFs = IMF2TimeDomain(IMFs=IMFs) # step4.转回到时域
EEG = ConstructEEG(IMFs=IMFs) # step5.选取有效范围的IMF并构建EEG
instantaneousFrequency = HHT(EEG) # step6.通过HHT获得实时频率
showRealtime3D(EEG, instantaneousFrequency) # step7.显示为实时三维图
```

3.2 基本数据类 BaseClass.py

BaseClass.py 主要实现了基本的数据类，包括数据、数据的采样频率、起始采样时间等。

3.2.1 所支持数据类型

这个函数中仅有一个实现好的类 Data，它支持一维数据和二维数据。

表 3.1: Data 类所支持的数据类型

数据类型	例
一维数据：[1, 2, 1, 2, 3, ...]	Example: data.shape = (500,)
二维数据：[[1, 2, 1, ..], [2, 6, 3, ..], ..]	Example: data.shape = (5, 500)

3.2.2 成员

成员包括变量和方法。

3.2.2.1 变量

表 3.2: Data 成员变量及其意义

成员变量	解释说明
data: np.ndarray	主要数据
dataLength	数据的长度。这里时指单条数据共采样了多少次
startTime	数据的开始采样时间
endTime	数据的结束采样时间
	[startTime, endTime)
FPS	每秒采样多少次

3.2.2.2 方法

表 3.3: Data 成员方法及其意义

成员方法	解释说明
getData() -> np.ndarray	获取采样数据
getDataLength(data: np.ndarray) -> int	获取数据的长度（采样次数）
getStartTime() -> float	获取数据的开始采样时间
getEndTime() -> float	获取数据的结束采样时间
getFPS() -> int	获取数据的采样频率
getTimeRangeArray() -> np.ndarray	获取时间的数组（数组中为所有的采样时刻）
setSubDataByPoints(frontSkippedPoints: int, backSkippedPoints: int) -> None	设置数据为子数据（通过采样点来设置）
setSubDataByTime(frontSkippedTime: float, backSkippedTime: float) -> None	设置数据为子数据（通过时间来设置）

3.2.3 构造函数的参数

1. startTime
2. endTime
3. FPS

具体含义可参考“成员变量”中的解释。

3.3 可视化 Visualize.py

Visualize.py 主要实现数据的可视化。

此模块一共实现了三种可视化功能：“可视化 EEG（一组数据）”、“可视化 IMF（多组数据）”和“结果 3D 可视化”。

可视化部分主要使用了 3 个库：

1. 一个是 numpy 库，用于 3D 显示中对数据的处理。
2. 一个是 matplotlib 库，用于数据的可视化和图形的显示。
3. 一个是我构造的 BaseClass 库中的 Data 类，因为整个程序中，数据都是在 Data 类中存储的。

3.3.1 显示 EEG 图像

EEG 数据有一个时间轴，剩下的就是数据了。因此，可以使用 matplotlib 很方便地显示。

```
def showEEG(EEG: Data, title="EEG") -> None:
    ax = plt.subplot()
    ax.set_title(title)
    ax.plot(EEG.getTimeRangeArray(), EEG.data)
    plt.show()
```

程序接受两个参数，一个是自定义 Data 类的待显示数据，一个是显示的图表的标题。

3.3.2 显示 IMF 图像

一般 IMF 不只一个，因此这个函数就是要将每一个 IMF 分别显示

```
def showIMFs(IMFs: Data, title="IMFs") -> None:
    fig, axes = plt.subplots(IMFs.data.shape[0], 1)
    if IMFs.data.shape[0] == 1:
        axes = list(axes)
    axes[0].set_title(title)
    for num, IMF in enumerate(IMFs.data):
        ax = axes[num]
        ax.plot(IMFs.getTimeRangeArray(), IMF)
        ax.set_ylabel("IMF " + str(num + 1))
    plt.show()
```

程序接受两个参数，一个是自定义 Data 类的待显示数据，一个是显示的图表的标题。

3.3.3 显示 3D 图

3D 图将显示最终信号的时间轴、频率、相位，由此可以较为明显地观察到算法实现的效果。

```
def show3d(F: Data, A: Data, ColorfulAndResize=False):
    x_F = F
    y_time = A.getTimeRangeArray()
    z_A = A

    fig = plt.figure()
    ax = fig.add_subplot(111, projection='3d')

    if ColorfulAndResize:
        colors = []
        for f in x_F.data:
            if f < 0.5:
```

```
13                 colors.append("red")
14             elif f < 4:
15                 colors.append("blue")
16             elif f < 8:
17                 colors.append("yellow")
18             elif f < 16:
19                 colors.append("green")
20             elif f < 32:
21                 colors.append("pink")
22             else:
23                 colors.append("red")
24         im = ax.scatter(np.abs(x_F.data), y_time, np.abs(z_A.data), c=colors, s=1)
               # s: 点的大小
25         ax.set_zlim(0, 1000)
26         ax.set_title("Colorful & Resize")
27     else:
28         im = ax.scatter(np.abs(x_F.data), y_time, np.abs(z_A.data), s=1) # s: 点的
               大小
29
30
31     ax.set_xlabel("x_F")
32     ax.set_ylabel("y_time")
33     ax.set_zlabel("z_A")
34     plt.show()
```

程序接受三个参数，一个是自定义 Data 类的频率数据，一个是自定义 Data 类的相位，最后一个参数决定是否像论文中的格式那样显示图表（以便和论文中的结果进行对照）。

其中,不同频率的数据显示不同的颜色的方法学习自https://www.jb51.net/article/258747.htm,绘制三维散点图的方法学习自https://blog.csdn.net/weixin_46287157/article/details/124784731

3.4 数据读取 Txt2numpy.py

Txt2numpy.py 主要实现数据的读取。

3.4.1 数据来源

这里不得不提一提数据的来源。论文中为注明数据的来源，我尝试了很多类型的数据都和论文中的数据差别较大。

我尝试向论文中的通讯地址发送了一封邮件，请求作者分享一份数据给我，但是邮件始终未得到回复。

最终，我在承勇老师提供的一个链接里下载到了一个合适的数据。

下载地址:https://figshare.com/articles/dataset/EEG_and_BIS_raw_data/5589841/1

case7.txt 是 case7.mat 读入 matlab 后，将 EEG 取第 1 到 4000 个元素，并保存到文件后得到的。

Matlab 保存数据到文件的代码为：

```
dlmwrite('case7.txt', EEG(1: 4000), 'delimiter', ' ')
```

此方法学习自 CSDN:https://blog.csdn.net/whb12345678feng/article/details/103108114

3.5 读取数据

OK，既然我们得到了数据，那么就可以开始愉快地将数据读入 Python 了。

```
import numpy as np
from BaseClass import Data
def txt2numpy(file="../Data/case7.txt") -> Data:
        arr = np.loadtxt(file)
        return Data(arr, 0, 100) # 开始时间、采样频率
```

3.5.1 说明

虽然代码不是很长，但是把这个功能专门抽象出一个 .py 文件出来，是为了方便日后程序的拓展与完善。如：支持多种数据格式的读取、支持频率的设定等。

3.6 显示原有 EEG 信号 ShowOriginalEEGdata.py

ShowOriginalEEGdata.py 可视化显示原始 EEG 信号（处理前）

3.6.1 简介

前面我们已经实现了 EEG 信号的展示函数，因此这里就会非常方便，只需要调用一下前面实现的函数即可。

如果想要手动实现也很简单：

```
from matplotlib import pyplot as plt
from BaseClass import Data

def showOriginalEEGdata(EEG: Data) -> None:
    ax = plt.subplot()
    ax.set_title('Original EEG')
    ax.plot(EEG.getTimeRangeArray(), EEG.getData())
    plt.show()
```

3.6.2 效果展示

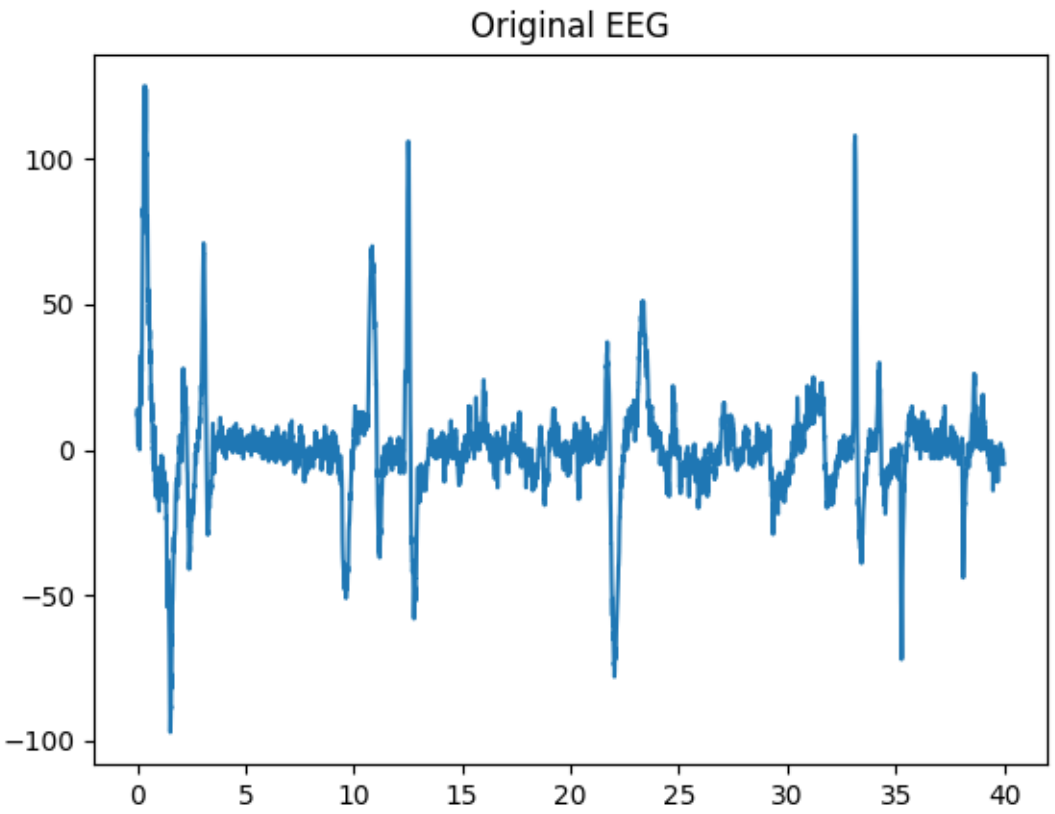

图 3.1: OriginalEEG

3.7 分解 EEG 为数个 IMF Data2IMFs.py

Data2IMFs.py 实现了一些 EEG 信号处理领域所需的功能

3.7.1 简介

在这里我花费了较多的精力，也写了较多的代码，我将 EMD 的实现抽象成了一个.py 文件，后面会讲具体实现原理。

这个文件的目的是调用另外一个文件 LetEMD.py 将原始 EEG 信号分解为数个 IMF 和“剩余部分”

同时，这个文件又手动实现了一遍数据的可视化展示，因为我实现的可视化展示类中，不支持显示“Residue”这个标题

```
1 from LetEMD import EMD
2 from matplotlib import pyplot as plt
3 from BaseClass import Data
```

```
4
5
6  def data2IMFs(data: Data):
7      emd = EMD()
8      IMFs, residue = emd.emd(data.getData())
9
10     fig, axes = plt.subplots(IMFs.shape[0] + 1, 1)
11     if IMFs.shape[0] == 1:
12         axes = list(axes)
13     axes[0].set_title("The IMFs")
14     for num, IMF in enumerate(IMFs):
15         axes[num].plot(data.getTimeRangeArray(), IMF)
16         axes[num].set_ylabel("IMF " + str(num + 1))
17     axes[IMFs.shape[0]].plot(data.getTimeRangeArray(), residue)
18     axes[IMFs.shape[0]].set_ylabel("Residue")
19
20     plt.show()
21
22     return Data(IMFs, data.getStartTime(), data.getFPS())
```

3.7.2 实现效果

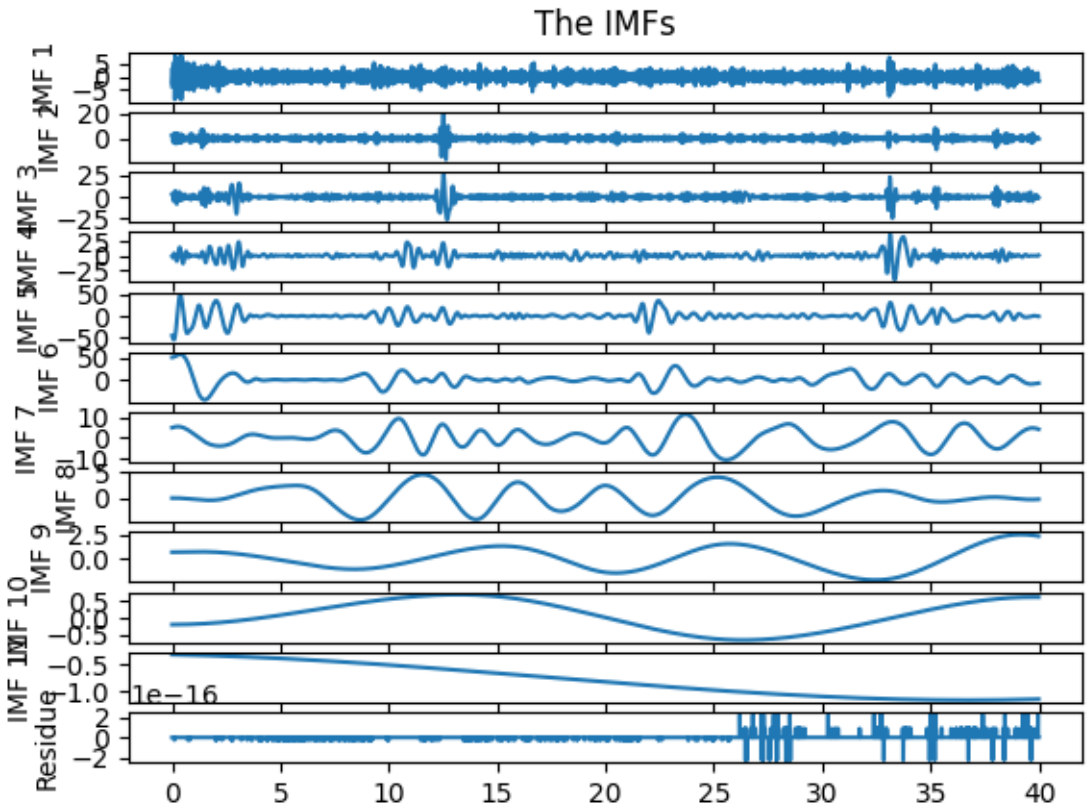

图 3.2: IMFs

3.8 将 IMF 转到频域 IMFs2FrequencyDomain.py

IMFs2FrequencyDomain.py 使用快速傅里叶变换将 IMF 由时域转换到了频域

3.8.1 简介

手动实现 EMD 花费了我很多的时间,在时间不是很充裕的前提下,我调用了 scipy.fftpack 中实现好的 FFT 函数完成了快速傅里叶变换，而没有再手动实现一遍。

```
1 from scipy.fftpack import fft
2 import numpy as np
3 from Visualize import showIMFs
4 from BaseClass import Data
5
6 def IMFs2FrequencyDomain(IMFs: Data):
7     for num, IMF in enumerate(IMFs.getData()):
8         frequency = fft(IMF)
```

```
        magnitude = np.abs(frequency)
        phase = np.angle(frequency)
        IMFs.data[num] = magnitude

    showIMFs(IMFs, "Change IMFs into frequency domain")
    return IMFs
```

3.8.2 实现效果

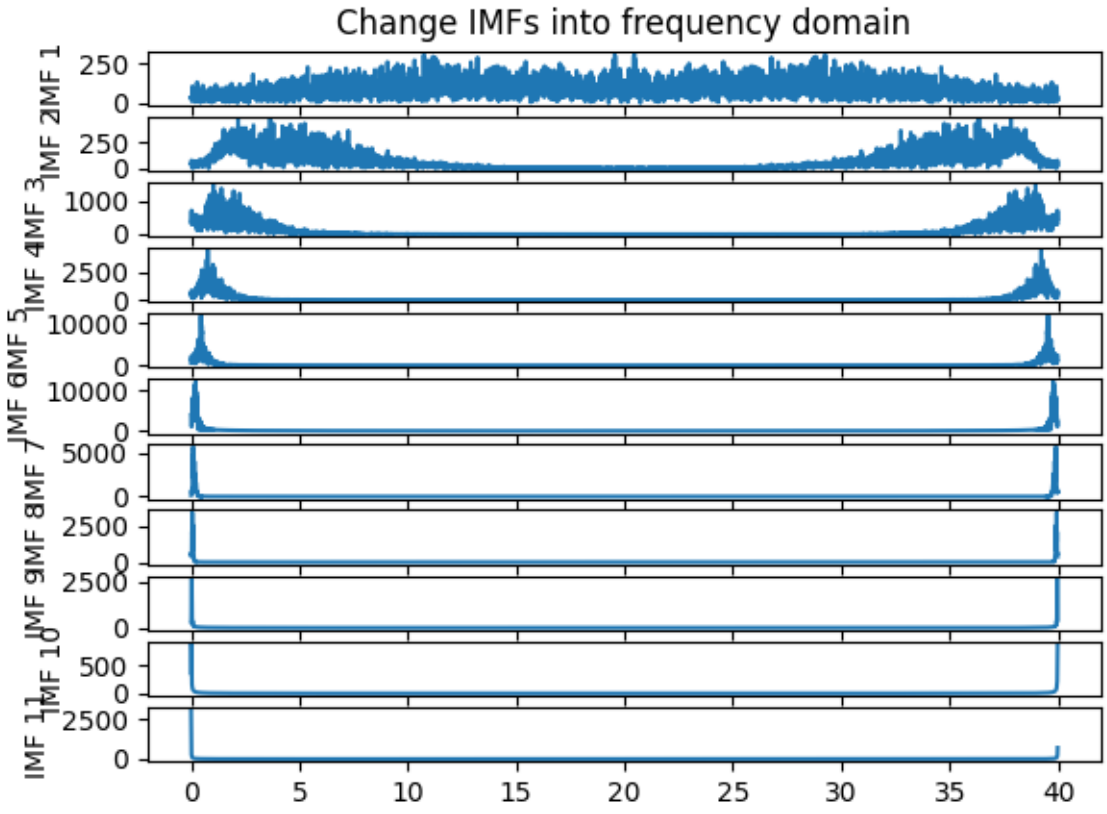

图 3.3: IMF2FrequencyDomain

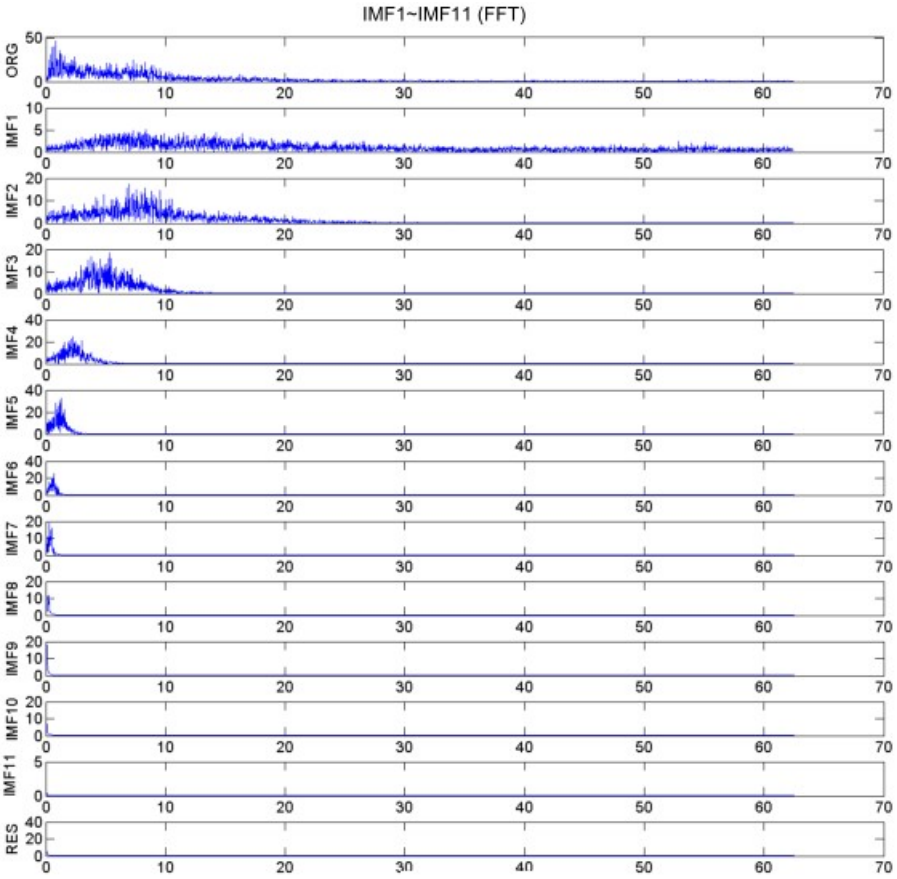

图 3.4: 论文中的结果

3.9 降噪 CutoffNoice.py

在频域中，很容易得到噪声信号和有效信号。CutoffNoice.py 将 IMF 中的噪声剔除。

3.9.1 简介

论文中已经给出了降噪的方式，因此这一步的主要任务就是将频率映射到给定范围内，并将不属于 0.5HZ 到 32HZ 的信号置零。

```
1  from Visualize import showIMFs
2  from BaseClass import Data
3
4  def cutoffNoise(IMFs: Data) -> Data:
5      originalMaxVal = []
6      resizeRate = 35
7      for th, IMF in enumerate(IMFs.data):
8          maxVal = IMF.max()
9          originalMaxVal.append(maxVal)
10         IMFs.data[th] = IMF * resizeRate / maxVal
```

```
    IMFs.data[IMFs.data < 0.5] = 0
    IMFs.data[IMFs.data > 32] = 0
    showIMFs(IMFs, "Cut off noise")
    for th, IMF in enumerate(IMFs.data):
        IMFs.data[th] = IMF * originalMaxVal[th] / resizeRate
    return IMFs
```

3.9.2 实现效果

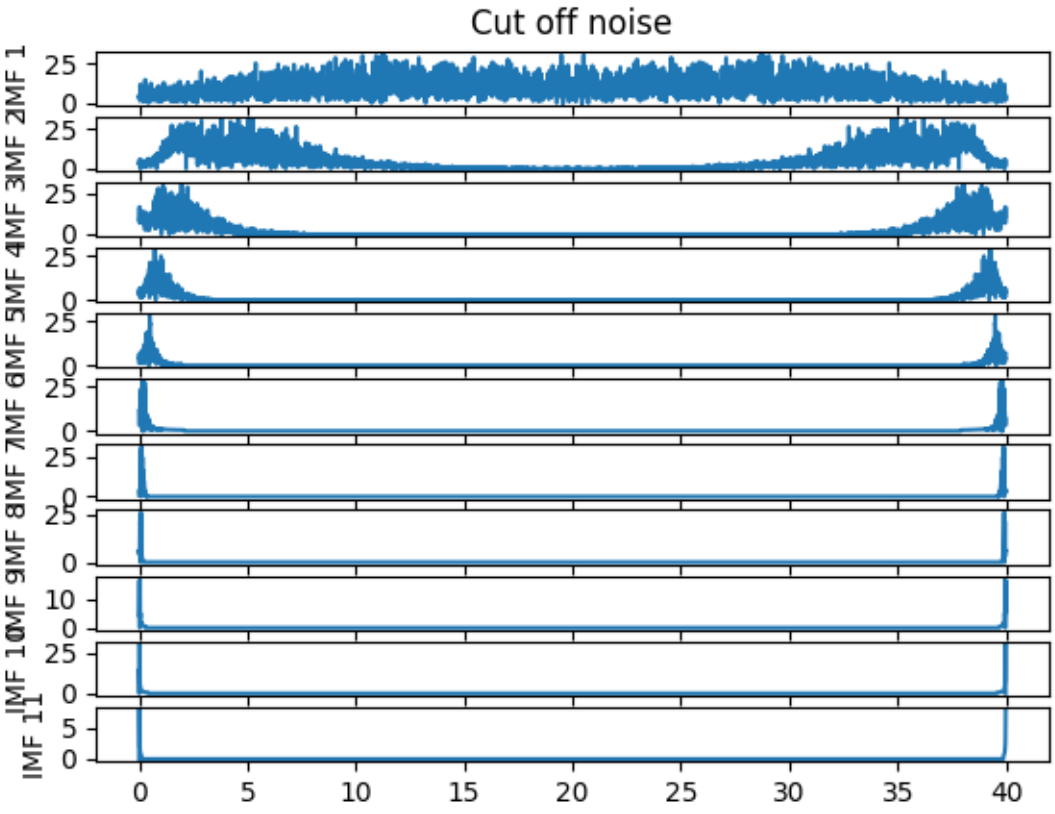

图 3.5: 降噪

3.10 将 IMF 转换回时域 IMF2TimeDomain.py

IMF2TimeDomain.py 将降噪后的 IMF 再次转换回时域。

3.11 简介

将 IMF 转换回时域主要使用快速傅里叶变换逆变换。这里同样使用了 numpy 中的 ifft 库。

可视化显示来自抽象出的显示函数。

```
import numpy as np
from BaseClass import Data
from Visualize import showIMFs

def IMF2TimeDomain(IMFs: Data) -> Data:
    for i in range(len(IMFs.data)):
        IMFs.data[i] = np.fft.ifft(IMFs.data[i])

    showIMFs(IMFs, "Back to time domain")
    return IMFs
```

3.11.1 实现效果

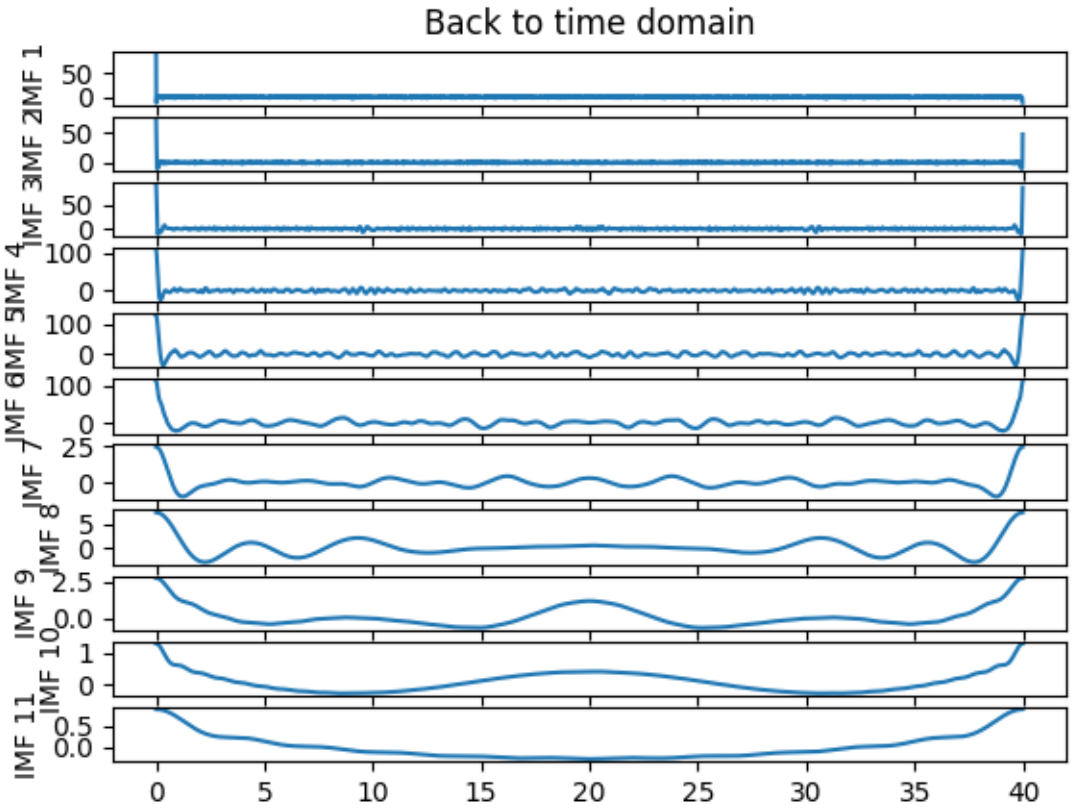

图 3.6: 转换回频域

这里和论文中一样，出现了边缘效应

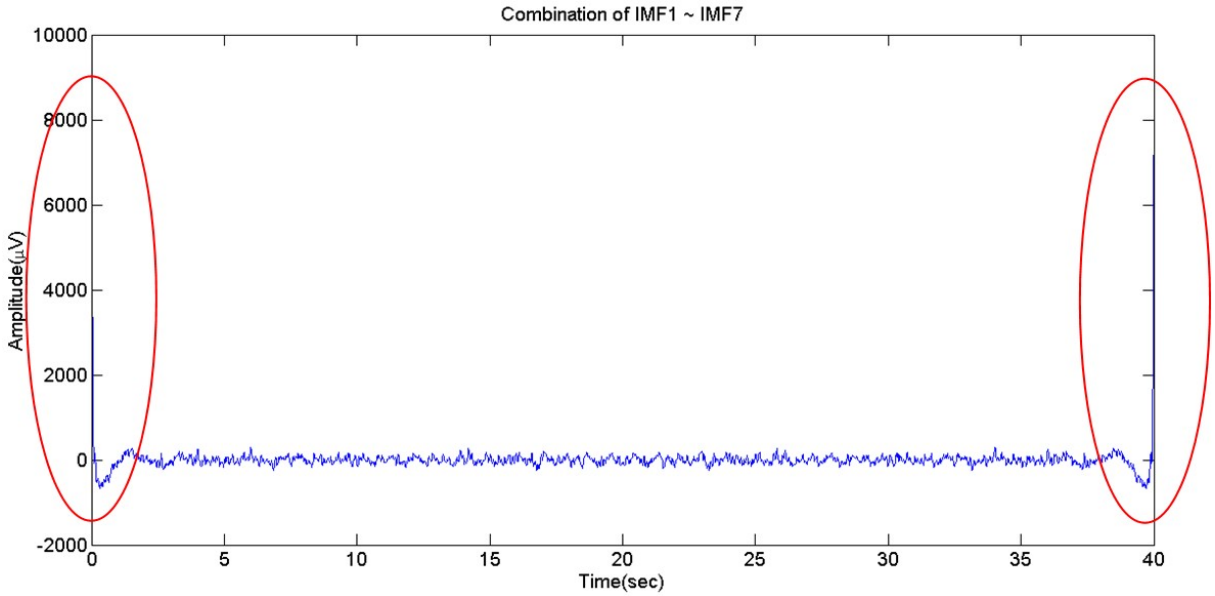

图 3.7: 论文中的边缘效应

3.12 将 IMF 构建回 EEG ConstructEEG.py

ConstructEEG.py 将 IMF 构建回 EEG

3.12.1 简介

将 IMF 构建回 EEG 相比于其他步骤而言就显得更加容易，只需要将 IMF 叠加即可。

在叠加之前，先去除 IFFT 产生的边缘效应，之后将较小的 IMF 丢弃。

```
1  from Visualize import showIMFs, showEEG
2  from BaseClass import Data
3
4  def ConstructEEG(IMFs: Data) -> Data:
5      # 丢掉国小的IMF
6      IMFs.setSubDataByPoints(200, 200)
7
8      showIMFs(IMFs, "Remove edge effects data")
9
10     # 丢弃过小的IMF
11     IMFs.data = IMFs.data[:6]
12     showIMFs(IMFs, "Desert small IMFs")
13
14     EEG = Data(IMFs.data.sum(axis=0), IMFs.getStartTime(), IMFs.getFPS())
15     showEEG(EEG, "Construct EEG")
16     # EEG = EEG[1000:]
17     # ax = plt.subplot()
18     # ax.set_title('Construct EEG')
```

```
19      # ax.plot(np.arange(0, end − start, 0.01), EEG)
20      # plt.show()
21      return EEG
```

3.12.2 实现效果

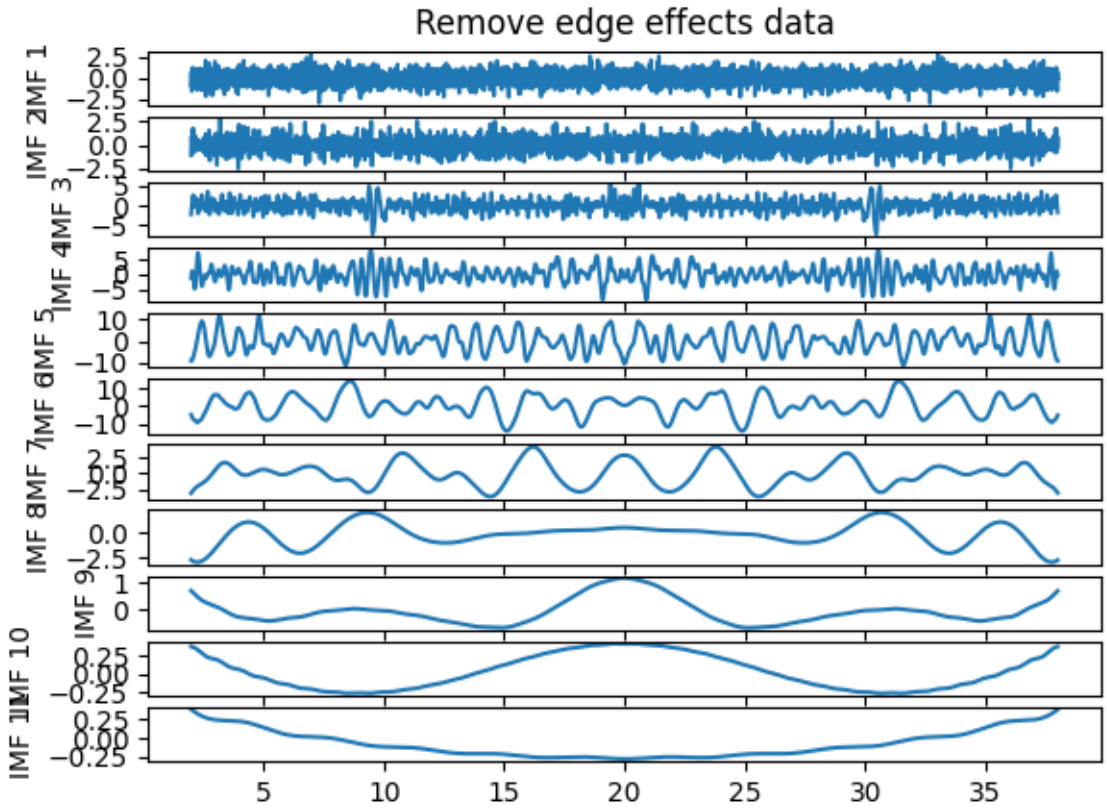

图 3.8: 去除边缘

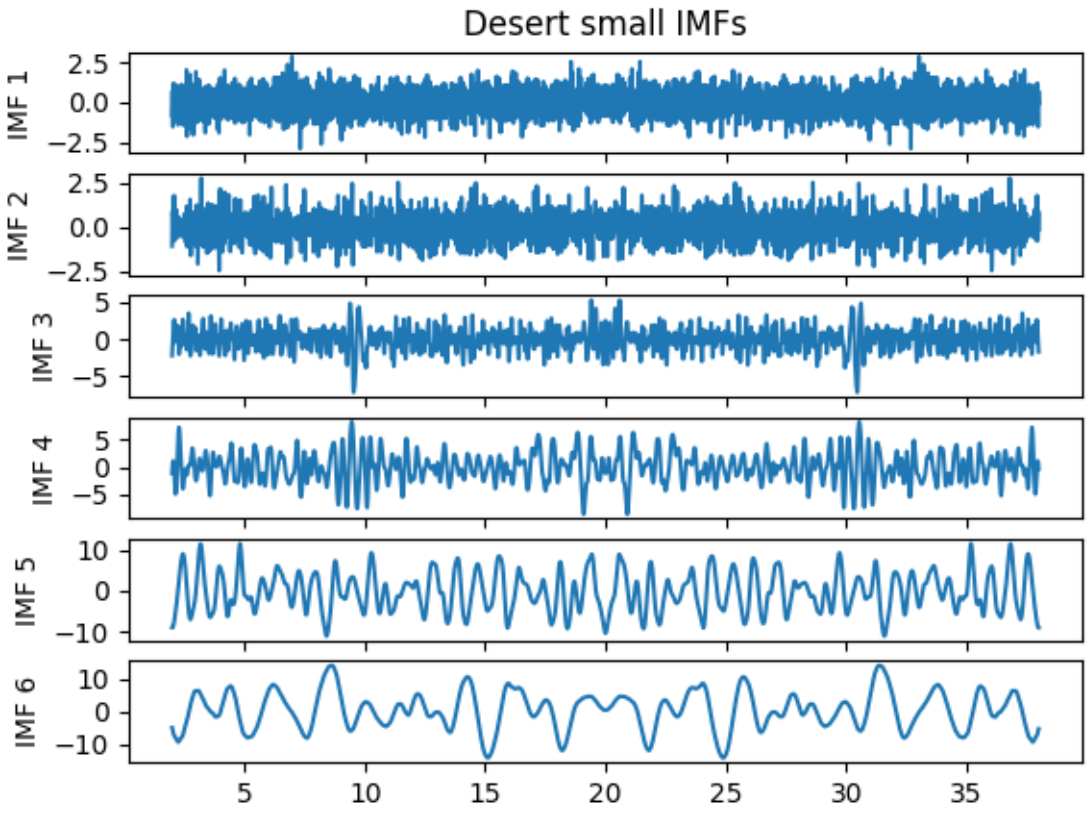

图 3.9: 丢弃较小的 IMF

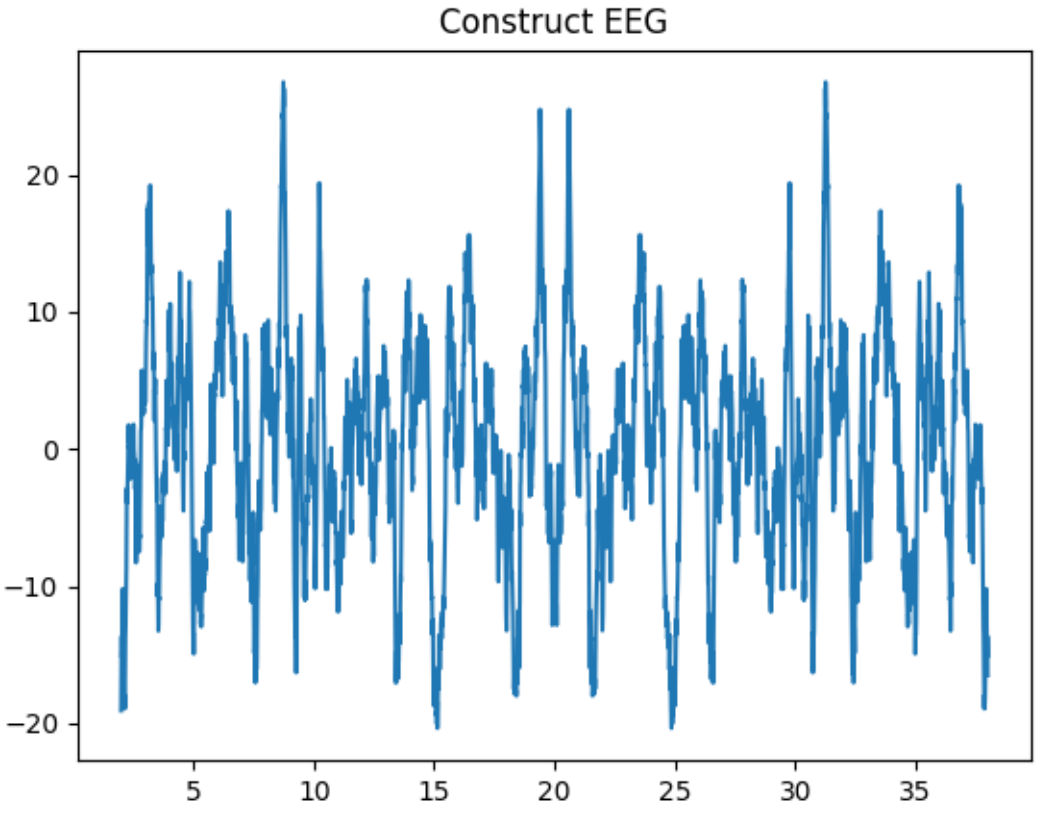

图 3.10: 合并成 EEG

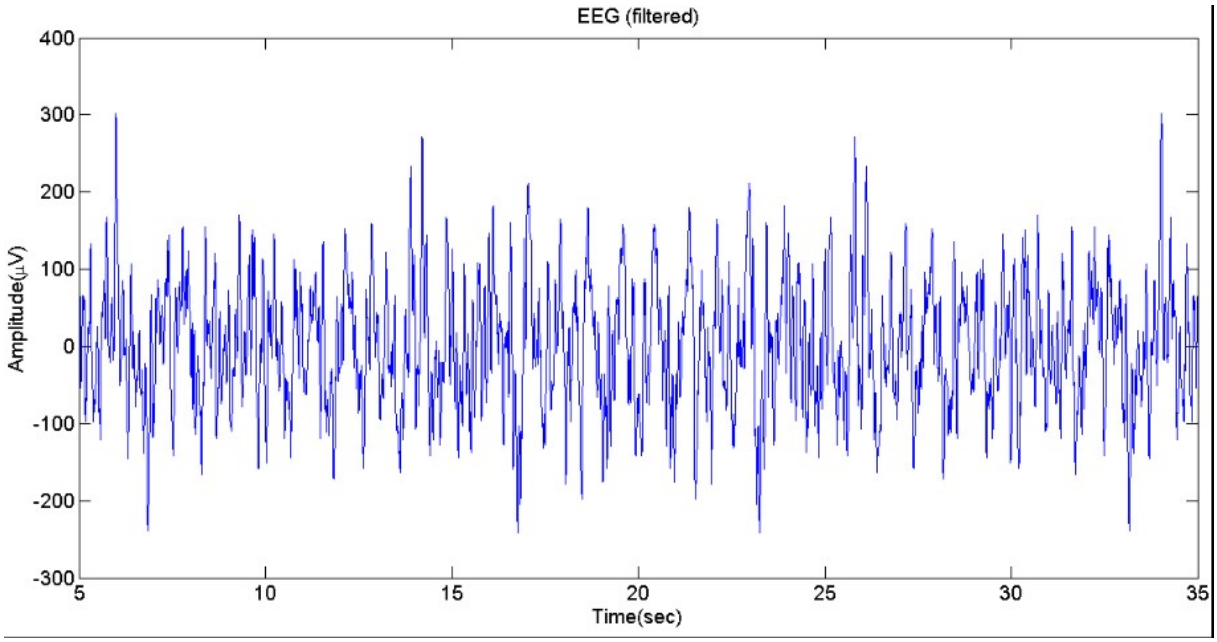

图 3.11: 论文中合并成的 EEG

3.13 希尔伯特-黄变换 HHT.py

HHT.py 通过 HHT 获得实时频率。

3.13.1 简介

希尔伯特-黄变换我也使用了 scipy 中的现成的库，学习自https://blog.csdn.net/rouranling/article/details/123071180

```python
from scipy.signal import hilbert
import numpy as np
from Visualize import showEEG
from BaseClass import Data

def HHT(EEG: Data) -> Data:
    analytic_signal = hilbert(EEG.data)
    amplitude_envelope = np.abs(analytic_signal)
    instantaneous_phase = np.unwrap(np.angle(analytic_signal))
    instantaneousFrequency = (np.diff(instantaneous_phase) / (2.0 * np.pi) * EEG.
        getFPS())
    # 上面instantaneousFrequency.shape = (3599,)
    instantaneousFrequency = np.append(instantaneousFrequency, 0)

    instantaneousFrequency = Data(instantaneousFrequency, EEG.getStartTime(), EEG
        .getFPS())
    showEEG(instantaneousFrequency, "HHT")

    return instantaneousFrequency
```

3.13.2 实现效果

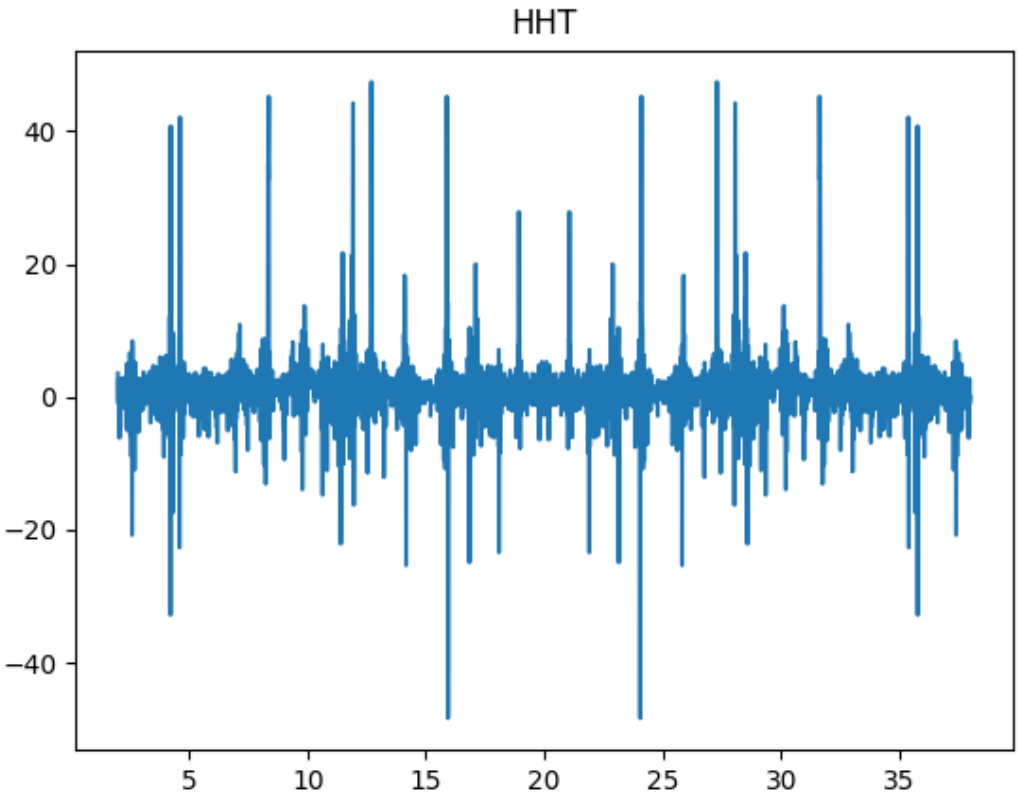

图 3.12: 希尔伯特黄变换

3.14 显示实时三维图 ShowRealtime3D.py

3.14.1 简介

ShowRealtime3D.py 将通过 HHT 变换后的信号显示为实时三维图，并和论文中的结果做对比。

```
1 from Visualize import show3d
2 import numpy as np
3 from BaseClass import Data
4
5 def showRealtime3D(EEG: Data, instantaneousFrequency: Data) -> None:
6     show3d(instantaneousFrequency, EEG)
7     show3d(instantaneousFrequency, EEG, ColorfulAndResize=True)
```

3.14.2 实现效果

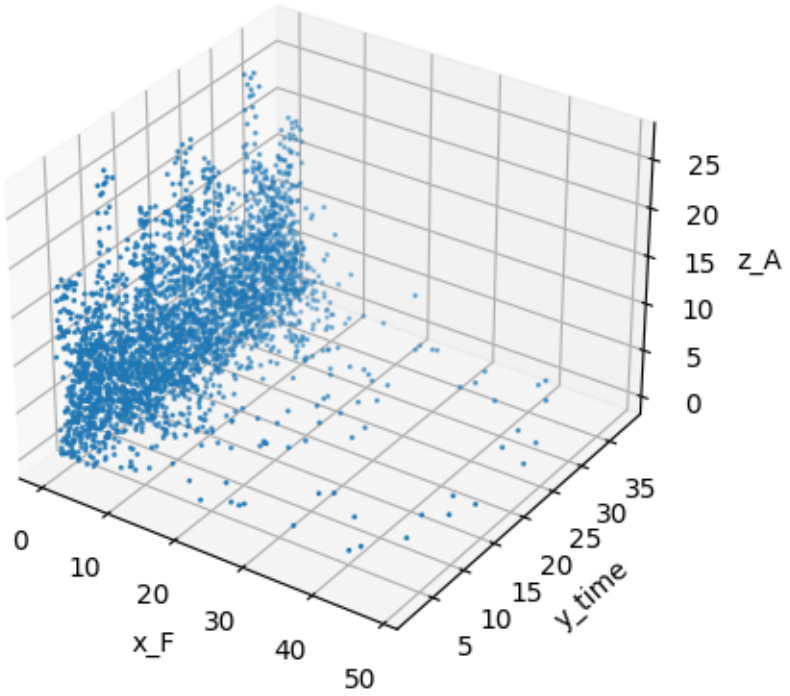

图 3.13: 3D 散点图

刚实现的效果猛的一看，还以为和论文中的不太一样。但是仔细分析后发现，不同之处主要来自两个方面

1. 论文中相位纵坐标很高，因此论文中图像整体较扁

2. 论文中的散点图颜色不同

因此，我将图像压扁并对不同种类的波形加上不同的颜色，发现和论文中的实现效果很像！

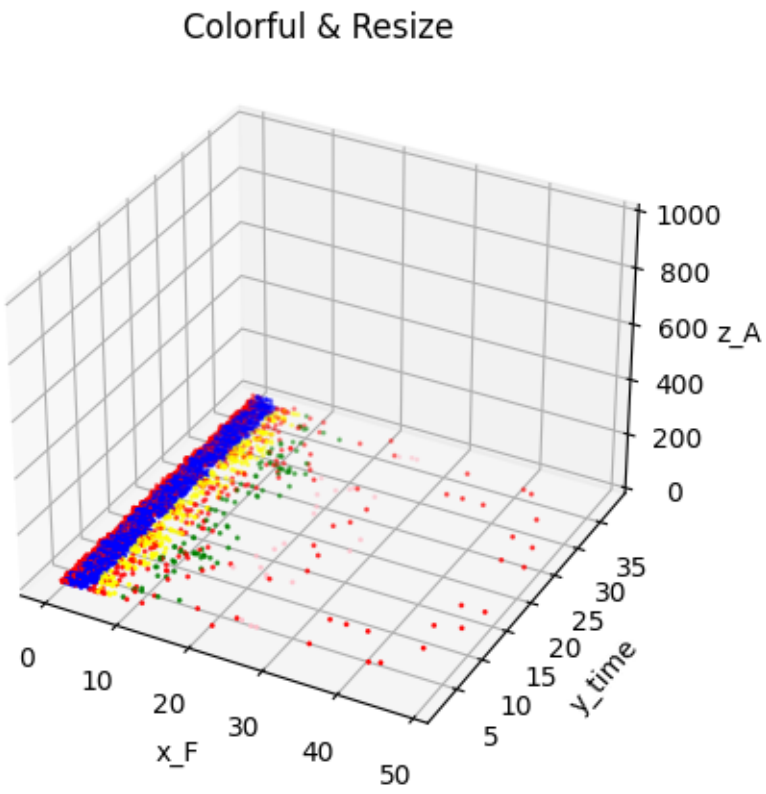

图 3.14: 格式化 3D 散点图

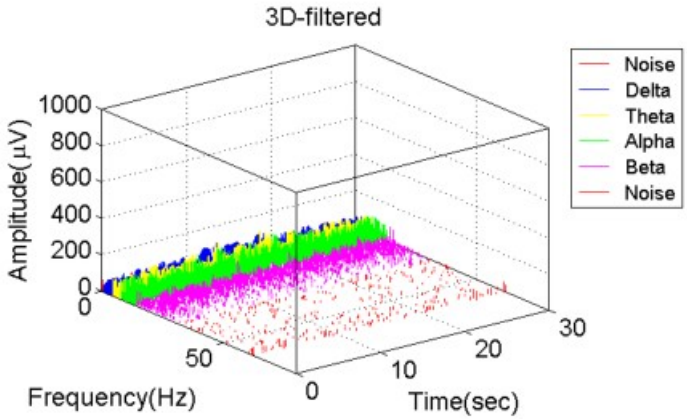

图 3.15: 论文中的最终 3D 散点图

3.15 基本功能类 BaseFunction.py

BaseFunction.py 实现了一些 EEG 信号处理领域所需的功能

3.15.1 简介

这里不得不提一句，本来专门抽象出这个 .py 文件是想着到后面可能会有很多地方再次用到这些较为基础的功能，但是到最后发现几乎只有 用到了这个文件里的函数

此文件中实现了：

1. ChangeToSameType：将两个数据转换为相同的 numpy 的类型
2. formatTimeArray：规范化时间数组，使其不会在微小值上爆炸。返回的数组从 0 开始，最小增量为 1。
3. MyCubicSpline：三次样条曲线，原因是 scipy.interpolate.interp1d 不支持少于 4 个点的三次样条曲线
4. peekDetection：极值检测

3.15.2 ChangeToSameType

原理很简单，调用 numpy 中的 find_common_type 函数，然后把不是这个类型的数据强制转换为这个类型

```
def ChangeToSameType(x: np.ndarray, y: np.ndarray):
    dtype = np.find_common_type([x.dtype, y.dtype], [])
    if x.dtype != dtype:
        x = x.astype(dtype)
    if y.dtype != dtype:
        y = y.astype(dtype)
    return x, y
```

3.15.3 formatTimeArray

规范化时间数组，使其不会在微小值上爆炸。返回的数组从 0 开始，最小增量为 1。

```
def formatTimeArray(t):
    d = np.diff(t)
    return (t - t[0]) / np.min(d)
```

3.15.4 MyCubicSpline

幸好之前数字图像处理课上学过三次样条插值曲线，但是还是参考了很多博客：https://blog.csdn.net/bodybo/article/details/77335129、https://www.pythonheidong.com/blog/article/327241/58759b5470b375eb4f6d

```
def MyCubicSpline(x, y, T):

    x0, x1, x2 = x
    y0, y1, y2 = y

    x1x0, x2x1 = x1 - x0, x2 - x1
    y1y0, y2y1 = y1 - y0, y2 - y1
    _x1x0, _x2x1 = 1.0 / x1x0, 1.0 / x2x1

    m11, m12, m13 = 2 * _x1x0, _x1x0, 0
    m21, m22, m23 = _x1x0, 2.0 * (_x1x0 + _x2x1), _x2x1
    m31, m32, m33 = 0, _x2x1, 2.0 * _x2x1

    v1 = 3 * y1y0 * _x1x0 * _x1x0
    v3 = 3 * y2y1 * _x2x1 * _x2x1
    v2 = v1 + v3

    M = np.array([[m11, m12, m13], [m21, m22, m23], [m31, m32, m33]])
    v = np.array([v1, v2, v3]).T
    k = np.array(np.linalg.inv(M).dot(v))

    a1 = k[0] * x1x0 - y1y0
    b1 = -k[1] * x1x0 + y1y0
    a2 = k[1] * x2x1 - y2y1
    b2 = -k[2] * x2x1 + y2y1

    t = T[np.r_[T >= x0] & np.r_[T <= x2]]
    t1 = (T[np.r_[T >= x0] & np.r_[T < x1]] - x0) / x1x0
    t2 = (T[np.r_[T >= x1] & np.r_[T <= x2]] - x1) / x2x1
    t11, t22 = 1.0 - t1, 1.0 - t2

    q1 = t11 * y0 + t1 * y1 + t1 * t11 * (a1 * t11 + b1 * t1)
    q2 = t22 * y1 + t2 * y2 + t2 * t22 * (a2 * t22 + b2 * t2)
    q = np.append(q1, q2)

    return t, q
```

3.15.5 peekDetection

我不是很懂这样做的意义，但还是跟着参考资料这也做了

```
def peekDetection(T: np.ndarray, S: np.ndarray):
    S1, S2 = S[:-1], S[1:]
    indexEr = np.nonzero(S1 * S2 < 0)[0]
    if np.any(S == 0):
        indexZero = np.nonzero(S == 0)[0]
        if np.any(np.diff(indexZero) == 1):
            zero = S == 0
```

```
        diff = np.diff(np.append(np.append(0, zero), 0))
        one = np.nonzero(diff == 1)[0]
        fuOne = np.nonzero(diff == -1)[0] - 1
        indexZero = np.round((one + fuOne) / 2.0)
    indexEr = np.sort(np.append(indexEr, indexZero))

# 找到局部极值
d = np.diff(S)
d1, d2 = d[:-1], d[1:]
indexMin = np.nonzero(np.r_[d1 * d2 < 0] & np.r_[d1 < 0])[0] + 1
indexMax = np.nonzero(np.r_[d1 * d2 < 0] & np.r_[d1 > 0])[0] + 1

# 如果有多个点具有相同的值
if np.any(d == 0):
    imax, imin = [], []
    bad = d == 0
    dd = np.diff(np.append(np.append(0, bad), 0))
    debs = np.nonzero(dd == 1)[0]
    fins = np.nonzero(dd == -1)[0]
    if debs[0] == 1:
        if len(debs) > 1:
            debs, fins = debs[1:], fins[1:]
        else:
            debs, fins = [], []

    if len(debs) > 0:
        if fins[-1] == len(S) - 1:
            if len(debs) > 1:
                debs, fins = debs[:-1], fins[:-1]
            else:
                debs, fins = [], []

    lc = len(debs)
    if lc > 0:
        for k in range(lc):
            if d[debs[k] - 1] > 0:
                if d[fins[k]] < 0:
                    imax.append(np.round((fins[k] + debs[k]) / 2.0))
            else:
                if d[fins[k]] > 0:
                    imin.append(np.round((fins[k] + debs[k]) / 2.0))

    if len(imax) > 0:
        indexMax = indexMax.tolist()
        for x in imax:
            indexMax.append(int(x))
        indexMax.sort()

```

```
56          if len(imin) > 0:
57              indexMin = indexMin.tolist()
58              for x in imin:
59                  indexMin.append(int(x))
60              indexMin.sort()
61
62      return T[indexMax], S[indexMax], T[indexMin], S[indexMin], indexEr # 局部最
            大值的下标、局部最大值、局部最小值的坐标、局部最小值、index
```

3.16 EEG 信号的 EMD 分解 LetEMD.py

LetEMD.py 实现了 EEG 信号的 EMD 分解

3.16.1 简介

这个代码我写了 200 多行，参考了https://github.com/laszukdawid/PyEMD很多，但是都是手打的,调试了好久。同时也参考了https://blog.csdn.net/The_Time_Runner/article/details/100882454、https://blog.csdn.net/weixin_39781323/article/details/115808594很多地方并不是很懂。

主要包括：

1. EMD 主函数
2. 提取最大和最小样条
3. 镜像操作
4. 三次样条曲线
5. 是否停止分解
6. 是否为 IMF

等功能。

```
1  import numpy as np
2  from scipy.interpolate import interp1d
3  import BaseFunction
4
5
6  class EMD():
7      def ___init___(self):
8          self.nbsym = 2
9          self.dtype = np.float64
10         self.IMFs = None
```

```
        self.residue = None

    def emd(self, S, T=None, max_imf=-1):
        T = np.arange(0, len(S), dtype=S.dtype)
        T = BaseFunction.formatTimeArray(T)
        S, T = BaseFunction.ChangeToSameType(S, T)
        oldIMF = np.nan
        IMFNum = 0
        EXTNum = -1
        self.dtype = S.dtype
        residue = S.astype(self.dtype)
        IMF = np.zeros(len(S), dtype=self.dtype)
        IMF2 = np.empty((IMFNum, len(S)))

        done = False
        while not done:
            residue[:] = S - np.sum(IMF2[:IMFNum], axis=0)
            IMF = residue.copy()
            mean = np.zeros(len(S), dtype=self.dtype)
            n = 0
            while True:
                n += 1
                if n >= 1000: # 设置为最多执行1000次
                    print("数据过大或发生了死循环！")
                    break
                residueEXT = BaseFunction.peekDetection(T, IMF)
                maxLoc, minLoc, indexer = residueEXT[0], residueEXT[2], residueEXT
                    [4]
                EXTNum = len(minLoc) + len(maxLoc)
                nzm = len(indexer)
                if EXTNum > 2:
                    envMax, envMin, eMax, eMin = self.extractMaxAndMinSpline(T,
                        IMF)
                    mean[:] = 0.5 * (envMax + envMin)
                    oldIMF = IMF.copy()
                    IMF[:] = IMF - mean
                    # ----------------
                    residueEXT = BaseFunction.peekDetection(T, IMF)
                    maxLoc, temp, minLoc, temp, indexer_ = residueEXT
                    EXTNum = len(maxLoc) + len(minLoc)
                    nzm = len(indexer_)
                    if oldIMF is np.nan:
                        continue
                    if self.ifIsIMF(IMF, oldIMF, eMax, eMin) and abs(EXTNum - nzm
                        ) < 2:
                        break
                else:
                    done = True
```

```
56                  break
57              IMF2 = np.vstack((IMF2, IMF.copy()))
58              IMFNum += 1
59              if self.timeToStop(S, IMF2) or IMFNum == max_imf:
60                  done = True
61                  break
62          self.IMFs = IMF2.copy()
63          self.residue = S - np.sum(self.IMFs, axis=0)
64          return self.IMFs, self.residue
65
66      def extractMaxAndMinSpline(self, T: np.ndarray, S: np.ndarray):
67          extremaLoc = BaseFunction.peekDetection(T, S)
68          maxLoc, maxVal = extremaLoc[0], extremaLoc[1]
69          minLoc, minVal = extremaLoc[2], extremaLoc[3]
70          if len(maxLoc) + len(minLoc) < 3:
71              return [-1] * 4
72          maxExtrema, minExtrema = self.preparePoints(T, S, maxLoc, maxVal, minLoc
                , minVal)
73          temp, maxSpline = self.cubicSpline(T, maxExtrema)
74          temp, minSpline = self.cubicSpline(T, minExtrema)
75          return maxSpline, minSpline, maxExtrema, minExtrema
76
77      """
78      镜像操作
79      """
80      def preparePoints(self, T, S, maxLoc, maxVal, minLoc, minVal):
81          # 至少需要两个极值
82          maxExtrema = np.zeros((2, len(maxLoc)), dtype=self.dtype)
83          minExtrema = np.zeros((2, len(minLoc)), dtype=self.dtype)
84          maxExtrema[0], minExtrema[0] = maxLoc, minLoc
85          maxExtrema[1], minExtrema[1] = maxVal, minVal
86          nbsym = self.nbsym
87          minEnd, maxEnd = len(minLoc), len(maxLoc)
88          # 左边界
89          locD = maxLoc[0] - minLoc[0]
90          ifLeftExtremaIsMax = locD < 0 # 如果locD < 0，那么leftExtremaMaxType就为True，就说
                明为左极值最大值；否则为最小值
91
92          # 左极值为最大值
93          if ifLeftExtremaIsMax:
94              if (S[0] > minVal[0]) and (np.abs(locD) > (maxLoc[0] - T[0])):
95                  # 第一个极值
96                  expandLeftMaxPos = 2 * maxLoc[0] - maxLoc[1 : nbsym + 1]
97                  expandLeftMinPos = 2 * maxLoc[0] - minLoc[0:nbsym]
98                  expandLeftMaxVal = maxVal[1 : nbsym + 1]
99                  expandLeftMinVal = minVal[0:nbsym]
100             else:
101                 # 起始位置
```

```
102             expandLeftMaxPos = 2 * T[0] - maxLoc[0:nbsym]
103             expandLeftMinPos = 2 * T[0] - np.append(T[0], minLoc[0 : nbsym -
                    1])
104             expandLeftMaxVal = maxVal[0:nbsym]
105             expandLeftMinVal = np.append(S[0], minVal[0 : nbsym - 1])
106         # 左极值为最小值
107         else:
108             if (S[0] < maxVal[0]) and (np.abs(locD) > (minLoc[0] - T[0])):
109                 # 第一个极值
110                 expandLeftMaxPos = 2 * minLoc[0] - maxLoc[0:nbsym]
111                 expandLeftMinPos = 2 * minLoc[0] - minLoc[1 : nbsym + 1]
112                 expandLeftMaxVal = maxVal[0:nbsym]
113                 expandLeftMinVal = minVal[1 : nbsym + 1]
114             else:
115                 # 起始位置
116                 expandLeftMaxPos = 2 * T[0] - np.append(T[0], maxLoc[0 : nbsym -
                        1])
117                 expandLeftMinPos = 2 * T[0] - minLoc[0:nbsym]
118                 expandLeftMaxVal = np.append(S[0], maxVal[0 : nbsym - 1])
119                 expandLeftMinVal = minVal[0:nbsym]
120
121         if not expandLeftMinPos.shape:
122             expandLeftMinPos, expandLeftMinVal = minLoc, minVal
123         if not expandLeftMaxPos.shape:
124             expandLeftMaxPos, expandLeftMaxVal = maxLoc, maxVal
125         expandLeftMin = np.vstack((expandLeftMinPos[::-1], expandLeftMinVal[::-1])
                )
126         expandLeftMax = np.vstack((expandLeftMaxPos[::-1], expandLeftMaxVal
                [::-1]))
127
128         # 右边界
129         locD = maxLoc[-1] - minLoc[-1]
130         rightExtremaMaxYype = locD > 0
131         # 右极值是最大值
132         if not rightExtremaMaxYype:
133             if (S[-1] < maxVal[-1]) and (np.abs(locD) > (T[-1] - minLoc[-1])):
134                 # 最后一个极值
135                 maxIndex = max(0, maxEnd - nbsym)
136                 minIndex = max(0, minEnd - nbsym - 1)
137                 expandRightMaxLoc = 2 * minLoc[-1] - maxLoc[maxIndex:]
138                 expandRightMinPos = 2 * minLoc[-1] - minLoc[minIndex:-1]
139                 expandRightMaxVal = maxVal[maxIndex:]
140                 expandRightMinVal = minVal[minIndex:-1]
141             else:
142                 # 终止位置
143                 maxIndex = max(0, maxEnd - nbsym + 1)
144                 minIndex = max(0, minEnd - nbsym)
145                 expandRightMaxLoc = 2 * T[-1] - np.append(maxLoc[maxIndex:], T
```

```
                    [-1])
146                 expandRightMinPos = 2 * T[-1] - minLoc[minIndex:]
147                 expandRightMaxVal = np.append(maxVal[maxIndex:], S[-1])
148                 expandRightMinVal = minVal[minIndex:]
149             # 右极值是最小值
150             else:
151                 if (S[-1] > minVal[-1]) and len(maxLoc) > 1 and (np.abs(locD) > (T[-1]
                        - maxLoc[-1])):
152                     # 最后一个极值
153                     maxIndex = max(0, maxEnd - nbsym - 1)
154                     minIndex = max(0, minEnd - nbsym)
155                     expandRightMaxLoc = 2 * maxLoc[-1] - maxLoc[maxIndex:-1]
156                     expandRightMinPos = 2 * maxLoc[-1] - minLoc[minIndex:]
157                     expandRightMaxVal = maxVal[maxIndex:-1]
158                     expandRightMinVal = minVal[minIndex:]
159                 else:
160                     # 终止位置
161                     maxIndex = max(0, maxEnd - nbsym)
162                     minIndex = max(0, minEnd - nbsym + 1)
163                     expandRightMaxLoc = 2 * T[-1] - maxLoc[maxIndex:]
164                     expandRightMinPos = 2 * T[-1] - np.append(minLoc[minIndex:], T
                            [-1])
165                     expandRightMaxVal = maxVal[maxIndex:]
166                     expandRightMinVal = np.append(minVal[minIndex:], S[-1])
167
168             if not expandRightMinPos.shape:
169                 expandRightMinPos, expandRightMinVal = minLoc, minVal
170             if not expandRightMaxLoc.shape:
171                 expandRightMaxLoc, expandRightMaxVal = maxLoc, maxVal
172
173             enpandRightMin = np.vstack((expandRightMinPos[::-1], expandRightMinVal
                    [::-1]))
174             expandRightMax = np.vstack((expandRightMaxLoc[::-1], expandRightMaxVal
                    [::-1]))
175             maxExtrema = np.hstack((expandLeftMax, maxExtrema, expandRightMax))
176             minExtrema = np.hstack((expandLeftMin, minExtrema, enpandRightMin))
177             return maxExtrema, minExtrema
178
179         """
180         三次样条
181         """
182         def cubicSpline(self, T: np.ndarray, extrema: np.ndarray):
183             t = T[np.r_[T >= extrema[0, 0]] & np.r_[T <= extrema[0, -1]]]
184             if extrema.shape[1] > 3: # 大于三个点，使用内置库
185                 return t, interp1d(extrema[0], extrema[1], kind="cubic")(t)
186             else: # 否则还得手动实现一波
187                 return BaseFunction.MyCubicSpline(extrema[0], extrema[1], t)
188
```

```
189     """
190     是否停止分解
191     对于剩下的数据，最大值和最小值之差小于0.001 或 绝对值之和小于0.005时，停止分解
192     """
193     def timeToStop(self, S: np.ndarray, IMF: np.ndarray) -> bool:
194         remain = S - np.sum(IMF, axis=0)
195         if np.max(remain) - np.min(remain) < 0.001: # 这里设置为0.001
                #----------------------
196             return True
197         if np.sum(np.abs(remain)) < 0.005: # 这里设置为0.005
                #----------------------
198             return True
199         return False
200
201     """
202     如果连续筛选不影响信号，则信号为IMF
203     """
204     def ifIsIMF(self, newIMF: np.ndarray, oldIMF: np.ndarray, eMax: np.ndarray,
            eMin: np.ndarray) -> bool:
205         if np.any(eMax[1] < 0) or np.any(eMin[1] > 0):
206             return False
207         if np.sum(newIMF ** 2) < 1e-10:
208             return False
209
210         IMFDiff = newIMF - oldIMF
211         PingFang = np.sum(IMFDiff ** 2)
212         svar = PingFang / (max(oldIMF) - min(oldIMF))
213         if svar < 0.001: # svar_thr，定义为0.001
214             return True
215
216         # 标准差检验
217         std = np.sum((IMFDiff / newIMF) ** 2)
218         if std < 0.2: # std_thr，定义为0.2
219             return True
220         energyRatio = PingFang / np.sum(oldIMF * oldIMF)
221         if energyRatio < 0.2: # energy_ratio_thr，定义为0.2
222             return True
223         return False
```

第 4 章　未来展望

1. 当前程序不支持命令行传参，也就是说默认采样频率是 100，并且只能读取名为“case7.txt”的数据文件。不过这不是大问题。

2. 在去除“很小”的 IMF 时，论文中并未提出一种有效的由程序自动识别“什么是很小的 IMF”的方法。未来可以在程序中添加这个功能，比如“最大值小于第一

个 IMF 的 1% 就视为小 IMF”、“极差不超过多少就视为小 IMF”等。

3. 我所实现的 EMD 分解不支持参数设置，如：分解为多少 IMF、分解的终止条件等。

4. IFFT 带来的边缘效应会使 IMF 两端的振幅特别地高。当前（论文中也类似）采用的办法是去除前 200 个采样点和后 200 个采样点。未来可以研究并使用一种更为“智能”的方法，让程序计算出前后应该分别去除多少个采样点以避免边缘效应。

参考文献

[1] *Mu-Tzu Shih, Faiyaz Doctor, Shou-Zen Fan, Kuo-Kuang Jen and Jiann-Shing Shieh. Instantaneous 3D EEG Signal Analysis Based on Empirical Mode Decomposition and the Hilbert-Huang Transform Applied to Depth of Anaesthesia*, 2022.

[2] *爱码网：matplotlib 消除 QT 警告*. https://www.likecs.com/ask-924744.html, 2022.

[3] *CSDN:matplotlib 显示三维散点图*. https://blog.csdn.net/weixin_46287157/article/details/12478
2022.

[4] *脚本之家：不同频率的数据显示不同的颜色*. https://www.jb51.net/article/258747.htm, 2022.

[5] *figshare:深度麻醉数据集下载*. https://figshare.com/articles/dataset/EEG_and_BIS_raw_data/
2022.

[6] *CSDN: Matlab 保存数据到文件*. https://blog.csdn.net/whb12345678feng/article/details/10310811
2022.

[7] *CSDN: Hilbert 变换提取信号特征*. https://blog.csdn.net/rouranling/article/details/123071180,
2022.

[8] *CSDN: Cubic spline(三次样条插值)*. https://blog.csdn.net/bodybo/article/details/77335129,
2022.

[9] *Python 黑洞网：三次样条曲线*. https://www.pythonheidong.com/blog/article/327241/58759b547
2022.

[10] *Github: PyEMD*. https://github.com/laszukdawid/PyEMD, 2022.

[11] *CSDN: scipy.interpolate.interp1d() 函数详解*. https://blog.csdn.net/The_Time_Runner/article/d 2022.

[12] *CSDN: 如何检测一个信号的变换趋势是上升还是下降*. https://blog.csdn.net/weixin_39781323/article/details/115808594, 2022.