北京理工大學

信号与信息处理课程设计实验报告

信号产生、分析与处理软件系统设计

学	院：	信息与电子学院
专	业：	电子信息工程（徐特立英才班）
班	级：	[redacted]
姓	名：	[redacted]
学	号：	[redacted]

目录

实验五 脑电 SSVEP 信号的目标分类

一、 实验题目

1. 基本知识

稳态视觉诱发电位(steady-state visual evoked potentials, SSVEP)是脑机接口系统经常使用的一种信号，当人眼受到固定频率超过 4Hz 的视觉刺激时，大脑皮质活动将被调节，诱发出类似于视觉刺激相同频率的周期性节律信号，这就是稳态视觉诱发电位（Steady-State Visual Evoked Potential, SSVEP）。基于 SSVEP 的脑-机接口系统，是通过识别大脑中的脑电信号的频率成分来检测目标指令。

2. 实验数据

脑电数据采集是通过观看 10 个不同频率刺激块诱发的脑电信号，对应目标 1-10，如图 3 所示。目标与刺激频率信息如下。

表 1：10 个目标对应的刺激频率

类别 1	类别 2	类别 3	类别 4	类别 5	类别 6	类别 7	类别 8	类别 9	类别 10
8Hz	8.5Hz	9Hz	9.5Hz	10Hz	10.5Hz	11Hz	11.5Hz	12Hz	12.5Hz

每类刺激诱发的脑电信号中包含了基频和倍频成分，例如类别 1 信号，信号所含频率为 8Hz、16Hz、32Hz，倍频成分的能量可能比基频的能量大，有时倍频比基频能量小，如图 1 所示，16Hz 幅度小于 8Hz。

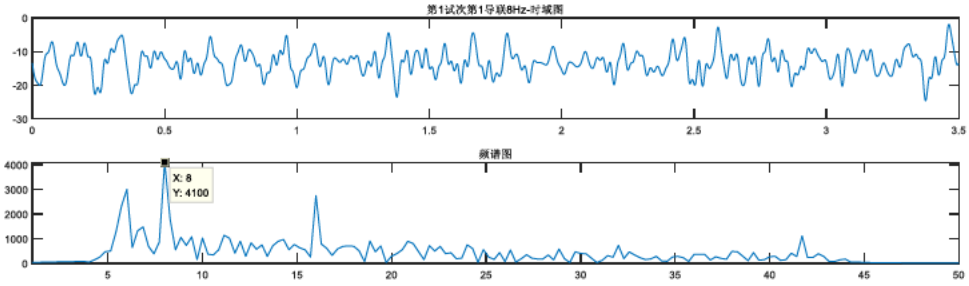


图 1：脑电信号时域波形及频域频谱

数据格式：

- eegmat.mat 文件中包含 x 变量，数据维度信息为 8*3500*20；
- 8 对应脑电信号的导联数；
- 3500 对应 3.5s 的时域采样点，采样频率 1000Hz；
- label.mat 为标签信息，其中 20 值对应 20 个目标试次。

数据加载：

- Load(eeg.mat);
- Load(label.mat);

3. 目标分类

(1) 对每一通道脑电信号进行带通滤波，滤波范围 5-40Hz。

(2) 利用 FFT 或功率谱 periodogram 对一个试次下每个通道的脑电信号进行频谱分析，查看 7-15Hz 范围内最高峰值是多少，并与所给刺激频率比对，8 个通道投票最多的目标即为该试次所分类出来的目标结果。另外，可加入倍频频率检测，提高目标识别准确率。

(3) 对 20 个试次分别进行目标分类，根据频谱信息判断目标类别并与 其真实标签 label 进行比较，计算准确率。

(4) GUI 界面呈现滤波器频响；20 个试次每一个通道下的频谱图，并标出峰值频率；呈现 20 个试次中每个导联目标识别结果，8 导联联合目标识别结果,以及导联和倍频联合的目标识别结果。最终给出 20 个试次识别的平均准确率。

二、 实验原理与方法

稳态视觉诱发电位(SSVEP)是当人眼接收固定频率(≥4Hz)视觉刺激时，大脑皮质活动被调节产生的周期性节律信号，其包含与刺激频率一致的基频成分以及 2 倍频、3 倍频等倍频成分。本实验首先对脑电信号进行 5-40Hz 的带通滤波处理，然后对处理后的信号进行 FFT 频谱分析。通过提取脑电信号中的基频和倍频特征，与 10 个类别的刺激频率进行比对，最后结合多通道投票机制实现 10 类目标的分类。

1. 滤波器设计

1.1 基本原理

要实现 5-40Hz 的带通滤波，首先要设计一个对应截止频率的带通滤波器。为了提高滤波器设计的灵活性，并限制滤波器的阶数不要太高，本实验采取低通滤波器和高通滤波器组合的方式设计带通滤波器。为设计方便起见，并保证在通带范围内滤波器的频率响应尽可能平坦，本实验将采用巴特沃斯滤波器进行 IIR 滤波器设计。

N 阶巴特沃斯低通滤波器的幅度平方响应为

$$|H_a(j\omega)|^2 = \frac{1}{1+\left(\frac{\omega}{\omega_c}\right)^{2N}} \tag{1}$$

其中，N为滤波器阶数，ω_c是 3dB 截止频率（单位：rad/s）。设定滤波器的截止频率、过渡带宽、通带波纹和阻带衰减等参数，根据公式(1)可计算出模拟巴特沃斯滤波器的最小阶数 N 和 3dB 截止频率 ω_c，据此可求出滤波器的系统函数$H_a(s)$。

对于 IIR 滤波器，可采用双线性变换法进行设计。根据实验要求可设定低通滤波器的通带截止频率ω_s为 40Hz，过渡带宽为 5Hz（即阻带截止频率ω_p为 45Hz）；高通滤

波器的通带截止频率ω_s为 5Hz，过渡带宽为 1Hz（即阻带截止频率ω_p为 4Hz），但上述的截止频率不能直接用于巴特沃斯模拟滤波器的设计。首先将实验中设定的模拟截止频率ω_p、ω_s转换为数字频率Ω_p、Ω_s，然后按照以下公式进行频率预畸：

$$\omega = \frac{2}{T} tan\left(\frac{\Omega}{2}\right) \tag{2}$$

由此得到模拟原型滤波器的截止频率ω_p'、ω_s'。根据预畸后的截止频率ω_p'和ω_s'、通带波纹和阻带衰减等参数按照上述的方法设计巴特沃斯模拟滤波器$H_a(s)$，再根据双线性变换法的映射关系设计数字滤波器$H(z)$，即为

$$H(z) = H_a\left(\frac{2}{T}\frac{1-z^{-1}}{1+z^{-1}}\right) \tag{3}$$

按照双线性变换法可分别设计低通滤波器$H_1(z)$和$H_2(z)$，则组合后的带通滤波器的系统函数为：

$$H(z) = H_1(z)H_2(z) \tag{4}$$

滤波器设计完成后，需要对待处理的脑电信号进行滤波处理，消除低频噪声和高频噪声对目标分类的影响。设输入的单试次单通道脑电信号为$x[n]$，滤波处理后的信号为输入信号与滤波器冲激响应的线性卷积：

$$y[n] = x[n] * h[n] \tag{5}$$

因此有

$$y[n] = -\sum_{k=1}^{M} a_k y[n-k] + \sum_{k=0}^{N} b_k x[n-k] \tag{6}$$

其中a_k和b_k分别为滤波器系统函数$H(z)$分母和分子的系数，$y[n]$即为滤波处理后的脑电信号。

1.2 关键 MATLAB 函数

- buttord：根据滤波器指标，计算巴特沃斯数字滤波器的最优阶数 N 和截止频率 Wn；
- butter：计算 IIR 巴特沃斯数字滤波器的分子、分母系数；
- freqz：计算滤波器频率响应；
- filtfilt：零相位滤波，应用滤波器系数对信号进行滤波处理，避免相位失真；

2. FFT 频谱分析

2.1. 基本原理

为了通过频率成分对脑电信号进行目标分类，需要对其进行 FFT 频谱分析。首先按照采样频率 1kHz 对单试次单通道的脑电信号进行采样，得到采样序列$x[n]$，对其进

行带通滤波处理之后再做 FFT 频谱分析。为抑制频谱泄漏造成的影响，提高分类的准确率，可以对采样信号进行加窗处理（本实验支持矩形窗、汉宁窗、汉明窗和布莱克曼窗）。对加窗后的信号$x[n]$进行 FFT，即为

$$X[k] = FFT(x[n]) \tag{7}$$

其中$X[k]$为采样信号的 L 点 FFT，为方便计算，通常取 FFT 点数 L 为大于信号采样点数的最小一个 2 的幂次。由于 FFT 仅是 DFT 的快速计算方法，根据 DFT 和 FT 之间的关系，当 k 取值在$0\sim\frac{L}{2}$时对应的模拟频率为$0\sim\frac{f_s}{2}$，则 FFT 序列第 k 个值对应的模拟频率应当为$\frac{kf_s}{L}$。记$X(j\omega)$为对应的连续频谱，则存在下述对应关系：

$$X\left(j\frac{kf_s}{L}\right) = \frac{X[k]}{f_s} \tag{8}$$

对于单边谱密度，还需对除 0 和奈奎斯特频率$\frac{f_s}{2}$的频谱幅度值均乘以 2 进行调整。

在进行 FFT 频谱分析之后，绘制信号的频谱图，并按照题目要求采用基频检测或倍频检测对脑电信号进行分类。若采用基频检测，在基频区间（7-15Hz）提取脑电信号频谱的最高峰值，通过比较峰值频率与 10 类目标对应的刺激频率的距离，取最接近的刺激频率对应目标为该试次该通道脑电信号所分类别。倍频检测的方法与基频检测类似，在倍频区间（15-30Hz）提取脑电信号频谱的最高峰值，若峰值大于信号频谱最大值的 0.1 倍即认为是有效峰值。如果峰值频率存在，将峰值频率除以 2 后再将其与 10 类目标对应的刺激频率的距离进行比较，取最接近的刺激频率对应目标为该试次该通道脑电信号所分类别。注意实验中将不再考虑三倍频及以上的倍频检测，因为此时的频率成分已经很微弱，难以反映脑电信号的类别特征。这就是单试次单通道的脑电信号分类方法。

2.2. 关键 MATLAB 函数

- fft：快速傅里叶变换，将时域信号转换为频域；
- rectwin/hann/hamming/blackman：生成各类窗函数；
- findpeaks：在指定频率区间内检测峰值，提取主峰值作为特征频率

3. 目标分类方法

为了提高目标分类的准确率，每个试次的脑电信号均采取多通道投票的方法，实验中探究了基频投票分类和基倍频联合投票分类两种方法。

若采用基频投票分类，对于待分类的试次，每个通道按照 2.1 中的方法进行基频分类，并将 8 个通道的基频分类结果汇总在一起进行投票（共 8 票），得票最多的目标即

为该试次脑电信号所分的类别。

若采用基倍频联合投票分类，对于待分类的试次，每个通道按照 2.1 中的方法分别进行基频分类和倍频分类，并将 8 个通道的基频分类和倍频分类的结果汇总在一起进行投票（共 16 票），得票最多的目标即为该试次脑电信号所分的类别。

在两种投票分类方式中，若出现不同类别的平票现象，一律取较低刺激频率对应目标为该试次脑电信号所分的类别。

三、 实验结果与分析

1. 软件系统界面设计

本系统基于 MATLAB App Designer 开发，采用标签页式模块化布局，通过标签页的切换可分别选择“滤波器设计”、“频谱分析”和“目标分类”这三大功能模块。软件系统界面简洁直观、功能分区明确，操作流程清晰，可以完成脑电 SSVEP 信号目标分类的基本功能。下面对软件系统界面进行简要介绍：

1.1. 标签页 1：滤波器设计

标签页 1 实现滤波器的设计并绘制滤波器频率响应的功能，布局主要分为顶部数据加载区、上侧参数控制区和下侧绘图展示区三大区域，如图 2 所示：

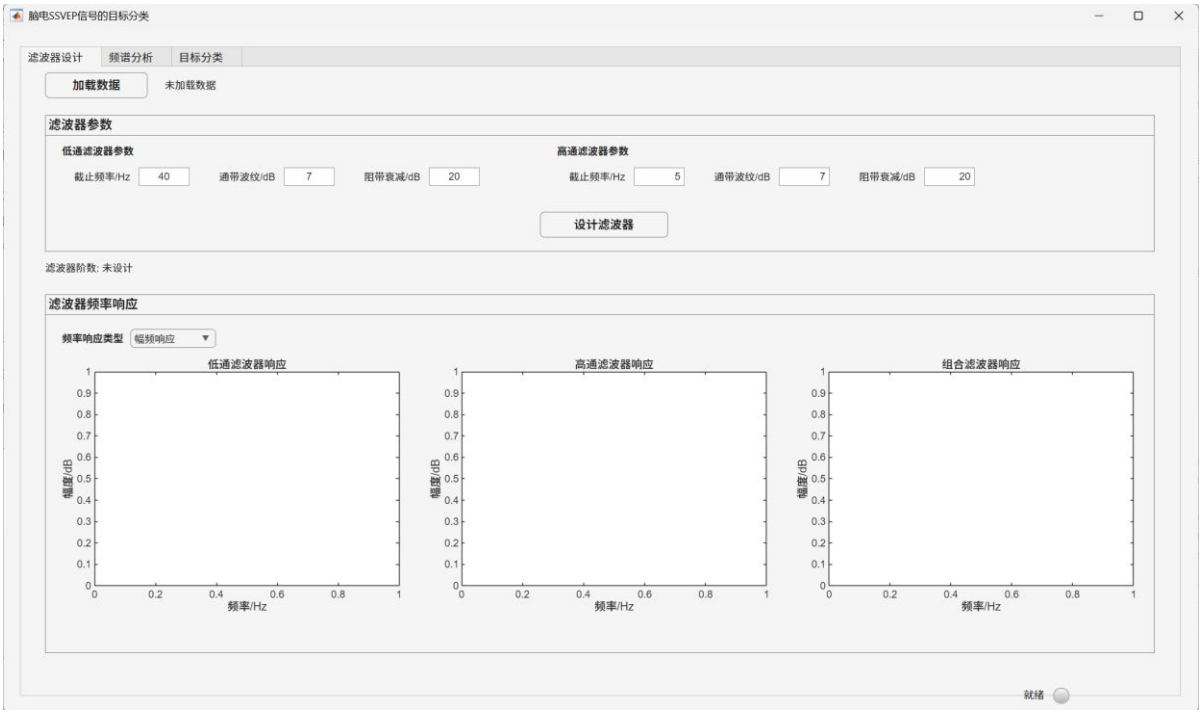


图 2：标签页 1 初始化界面

上图所示界面为直接打开系统，未经任何操作的初始化界面。

1.1.1. 顶部数据加载区

标签页 1 的顶部包含“加载数据”的按钮，点击该按钮即可选择文件夹加载其中

的 eeg.mat（脑电数据）和 label.mat（标签数据），数据加载成功后旁边的标签会显示数据维度信息，若加载失败则会弹窗提示错误原因，只有在加载数据后才能进行后续的频谱分析和目标分类操作。

1.1.2. 上侧参数控制区

标签页 1 的上侧为滤波器参数面板。滤波器参数面板包含所设计的高通滤波器和低通滤波器的截止频率、通带波纹和阻带衰减等指标，默认输入参数如上图所示，实验中保持默认参数即可。完成滤波器参数的输入后，点击“设计滤波器”按钮即可按照输入的指标进行滤波器的设计，并在下侧绘图展示区绘制滤波器的频率响应。

1.1.3. 下侧绘图展示区

标签页 1 的下侧为滤波器频率响应面板，面板中包含一个“频率响应”下拉框和三个水平排列的坐标区，这三个坐标区分别绘制所设计的低通滤波器、高通滤波器和组合的带通滤波器的频率响应。通过切换“频率响应类型”下拉框，可选择绘制所设计滤波器的幅频响应和相频相应。

1.2. 标签页 2：频谱分析

标签页 2 实现单试次单通道滤波处理后的脑电信号的时域波形绘制和频谱分析，寻找频谱峰值进行基频检测或倍频检测，布局主要分为顶部参数控制区、左侧绘图展示区和右侧信息展示区三大区域，如图 3 所示：

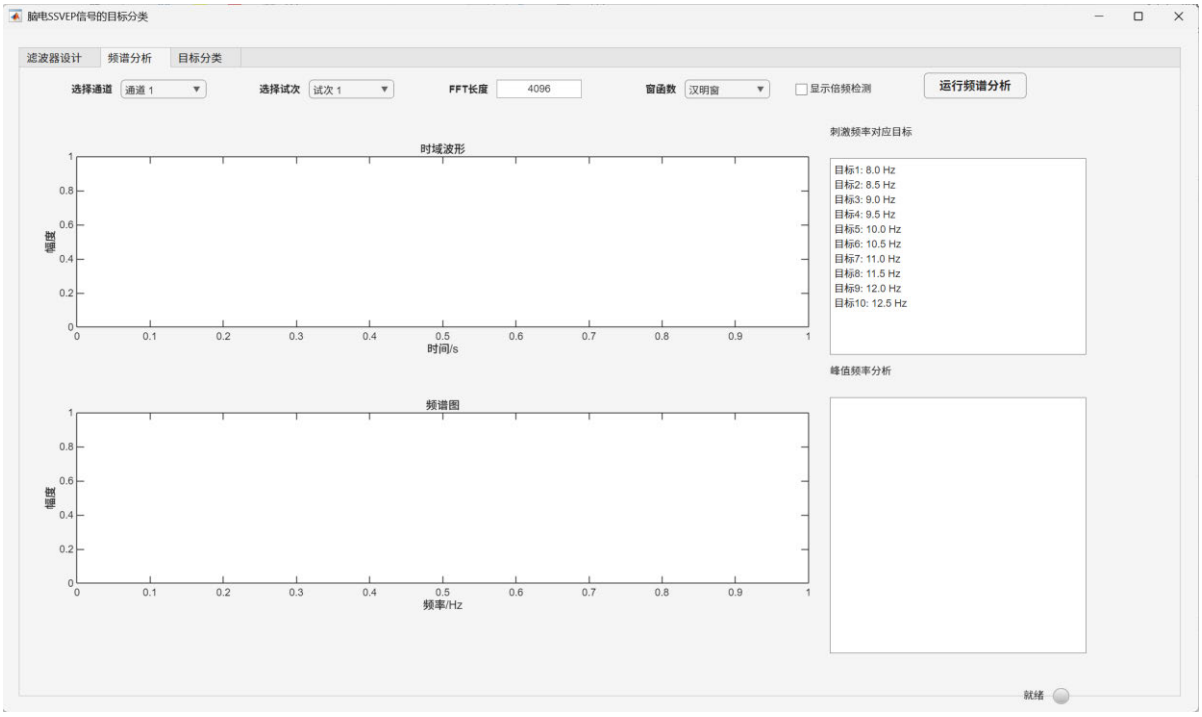


图 3：标签页 2 初始化界面

上图所示界面为直接打开系统，未经任何操作的初始化界面。

1.2.1. 顶部参数控制区

标签页 2 的顶部包含“选择通道”和“选择试次”下拉框，用于选择进行分析的脑电信号的试次和通道。FFT 点数可以直接以数值的形式输入，默认点数为 4096。为减轻频谱泄漏在 FFT 之前应进行加窗处理，窗函数类型可通过“窗函数”下拉框选择，默认选择汉明窗。注意，系统在默认情况下只进行基频检测，如果要加入倍频检测需要选中“显示倍频检测”复选框。在完成上述所有参数设置后，点击“运行频谱分析”按钮即可对选中的试次通道的脑电信用于绘制号进行频谱分析和类别检测。

1.2.2. 左侧绘图展示区

标签页 2 的左侧包含两个垂直排列的坐标区，上侧坐标区用于绘制滤波处理后脑电信号的时域波形，下侧坐标区用于绘制滤波处理后脑电信号的幅度谱，并标注基频区间、倍频区间以及区间内的峰值。

1.2.3. 右侧信息展示区

标签页 2 的右侧包含两个垂直排列的文本区，上侧文本区固定显示 10 类目标对应的刺激频率，方便用户对照观察。下侧文本区显示基频分析和倍频分析的峰值频率、对应目标类别和频率差等内容。

1.3. 标签页 3：目标分类

对所有 20 个试次分别进行基频投票分类和基倍频联合投票分类，展示详细分类结果和准确率统计，布局主要分为左侧分类结果表格和右侧准确率统计区，如下图所示：

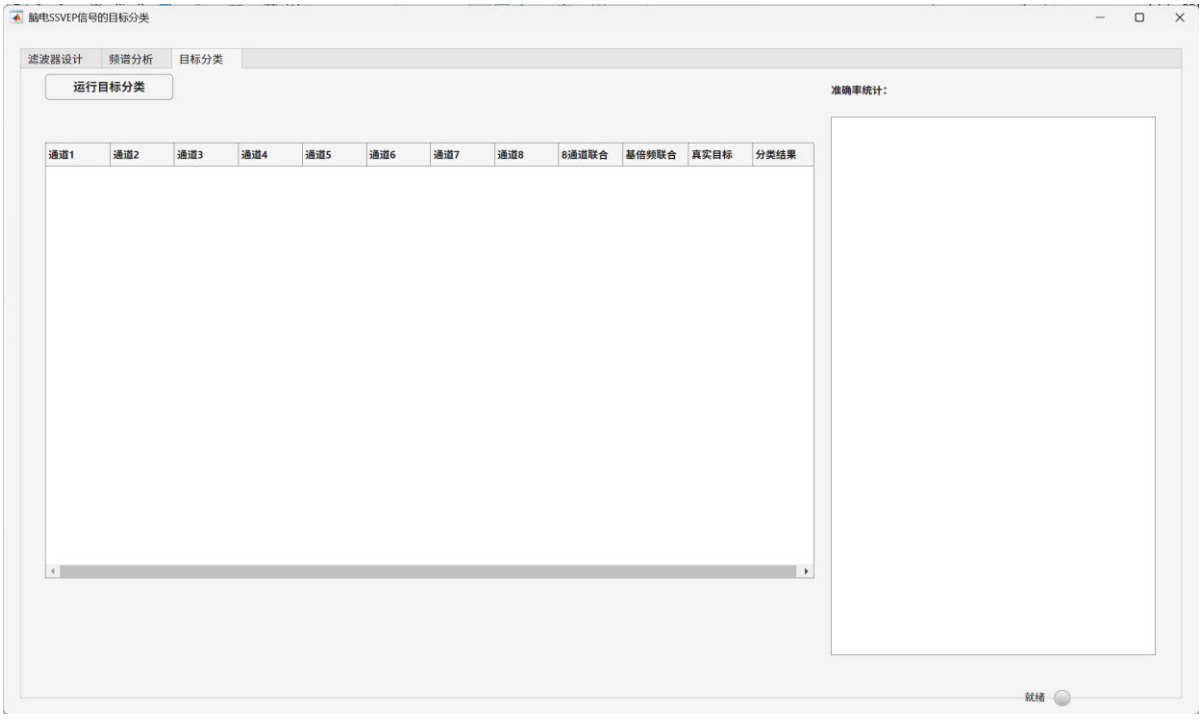


图 4：标签页 3 初始化界面

上图所示界面为直接打开系统，未经任何操作的初始化界面。

1.3.1. 左侧分类结果表格

标签页 3 的左侧是一个 20×20 的表格，其中 20 行对应 20 个试次，20 列分别对应 8 通道基频预测结果、8 通道倍频预测结果、基频投票分类结果、基倍频联合投票分类结果、真实目标类别、投票分类正确性判断。

1.3.2. 右侧准确率统计区

标签页 3 的右侧是一个文本区，分模块展示准确率统计信息，包含基频检测、倍频检测每个通道的准确率以及总体准确率、基频投票分类准确率、基倍频联合投票准确率以及性能分析等内容。

注意，只有当点击最上方的“运行目标分类”按钮才会进行目标分类，并在表格和文本区中显示对应的分类信息。

2. 滤波器设计

2.1. 设计结果

按照默认参数进行滤波器设计，滤波器的幅频响应和相频响应分别如图 5、图 6 所示：

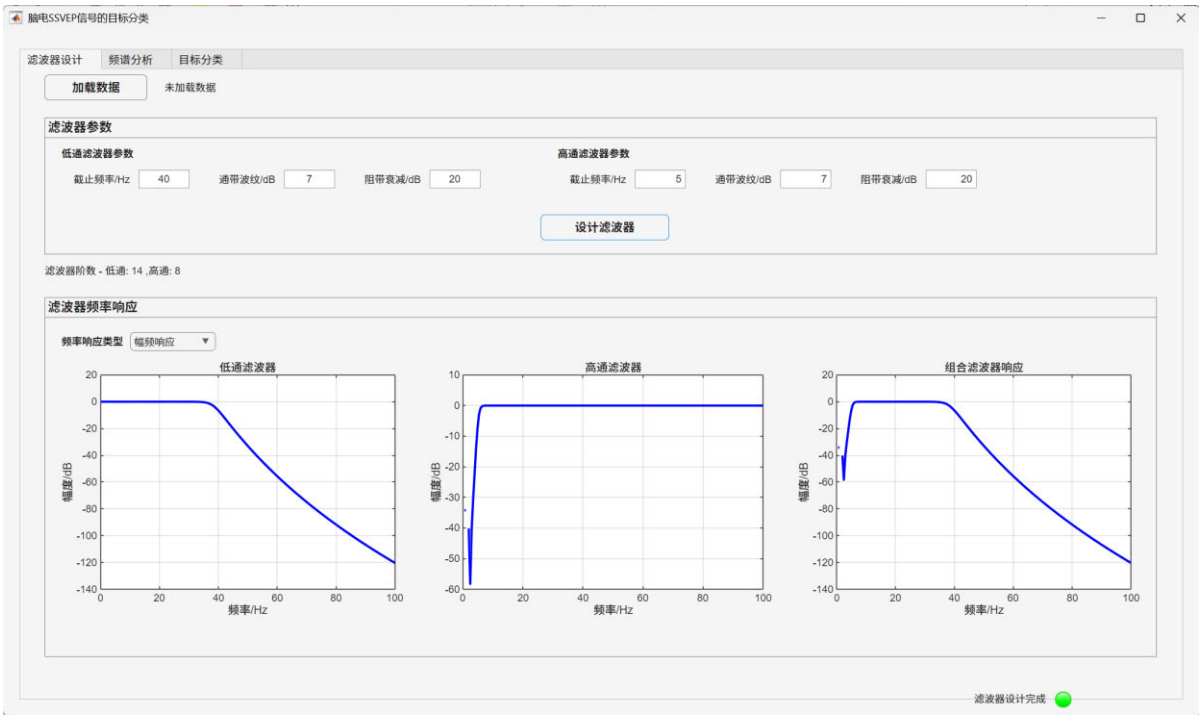


图 5：滤波器幅频响应

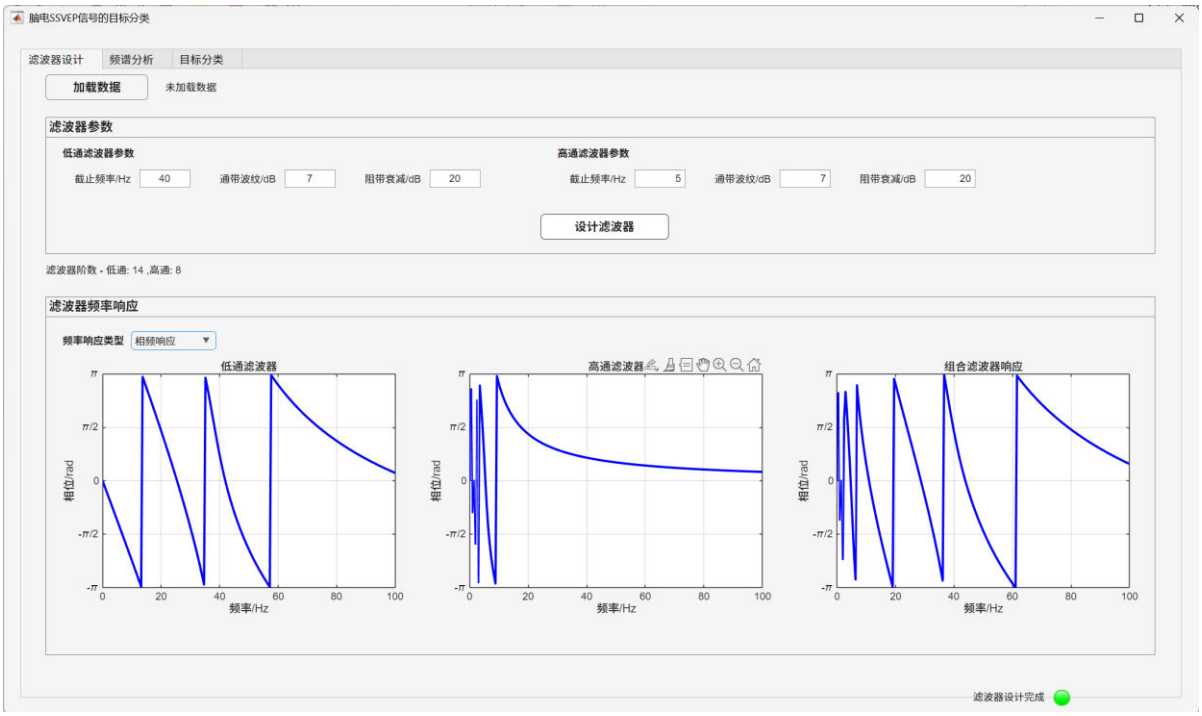


图 6：滤波器相频响应

2.2. 分析讨论

通过观察滤波器的频率响应曲线，可发现低通滤波器和高通滤波器均有较为平坦的通带，并且截止频率符合设计参数要求。将低通滤波器和高通滤波器组合在一起，即可得到对应的带通滤波器。这种通过低通和高通滤波器组合构成带通滤波器的设计方法不仅可控制滤波器的阶数不会太大，更可以提高滤波器设计的灵活性，与直接设计带通滤波器相比具有明显的优势。从所设计滤波器的幅频响应曲线可以看出，所设计的巴特沃斯滤波器具有非常平坦的通带，这样使得通带内的频率响应不会出现很大的起伏，可以更好地保存所需要的基频和倍频频率分量，有利于后续的进一步分析。

3. 频谱分析

3.1. 频谱分析结果

加载数据文件后，在标签页 2 选择试次 1 通道 1 的脑电信号，按照默认的参数进行 FFT 频谱分析，结果如图 7 所示：

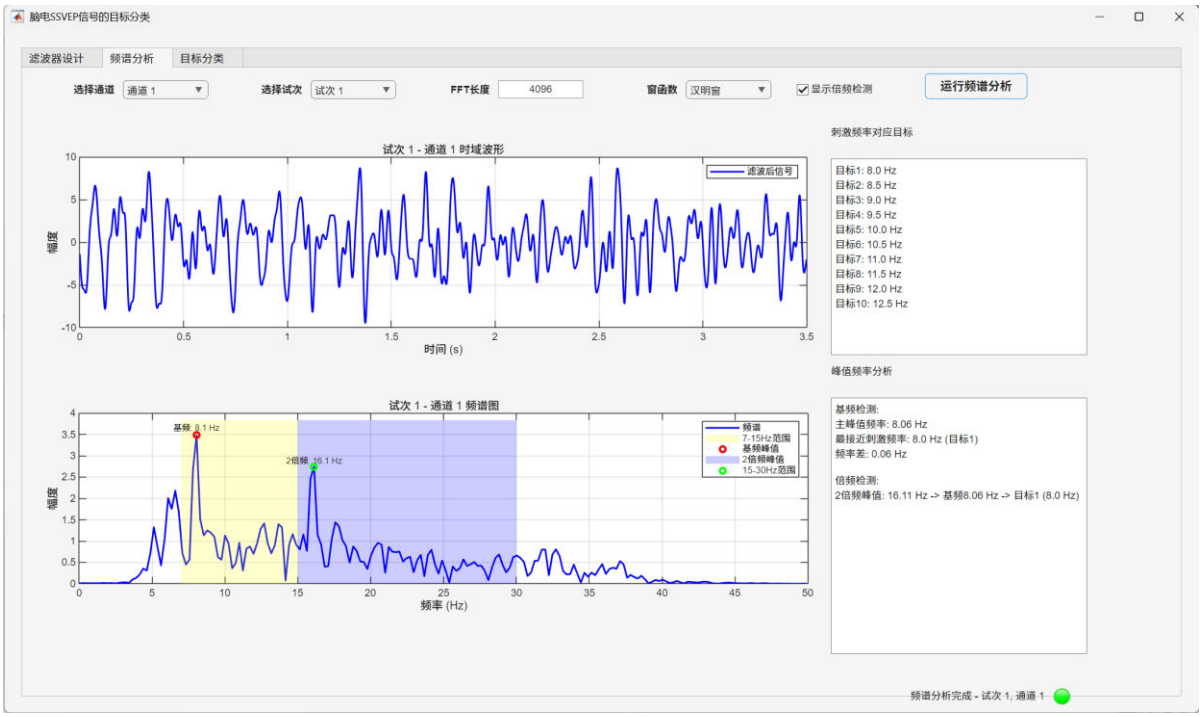


图 7：脑电信号的频谱分析

因为基频检测的结果已包含在基倍频联合检测的结果中，所以在此仅给出显示倍频检测的结果图。

3.2. 分析讨论

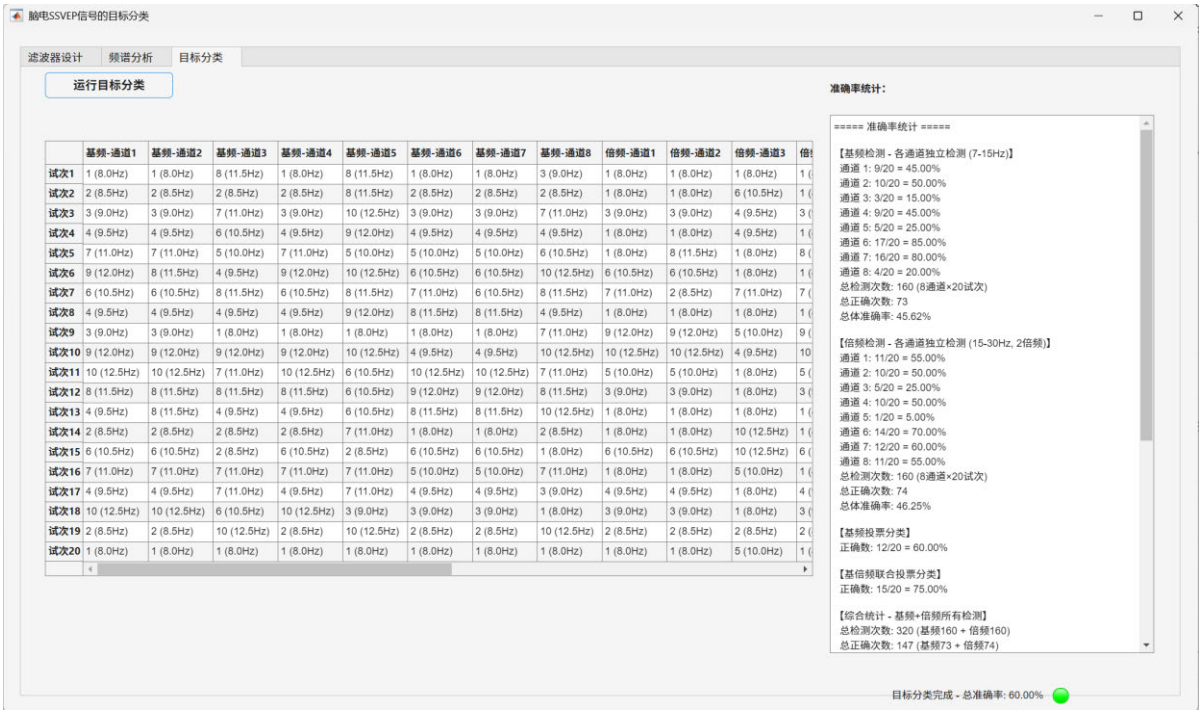
通过观察上图可以看出，试次 1 通道 1 脑电信号的基频峰值位于 8.1Hz，倍频峰值位于 16.1Hz，通过基频检测和倍频检测均可将试次 1 通道 1 脑电信号分类为类别 1，这也与实际情况相符，初步验证了频谱分析的正确性。

若需要查看其他试次通道脑电信号的频谱分析情况，可以更改试次和通道再次运行频谱分析，即可相应的分析结果。

4. 目标分类

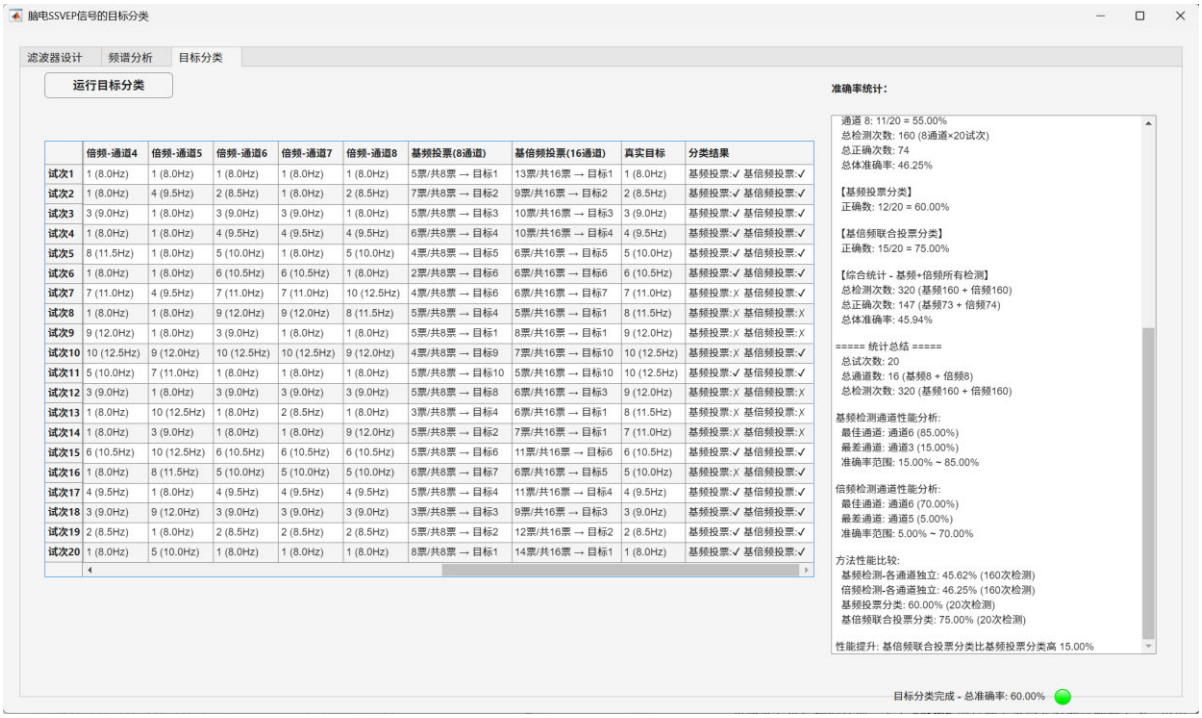
4.1. 目标分类结果

加载数据文件后，在标签页 3 直接点击“运行目标分类”按钮，目标分类的结果如图 8、图 9 所示：



	基频-通道1	基频-通道2	基频-通道3	基频-通道4	基频-通道5	基频-通道6	基频-通道7	基频-通道8	倍频-通道1	倍频-通道2	倍频-通道3	倍
试次1	1 (8.0Hz)	1 (8.0Hz)	8 (11.5Hz)	1 (8.0Hz)	8 (11.5Hz)	1 (8.0Hz)	1 (8.0Hz)	3 (9.0Hz)	1 (8.0Hz)	1 (8.0Hz)	1 (8.0Hz)	1 (
试次2	2 (8.5Hz)	2 (8.5Hz)	2 (8.5Hz)	2 (8.5Hz)	8 (11.5Hz)	2 (8.5Hz)	2 (8.5Hz)	2 (8.5Hz)	1 (8.0Hz)	1 (8.0Hz)	6 (10.5Hz)	1 (
试次3	3 (9.0Hz)	3 (9.0Hz)	7 (11.0Hz)	3 (9.0Hz)	10 (12.5Hz)	3 (9.0Hz)	3 (9.0Hz)	7 (11.0Hz)	3 (9.0Hz)	3 (9.0Hz)	4 (9.5Hz)	3 (
试次4	4 (9.5Hz)	4 (9.5Hz)	6 (10.5Hz)	4 (9.5Hz)	9 (12.0Hz)	4 (9.5Hz)	4 (9.5Hz)	4 (9.5Hz)	1 (8.0Hz)	1 (8.0Hz)	4 (9.5Hz)	1 (
试次5	7 (11.0Hz)	7 (11.0Hz)	5 (10.0Hz)	7 (11.0Hz)	5 (10.0Hz)	5 (10.0Hz)	5 (10.0Hz)	6 (10.5Hz)	1 (8.0Hz)	8 (11.5Hz)	1 (8.0Hz)	8 (
试次6	9 (12.0Hz)	8 (11.5Hz)	4 (9.5Hz)	9 (12.0Hz)	10 (12.5Hz)	6 (10.5Hz)	6 (10.5Hz)	10 (12.5Hz)	6 (10.5Hz)	6 (10.5Hz)	1 (8.0Hz)	1 (
试次7	6 (10.5Hz)	6 (10.5Hz)	8 (11.5Hz)	6 (10.5Hz)	8 (11.5Hz)	7 (11.0Hz)	6 (10.5Hz)	8 (11.5Hz)	7 (11.0Hz)	2 (8.5Hz)	7 (11.0Hz)	7 (
试次8	4 (9.5Hz)	4 (9.5Hz)	4 (9.5Hz)	4 (9.5Hz)	9 (12.0Hz)	8 (11.5Hz)	8 (11.5Hz)	4 (9.5Hz)	1 (8.0Hz)	1 (8.0Hz)	1 (8.0Hz)	1 (
试次9	3 (9.0Hz)	3 (9.0Hz)	1 (8.0Hz)	1 (8.0Hz)	1 (8.0Hz)	1 (8.0Hz)	1 (8.0Hz)	7 (11.0Hz)	9 (12.0Hz)	9 (12.0Hz)	5 (10.0Hz)	9 (
试次10	9 (12.0Hz)	9 (12.0Hz)	9 (12.0Hz)	9 (12.0Hz)	10 (12.5Hz)	4 (9.5Hz)	4 (9.5Hz)	10 (12.5Hz)	10 (12.5Hz)	10 (12.5Hz)	4 (9.5Hz)	10
试次11	10 (12.5Hz)	10 (12.5Hz)	7 (11.0Hz)	10 (12.5Hz)	6 (10.5Hz)	10 (12.5Hz)	10 (12.5Hz)	7 (11.0Hz)	5 (10.0Hz)	5 (10.0Hz)	1 (8.0Hz)	5 (
试次12	8 (11.5Hz)	8 (11.5Hz)	8 (11.5Hz)	8 (11.5Hz)	6 (10.5Hz)	9 (12.0Hz)	9 (12.0Hz)	8 (11.5Hz)	3 (9.0Hz)	3 (9.0Hz)	1 (8.0Hz)	3 (
试次13	4 (9.5Hz)	8 (11.5Hz)	4 (9.5Hz)	4 (9.5Hz)	6 (10.5Hz)	8 (11.5Hz)	8 (11.5Hz)	10 (12.5Hz)	1 (8.0Hz)	1 (8.0Hz)	1 (8.0Hz)	1 (
试次14	2 (8.5Hz)	2 (8.5Hz)	2 (8.5Hz)	2 (8.5Hz)	7 (11.0Hz)	1 (8.0Hz)	1 (8.0Hz)	2 (8.5Hz)	1 (8.0Hz)	1 (8.0Hz)	10 (12.5Hz)	1 (
试次15	6 (10.5Hz)	6 (10.5Hz)	2 (8.5Hz)	6 (10.5Hz)	2 (8.5Hz)	6 (10.5Hz)	6 (10.5Hz)	1 (8.0Hz)	6 (10.5Hz)	6 (10.5Hz)	10 (12.5Hz)	6 (
试次16	7 (11.0Hz)	7 (11.0Hz)	7 (11.0Hz)	7 (11.0Hz)	7 (11.0Hz)	5 (10.0Hz)	5 (10.0Hz)	7 (11.0Hz)	1 (8.0Hz)	1 (8.0Hz)	5 (10.0Hz)	1 (
试次17	4 (9.5Hz)	4 (9.5Hz)	7 (11.0Hz)	4 (9.5Hz)	7 (11.0Hz)	4 (9.5Hz)	4 (9.5Hz)	3 (9.0Hz)	4 (9.5Hz)	4 (9.5Hz)	1 (8.0Hz)	4 (
试次18	10 (12.5Hz)	10 (12.5Hz)	6 (10.5Hz)	10 (12.5Hz)	3 (9.0Hz)	3 (9.0Hz)	3 (9.0Hz)	1 (8.0Hz)	3 (9.0Hz)	3 (9.0Hz)	1 (8.0Hz)	3 (
试次19	2 (8.5Hz)	2 (8.5Hz)	10 (12.5Hz)	2 (8.5Hz)	10 (12.5Hz)	2 (8.5Hz)	2 (8.5Hz)	10 (12.5Hz)	2 (8.5Hz)	2 (8.5Hz)	2 (8.5Hz)	2 (
试次20	1 (8.0Hz)	1 (8.0Hz)	1 (8.0Hz)	1 (8.0Hz)	1 (8.0Hz)	1 (8.0Hz)	1 (8.0Hz)	1 (8.0Hz)	1 (8.0Hz)	1 (8.0Hz)	5 (10.0Hz)	1 (

图 8：目标分类结果(1)



	倍频-通道4	倍频-通道5	倍频-通道6	倍频-通道7	倍频-通道8	基频投票(8通道)	基倍频投票(16通道)	真实目标	分类结果
试次1	1 (8.0Hz)	1 (8.0Hz)	1 (8.0Hz)	1 (8.0Hz)	1 (8.0Hz)	5票/共8票 → 目标1	13票/共16票 → 目标1	1 (8.0Hz)	基频投票:✓ 基倍频投票:✓
试次2	1 (8.0Hz)	4 (9.5Hz)	2 (8.5Hz)	1 (8.0Hz)	2 (8.5Hz)	7票/共8票 → 目标2	9票/共16票 → 目标2	2 (8.5Hz)	基频投票:✓ 基倍频投票:✓
试次3	3 (9.0Hz)	1 (8.0Hz)	3 (9.0Hz)	3 (9.0Hz)	1 (8.0Hz)	5票/共8票 → 目标3	10票/共16票 → 目标3	3 (9.0Hz)	基频投票:✓ 基倍频投票:✓
试次4	1 (8.0Hz)	1 (8.0Hz)	4 (9.5Hz)	4 (9.5Hz)	4 (9.5Hz)	6票/共8票 → 目标4	10票/共16票 → 目标4	4 (9.5Hz)	基频投票:✓ 基倍频投票:✓
试次5	8 (11.5Hz)	1 (8.0Hz)	5 (10.0Hz)	1 (8.0Hz)	5 (10.0Hz)	4票/共8票 → 目标5	6票/共16票 → 目标5	5 (10.0Hz)	基频投票:✓ 基倍频投票:✓
试次6	1 (8.0Hz)	1 (8.0Hz)	6 (10.5Hz)	6 (10.5Hz)	1 (8.0Hz)	2票/共8票 → 目标6	6票/共16票 → 目标6	6 (10.5Hz)	基频投票:✓ 基倍频投票:✓
试次7	7 (11.0Hz)	4 (9.5Hz)	7 (11.0Hz)	7 (11.0Hz)	10 (12.5Hz)	4票/共8票 → 目标6	6票/共16票 → 目标7	7 (11.0Hz)	基频投票:X 基倍频投票:✓
试次8	1 (8.0Hz)	1 (8.0Hz)	9 (12.0Hz)	9 (12.0Hz)	8 (11.5Hz)	5票/共8票 → 目标4	5票/共16票 → 目标1	8 (11.5Hz)	基频投票:X 基倍频投票:X
试次9	9 (12.0Hz)	1 (8.0Hz)	3 (9.0Hz)	1 (8.0Hz)	1 (8.0Hz)	5票/共8票 → 目标1	8票/共16票 → 目标1	9 (12.0Hz)	基频投票:X 基倍频投票:X
试次10	10 (12.5Hz)	9 (12.0Hz)	10 (12.5Hz)	10 (12.5Hz)	9 (12.0Hz)	4票/共8票 → 目标9	7票/共16票 → 目标10	10 (12.5Hz)	基频投票:X 基倍频投票:✓
试次11	5 (10.0Hz)	7 (11.0Hz)	1 (8.0Hz)	1 (8.0Hz)	1 (8.0Hz)	5票/共8票 → 目标10	5票/共16票 → 目标10	10 (12.5Hz)	基频投票:✓ 基倍频投票:✓
试次12	3 (9.0Hz)	1 (8.0Hz)	3 (9.0Hz)	3 (9.0Hz)	3 (9.0Hz)	5票/共8票 → 目标8	6票/共16票 → 目标3	9 (12.0Hz)	基频投票:X 基倍频投票:X
试次13	1 (8.0Hz)	10 (12.5Hz)	1 (8.0Hz)	2 (8.5Hz)	1 (8.0Hz)	3票/共8票 → 目标4	6票/共16票 → 目标1	8 (11.5Hz)	基频投票:X 基倍频投票:X
试次14	1 (8.0Hz)	3 (9.0Hz)	1 (8.0Hz)	1 (8.0Hz)	9 (12.0Hz)	5票/共8票 → 目标2	7票/共16票 → 目标1	7 (11.0Hz)	基频投票:X 基倍频投票:X
试次15	6 (10.5Hz)	10 (12.5Hz)	6 (10.5Hz)	6 (10.5Hz)	6 (10.5Hz)	5票/共8票 → 目标6	11票/共16票 → 目标6	6 (10.5Hz)	基频投票:✓ 基倍频投票:✓
试次16	1 (8.0Hz)	8 (11.5Hz)	5 (10.0Hz)	5 (10.0Hz)	5 (10.0Hz)	6票/共8票 → 目标7	6票/共16票 → 目标5	5 (10.0Hz)	基频投票:X 基倍频投票:✓
试次17	4 (9.5Hz)	1 (8.0Hz)	4 (9.5Hz)	4 (9.5Hz)	4 (9.5Hz)	5票/共8票 → 目标4	11票/共16票 → 目标4	4 (9.5Hz)	基频投票:✓ 基倍频投票:✓
试次18	3 (9.0Hz)	9 (12.0Hz)	3 (9.0Hz)	3 (9.0Hz)	3 (9.0Hz)	3票/共8票 → 目标3	9票/共16票 → 目标3	3 (9.0Hz)	基频投票:✓ 基倍频投票:✓
试次19	2 (8.5Hz)	1 (8.0Hz)	2 (8.5Hz)	2 (8.5Hz)	2 (8.5Hz)	5票/共8票 → 目标2	12票/共16票 → 目标2	2 (8.5Hz)	基频投票:✓ 基倍频投票:✓
试次20	1 (8.0Hz)	5 (10.0Hz)	1 (8.0Hz)	1 (8.0Hz)	1 (8.0Hz)	8票/共8票 → 目标1	14票/共16票 → 目标1	1 (8.0Hz)	基频投票:✓ 基倍频投票:✓

图 9：目标分类结果(2)

注意，由于表格和文本框较长，目标分类结果将通过两张图展示。

4.2. 分析讨论

观察图 8、图 9 的统计结果可以发现，不论是基频检测还是倍频检测，所有 20 个试次 8 个通道的分析结果将不再像图 7 一样理想，而是会发生一定的分类错误。分类

错误可能是由于检测到的脑电信号受噪声影响比较严重，更重要的是脑电信号不光只有 SSVEP 信号，还含有其他不需要的干扰信号，这些干扰信号的频率分量会对目标分类造成影响，从而导致分类出现错误。观察文本框中基频和倍频独立检测的总体准确率还不到 50%，这说明脑电信号受到的其他干扰是非常严重的。但进行基频 8 通道的联合投票后，分类的准确率达到 60%，这是因为投票可以汇总各通道的分类信息，从而减小其他干扰，提高分类的准确率。当进行基倍频联合投票时，此时分类又汇总了倍频的信息，投票的总票数增加到 16，从而进一步将分类的准确率提高到 75%。观察文本框中基频和倍频检测各个通道独立检测的分类准确率，可以发现各个通道的分类准确率差异较大，这是因为每个通道导联的位置不同，采集到信号的频率特征和受到的干扰也有所不同。可以发现通道 6 在基频检测和倍频检测的准确率均是最高的，这说明通道 6 可以更好地反映了脑电 SSVEP 信号的频谱特征，因此在后续的脑电信号目标分类实践中可以更多地以通道 6 检测到的信号为标准进行分类。

四、 实验总结

通过本次脑电 SSVEP 信号目标分类实验，我收获了扎实的 SSVEP 信号处理专业知识与 MATLAB App Designer 开发实践经验，对脑机接口核心技术的原理与信号处理技术有了更为透彻的理解。实验过程中，我不仅深入掌握了 SSVEP 信号的产生机制、基频和倍频特征提取方法，还熟练运用 MATLAB 实现了 IIR 巴特沃斯带通滤波设计、加窗 FFT 频谱分析、多通道投票分类等核心算法，成功开发出脑电 SSVEP 信号的目标分类的模块化 GUI 系统。

其中，SSVEP 信号处理算法优化与 App 跨模块数据交互是我在实验中遇到的主要困难。第一个问题是频谱泄漏导致峰值检测准确率偏低。刚开始进行 FFT 分析时，我并没有进行加窗处理，由于 SSVEP 信号频率未完全对准离散频率点，出现了频谱泄漏现象，导致基频与倍频峰值识别偏差。为解决这一问题，我通过对比不同窗函数性能，在进行 FFT 分析之前对脑电信号使用汉明窗进行加窗处理，同时增加 FFT 补零长度至 4096 点，最终有效抑制了频谱泄漏，从而提高分类准确率。第二个问题是多通道投票算法设计错误。在程序开发初期，我对倍频有效判断标准不明确，导致部分无效倍频数据参与投票，降低了分类准确率。最后我通过设定倍频峰值不小于基频峰值 10% 的有效阈值，在 App 私有属性中添加投票计数变量，并通过逻辑判断实现了基频单独投票与基频+倍频的联合投票的灵活切换，解决了分类逻辑冲突问题。

本次实验让我深刻体会到理论与实践结合的重要意义。之前在课堂上学习的 FFT 频谱分析、IIR 滤波器设计等理论知识，通过这款 SSVEP 信号分类系统的开发得到了

全面应用。实验中，我不仅验证了 5-40Hz 带通滤波对脑电信号噪声与其他干扰的抑制效果，观察到了频谱泄漏对频率检测的影响，还通过多通道数据融合验证了基倍频联合分类的优势，进一步加深了对 SSVEP 信号特征与分类算法的理解，真切感受到该技术在脑机接口领域的应用价值。

最后，我要衷心感谢[redacted]的悉心指导。[redacted]对 SSVEP 信号处理原理和基倍频投票逻辑等算法的细致讲解，帮助我快速理清了系统开发思路。通过本次实验，我不仅夯实了数字信号处理与脑机接口的基础理论，积累了算法优化与 GUI 开发的实践经验，更培养了科学严谨的实验态度和自主调试的问题解决思维。这些收获将为我后续深入学习信号处理进阶算法、开展相关科研实践奠定坚实基础，对未来的专业学习与发展产生积极而深远的影响。

五、 程序代码

```
classdef exp5 < matlab.apps.AppBase

    % Properties that correspond to app components
    properties (Access = public)
        UIFigure                    matlab.ui.Figure
        ProgressLamp                matlab.ui.control.Lamp
        StatusLabel                 matlab.ui.control.Label
        TabGroup                    matlab.ui.container.TabGroup
        DataFilterTab               matlab.ui.container.Tab
        FilterOrderInfoLabel        matlab.ui.control.Label
        FilterResponsePanel         matlab.ui.container.Panel
        FreqzDropDown               matlab.ui.control.DropDown
        FreqzLabel                  matlab.ui.control.Label
        CombinedResponseAxes        matlab.ui.control.UIAxes
        LowpassResponseAxes         matlab.ui.control.UIAxes
        HighpassResponseAxes        matlab.ui.control.UIAxes
        FilterParametersPanel       matlab.ui.container.Panel
        DesignFilterButton          matlab.ui.control.Button
        HighpassStopAttenEditField  matlab.ui.control.NumericEditField
        HighpassStopAttenLabel      matlab.ui.control.Label
        HighpassCutoffEditField     matlab.ui.control.NumericEditField
        HighpassCutoffLabel         matlab.ui.control.Label
        HighpassPassRippleEditField matlab.ui.control.NumericEditField
        HighpassPassRippleLabel     matlab.ui.control.Label
        HighpassParametersLabel     matlab.ui.control.Label
        LowpassStopAttenEditField   matlab.ui.control.NumericEditField
        LowpassStopAttenLabel       matlab.ui.control.Label
        LowpassCutoffEditField      matlab.ui.control.NumericEditField
        LowpassCutoffLabel          matlab.ui.control.Label
```

```
        LowpassPassRippleEditField   matlab.ui.control.NumericEditField
        LowpassPassRippleLabel       matlab.ui.control.Label
        LowpassParametersLabel       matlab.ui.control.Label
        DataInfoLabel                matlab.ui.control.Label
        LoadDataButton               matlab.ui.control.Button
        AnalysisTab                  matlab.ui.container.Tab
        PeakFrequencyText            matlab.ui.control.TextArea
        PeakFrequencyLabel           matlab.ui.control.Label
        StimulusFrequenciesText      matlab.ui.control.TextArea
        StimulusFrequenciesLabel     matlab.ui.control.Label
        RunAnalysisButton            matlab.ui.control.Button
        ShowHarmonicCheckBox         matlab.ui.control.CheckBox
        WindowTypeDropDown           matlab.ui.control.DropDown
        WindowTypeLabel              matlab.ui.control.Label
        FFTEditField                 matlab.ui.control.NumericEditField
        FFTLengthLabel               matlab.ui.control.Label
        TrialSelectorDropDown        matlab.ui.control.DropDown
        TrialSelectorLabel           matlab.ui.control.Label
        ChannelSelectorDropDown      matlab.ui.control.DropDown
        ChannelSelectorLabel         matlab.ui.control.Label
        SpectrumAxes                 matlab.ui.control.UIAxes
        TimeDomainAxes               matlab.ui.control.UIAxes
        ResultsTab                   matlab.ui.container.Tab
        AccuracyTextArea             matlab.ui.control.TextArea
        AccuracyLabel                matlab.ui.control.Label
        ResultsTable                 matlab.ui.control.Table
        RunClassificationButton      matlab.ui.control.Button
    end


    properties (Access = private)
        Property % Description
        % 数据属性
        RawData                  % 原始脑电数据 (8×3500×20)
        Labels                   % 真实标签 (20×1)
        FilteredData             % 滤波后数据
        Fs = 1000                % 采样频率 (Hz)

        % 滤波器系数
        lpCoeff                  % 低通滤波器系数
        hpCoeff                  % 高通滤波器系数
        lpH                      % 低通滤波器频率响应
        hpH                      % 高通滤波器频率响应
        combinedH                % 组合滤波器频率响应
```

```
        combinedF                   % 滤波器频率范围
        
        % 频率参数
        StimFreqs = [8, 8.5, 9, 9.5, 10, 10.5, 11, 11.5, 12, 12.5]  % 刺激频率
        Harmonics = [2, 3, 4]       % 倍频阶数
        
        % 分析结果
        ChannelPredictions          % 每个通道的基频预测 (20×8)
        ChannelHarmonicPredictions % 每个通道的基倍频预测 (20×8)
        JointPredictions            % 8 通道联合预测 (20×1)
        HarmonicPredictions         % 基频+倍频预测 (20×1)
        PeakFrequencies             % 峰值频率 (20×8)
        
        % 准确率
        ChannelAccuracies          % 每个通道的基频准确率 (1×8)
        ChannelHarmonicAccuracies % 每个通道的基倍频准确率 (1×8)
        JointAccuracy              % 联合准确率
        HarmonicAccuracy           % 基频+倍频准确率
        
        % 滤波器参数
        LowpassCutoff = 40         % 低通截止频率 (Hz)
        LowpassPassRipple = 7      % 低通通带波纹 (dB)
        LowpassStopAtten = 20      % 低通阻带衰减 (dB)
        
        HighpassCutoff = 5         % 高通截止频率 (Hz)
        HighpassPassRipple = 7     % 高通通带波纹 (dB)
        HighpassStopAtten = 20     % 高通阻带衰减 (dB)
        
        HarmonicPeakFrequencies
    end

    methods (Access = private)

        function filterSignal = applyCascadeFilter(app, signal)
            % 应用级联滤波器（先低通后高通）
            % 第一步：低通滤波
            lpFiltered = filtfilt(app.lpCoeff.b, app.lpCoeff.a, signal);

            % 第二步：高通滤波
            filterSignal = filtfilt(app.hpCoeff.b, app.hpCoeff.a, lpFiltered);
        end

        function [spectrum, freq] = computeFFTSpectrum(app, signal)
            % 计算信号的 FFT 频谱
            N = app.FFTEditField.Value;
```

```
119.
120.             % 加窗
121.             switch app.WindowTypeDropDown.Value
122.                 case '矩形窗'
123.                     window = rectwin(length(signal));
124.                 case '汉宁窗'
125.                     window = hann(length(signal));
126.                 case '汉明窗'
127.                     window = hamming(length(signal));
128.                 case '布莱克曼窗'
129.                     window = blackman(length(signal));
130.                 otherwise
131.                     window = hamming(length(signal));
132.             end
133.
134.             % 应用窗函数
135.             windowedSignal = signal(:) .* window;
136.
137.             % 补零到 FFT 长度
138.             if length(windowedSignal) < N
139.                 paddedSignal = [windowedSignal; zeros(N - length(windowedSignal), 1)];

140.             else
141.                 paddedSignal = windowedSignal(1:N);
142.             end
143.
144.             % 计算 FFT
145.             fftResult = fft(paddedSignal, N);
146.
147.             % 计算单边频谱
148.             spectrum = abs(fftResult(1:N/2+1)/app.Fs);
149.             spectrum(2:end-1) = spectrum(2:end-1) * 2;
150.             freq = (0:N/2) * app.Fs / N;
151.         end
152.
153.         function [targetIdx, peakFreq] = detectFundamentalFrequency(app, spectrum, fre
    q)
154.             % 在 7-15Hz 范围内寻找基频峰值
155.             targetIdx = 0;
156.             peakFreq = 0;
157.
158.             % 在 7-15Hz 范围内寻找峰值
159.             freqRange = (freq >= 7 & freq <= 15);
160.             spectrumRange = spectrum(freqRange);
161.             freqRangeValues = freq(freqRange);
```

```
162.
163.             [peaks, locs] = findpeaks(spectrumRange, 'NPeaks', 3, 'SortStr', 'descend'
    );
164.
165.             if ~isempty(peaks)
166.                 % 主峰值频率
167.                 mainPeakFreq = freqRangeValues(locs(1));
168.                 peakFreq = mainPeakFreq;
169.
170.                 % 找到最接近的刺激频率
171.                 [~, targetIdx] = min(abs(mainPeakFreq - app.StimFreqs));
172.             end
173.         end
174.
175.         function [targetIdx, peakFreq] = detectHarmonicFrequency(app, spectrum, freq,
    basePeakVal)
176.             % 在 15-30Hz 范围内寻找 2 倍频峰值
177.             targetIdx = 0;
178.             peakFreq = 0;
179.
180.             % 在 15-30Hz 范围内寻找峰值
181.             freqRangeHarmonic = (freq >= 15 & freq <= 30);
182.             if any(freqRangeHarmonic)
183.                 % 在倍频范围内找最高峰值
184.                 [maxValHarmonic, maxIdxHarmonic] = max(spectrum(freqRangeHarmonic));
185.                 harmonicFreqPeak = freq(freqRangeHarmonic);
186.                 harmonicPeakFreq = harmonicFreqPeak(maxIdxHarmonic);
187.                 peakFreq = harmonicPeakFreq;
188.
189.                 % 将倍频频率转换为基频频率（除以 2）
190.                 baseFreqFromHarmonic = harmonicPeakFreq / 2;
191.
192.                 % 找到最接近的刺激频率
193.                 [~, targetIdx] = min(abs(baseFreqFromHarmonic - app.StimFreqs));
194.
195.                 % 检查峰值是否超过阈值（基频峰值的 10%）
196.                 if maxValHarmonic <= basePeakVal * 0.1  % 阈值
197.                     targetIdx = 0;  % 不采用
198.                 end
199.             end
200.         end
201.
202.         function updateResultsDisplay(app)
203.             % 更新结果显示
204.             if isempty(app.ChannelPredictions)
```

```
205.                 return;
206.             end
207.
208.             numTrials = length(app.Labels);
209.             numChannels = size(app.ChannelPredictions, 2);
210.
211.             % 创建结果显示表格
212.             resultData = cell(numTrials, 20);
213.
214.             % 重新计算每个试次的投票数量，用于显示
215.             for trial = 1:numTrials
216.                 % ===== 计算基频投票数量 =====
217.                 baseVotes = zeros(1, length(app.StimFreqs));
218.                 for channel = 1:numChannels
219.                     pred = app.ChannelPredictions(trial, channel);
220.                     if pred > 0
221.                         baseVotes(pred) = baseVotes(pred) + 1;
222.                     end
223.                 end
224.
225.                 % 找到基频投票的最高票数
226.                 % ===== 计算基倍频投票数量 =====
227.                 combinedVotes = zeros(1, length(app.StimFreqs));
228.                 for channel = 1:numChannels
229.                     % 基频投票
230.                     basePred = app.ChannelPredictions(trial, channel);
231.                     if basePred > 0
232.                         combinedVotes(basePred) = combinedVotes(basePred) + 1;
233.                     end
234.
235.                     % 倍频投票
236.                     harmonicPred = app.ChannelHarmonicPredictions(trial, channel);
237.                     if harmonicPred > 0
238.                         combinedVotes(harmonicPred) = combinedVotes(harmonicPred) + 1;
239.                     end
240.                 end
241.
242.                 % 找到基倍频投票的最高票数
243.
244.                 % 各通道基频检测结果 (列 1-8)
245.                 for channel = 1:8
246.                     if channel <= size(app.ChannelPredictions, 2)
247.                         pred = app.ChannelPredictions(trial, channel);
248.                         freqVal = app.PeakFrequencies(trial, channel);
```

```
                        if pred > 0 && freqVal > 0
                            resultData{trial, channel} = sprintf('%d (%.1fHz)', pred, app.StimFreqs(pred));
                        else
                            resultData{trial, channel} = '未检测';
                        end
                    else
                        resultData{trial, channel} = 'N/A';
                    end
                end

                % 各通道倍频检测结果 (列 9-16)
                for channel = 1:8
                    colIdx = 8 + channel;
                    if channel <= size(app.ChannelHarmonicPredictions, 2)
                        pred = app.ChannelHarmonicPredictions(trial, channel);
                        freqVal = app.HarmonicPeakFrequencies(trial, channel);
                        if pred > 0 && freqVal > 0
                            resultData{trial, colIdx} = sprintf('%d (%.1fHz)', pred, app.StimFreqs(pred));
                        else
                            resultData{trial, colIdx} = '未检测';
                        end
                    else
                        resultData{trial, colIdx} = 'N/A';
                    end
                end

                % 基频检测-8 通道联合投票结果 (列 17)
                jointPred = app.JointPredictions(trial);
                baseVoteCounts = zeros(1, length(app.StimFreqs));
                for ch = 1:numChannels
                    pred = app.ChannelPredictions(trial, ch);
                    if pred > 0
                        baseVoteCounts(pred) = baseVoteCounts(pred) + 1;
                    end
                end
                voteCount = baseVoteCounts(jointPred);
                totalBaseVotes = sum(baseVoteCounts);
                resultData{trial, 17} = sprintf('%d 票/共%d 票 → 目标%d', ...
                    voteCount, totalBaseVotes, jointPred);

                % 基频+倍频-16 通道联合投票结果 (列 18)
                harmonicPred = app.HarmonicPredictions(trial);
                combinedVoteCounts = zeros(1, length(app.StimFreqs));
```

```
                for ch = 1:numChannels
                    % 基频投票
                    basePred = app.ChannelPredictions(trial, ch);
                    if basePred > 0
                        combinedVoteCounts(basePred) = combinedVoteCounts(basePred) + 1;
                    end

                    % 倍频投票
                    harmonicPredCh = app.ChannelHarmonicPredictions(trial, ch);
                    if harmonicPredCh > 0
                        combinedVoteCounts(harmonicPredCh) = combinedVoteCounts(harmonicPredCh) + 1;
                    end
                end
                voteCountCombined = combinedVoteCounts(harmonicPred);
                totalCombinedVotes = sum(combinedVoteCounts);
                resultData{trial, 18} = sprintf('%d 票/共%d 票 → 目标%d', ...
                    voteCountCombined, totalCombinedVotes, harmonicPred);

                % 真实标签 (列 19)
                trueLabel = app.Labels(trial);
                resultData{trial, 19} = sprintf('%d (%.1fHz)', trueLabel, app.StimFreqs(trueLabel));

                % 检查是否正确 (列 20)
                isJointCorrect = (jointPred == trueLabel);
                isHarmonicCorrect = (harmonicPred == trueLabel);

                resultData{trial, 20} = sprintf('基频投票:%s 基倍频投票:%s', ...
                    char(10004*isJointCorrect + '╳'*~isJointCorrect), ...
                    char(10004*isHarmonicCorrect + '╳'*~isHarmonicCorrect));
            end

            % 设置表格数据
            columnNames = cell(1, 20);
            for i = 1:8
                columnNames{i} = sprintf('基频-通道%d', i);
                columnNames{i+8} = sprintf('倍频-通道%d', i);
            end
            columnNames{17} = '基频投票(8 通道)';
            columnNames{18} = '基倍频投票(16 通道)';
            columnNames{19} = '真实目标';
            columnNames{20} = '分类结果';

```

```
            app.ResultsTable.Data = resultData;
            app.ResultsTable.ColumnName = columnNames;

            % ===== 计算准确率统计 =====
            accuracyText = sprintf('===== 准确率统计 =====\n\n');

            % 1. 基频检测统计
            accuracyText = sprintf('%s【基频检测 - 各通道独立检测 (7-15Hz)】\n', accuracyText);
            totalCorrectBase = 0;

            for channel = 1:numChannels
                channelCorrect = sum(app.ChannelPredictions(:, channel) == app.Labels);
                totalCorrectBase = totalCorrectBase + channelCorrect;

                accuracyText = sprintf('%s  通道 %d: %d/%d = %.2f%%\n', accuracyText, ...
                    channel, channelCorrect, numTrials, app.ChannelAccuracies(channel));
            end

            totalBaseDetections = numTrials * numChannels;
            overallAccuracyBase = totalCorrectBase / totalBaseDetections * 100;
            accuracyText = sprintf('%s  总检测次数: %d (8通道×20试次)\n', accuracyText, totalBaseDetections);
            accuracyText = sprintf('%s  总正确次数: %d\n', accuracyText, totalCorrectBase);
            accuracyText = sprintf('%s  总体准确率: %.2f%%\n\n', accuracyText, overallAccuracyBase);

            % 2. 倍频检测统计
            accuracyText = sprintf('%s【倍频检测 - 各通道独立检测 (15-30Hz, 2倍频)】\n', accuracyText);
            totalCorrectHarmonic = 0;

            for channel = 1:numChannels
                channelCorrectHarmonic = sum(app.ChannelHarmonicPredictions(:, channel) == app.Labels);
                totalCorrectHarmonic = totalCorrectHarmonic + channelCorrectHarmonic;


                accuracyText = sprintf('%s  通道 %d: %d/%d = %.2f%%\n', accuracyText, ...
```

```
                    channel, channelCorrectHarmonic, numTrials, app.ChannelHarmonicAccuracies(channel));
            end

            totalHarmonicDetections = numTrials * numChannels;
            overallAccuracyHarmonic = totalCorrectHarmonic / totalHarmonicDetections * 100;
            accuracyText = sprintf('%s  总检测次数: %d (8 通道×20 试次)\n', accuracyText, totalHarmonicDetections);
            accuracyText = sprintf('%s  总正确次数: %d\n', accuracyText, totalCorrectHarmonic);
            accuracyText = sprintf('%s  总体准确率: %.2f%%\n\n', accuracyText, overallAccuracyHarmonic);

            % 3. 基频检测-8 通道联合投票
            jointCorrect = sum(app.JointPredictions == app.Labels);
            accuracyText = sprintf('%s【基频投票分类】\n', accuracyText);
            accuracyText = sprintf('%s  正确数: %d/%d = %.2f%%\n\n', accuracyText, ...
                jointCorrect, numTrials, app.JointAccuracy);

            % 4. 基频+倍频-16 通道联合投票
            harmonicCorrect = sum(app.HarmonicPredictions == app.Labels);
            accuracyText = sprintf('%s【基倍频联合投票分类】\n', accuracyText);
            accuracyText = sprintf('%s  正确数: %d/%d = %.2f%%\n\n', accuracyText, ...
                harmonicCorrect, numTrials, app.HarmonicAccuracy);

            % 5. 综合统计
            totalAllDetections = totalBaseDetections + totalHarmonicDetections;
            totalAllCorrect = totalCorrectBase + totalCorrectHarmonic;
            overallAllAccuracy = totalAllCorrect / totalAllDetections * 100;
            accuracyText = sprintf('%s【综合统计 - 基频+倍频所有检测】\n', accuracyText);
            accuracyText = sprintf('%s  总检测次数: %d (基频 160 + 倍频 160)\n', accuracyText, totalAllDetections);
            accuracyText = sprintf('%s  总正确次数: %d (基频%d + 倍频%d)\n', accuracyText, ...
                totalAllCorrect, totalCorrectBase, totalCorrectHarmonic);
            accuracyText = sprintf('%s  总体准确率: %.2f%%\n\n', accuracyText, overallAllAccuracy);

            % 6. 统计总结
            accuracyText = sprintf('%s===== 统计总结 =====\n', accuracyText);
```

```
            accuracyText = sprintf('%s  总试次数: %d\n', accuracyText, numTrials);
            accuracyText = sprintf('%s  总通道数: %d (基频%d + 倍频%d)\n', accuracyText, ...
                numChannels*2, numChannels, numChannels);
            accuracyText = sprintf('%s  总检测次数: %d (基频%d + 倍频%d)\n\n', accuracyText, ...
                totalAllDetections, totalBaseDetections, totalHarmonicDetections);

            % 基频检测通道性能分析
            [bestAccuracyBase, bestChannelBase] = max(app.ChannelAccuracies);
            [worstAccuracyBase, worstChannelBase] = min(app.ChannelAccuracies);

            accuracyText = sprintf('%s基频检测通道性能分析:\n', accuracyText);
            accuracyText = sprintf('%s  最佳通道: 通道%d (%.2f%%)\n', accuracyText, bestChannelBase, bestAccuracyBase);
            accuracyText = sprintf('%s  最差通道: 通道%d (%.2f%%)\n', accuracyText, worstChannelBase, worstAccuracyBase);
            accuracyText = sprintf('%s  准确率范围: %.2f%% ~ %.2f%%\n', accuracyText, worstAccuracyBase, bestAccuracyBase);

            % 倍频检测通道性能分析
            [bestAccuracyHarmonic, bestChannelHarmonic] = max(app.ChannelHarmonicAccuracies);
            [worstAccuracyHarmonic, worstChannelHarmonic] = min(app.ChannelHarmonicAccuracies);

            accuracyText = sprintf('%s\n倍频检测通道性能分析:\n', accuracyText);
            accuracyText = sprintf('%s  最佳通道: 通道%d (%.2f%%)\n', accuracyText, bestChannelHarmonic, bestAccuracyHarmonic);
            accuracyText = sprintf('%s  最差通道: 通道%d (%.2f%%)\n', accuracyText, worstChannelHarmonic, worstAccuracyHarmonic);
            accuracyText = sprintf('%s  准确率范围: %.2f%% ~ %.2f%%\n', accuracyText, worstAccuracyHarmonic, bestAccuracyHarmonic);

            % 方法性能比较
            accuracyText = sprintf('%s\n方法性能比较:\n', accuracyText);
            accuracyText = sprintf('%s  基频检测-各通道独立: %.2f%% (160次检测)\n', accuracyText, overallAccuracyBase);
            accuracyText = sprintf('%s  倍频检测-各通道独立: %.2f%% (160次检测)\n', accuracyText, overallAccuracyHarmonic);
            accuracyText = sprintf('%s  基频投票分类: %.2f%% (20次检测)\n', accuracyText, app.JointAccuracy);
            accuracyText = sprintf('%s  基倍频联合投票分类: %.2f%% (20次检测)\n\n', accuracyText, app.HarmonicAccuracy);

```

```
            % 性能提升分析
            if app.HarmonicAccuracy > app.JointAccuracy
                improvement = app.HarmonicAccuracy - app.JointAccuracy;
                accuracyText = sprintf('%s 性能提升：基倍频联合投票分类比基频投票分类高 %.2f%%\n', accuracyText, improvement);
            elseif app.HarmonicAccuracy < app.JointAccuracy
                decline = app.JointAccuracy - app.HarmonicAccuracy;
                accuracyText = sprintf('%s 性能下降：基倍频联合投票分类比基频投票分类低 %.2f%%\n', accuracyText, decline);
            else
                accuracyText = sprintf('%s 性能相同：两种投票方法准确率相同\n', accuracyText);
            end

            app.AccuracyTextArea.Value = accuracyText;
        end


        function PlotAmpFreqz(app)
                % 幅频响应
                % 显示低通滤波器响应
                plot(app.LowpassResponseAxes, app.combinedF, 20*log10(abs(app.lpH)), 'b', 'LineWidth', 2);
                app.LowpassResponseAxes.Title.String = sprintf('低通滤波器');
                app.LowpassResponseAxes.XLabel.String = '频率/Hz';
                app.LowpassResponseAxes.YLabel.String = '幅度/dB';
                app.LowpassResponseAxes.XGrid = 'on';
                app.LowpassResponseAxes.YGrid = 'on';
                xlim(app.LowpassResponseAxes, [0 100]);
                ylim(app.LowpassResponseAxes, 'auto')
                yticks(app.LowpassResponseAxes, 'auto')
                yticklabels(app.LowpassResponseAxes, 'auto')

                % 显示高通滤波器响应
                plot(app.HighpassResponseAxes, app.combinedF, 20*log10(abs(app.hpH)), 'b', 'LineWidth', 2);
                app.HighpassResponseAxes.Title.String = sprintf('高通滤波器');
                app.HighpassResponseAxes.XLabel.String = '频率/Hz';
                app.HighpassResponseAxes.YLabel.String = '幅度/dB';
                app.HighpassResponseAxes.XGrid = 'on';
                app.HighpassResponseAxes.YGrid = 'on';
                xlim(app.HighpassResponseAxes, [0 100]);
                ylim(app.HighpassResponseAxes, 'auto')
                yticks(app.HighpassResponseAxes, 'auto')
                yticklabels(app.HighpassResponseAxes, 'auto')
```

```
471.
472.                 % 显示组合滤波器响应
473.                 plot(app.CombinedResponseAxes, app.combinedF, 20*log10(abs(app.combine
    dH)), 'b', 'LineWidth', 2);
474.                 app.CombinedResponseAxes.XLabel.String = '频率/Hz';
475.                 app.CombinedResponseAxes.YLabel.String = '幅度/dB';
476.                 app.CombinedResponseAxes.XGrid = 'on';
477.                 app.CombinedResponseAxes.YGrid = 'on';
478.                 xlim(app.CombinedResponseAxes, [0 100]);
479.                 ylim(app.CombinedResponseAxes, 'auto')
480.                 yticks(app.CombinedResponseAxes, 'auto')
481.                 yticklabels(app.CombinedResponseAxes, 'auto')
482.         end
483.         function PlotPhasedFreqz(app)
484.                 % 相频响应
485.                 % 显示低通滤波器响应
486.                 plot(app.LowpassResponseAxes, app.combinedF, angle(app.lpH), 'b', 'Lin
    eWidth', 2);
487.                 app.LowpassResponseAxes.Title.String = sprintf('低通滤波器');
488.                 app.LowpassResponseAxes.XLabel.String = '频率/Hz';
489.                 app.LowpassResponseAxes.YLabel.String = '相位/rad';
490.                 app.LowpassResponseAxes.XGrid = 'on';
491.                 app.LowpassResponseAxes.YGrid = 'on';
492.                 xlim(app.LowpassResponseAxes, [0 100]);
493.                 ylim(app.LowpassResponseAxes, [-pi pi]);
494.                 yticks(app.LowpassResponseAxes, -pi:pi/2:pi);
495.                 yticklabels(app.LowpassResponseAxes, {'-\pi','-\pi/2','0','\pi/2','\pi
    '});
496.
497.                 % 显示高通滤波器响应
498.                 plot(app.HighpassResponseAxes, app.combinedF, angle(app.hpH), 'b', 'Li
    neWidth', 2);
499.                 app.HighpassResponseAxes.Title.String = sprintf('高通滤波器');
500.                 app.HighpassResponseAxes.XLabel.String = '频率/Hz';
501.                 app.HighpassResponseAxes.YLabel.String = '相位/rad';
502.                 app.HighpassResponseAxes.XGrid = 'on';
503.                 app.HighpassResponseAxes.YGrid = 'on';
504.                 xlim(app.HighpassResponseAxes, [0 100]);
505.                 ylim(app.HighpassResponseAxes, [-pi pi]);
506.                 yticks(app.HighpassResponseAxes, -pi:pi/2:pi);
507.                 yticklabels(app.HighpassResponseAxes, {'-\pi','-\pi/2','0','\pi/2','\p
    i'});
508.
509.                 % 显示组合滤波器响应
```

```
510.                 plot(app.CombinedResponseAxes, app.combinedF, angle(app.combinedH), 'b
    ', 'LineWidth', 2);
511.                 app.CombinedResponseAxes.XLabel.String = '频率/Hz';
512.                 app.CombinedResponseAxes.YLabel.String = '相位/rad';
513.                 app.CombinedResponseAxes.XGrid = 'on';
514.                 app.CombinedResponseAxes.YGrid = 'on';
515.                 xlim(app.CombinedResponseAxes, [0 100]);
516.                 ylim(app.CombinedResponseAxes, [-pi pi]);
517.                 yticks(app.CombinedResponseAxes, -pi:pi/2:pi);
518.                 yticklabels(app.CombinedResponseAxes, {'-\pi','-\pi/2','0','\pi/2','\p
    i'});
519.
520.         end
521.     end
522.
523.     % Callbacks that handle component events
524.     methods (Access = private)
525.
526.         % Button pushed function: LoadDataButton
527.         function LoadDataButtonPushed(app, event)
528.             % 加载数据函数 - 一次性导入两个文件
529.             try
530.                 app.StatusLabel.Text = '正在加载数据...';
531.                 app.ProgressLamp.Color = 'yellow';
532.                 drawnow;
533.
534.                 % 选择文件夹路径
535.                 folderPath = uigetdir('', '选择包含 eeg.mat 和 label.mat 文件的文件夹');
536.                 if folderPath == 0
537.                     app.StatusLabel.Text = '数据加载已取消';
538.                     app.ProgressLamp.Color = [0.8 0.8 0.8];
539.                     return;
540.                 end
541.
542.                 % 自动查找 eeg.mat 文件
543.                 eegFiles = dir(fullfile(folderPath, 'eeg.mat'));
544.                 if isempty(eegFiles)
545.                     uialert(app.UIFigure, '在选择的文件夹中未找到 eeg.mat 文件', '文件未找
    到');
546.                     return;
547.                 end
548.
549.                 % 加载脑电数据
550.                 eegData = load(fullfile(folderPath, 'eeg.mat'));
551.
```

```
552.                 % 检查数据格式
553.                 if ~isfield(eegData, 'x')
554.                     uialert(app.UIFigure, 'eeg.mat 文件必须包含名为"x"的变量', '数据格式
    错误');
555.                     return;
556.                 end
557.
558.                 app.RawData = eegData.x;
559.
560.                 % 检查维度
561.                 if ndims(app.RawData) ~= 3
562.                     uialert(app.UIFigure, '数据应为三维数组(通道×采样点×试次)', '维度错误
    ');
563.                     return;
564.                 end
565.
566.                 % 自动查找 label.mat 文件
567.                 labelFiles = dir(fullfile(folderPath, 'label.mat'));
568.                 if ~isempty(labelFiles)
569.                     % 加载标签文件
570.                     labelData = load(fullfile(folderPath, 'label.mat'));
571.                     app.Labels = labelData.x_label;
572.                 else
573.                     % 如果没有标签文件，创建默认标签
574.                     app.Labels = repmat([1:10, 1:10], 1)';
575.                 end
576.
577.                 app.Labels = app.Labels(:);
578.
579.                 % 更新数据显示
580.                 [numChannels, numSamples, numTrials] = size(app.RawData);
581.                 app.DataInfoLabel.Text = sprintf('数据已加载: %d 通道 × %d 采样点 × %d 试
    次', ...
582.                     numChannels, numSamples, numTrials);
583.
584.                 % 更新通道选择器
585.                 app.ChannelSelectorDropDown.Items = arrayfun(@(x) sprintf('通
    道 %d', x), 1:numChannels, 'UniformOutput', false);
586.                 app.ChannelSelectorDropDown.Value = '通道 1';
587.
588.                 % 更新试次选择器
589.                 app.TrialSelectorDropDown.Items = arrayfun(@(x) sprintf('试
    次 %d', x), 1:numTrials, 'UniformOutput', false);
590.                 app.TrialSelectorDropDown.Value = '试次 1';
591.
```

```
592.                 % 更新刺激频率显示
593.                 freqText = sprintf('目标 1: %.1f Hz\n 目标 2: %.1f Hz\n 目标 3: %.1f Hz\n
    目标 4: %.1f Hz\n 目标 5: %.1f Hz\n 目标 6: %.1f Hz\n 目标 7: %.1f Hz\n 目标 8: %.1f Hz\n 目标
    9: %.1f Hz\n 目标 10: %.1f Hz', ...
594.                     app.StimFreqs);
595.                 app.StimulusFrequenciesText.Value = freqText;
596.
597.                 app.StatusLabel.Text = '数据加载完成';
598.                 app.ProgressLamp.Color = 'green';
599.
600.             catch ME
601.                 uialert(app.UIFigure, ME.message, '数据加载错误');
602.                 app.StatusLabel.Text = '数据加载失败';
603.                 app.ProgressLamp.Color = 'red';
604.             end
605.         end
606.
607.         % Button pushed function: DesignFilterButton
608.         function DesignFilterButtonPushed(app, event)
609.             % 设计巴特沃斯 IIR 滤波器（先低通后高通）
610.             try
611.                 app.StatusLabel.Text = '正在设计滤波器...';
612.                 app.ProgressLamp.Color = 'yellow';
613.                 drawnow;
614.
615.                 % 获取滤波器参数 - 分别设置低通和高通参数
616.                 app.LowpassCutoff = app.LowpassCutoffEditField.Value;
617.                 app.LowpassPassRipple = app.LowpassPassRippleEditField.Value;
618.                 app.LowpassStopAtten = app.LowpassStopAttenEditField.Value;
619.
620.                 app.HighpassCutoff = app.HighpassCutoffEditField.Value;
621.                 app.HighpassPassRipple = app.HighpassPassRippleEditField.Value;
622.                 app.HighpassStopAtten = app.HighpassStopAttenEditField.Value;
623.
624.                 % 归一化频率
625.                 nyquist = app.Fs / 2;
626.                 lpWp = app.LowpassCutoff / nyquist;
627.                 lpWs = (app.LowpassCutoff + 5) / nyquist; % 阻带频率比通带高 5Hz
628.
629.                 hpWp = app.HighpassCutoff / nyquist;
630.                 hpWs = (app.HighpassCutoff - 1) / nyquist; % 阻带频率比通带低 1Hz
631.
632.                 % 设计低通滤波器
633.                 [lpN, lpWn] = buttord(lpWp, lpWs, app.LowpassPassRipple, app.LowpassSt
    opAtten);
```

```
634.                 [lpB, lpA] = butter(lpN, lpWn, 'low');
635.
636.                 % 设计高通滤波器
637.                 [hpN, hpWn] = buttord(hpWp, hpWs, app.HighpassPassRipple, app.Highpass
    StopAtten);
638.                 [hpB, hpA] = butter(hpN, hpWn, 'high');
639.
640.                 % 保存滤波器系数
641.                 app.lpCoeff.b = lpB;
642.                 app.lpCoeff.a = lpA;
643.                 app.lpCoeff.order = lpN;
644.
645.                 app.hpCoeff.b = hpB;
646.                 app.hpCoeff.a = hpA;
647.                 app.hpCoeff.order = hpN;
648.
649.                 % 计算频率响应
650.                 N = 1024;
651.                 [app.lpH, lpF] = freqz(lpB, lpA, N, app.Fs);
652.                 [app.hpH, ~] = freqz(hpB, hpA, N, app.Fs);
653.                 app.combinedF = lpF;
654.
655.                 % 计算组合响应
656.                 app.combinedH = app.lpH .* app.hpH;
657.
658.                 % 绘制频率响应
659.                 switch app.FreqzDropDown.Value
660.                     case '幅频响应'
661.                         PlotAmpFreqz(app)
662.                     case '相频响应'
663.                         PlotPhasedFreqz(app)
664.                     otherwise
665.                         PlotAmpFreqz(app)
666.                 end
667.
668.                 % 显示滤波器阶数信息
669.                 app.FilterOrderInfoLabel.Text = sprintf('滤波器阶数 - 低通: %d ,高
    通: %d', lpN ,hpN);
670.                 app.StatusLabel.Text = '滤波器设计完成';
671.                 app.ProgressLamp.Color = 'green';
672.
673.             catch ME
674.                 uialert(app.UIFigure, ME.message, '滤波器设计错误');
675.                 app.StatusLabel.Text = '滤波器设计失败';
676.                 app.ProgressLamp.Color = 'red';
```

```
            end
        end

        % Button pushed function: RunAnalysisButton
        function RunAnalysisButtonPushed(app, event)
            % 处理单个试次的频谱分析，同时显示时域波形
            try
                if isempty(app.RawData)
                    uialert(app.UIFigure, '请先加载数据', '数据未加载');
                    return;
                end

                if isempty(app.lpCoeff) || isempty(app.hpCoeff)
                    uialert(app.UIFigure, '请先设计滤波器', '滤波器未设计');
                    return;
                end

                app.StatusLabel.Text = '正在进行频谱分析...';
                app.ProgressLamp.Color = 'yellow';
                drawnow;

                % 获取选中的通道和试次
                selectedChannel = str2double(app.ChannelSelectorDropDown.Value(end));

                if length(app.TrialSelectorDropDown.Value) == 3
                    selectedTrial = str2double(app.TrialSelectorDropDown.Value(end));

                else
                    selectedTrial = str2double(app.TrialSelectorDropDown.Value(end-1:e
    nd));
                end

                % 提取原始信号
                rawSignal = squeeze(app.RawData(selectedChannel, :, selectedTrial));

                % 应用级联滤波
                filteredSignal = app.applyCascadeFilter(rawSignal);

                % 保存滤波后数据
                app.FilteredData(selectedChannel, :, selectedTrial) = filteredSignal;

                % 创建时间轴
                timeAxis = (0:length(rawSignal)-1) / app.Fs;
```

```
718.                % ===== 显示时域波形 =====
719.                plot(app.TimeDomainAxes, timeAxis, filteredSignal, 'b', 'LineWidth', 1
    .5);
720.                app.TimeDomainAxes.Title.String = sprintf('试次 %d - 通道 %d 时域波形
    ', ...
721.                    selectedTrial, selectedChannel);
722.                app.TimeDomainAxes.XLabel.String = '时间 (s)';
723.                app.TimeDomainAxes.YLabel.String = '幅度';
724.                app.TimeDomainAxes.XGrid = 'on';
725.                app.TimeDomainAxes.YGrid = 'on';
726.                legend(app.TimeDomainAxes, '滤波后信号', 'Location', 'northeast');
727.
728.                % 计算 FFT 频谱
729.                [spectrum, freq] = app.computeFFTSpectrum(filteredSignal);
730.
731.                % ===== 基频检测 =====
732.                [baseTargetIdx, basePeakFreq] = app.detectFundamentalFrequency(spectru
    m, freq);
733.
734.                % ===== 倍频检测 =====
735.                if app.ShowHarmonicCheckBox.Value && basePeakFreq > 0
736.                    [harmonicTargetIdx, harmonicPeakFreq] = app.detectHarmonicFrequenc
    y(spectrum, freq, spectrum(freq == basePeakFreq));
737.                end
738.
739.                % 显示频谱图
740.                plot(app.SpectrumAxes, freq, spectrum, 'b', 'LineWidth', 1.5);
741.                hold(app.SpectrumAxes, 'on');
742.
743.                % 标记 7-15Hz 范围
744.                fill(app.SpectrumAxes, [7 15 15 7], [0 0 max(spectrum)*1.1 max(spectru
    m)*1.1], ...
745.                    'y', 'FaceAlpha', 0.2, 'EdgeColor', 'none');
746.
747.                if basePeakFreq > 0
748.                    % 标记基频峰值
749.                    plot(app.SpectrumAxes, basePeakFreq, spectrum(freq == basePeakFreq
    ), 'ro', ...s
750.                        'MarkerSize', 5, 'LineWidth', 2);
751.                    text(app.SpectrumAxes, basePeakFreq, spectrum(freq == basePeakFreq
    )*1.05, ...
752.                        sprintf('基频: %.1f Hz', basePeakFreq), ...
753.                        'HorizontalAlignment', 'center', 'FontSize', 10);
754.                    % 保存峰值频率
```

```
755.                     app.PeakFrequencies(selectedTrial, selectedChannel) = basePeakFreq
    ;
756.
757.                     % 显示基频检测信息
758.                     peakText = sprintf('基频检测:\n 主峰值频率: %.2f Hz\n 最接近刺激频
    率: %.1f Hz (目标%d)\n 频率差: %.2f Hz\n', ...
759.                         basePeakFreq, app.StimFreqs(baseTargetIdx), baseTargetIdx, abs
    (basePeakFreq - app.StimFreqs(baseTargetIdx)));
760.
761.                     % 显示倍频检测信息
762.                     if app.ShowHarmonicCheckBox.Value
763.                         % 标记 15-30Hz 范围
764.                         fill(app.SpectrumAxes, [15 30 30 15], [0 0 max(spectrum)*1.1 m
    ax(spectrum)*1.1], ...
765.                             'b', 'FaceAlpha', 0.2, 'EdgeColor', 'none');
766.                         if harmonicPeakFreq > 0
767.                             % 标记倍频峰值
768.                             plot(app.SpectrumAxes, harmonicPeakFreq, spectrum(freq ==
    harmonicPeakFreq), 'go', ...
769.                                 'MarkerSize', 5, 'LineWidth', 2);
770.                             text(app.SpectrumAxes, harmonicPeakFreq, spectrum(freq ==
    harmonicPeakFreq)*1.05, ...
771.                                 sprintf('2 倍频: %.1f Hz', harmonicPeakFreq), ...
772.                                 'HorizontalAlignment', 'center', 'FontSize', 10);
773.
774.                             peakText = sprintf('%s\n 倍频检测:\n2 倍频峰值: %.2f Hz -> 基
    频%.2f Hz -> 目标%d (%.1f Hz)\n', ...
775.                                 peakText, harmonicPeakFreq, harmonicPeakFreq/2, harmon
    icTargetIdx, app.StimFreqs(harmonicTargetIdx));
776.                         else
777.                             peakText = sprintf('%s\n 倍频检测:\n 在 15-30Hz 范围内未检测到
    有效峰值\n', peakText);
778.                         end
779.                     end
780.                 else
781.                     peakText = '在 7-15Hz 范围内未检测到明显峰值';
782.                 end
783.
784.                 app.PeakFrequencyText.Value = peakText;
785.
786.                 % 设置图形属性
787.                 app.SpectrumAxes.Title.String = sprintf('试次 %d - 通道 %d 频谱图
    ', ...
788.                     selectedTrial, selectedChannel);
```

```
789.                 app.SpectrumAxes.XLabel.String = '频率 (Hz)';
790.                 app.SpectrumAxes.YLabel.String = '幅度';
791.                 app.SpectrumAxes.XGrid = 'on';
792.                 app.SpectrumAxes.YGrid = 'on';
793.                 xlim(app.SpectrumAxes, [0 50]);
794.
795.                 % 设置图例
796.                 if app.ShowHarmonicCheckBox.Value && harmonicPeakFreq > 0
797.                     legend(app.SpectrumAxes, {'频谱', '7-15Hz 范围', '基频峰值', '2 倍频峰值', '15-30Hz 范围'}, 'Location', 'northeast');
798.                 else
799.                     legend(app.SpectrumAxes, {'频谱', '7-15Hz 范围', '基频峰值'}, 'Location', 'northeast');
800.                 end
801.
802.                 hold(app.SpectrumAxes, 'off');
803.
804.                 app.StatusLabel.Text = sprintf('频谱分析完成 - 试次 %d, 通道 %d', ...
805.                     selectedTrial, selectedChannel);
806.                 app.ProgressLamp.Color = 'green';
807.
808.             catch ME
809.                 uialert(app.UIFigure, ME.message, '频谱分析错误');
810.                 app.StatusLabel.Text = '频谱分析失败';
811.                 app.ProgressLamp.Color = 'red';
812.             end
813.         end
814.
815.         % Button pushed function: RunClassificationButton
816.         function RunClassificationButtonPushed(app, event)
817.             % 对所有试次进行分类
818.             try
819.                 if isempty(app.RawData)
820.                     uialert(app.UIFigure, '请先加载数据', '数据未加载');
821.                     return;
822.                 end
823.
824.                 if isempty(app.lpCoeff) || isempty(app.hpCoeff)
825.                     uialert(app.UIFigure, '请先设计滤波器', '滤波器未设计');
826.                     return;
827.                 end
828.
829.                 app.StatusLabel.Text = '正在进行目标分类...';
830.                 app.ProgressLamp.Color = 'yellow';
831.                 drawnow;
```

```
832.
833.                 % 获取数据维度
834.                 [numChannels, ~, numTrials] = size(app.RawData);
835.
836.                 % 初始化结果存储
837.                 app.ChannelPredictions = zeros(numTrials, numChannels);
838.                 app.ChannelHarmonicPredictions = zeros(numTrials, numChannels);
839.                 app.JointPredictions = zeros(numTrials, 1);
840.                 app.HarmonicPredictions = zeros(numTrials, 1);
841.                 app.PeakFrequencies = zeros(numTrials, numChannels);
842.                 app.HarmonicPeakFrequencies = zeros(numTrials, numChannels);
843.
844.                 % 处理每个试次
845.                 for trial = 1:numTrials
846.                     channelVotes = zeros(1, length(app.StimFreqs));
847.                     channelVotesHarmonic = zeros(1, length(app.StimFreqs));
848.
849.                     for channel = 1:numChannels
850.                         % 提取信号并滤波
851.                         signal = squeeze(app.RawData(channel, :, trial));
852.                         filteredSignal = app.applyCascadeFilter(signal);
853.
854.                         % 计算 FFT 频谱
855.                         [spectrum, freq] = app.computeFFTSpectrum(filteredSignal);
856.
857.                         % ===== 基频检测 =====
858.                         [baseTargetIdx, basePeakFreq] = app.detectFundamentalFrequency
    (spectrum, freq);
859.
860.                         if basePeakFreq > 0
861.                             app.PeakFrequencies(trial, channel) = basePeakFreq;
862.                             app.ChannelPredictions(trial, channel) = baseTargetIdx;
863.                             channelVotes(baseTargetIdx) = channelVotes(baseTargetIdx)
    + 1;
864.
865.                             % ===== 倍频检测 =====
866.                             [harmonicTargetIdx, harmonicPeakFreq] = app.detectHarmonic
    Frequency(spectrum, freq, spectrum(freq == basePeakFreq));
867.
868.                             if harmonicPeakFreq > 0
869.                                 app.HarmonicPeakFrequencies(trial, channel) = harmonic
    PeakFreq;
870.                                 app.ChannelHarmonicPredictions(trial, channel) = harmo
    nicTargetIdx;
```

```
                                channelVotesHarmonic(harmonicTargetIdx) = channelVotes
Harmonic(harmonicTargetIdx) + 1;
                            else
                                app.ChannelHarmonicPredictions(trial, channel) = 0;
                            end
                        else
                            app.ChannelPredictions(trial, channel) = 0;
                            app.ChannelHarmonicPredictions(trial, channel) = 0;
                        end
                    end

                    % 基频检测-8通道联合投票
                    [~, app.JointPredictions(trial)] = max(channelVotes);

                    % 基频+倍频-16通道联合投票
                    combinedVotes = channelVotes + channelVotesHarmonic;
                    [~, app.HarmonicPredictions(trial)] = max(combinedVotes);
                end

                % 计算准确率
                app.ChannelAccuracies = zeros(1, numChannels);
                app.ChannelHarmonicAccuracies = zeros(1, numChannels);

                for channel = 1:numChannels
                    % 基频检测准确率
                    correctBase = sum(app.ChannelPredictions(:, channel) == app.Labels
);
                    app.ChannelAccuracies(channel) = correctBase / numTrials * 100;

                    % 倍频检测准确率
                    correctHarmonic = sum(app.ChannelHarmonicPredictions(:, channel) =
= app.Labels);
                    app.ChannelHarmonicAccuracies(channel) = correctHarmonic / numTria
ls * 100;
                end

                app.JointAccuracy = sum(app.JointPredictions == app.Labels) / numTrial
s * 100;
                app.HarmonicAccuracy = sum(app.HarmonicPredictions == app.Labels) / nu
mTrials * 100;

                % 显示结果
                app.updateResultsDisplay();

```

```
909.                 app.StatusLabel.Text = sprintf('目标分类完成 - 总准确
    率: %.2f%%', app.JointAccuracy);
910.                 app.ProgressLamp.Color = 'green';
911.
912.             catch ME
913.                 uialert(app.UIFigure, ME.message, '分类错误');
914.                 app.StatusLabel.Text = '分类失败';
915.                 app.ProgressLamp.Color = 'red';
916.             end
917.         end
918.
919.         % Value changed function: FreqzDropDown
920.         function FreqzValueChanged(app, event)
921.             % 绘制频率响应
922.             switch app.FreqzDropDown.Value
923.                 case '幅频响应'
924.                     PlotAmpFreqz(app)
925.                 case '相频响应'
926.                     PlotPhasedFreqz(app)
927.                 otherwise
928.                     PlotAmpFreqz(app)
929.             end
930.         end
931.     end
932.
933.     % Component initialization
934.     methods (Access = private)
935.
936.         % Create UIFigure and components
937.         function createComponents(app)
938.
939.             % Create UIFigure and hide until all components are created
940.             app.UIFigure = uifigure('Visible', 'off');
941.             app.UIFigure.Position = [100 100 1400 800];
942.             app.UIFigure.Name = '脑电 SSVEP 信号的目标分类';
943.
944.             % Create TabGroup
945.             app.TabGroup = uitabgroup(app.UIFigure);
946.             app.TabGroup.Position = [20 20 1360 760];
947.
948.             % Create DataFilterTab
949.             app.DataFilterTab = uitab(app.TabGroup);
950.             app.DataFilterTab.Title = '滤波器设计';
951.
952.             % Create LoadDataButton
```

```
953.             app.LoadDataButton = uibutton(app.DataFilterTab, 'push');
954.             app.LoadDataButton.ButtonPushedFcn = createCallbackFcn(app, @LoadDataButto
    nPushed, true);
955.             app.LoadDataButton.FontSize = 14;
956.             app.LoadDataButton.FontWeight = 'bold';
957.             app.LoadDataButton.Position = [30 700 120 30];
958.             app.LoadDataButton.Text = '加载数据';
959. 
960.             % Create DataInfoLabel
961.             app.DataInfoLabel = uilabel(app.DataFilterTab);
962.             app.DataInfoLabel.Position = [170 700 400 30];
963.             app.DataInfoLabel.Text = '未加载数据';
964. 
965.             % Create FilterParametersPanel
966.             app.FilterParametersPanel = uipanel(app.DataFilterTab);
967.             app.FilterParametersPanel.Title = '滤波器参数';
968.             app.FilterParametersPanel.FontWeight = 'bold';
969.             app.FilterParametersPanel.FontSize = 15;
970.             app.FilterParametersPanel.Position = [30 520 1300 160];
971. 
972.             % Create LowpassParametersLabel
973.             app.LowpassParametersLabel = uilabel(app.FilterParametersPanel);
974.             app.LowpassParametersLabel.FontWeight = 'bold';
975.             app.LowpassParametersLabel.Position = [20 107 120 22];
976.             app.LowpassParametersLabel.Text = '低通滤波器参数';
977. 
978.             % Create LowpassPassRippleLabel
979.             app.LowpassPassRippleLabel = uilabel(app.FilterParametersPanel);
980.             app.LowpassPassRippleLabel.HorizontalAlignment = 'right';
981.             app.LowpassPassRippleLabel.Position = [190 77 80 22];
982.             app.LowpassPassRippleLabel.Text = '通带波纹/dB';
983. 
984.             % Create LowpassPassRippleEditField
985.             app.LowpassPassRippleEditField = uieditfield(app.FilterParametersPanel, 'n
    umeric');
986.             app.LowpassPassRippleEditField.Limits = [0.1 10];
987.             app.LowpassPassRippleEditField.HorizontalAlignment = 'center';
988.             app.LowpassPassRippleEditField.Position = [280 77 60 22];
989.             app.LowpassPassRippleEditField.Value = 7;
990. 
991.             % Create LowpassCutoffLabel
992.             app.LowpassCutoffLabel = uilabel(app.FilterParametersPanel);
993.             app.LowpassCutoffLabel.HorizontalAlignment = 'right';
994.             app.LowpassCutoffLabel.Position = [20 77 80 22];
```

```
995.             app.LowpassCutoffLabel.Text = '截止频率/Hz';
996.
997.             % Create LowpassCutoffEditField
998.             app.LowpassCutoffEditField = uieditfield(app.FilterParametersPanel, 'numer
    ic');
999.             app.LowpassCutoffEditField.Limits = [10 200];
1000.                app.LowpassCutoffEditField.HorizontalAlignment = 'center';
1001.                app.LowpassCutoffEditField.Position = [110 77 60 22];
1002.                app.LowpassCutoffEditField.Value = 40;
1003.
1004.                % Create LowpassStopAttenLabel
1005.                app.LowpassStopAttenLabel = uilabel(app.FilterParametersPanel);
1006.                app.LowpassStopAttenLabel.HorizontalAlignment = 'right';
1007.                app.LowpassStopAttenLabel.Position = [360 77 80 22];
1008.                app.LowpassStopAttenLabel.Text = '阻带衰减/dB';
1009.
1010.                % Create LowpassStopAttenEditField
1011.                app.LowpassStopAttenEditField = uieditfield(app.FilterParametersPanel,
     'numeric');
1012.                app.LowpassStopAttenEditField.Limits = [5 100];
1013.                app.LowpassStopAttenEditField.HorizontalAlignment = 'center';
1014.                app.LowpassStopAttenEditField.Position = [450 77 60 22];
1015.                app.LowpassStopAttenEditField.Value = 20;
1016.
1017.                % Create HighpassParametersLabel
1018.                app.HighpassParametersLabel = uilabel(app.FilterParametersPanel);
1019.                app.HighpassParametersLabel.FontWeight = 'bold';
1020.                app.HighpassParametersLabel.Position = [600 107 120 22];
1021.                app.HighpassParametersLabel.Text = '高通滤波器参数';
1022.
1023.                % Create HighpassPassRippleLabel
1024.                app.HighpassPassRippleLabel = uilabel(app.FilterParametersPanel);
1025.                app.HighpassPassRippleLabel.HorizontalAlignment = 'right';
1026.                app.HighpassPassRippleLabel.Position = [770 77 80 22];
1027.                app.HighpassPassRippleLabel.Text = '通带波纹/dB';
1028.
1029.                % Create HighpassPassRippleEditField
1030.                app.HighpassPassRippleEditField = uieditfield(app.FilterParametersPane
    l, 'numeric');
1031.                app.HighpassPassRippleEditField.Limits = [0.1 10];
1032.                app.HighpassPassRippleEditField.Position = [860 77 60 22];
1033.                app.HighpassPassRippleEditField.Value = 7;
1034.
1035.                % Create HighpassCutoffLabel
1036.                app.HighpassCutoffLabel = uilabel(app.FilterParametersPanel);
```

```
1037.                 app.HighpassCutoffLabel.HorizontalAlignment = 'right';
1038.                 app.HighpassCutoffLabel.Position = [600 77 80 22];
1039.                 app.HighpassCutoffLabel.Text = '截止频率/Hz';
1040. 
1041.                 % Create HighpassCutoffEditField
1042.                 app.HighpassCutoffEditField = uieditfield(app.FilterParametersPanel, '
    numeric');
1043.                 app.HighpassCutoffEditField.Limits = [0.1 20];
1044.                 app.HighpassCutoffEditField.Position = [690 77 60 22];
1045.                 app.HighpassCutoffEditField.Value = 5;
1046. 
1047.                 % Create HighpassStopAttenLabel
1048.                 app.HighpassStopAttenLabel = uilabel(app.FilterParametersPanel);
1049.                 app.HighpassStopAttenLabel.HorizontalAlignment = 'right';
1050.                 app.HighpassStopAttenLabel.Position = [940 77 80 22];
1051.                 app.HighpassStopAttenLabel.Text = '阻带衰减/dB';
1052. 
1053.                 % Create HighpassStopAttenEditField
1054.                 app.HighpassStopAttenEditField = uieditfield(app.FilterParametersPanel
    , 'numeric');
1055.                 app.HighpassStopAttenEditField.Limits = [5 100];
1056.                 app.HighpassStopAttenEditField.Position = [1030 77 60 22];
1057.                 app.HighpassStopAttenEditField.Value = 20;
1058. 
1059.                 % Create DesignFilterButton
1060.                 app.DesignFilterButton = uibutton(app.FilterParametersPanel, 'push');

1061.                 app.DesignFilterButton.ButtonPushedFcn = createCallbackFcn(app, @Desig
    nFilterButtonPushed, true);
1062.                 app.DesignFilterButton.FontSize = 14;
1063.                 app.DesignFilterButton.FontWeight = 'bold';
1064.                 app.DesignFilterButton.Position = [580 17 150 30];
1065.                 app.DesignFilterButton.Text = '设计滤波器';
1066. 
1067.                 % Create FilterResponsePanel
1068.                 app.FilterResponsePanel = uipanel(app.DataFilterTab);
1069.                 app.FilterResponsePanel.Title = '滤波器频率响应';
1070.                 app.FilterResponsePanel.FontWeight = 'bold';
1071.                 app.FilterResponsePanel.FontSize = 15;
1072.                 app.FilterResponsePanel.Position = [30 50 1300 420];
1073. 
1074.                 % Create HighpassResponseAxes
1075.                 app.HighpassResponseAxes = uiaxes(app.FilterResponsePanel);
1076.                 title(app.HighpassResponseAxes, '高通滤波器响应')
```

```
1077.                 xlabel(app.HighpassResponseAxes, '频率/Hz')
1078.                 ylabel(app.HighpassResponseAxes, '幅度/dB')
1079.                 zlabel(app.HighpassResponseAxes, 'Z')
1080.                 app.HighpassResponseAxes.Box = 'on';
1081.                 app.HighpassResponseAxes.Position = [450 47 400 300];
1082. 
1083.                 % Create LowpassResponseAxes
1084.                 app.LowpassResponseAxes = uiaxes(app.FilterResponsePanel);
1085.                 title(app.LowpassResponseAxes, '低通滤波器响应')
1086.                 xlabel(app.LowpassResponseAxes, '频率/Hz')
1087.                 ylabel(app.LowpassResponseAxes, '幅度/dB')
1088.                 zlabel(app.LowpassResponseAxes, 'Z')
1089.                 app.LowpassResponseAxes.Box = 'on';
1090.                 app.LowpassResponseAxes.Position = [20 47 400 300];
1091. 
1092.                 % Create CombinedResponseAxes
1093.                 app.CombinedResponseAxes = uiaxes(app.FilterResponsePanel);
1094.                 title(app.CombinedResponseAxes, '组合滤波器响应')
1095.                 xlabel(app.CombinedResponseAxes, '频率/Hz')
1096.                 ylabel(app.CombinedResponseAxes, '幅度/dB')
1097.                 zlabel(app.CombinedResponseAxes, 'Z')
1098.                 app.CombinedResponseAxes.Box = 'on';
1099.                 app.CombinedResponseAxes.Position = [880 47 400 300];
1100. 
1101.                 % Create FreqzLabel
1102.                 app.FreqzLabel = uilabel(app.FilterResponsePanel);
1103.                 app.FreqzLabel.FontWeight = 'bold';
1104.                 app.FreqzLabel.Position = [20 358 77 22];
1105.                 app.FreqzLabel.Text = '频率响应类型';
1106. 
1107.                 % Create FreqzDropDown
1108.                 app.FreqzDropDown = uidropdown(app.FilterResponsePanel);
1109.                 app.FreqzDropDown.Items = {'幅频响应', '相频响应'};
1110.                 app.FreqzDropDown.ValueChangedFcn = createCallbackFcn(app, @FreqzValue
    Changed, true);
1111.                 app.FreqzDropDown.Position = [100 358 100 22];
1112.                 app.FreqzDropDown.Value = '幅频响应';
1113. 
1114.                 % Create FilterOrderInfoLabel
1115.                 app.FilterOrderInfoLabel = uilabel(app.DataFilterTab);
1116.                 app.FilterOrderInfoLabel.Position = [30 490 1200 22];
1117.                 app.FilterOrderInfoLabel.Text = '滤波器阶数: 未设计';
1118. 
1119.                 % Create AnalysisTab
1120.                 app.AnalysisTab = uitab(app.TabGroup);
```

```
1121.                 app.AnalysisTab.Title = '频谱分析';
1122.
1123.                 % Create TimeDomainAxes
1124.                 app.TimeDomainAxes = uiaxes(app.AnalysisTab);
1125.                 title(app.TimeDomainAxes, '时域波形')
1126.                 xlabel(app.TimeDomainAxes, '时间/s')
1127.                 ylabel(app.TimeDomainAxes, '幅度')
1128.                 zlabel(app.TimeDomainAxes, 'Z')
1129.                 app.TimeDomainAxes.Box = 'on';
1130.                 app.TimeDomainAxes.Position = [30 400 900 250];
1131.
1132.                 % Create SpectrumAxes
1133.                 app.SpectrumAxes = uiaxes(app.AnalysisTab);
1134.                 title(app.SpectrumAxes, '频谱图')
1135.                 xlabel(app.SpectrumAxes, '频率/Hz')
1136.                 ylabel(app.SpectrumAxes, '幅度')
1137.                 zlabel(app.SpectrumAxes, 'Z')
1138.                 app.SpectrumAxes.Box = 'on';
1139.                 app.SpectrumAxes.Position = [30 100 900 250];
1140.
1141.                 % Create ChannelSelectorLabel
1142.                 app.ChannelSelectorLabel = uilabel(app.AnalysisTab);
1143.                 app.ChannelSelectorLabel.HorizontalAlignment = 'right';
1144.                 app.ChannelSelectorLabel.FontWeight = 'bold';
1145.                 app.ChannelSelectorLabel.Position = [30 700 80 22];
1146.                 app.ChannelSelectorLabel.Text = '选择通道';
1147.
1148.                 % Create ChannelSelectorDropDown
1149.                 app.ChannelSelectorDropDown = uidropdown(app.AnalysisTab);
1150.                 app.ChannelSelectorDropDown.Items = {'通道 1', '通道 2', '通道 3', '通
    道 4', '通道 5', '通道 6', '通道 7', '通道 8'};
1151.                 app.ChannelSelectorDropDown.Position = [120 700 100 22];
1152.                 app.ChannelSelectorDropDown.Value = '通道 1';
1153.
1154.                 % Create TrialSelectorLabel
1155.                 app.TrialSelectorLabel = uilabel(app.AnalysisTab);
1156.                 app.TrialSelectorLabel.HorizontalAlignment = 'right';
1157.                 app.TrialSelectorLabel.FontWeight = 'bold';
1158.                 app.TrialSelectorLabel.Position = [250 700 80 22];
1159.                 app.TrialSelectorLabel.Text = '选择试次';
1160.
1161.                 % Create TrialSelectorDropDown
1162.                 app.TrialSelectorDropDown = uidropdown(app.AnalysisTab);
```

```
1163.                 app.TrialSelectorDropDown.Items = {'试次 1', '试次 2', '试次 3', '试
      次 4', '试次 5', '试次 6', '试次 7', '试次 8', '试次 9', '试次 10', '试次 11', '试
      次 12', '试次 13', '试次 14', '试次 15', '试次 16', '试次 17', '试次 18', '试次 19', '试
      次 20'};
1164.                 app.TrialSelectorDropDown.Position = [340 700 100 22];
1165.                 app.TrialSelectorDropDown.Value = '试次 1';
1166.
1167.                 % Create FFTLengthLabel
1168.                 app.FFTLengthLabel = uilabel(app.AnalysisTab);
1169.                 app.FFTLengthLabel.HorizontalAlignment = 'right';
1170.                 app.FFTLengthLabel.FontWeight = 'bold';
1171.                 app.FFTLengthLabel.Position = [470 700 80 22];
1172.                 app.FFTLengthLabel.Text = 'FFT 长度';
1173.
1174.                 % Create FFTEditField
1175.                 app.FFTEditField = uieditfield(app.AnalysisTab, 'numeric');
1176.                 app.FFTEditField.Limits = [4096 65536];
1177.                 app.FFTEditField.RoundFractionalValues = 'on';
1178.                 app.FFTEditField.ValueDisplayFormat = '%.0f';
1179.                 app.FFTEditField.HorizontalAlignment = 'center';
1180.                 app.FFTEditField.Position = [560 700 100 22];
1181.                 app.FFTEditField.Value = 4096;
1182.
1183.                 % Create WindowTypeLabel
1184.                 app.WindowTypeLabel = uilabel(app.AnalysisTab);
1185.                 app.WindowTypeLabel.HorizontalAlignment = 'right';
1186.                 app.WindowTypeLabel.FontWeight = 'bold';
1187.                 app.WindowTypeLabel.Position = [690 700 80 22];
1188.                 app.WindowTypeLabel.Text = '窗函数';
1189.
1190.                 % Create WindowTypeDropDown
1191.                 app.WindowTypeDropDown = uidropdown(app.AnalysisTab);
1192.                 app.WindowTypeDropDown.Items = {'矩形窗', '汉宁窗 ', '汉明窗', '布莱克曼
      窗'};
1193.                 app.WindowTypeDropDown.Position = [780 700 100 22];
1194.                 app.WindowTypeDropDown.Value = '汉明窗';
1195.
1196.                 % Create ShowHarmonicCheckBox
1197.                 app.ShowHarmonicCheckBox = uicheckbox(app.AnalysisTab);
1198.                 app.ShowHarmonicCheckBox.Text = '显示倍频检测';
1199.                 app.ShowHarmonicCheckBox.Position = [910 700 120 22];
1200.
1201.                 % Create RunAnalysisButton
1202.                 app.RunAnalysisButton = uibutton(app.AnalysisTab, 'push');
```

```
1203.                 app.RunAnalysisButton.ButtonPushedFcn = createCallbackFcn(app, @RunAna
    lysisButtonPushed, true);
1204.                 app.RunAnalysisButton.FontSize = 14;
1205.                 app.RunAnalysisButton.FontWeight = 'bold';
1206.                 app.RunAnalysisButton.Position = [1060 700 120 30];
1207.                 app.RunAnalysisButton.Text = '运行频谱分析';
1208.
1209.                 % Create StimulusFrequenciesLabel
1210.                 app.StimulusFrequenciesLabel = uilabel(app.AnalysisTab);
1211.                 app.StimulusFrequenciesLabel.Position = [950 650 150 22];
1212.                 app.StimulusFrequenciesLabel.Text = '刺激频率对应目标';
1213.
1214.                 % Create StimulusFrequenciesText
1215.                 app.StimulusFrequenciesText = uitextarea(app.AnalysisTab);
1216.                 app.StimulusFrequenciesText.Position = [950 400 300 230];
1217.                 app.StimulusFrequenciesText.Value = {'目标 1: 8.0 Hz'; '目标
    2: 8.5 Hz'; '目标 3: 9.0 Hz'; '目标 4: 9.5 Hz'; '目标 5: 10.0 Hz'; '目标 6: 10.5 Hz '; '目
    标 7: 11.0 Hz'; '目标 8: 11.5 Hz'; '目标 9: 12.0 Hz'; '目标 10: 12.5 Hz'};
1218.
1219.                 % Create PeakFrequencyLabel
1220.                 app.PeakFrequencyLabel = uilabel(app.AnalysisTab);
1221.                 app.PeakFrequencyLabel.Position = [950 370 150 22];
1222.                 app.PeakFrequencyLabel.Text = '峰值频率分析';
1223.
1224.                 % Create PeakFrequencyText
1225.                 app.PeakFrequencyText = uitextarea(app.AnalysisTab);
1226.                 app.PeakFrequencyText.Position = [950 50 300 300];
1227.
1228.                 % Create ResultsTab
1229.                 app.ResultsTab = uitab(app.TabGroup);
1230.                 app.ResultsTab.Title = '目标分类';
1231.
1232.                 % Create RunClassificationButton
1233.                 app.RunClassificationButton = uibutton(app.ResultsTab, 'push');
1234.                 app.RunClassificationButton.ButtonPushedFcn = createCallbackFcn(app, @
    RunClassificationButtonPushed, true);
1235.                 app.RunClassificationButton.FontSize = 14;
1236.                 app.RunClassificationButton.FontWeight = 'bold';
1237.                 app.RunClassificationButton.Position = [30 700 150 30];
1238.                 app.RunClassificationButton.Text = '运行目标分类';
1239.
1240.                 % Create ResultsTable
1241.                 app.ResultsTable = uitable(app.ResultsTab);
```

```
1242.                 app.ResultsTable.ColumnName = {'通道 1'; '通道 2'; '通道 3'; '通道 4'; '通
    道 5'; '通道 6'; '通道 7'; '通道 8'; '8 通道联合'; '基倍频联合'; '真实目标'; '分类结果'};
1243.                 app.ResultsTable.RowName = {'试次 1'; '试次 2'; '试次 3'; '试次 4'; '试次
    5'; '试次 6'; '试次 7'; '试次 8'; '试次 9'; '试次 10'; '试次 11'; '试次 12'; '试次 13'; '试次
    14'; '试次 15'; '试次 16'; '试次 17'; '试次 18'; '试次 19'; '试次 20'};
1244.                 app.ResultsTable.Position = [30 140 900 510];
1245.
1246.                 % Create AccuracyLabel
1247.                 app.AccuracyLabel = uilabel(app.ResultsTab);
1248.                 app.AccuracyLabel.FontWeight = 'bold';
1249.                 app.AccuracyLabel.Position = [950 700 80 22];
1250.                 app.AccuracyLabel.Text = '准确率统计：';
1251.
1252.                 % Create AccuracyTextArea
1253.                 app.AccuracyTextArea = uitextarea(app.ResultsTab);
1254.                 app.AccuracyTextArea.Position = [950 50 380 630];
1255.
1256.                 % Create StatusLabel
1257.                 app.StatusLabel = uilabel(app.UIFigure);
1258.                 app.StatusLabel.HorizontalAlignment = 'right';
1259.                 app.StatusLabel.Position = [20 10 1200 22];
1260.                 app.StatusLabel.Text = '就绪';
1261.
1262.                 % Create ProgressLamp
1263.                 app.ProgressLamp = uilamp(app.UIFigure);
1264.                 app.ProgressLamp.Position = [1230 10 20 20];
1265.                 app.ProgressLamp.Color = [0.8 0.8 0.8];
1266.
1267.                 % Show the figure after all components are created
1268.                 app.UIFigure.Visible = 'on';
1269.             end
1270.         end
1271.
1272.         % App creation and deletion
1273.         methods (Access = public)
1274.
1275.             % Construct app
1276.             function app = exp5
1277.
1278.                 % Create UIFigure and components
1279.                 createComponents(app)
1280.
1281.                 % Register the app with App Designer
1282.                 registerApp(app, app.UIFigure)
1283.
```

```
            if nargout == 0
                clear app
            end
        end

        % Code that executes before app deletion
        function delete(app)

            % Delete UIFigure when app is deleted
            delete(app.UIFigure)
        end
    end
end
```