中南大學
CENTRAL SOUTH UNIVERSITY

本科生课程设计报告

课程名称	数字信号处理课程设计
专业班级	通信工程 1802 班
姓　　名	叶伟伟
学　　号	8203181019
指导教师	张昊

目录

1 课题介绍

1.1 课程设计题目

设计实现一个流程如图 1 所示的数字信号处理演示系统，该系统包含信号生成、频谱分析、滤波器设计、数字滤波和输出信号分析 5 个主要模块，各模块的具体功能要求如下：

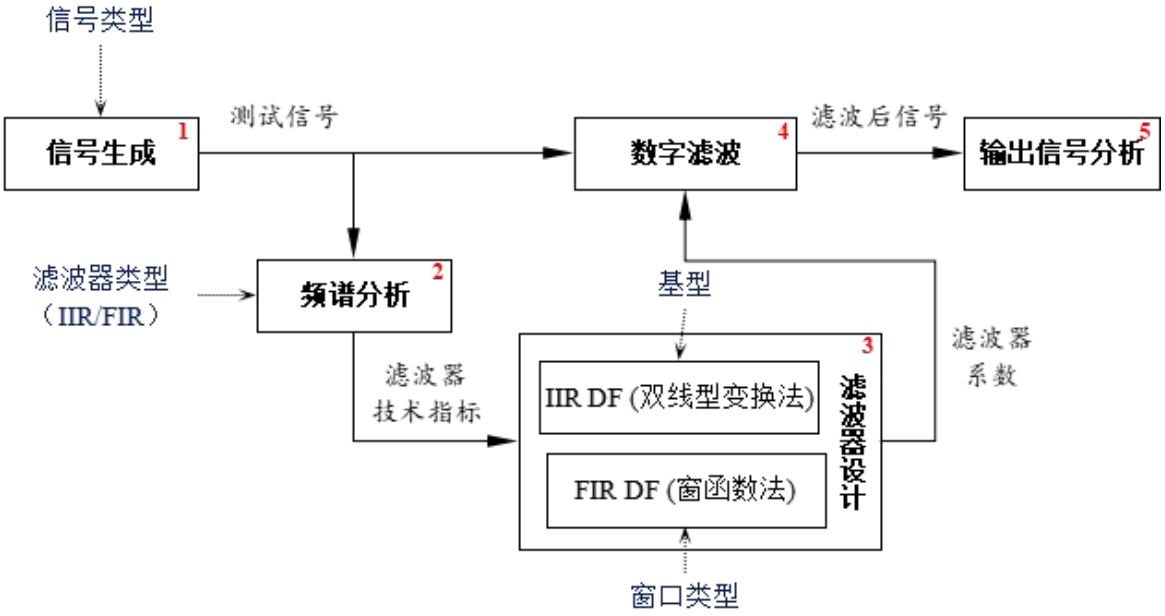

图 1.1-1 数字信号处理演示系统流程

⑴ 信号生成

根据信号类型不同可分为两大类：

① 多频正弦：输入如式 1 所示的由多个不同频率正弦信号叠加组合而成的模拟信号公式，指定**采样频率**（Hz）以及**采样点数**，动态生成该信号的采样序列作为测试信号，用于理论验证滤波器性能。

$$100\sin(2\pi f_1 t)+100\sin(2\pi f_2 t)+\cdots+100\sin(2\pi f_n t) \quad \text{(式 1)}$$

② 数字音频：直接读取音频信号作为测试信号，支持加入**白噪声**、**单频噪声**（正弦干扰）和**多频噪声**（多正弦干扰），用于感官验证滤波器性能。

测试信号生成后，展示其时域波形（第二类信号需支持加噪前后的音频回放）。

⑵ 频谱分析

使用 FFT 对产生的测试信号进行频域变换，展示其幅频、相频特性，指定

需要滤除或保留的频带，通过选择滤波器类型（IIR/FIR），确定对应的滤波器（低通、高通、带通、带阻）技术指标。

⑶ 滤波器设计

根据 IIR/FIR 数字滤波器技术指标设计滤波器，生成相应的滤波器系数，并展示对应的滤波器幅频（衰减）、相频特性。

① IIR DF 设计：使用**双线性变换法**，可选择滤波器基型（巴特沃斯/切比雪夫型）；

② FIR DF 设计：使用**窗口法**，可选择窗口类型。

⑷ 数字滤波

根据设计的滤波器系数，对测试信号进行滤波，得到滤波后信号。

① IIR DF：要求通过**差分方程迭代**实现滤波，未知初值置零处理；

② FIR DF：要求通过**快速卷积**实现滤波。可以选择使用**重叠相加**或**重叠保留**法进行卷积运算，并动态展示卷积运算的详细过程。

⑸ 输出信号分析

展示滤波后信号的时域波形（音频信号需要支持回放）、幅频与相频特性，分析是否满足滤波要求。对同一滤波要求，根据输出信号频谱，对比分析各类滤波器的差异。

为便于分析，模块⑵～⑸中涉及的所有频率参数均对折叠频率归一化。

1.2 题目设计要求

1、使用 MATLAB（或其它开发工具）编程实现上述信号处理演示系统，具体要求如下：

⑴ 系统应使用图形用户界面（**GUI**）；

⑵ 系统功能至少包括**信号的低通与高通滤波**；

⑶ 滤波器设计模块应**避免使用 MATLAB 工具箱函数**；

⑷ IIR DF 设计可选基于**巴特沃斯**或**切比雪夫 I 型**；

⑸ FIR DF 设计**可选择各类窗口**，且 **FIR 滤波可选长序列卷积方法**。

2、独立撰写课程设计报告，课程设计报告的内容包括：

⑴ 课程设计题目和题目设计要求；

⑵ 设计思想和系统功能结构及功能说明；

⑶ 设计中关键部分的详细描述和介绍，采用流程图描述关键模块的设计思路；

⑷ 总结，包括设计过程中遇到的问题和解决方法，心得体会等；

⑸ 参考文献；

⑹ 程序源代码清单。

2 系统设计

2.1 GUI 设计

2.1.1 主界面

主界面 GUI 如图 2.1-1 数字信号处理演示系统主界面如图 2.1-1 数字信号处理演示系统主界面所示，主界面中点击各个模块跳转到对应模块的 GUI 界面，其中 IIR 与 FIR 单选框选择要设计哪种滤波器。

(1) 点击测试信号生成模块按钮，打开**测试信号生成模块**，界面如图 2.1-2 所示

(2) 在滤波器设计栏的右侧单选框中,**选择 IIR,**点击滤波器设计模块按钮，打开 **IIR 滤波器设计模块**,如图 2.1-5 IIR 滤波器设计模块界面所示

(3) 在滤波器设计栏的右侧单选框中,**选择 FIR**,点击滤波器设计模块按钮，打开 **FIR 滤波器设计模块**,如图 2.1-6 FIR 滤波器设计模块界面所示

(4) 点击数字滤波与输出信号分析模块按钮，打开该模块，如图 2.1-7 FIR 滤波与输出信号分析界面与图 2.1-8 IIR 滤波与输出信号分析界面所示

图 2.1-1 数字信号处理演示系统主界面

2.1.2 测试信号生成模块界面

测试信号生成界面如图 2.1-2 所示。

(1) 信号类型选择**多频正弦，**点击**正弦信号生成**按钮打开如图 2.1-3 所示的

正弦信号生成模块。

(2) 信号类型选择**数字音频**，录入/从文件选择输入信号后， 可以点击**噪音生成**按钮打开噪音生成模块。

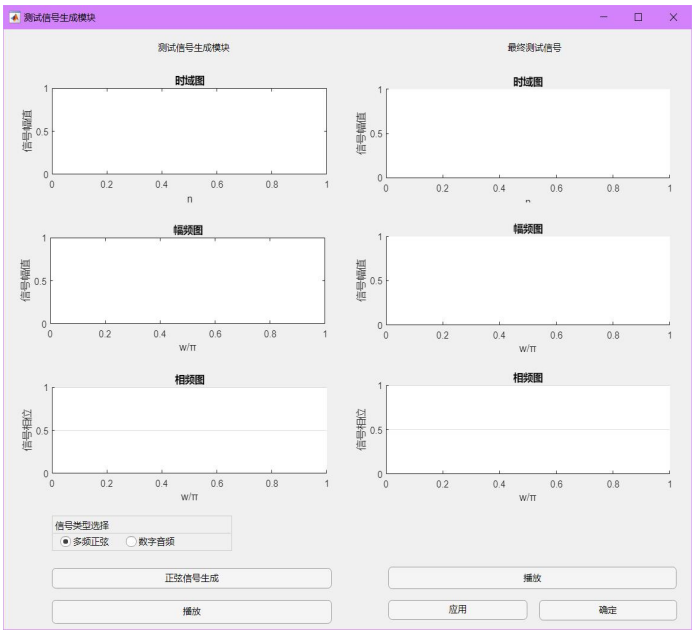

图 2.1-2 测试信号生成模块界面

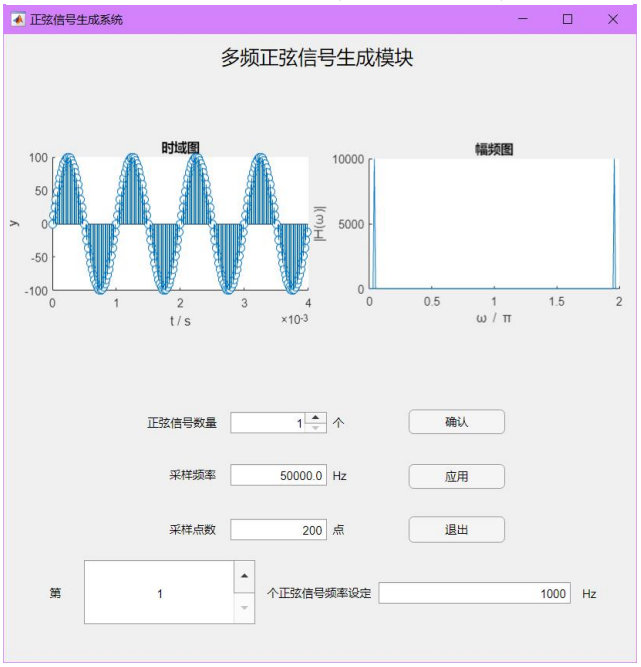

图 2.1-3 多频正弦信号生成模块界面

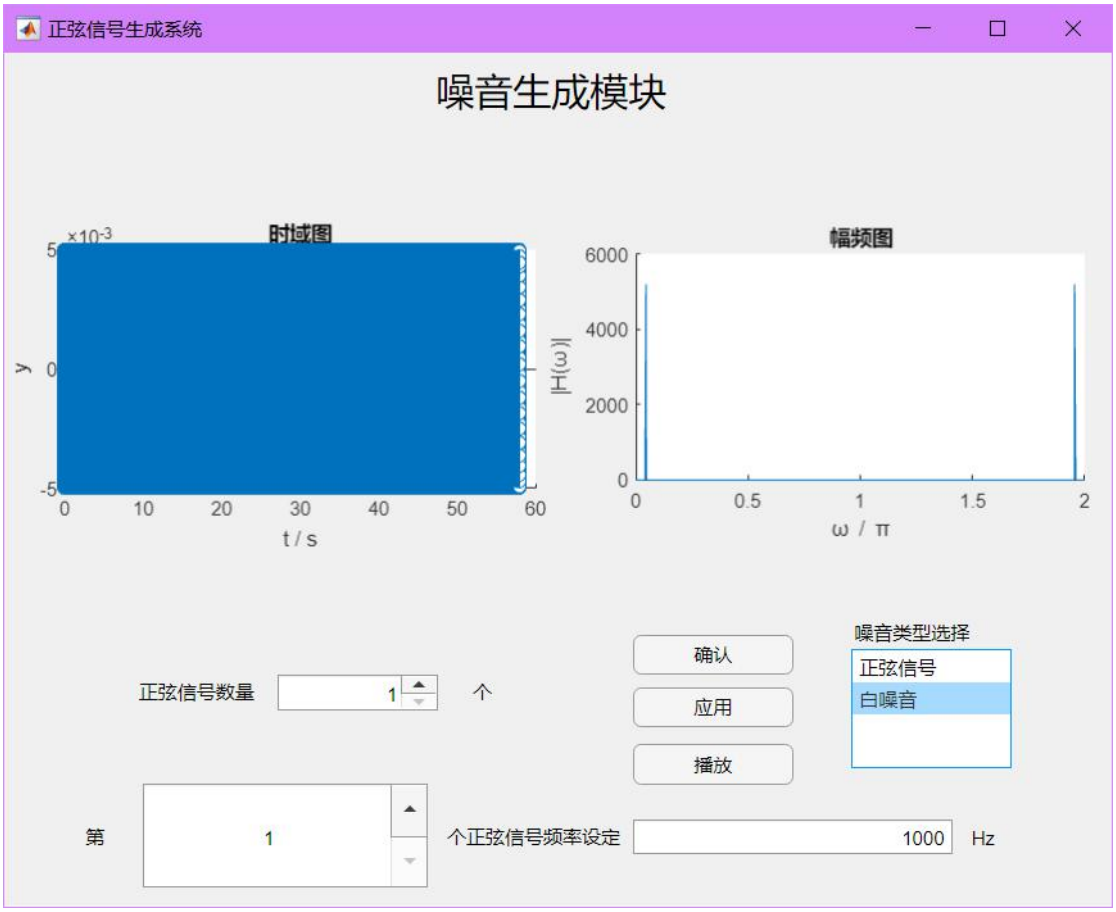

图 2.1-4 噪音生成模块界面

2.1.3 IIR 滤波器设计模块界面

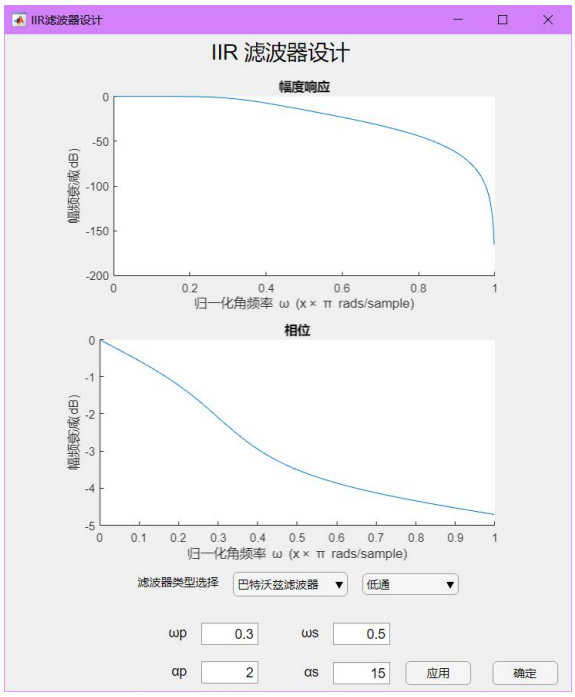

图 2.1-5 IIR 滤波器设计模块界面

2.1.4 FIR 滤波器设计模块界面

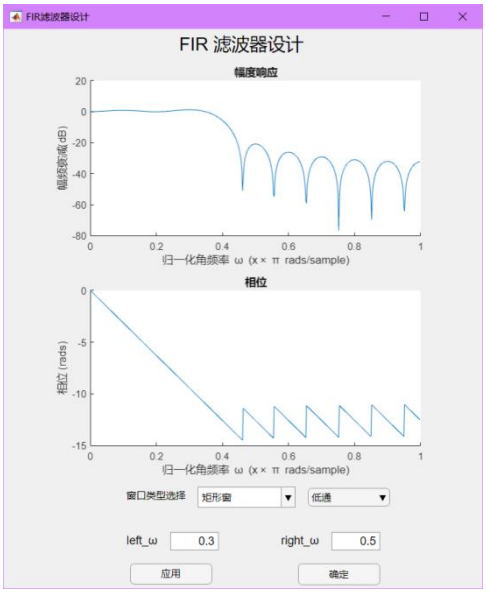

图 2.1-6 FIR 滤波器设计模块界面

2.1.5 滤波与输出信号分析界面

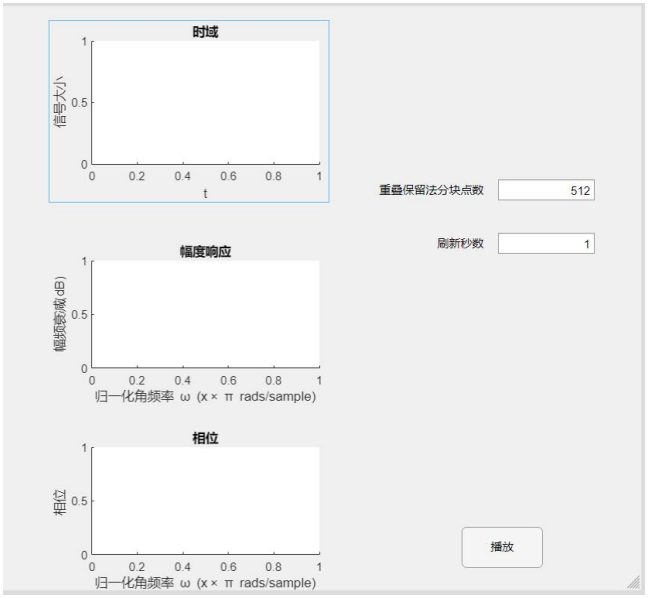

图 2.1-7 FIR 滤波与输出信号分析界面

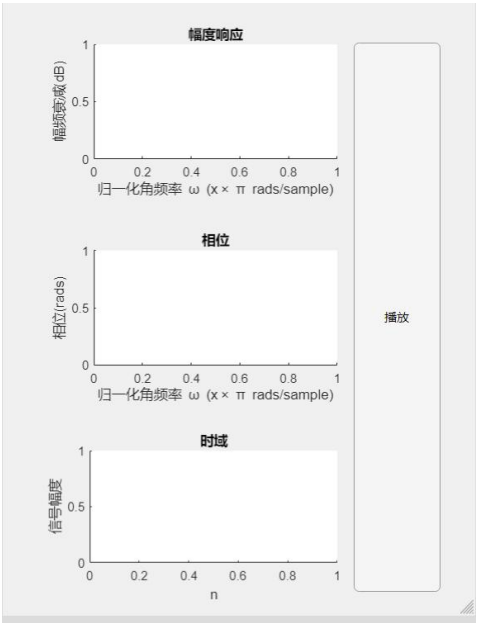

图 2.1-8 IIR 滤波与输出信号分析界面

2.2 模块功能设计

根据实验指导书，应该至少有信号生成,频谱分析,滤波器设计,数字滤波,输出信号分析五个模块,这里多了一个主模块进行模块的控制与模块间的解耦，模块

功能结构图如图 2.2-1 所示

2.2.1 模块功能结构图

主界面的三个按钮用来分别打开对应模块， 各个模块详细功能见图 2.2-1 及对应模块的流程图，见 2.2.2 ,2.2.3 ,2.2.4 ,2.2.5

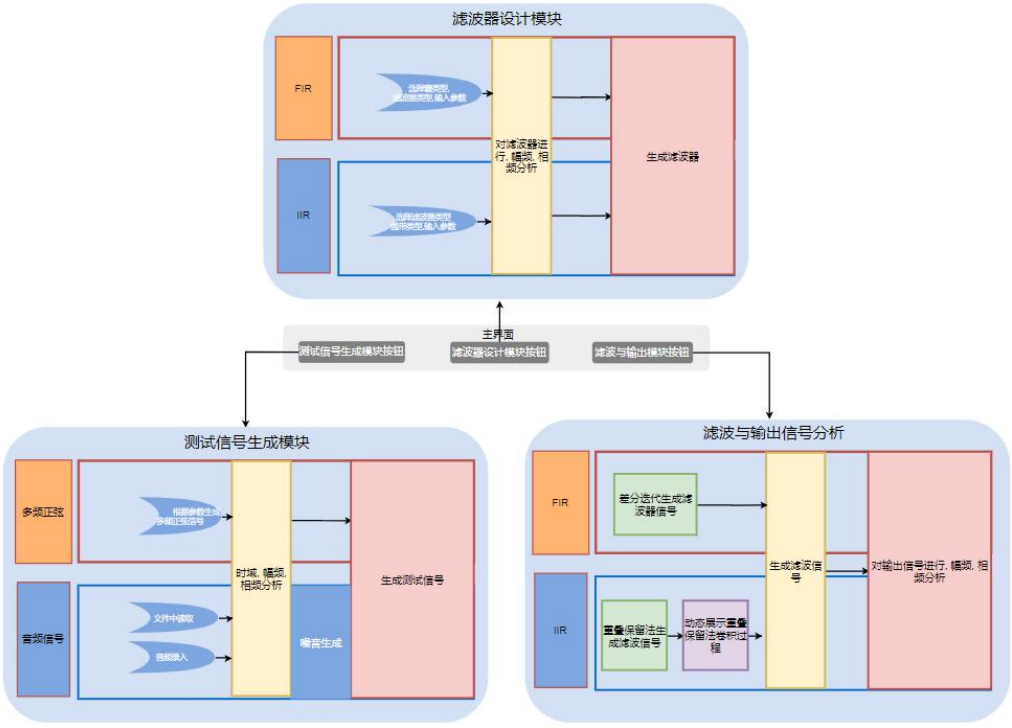

图 2.2-1 模块功能结构图

2.2.2 测试信号生成模块功能设计

测试信号生成模块的功能是为了产生测试信号， 根据**题目设计要求**，测试信号应该至少有**多频正弦**(图 2.2-2)**与音频信号**(图 2.2-3)。

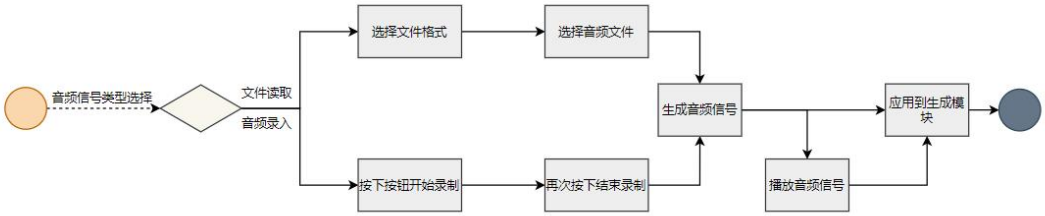

图 2.2-2 多频正弦信号生成功能流程图

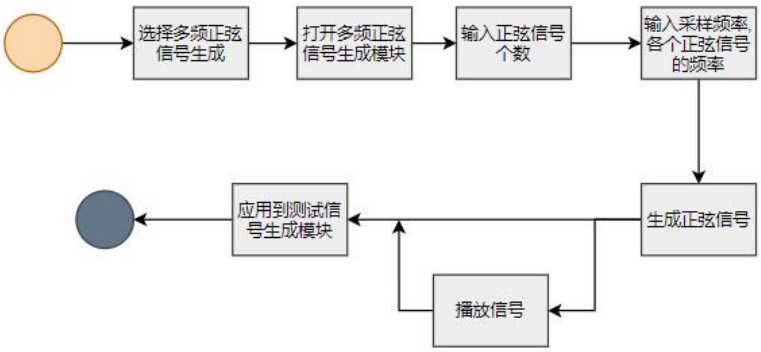

图 2.2-3 音频信号生成功能流程图

2.2.3 IIR 滤波器设计模块功能设计

IIR 滤波器设计模块功能模拟滤波器部分实现巴特沃兹滤波器与切比雪夫 I 型滤波器，对于低通，高通，带通带阻类型滤波器全部实现，并且展示对于幅频，相频特性见图 2.1-5 与图 2.2-4

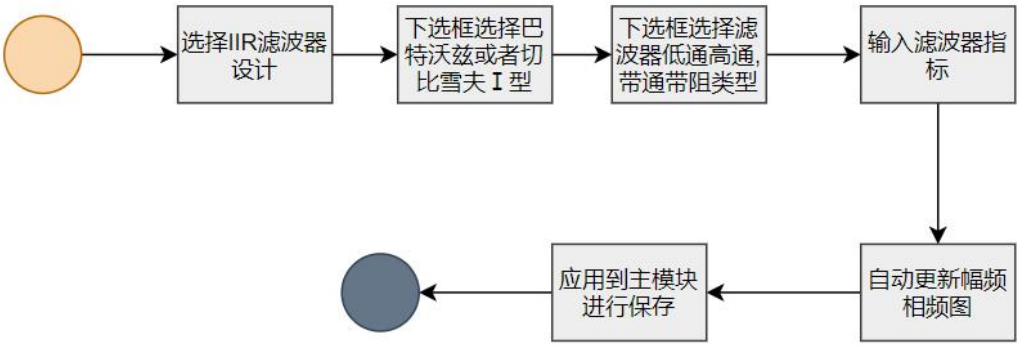

图 2.2-4 IIR 滤波器设计模块功能流程图

2.2.4 FIR 滤波器设计模块功能设计

FIR 滤波器设计模块使用**窗函数法进行滤波器设计**， 共实现了矩形窗，三角窗，汉明窗，哈明窗，布莱克曼窗五个窗口类型， 对于低通高通带通带阻滤波器类型全部实现， 用户输入左边截止频率与右边截止频率来进行滤波器的设计，见图 2.1-6 与图 2.2-5。

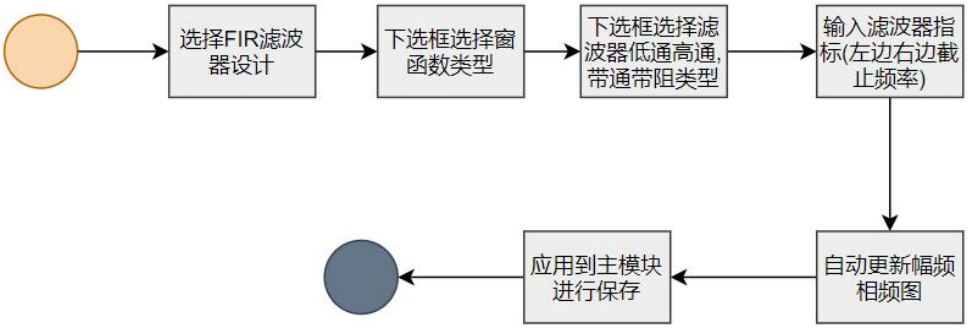

图 2.2-5 FIR 滤波器设计模块功能流程图

2.2.5 滤波与输出信号分析模块功能设计

滤波与输出信号分析模块根据选择的滤波器类型进行滤波的分析

1. 对于 IIR 滤波器， 采用**差分方程迭代**的形式来进行滤波， 最终展现时域图与幅频相频特性图，并且可以播放输出信号。
2. 对于 FIR 滤波器，采用**重叠保留法进行快速卷积**，动态展示卷积过程，最终呈现时域图与幅频相频特性图，并且可以播放输出信号。

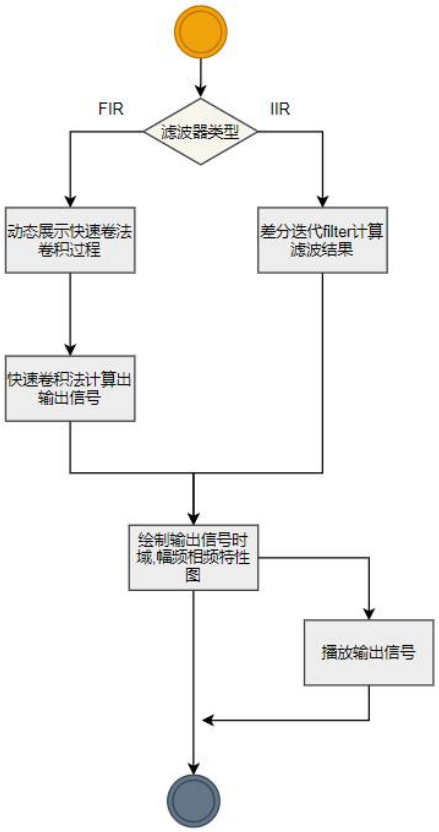

图 2.2-6 滤波与输出信号分析模块功能流程图

3 关键功能实现与展示

3.1 GUI 的实现

3.1.1 技术选型

在 Matlab 2018a 中， GUI 的实现有两种方法

1. GUIDE
2. App Designer

GUIDE 使用面向过程的开发方式， App Designer 使用面向对象的开发方式，由于本项目较大，为了尽可能做到代码间的解耦与分块，这里**我使用 App Designer 进行 GUI 开发**。

3.1.2 GUI 实现

App Designer 中， GUI 的实现均通过拖拉缩放的形式完成，由 matlab codeGenerator 生成相应代码。

图 3.1- 1 Matlab App Designer 设计工具

3.2 信号的分析

信号的分析实现两种功能

1. 基于时域图, 幅频图,相频图的分析
2. 基于感官的分析,即播放功能

3.2.1 时域图,频谱图,相频图的绘制

信号的分析基于时域图,幅频图与相频图时,时域图的绘制通过计算信号长度与信号大小得到, 幅频图与相频图的绘制主要是利用 fft 进行离散的快速傅里叶变换,再利用 abs()函数求得幅度响应及利用 angle 求得幅角进行求解。

三图绘制的一般流程代码如下:

```
% 绘制原信号音频--时域图
plot(app.testSigUIAxes,app.testSig);

% 更新数据
N = length(app.testSig); % 原信号点数
fft_y = fft(app.testSig); % 计算 fft

n = linspace(0 , N-1,N); % 第 n 点
w = 2*pi*n./N; % 角频率(未归一化)

% 绘制原信号音频--频谱图
plot(app.testSigSpectrum,w/pi,abs(fft_y));  % w 归一化

% 绘制原信号音频--相频图
plot(app.testSIgPhaseFig,w/pi,unwrap(angle(fft_y)));
```

如一个音频采样信号, 最终绘制结果如下

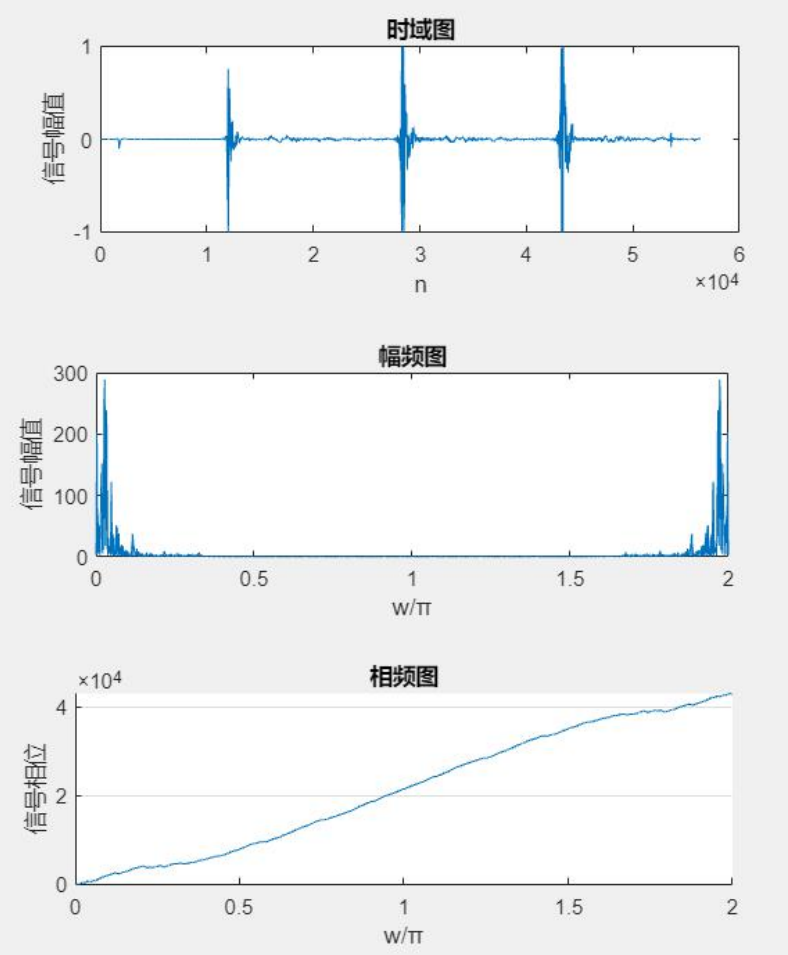

图 3.2-1 时频域分析图绘制结果

值得注意的是，一般在我们计算一个系统相频特性时，就要用到反正切函数提取相位，计算机中反正切函数规定，在一、二象限中的角度为 0～pi，三四象限的角度为 0～-pi。但实际得到的结果会发生相位跳变，跳变幅度为 2pi，这就叫相位的卷绕。unwrap 函数的作用就是解卷绕，使相位在 pi 处不发生跳变，从而反应出真实的相位变化。

实际在默认的情况下，unwrap 在检查到数据前后两点的差距在超过 pi 的时候，就认为有跳变，当然其阈值也能通过函数设置。

图 3.2-2 幅度角添加 unwrap

3.2.2 音频的播放

信号分析的第二种方式为音频播放的感官感受。

这里音频的播放使用 maltab 的 sound 函数进行，函数 sound(y,Fs)发送给扬声器一个信号，以 Fs 的频率进行播放，这里音频的播放频率全部约定为 44100Hz，参数 y 为一列的数据时,代表单声道播放信号，当 y 为两列数据时，代表双声道播放数据，**由于录制时默认为单声道,因此这里约定所有的信号都为单声道方式的播放。**

各个界面播放按钮对应的位置请看图 2.1-2 测试信号生成模块界面,图 2.1-8 IIR 滤波与输出信号分析界面,图 2.1-7 FIR 滤波与输出信号分析界面,图 2.1-4 噪音生成模块界面。

3.3 信号的产生

由于计算机无法保存无限的结果，实际上我们保存的都是离散的取值

1. 这里对正弦信号的采样全部采用 matlab 的 sin 函数进行。
2. 与播放约定一致，在进行数字音频信号的录音取值采样时， 选择的频率为 44100Hz,并且都为单声道的声音信号采样。

3.3.1 多频正弦信号的产生

多频正弦信号的产生利用多频正弦信号生成模块(附录三 多频正弦信号生成模块)完成,多频正弦信号的产生使用单一的，根据式 1 产生多个正弦信号，并且对其进行正弦函数个数的设置，采样点数的设置，采样频率的设置等,**模块整体算法输入参数共有 5 个, 输出为 1 个生成的正弦信号**

输入:

1. sinFunc: 指定的正弦函数,在 matlab 中使用匿名函数进行表示, 这里根据式 1 确定为: sinFunc = @(f, t) 100*sin(2*pi*f*t)
2. sn: 要设置的正弦信号的个数
3. f: 频率列表,即每个正弦信号的频率,长度不能大于 sn
4. N: 采样点数
5. Fs: 采样频率

输出:

sigGen: 1 个生成正弦信号

算法伪代码:

输入数据

根据采样点数初始化生成的正弦信号 sinSig

根据采样点数 N 与采样频率 Fs, 计算出各个采样位置

对采样频率列表 f 进行遍历 i=1:sn

计算第 i 个采样频率对应的正弦信号值 sin_i

将 sin_i 叠加到 sinSig 上

输出正弦信号 sinSig’

显示结果如图 3.3-1 正弦信号生成结果所示

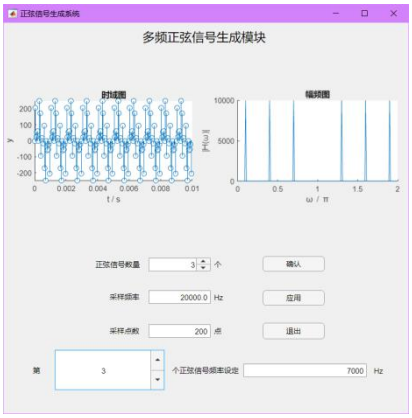

图 3.3-1 正弦信号生成结果

3.3.2 音频信号的产生

音频信号的产生分为**音频录入**与**从音频文件中读取**。

1. 音频录入

Matlab 实现音频输入需要通过 audioRecorder 对象来完成，audioRecorder，audiorecorder(Fs,nBits,NumChannels)通过指定采样频率 Fs，采样率 nBits，声道个数 numCahnnels 来进行声音的录制。

audioRecorderObj= audiorecorder(Fs,nBits,NumChannels) 语句保存一个 audiorecorder 对象到变量 recorder 中，由于需要多次多地使用该变量，在附录二 测试信号生成模块中添加一个变量，**在初始化该模块时保存该 recorder 对象，用于进行音频的录制及录制完成后音频信号的输出。**

音频录制流程图如下，结果如图 3.3-3 音频信号录入结果所示。

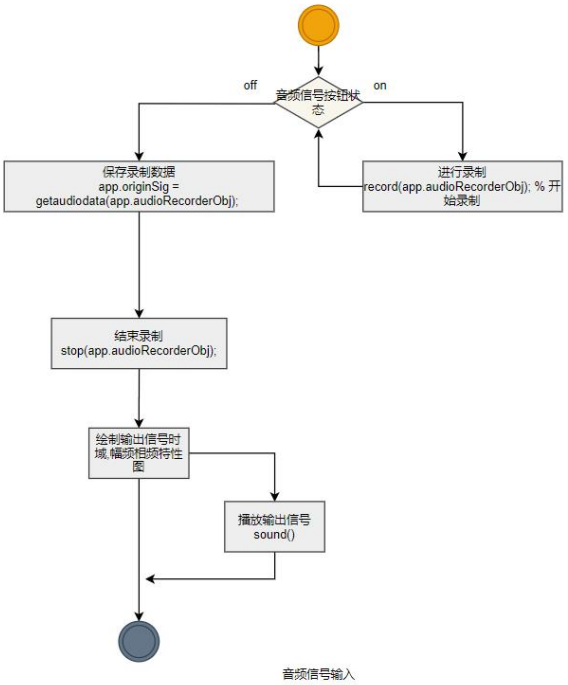

图 3.3-2 音频录制算法流程图

2. 音频读取

若要从音频文件中读取，则需要用到 matlab 的[sig，Fs] = audioread(path)函数，path 指定一个路径，返回对应的时域离散信号与该音频的采样频率。

值得注意的是，如果输入的音频文件本身为双声道音频文件，那么他对应的就会有两列离散数据，在进行数据处理时要特别留意，否则会导致出错，音频读取最终产生的信号如图 3.3-4 音频文件读取结果所示

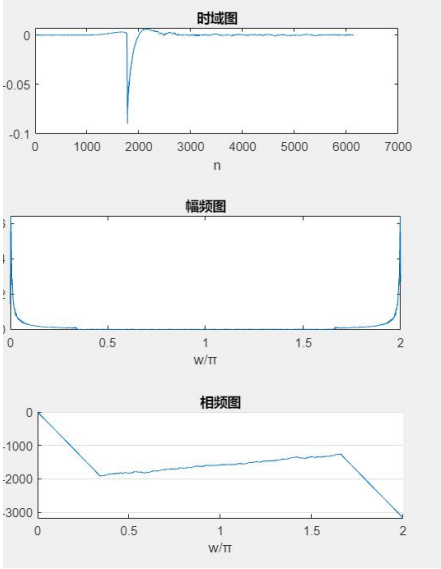

图 3.3-3 音频信号录入结果

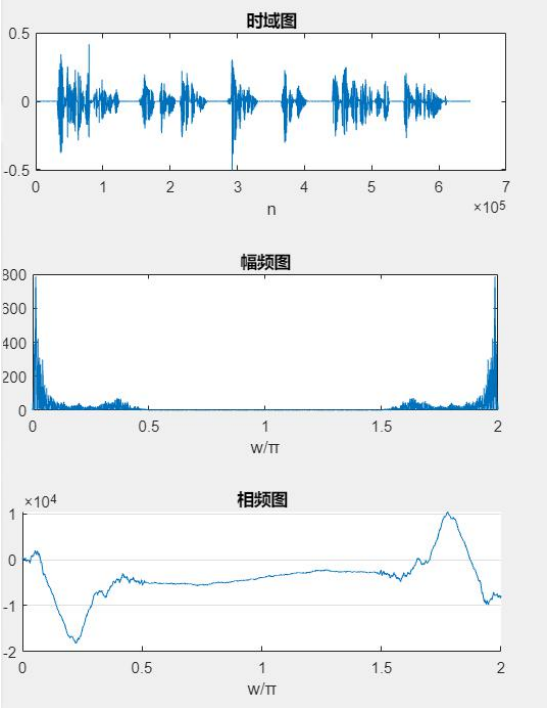

图 3.3-4 音频文件读取结果

3.3.3 噪音信号的产生

噪音的产生利用噪音信号模块完成(附录四 噪音生成模块)可选择产生**高斯白噪声**或者产生**多频正弦信号**噪声。

1. 对于多频正弦信号噪声，采样频率设置为 44100Hz，因此截止频率为 22050Hz，算法输入输出流程与多频正弦信号的产生类似，描述如下

模块整体算法输入参数共有 5 个，输出为 1 个生成的正弦信号

输入：

1. sn: 要设置的正弦噪音信号的个数

输出：

sinNoiseSig: 1 个生成正弦信号

算法伪代码：

输入数据

根据采样点数初始化生成的正弦信号 sinNoiseSig

根据采样频率 Fs=44100Hz，计算出各个采样位置

对 sn 个数进行遍历 i=1:sn

计算第 i 个采样频率对应的正弦信号值 sin_i

将 sin_i 叠加到 sinNoiseSig 上

输出正弦噪音信号 sinNoiseSig

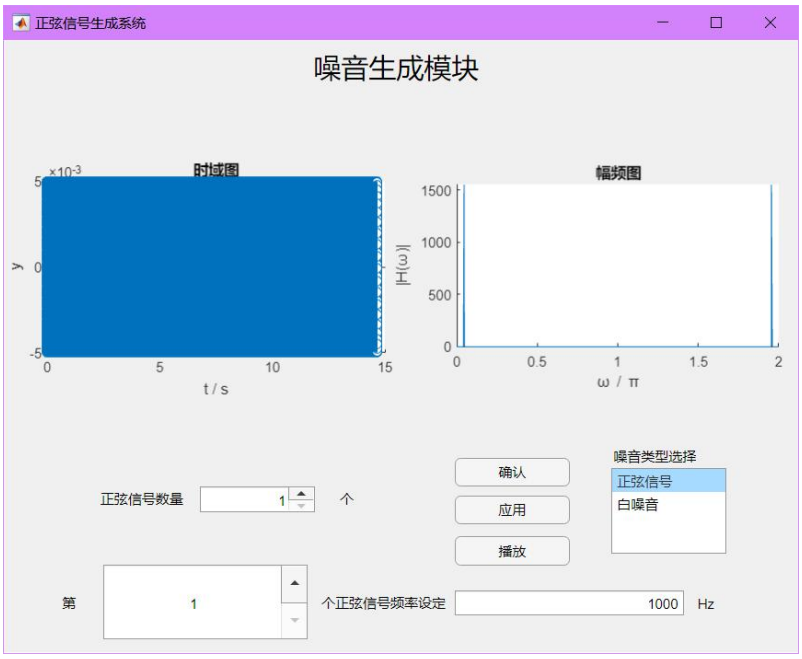

图 3.3-5 正弦噪音信号生成结果

2. 对于高斯白噪声，采用 matlab 的 randn 函数进行生成，产生结果如下

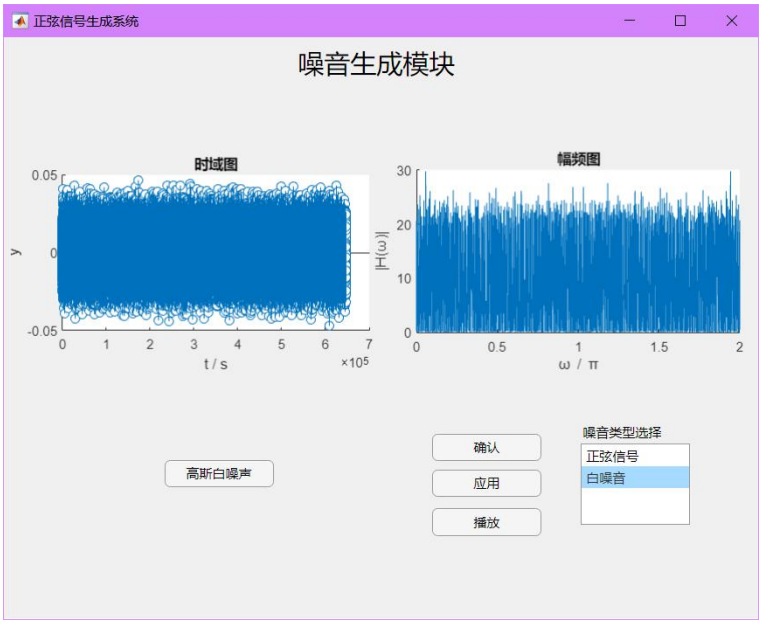

图 3.3-6 高斯白噪声生成结果

3.4 IIR 滤波器设计与滤波

IIR 数字滤波器的设计方法有两类：间接设计法和直接设计法。间接设计法是借助模拟滤波器设计方法进行设计的，先根据数字滤波器设计指标设计相应的过渡模拟滤波器，再将过渡模拟滤波器转换为数字滤波器。直接设计法在时域或频域直接设计数字滤波器。

根据实验指导书要求，这里通过双线性变换法的间接设计法来进行巴特沃兹滤波器的设计，这里自实现生成巴特沃兹滤波器与切比雪夫Ⅰ型滤波器的阶数与截止频率，由于通过阶数与截止频率生成数字滤波器极点参数较为复杂，这里直接调用 matlab 的 butter()函数与 cheby1()函数生成滤波器系统函数 H(z)的分子 b 与分母 a,用于分析显示滤波器及进行滤波。

3.4.1 双线性变换法设计巴特沃兹滤波器

巴特沃兹双线性变换法的数字滤波器的设计源代码见 附录十 巴特沃兹自实现，**该算法利用双线性变换法通过指定的数字指标直接设计数字滤波器,算法描述如下，**如图 3.4-2 所示为一巴特沃兹带通滤波器的设计。

输入

wp ： 通带截止频率， 低高通 1x1 带通带阻为 1x2

ws ： 阻带截止频率， 低高通 1x1 带通带阻为 1x2

rp ： 通带最大衰减

rs ： 阻带最大衰减

opt: 默认为 'z'，设计数字滤波， 指定为's'时设计模拟滤波

输出

Order: 满足指标的巴特沃兹滤波器阶数

Wn: 满足指标的巴特沃兹滤波器 3dB 截频

流程图

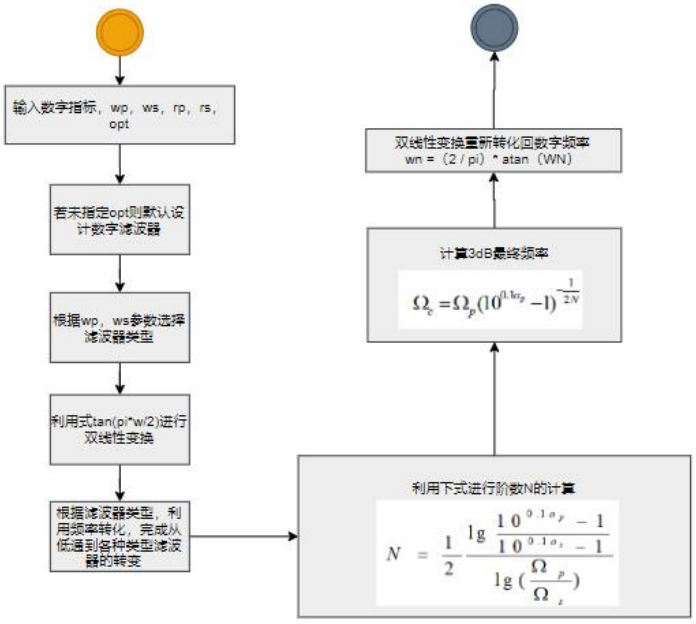

图 3.4-1 巴特沃兹数字滤波器设计算法流程图

3.4.2 双线性变换法设计切比雪夫Ⅰ型滤波器

切比雪夫双线性变换法的数字滤波器的设计源代码见 附录十一 切比雪夫Ⅰ型自实现，**该算法利用双线性变换法通过指定的数字指标直接设计数字滤波器,算法与 3.4.1 巴特沃兹算法相似，算法描述如下。**如图 3.4-4 所示，为一切比雪夫Ⅰ型带通滤波器的设计。

输入

wp ： 通带截止频率， 低高通 1x1 带通带阻为 1x2

ws ： 阻带截止频率， 低高通 1x1 带通带阻为 1x2

rp ： 通带最大衰减

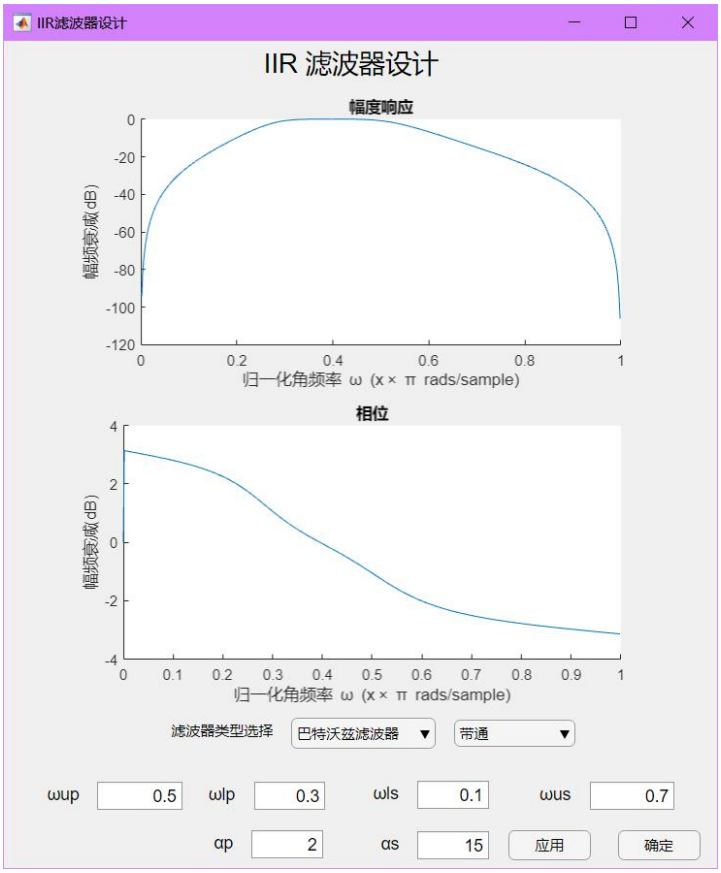

图 3.4-2 巴特沃兹带通滤波器设计

rs ：阻带最大衰减

opt: 默认为 'z'，设计数字滤波， 指定为's'时设计模拟滤波

输出

Order: 满足指标的切比雪夫滤波器阶数

Wn: 满足指标的切比雪夫滤波器 3dB 截频

流程图

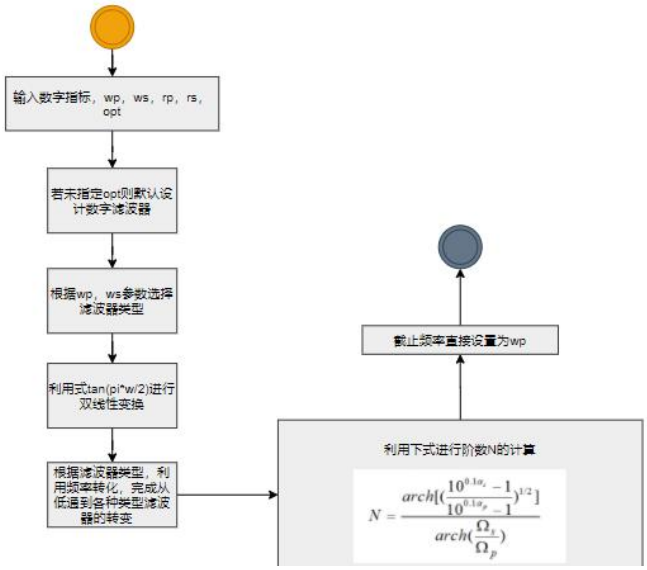

图 3.4-3 切比雪夫数字滤波器设计算法流程图

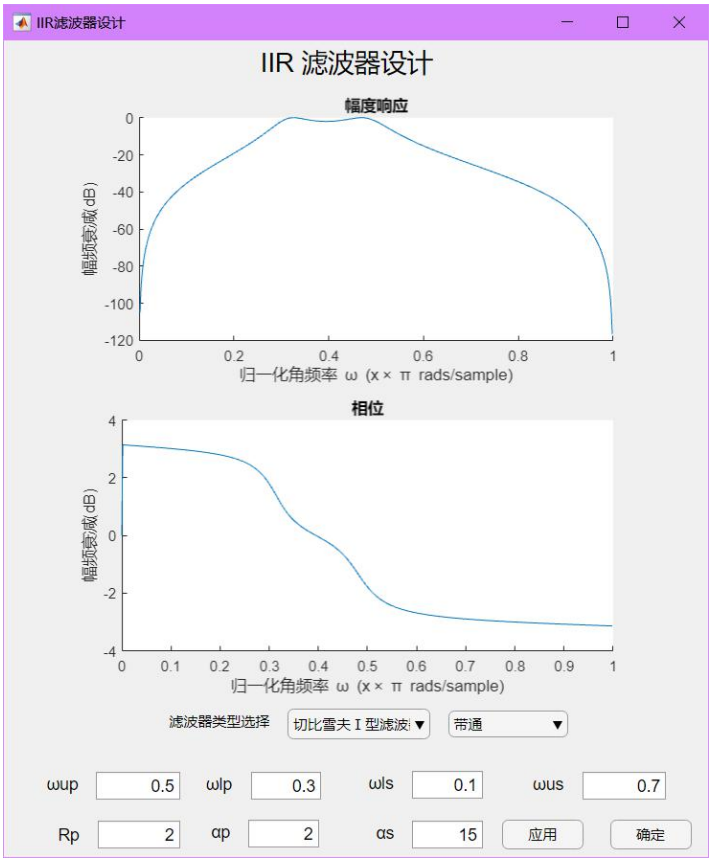

图 3.4-4 切比雪夫 I 型带通滤波器设计

3.4.3 差分迭代实现滤波

包含未知函数的差分及自变数的方程。在求微分方程的数值解时，常把其中的微分用相应的差分来近似，所导出的方程就是差分方程。

利用 IIR 滤波器系统函数分子分母参数 b,a 与时域离散信号进行差分迭代，是实现滤波的一种方式，matlab 提供了 filter()函数用来计算差分方程，因此这里使用 filter()函数进行滤波。使用方法为：y = filter(b,a,X)，X 是时域离散信号，b,a 分别是滤波器系统函数的分子与分母值，y 为最终滤波输出信号，之后交由信号分析模块进行图谱的绘制与音频 的播放,详见 3.2 信号的分析。

如某一多频正弦信号(图 3.4-5 某多频正弦信号),利用巴特沃兹 IIR 低通滤波器(图 3.4-6 巴特沃兹 IIR 低通滤波器)进行差分迭代滤波后的结果如图 3.4-7 所示。

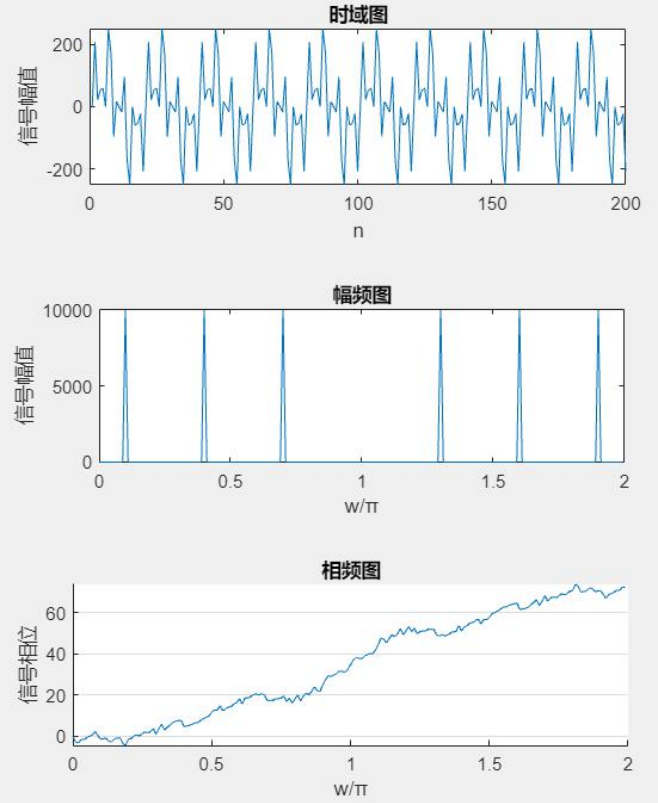

图 3.4-5 某多频正弦信号

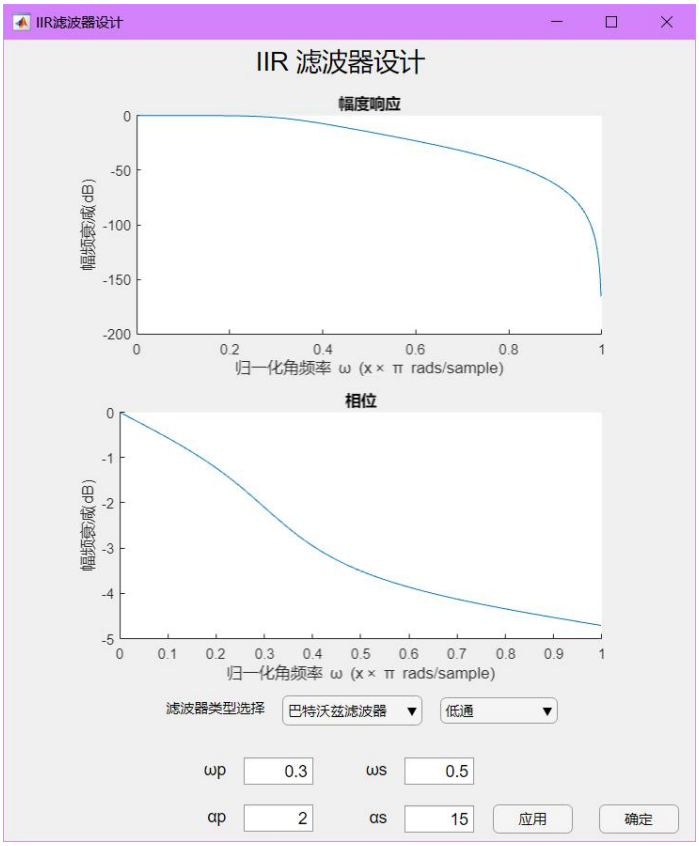

图 3.4-6 巴特沃兹 IIR 低通滤波器

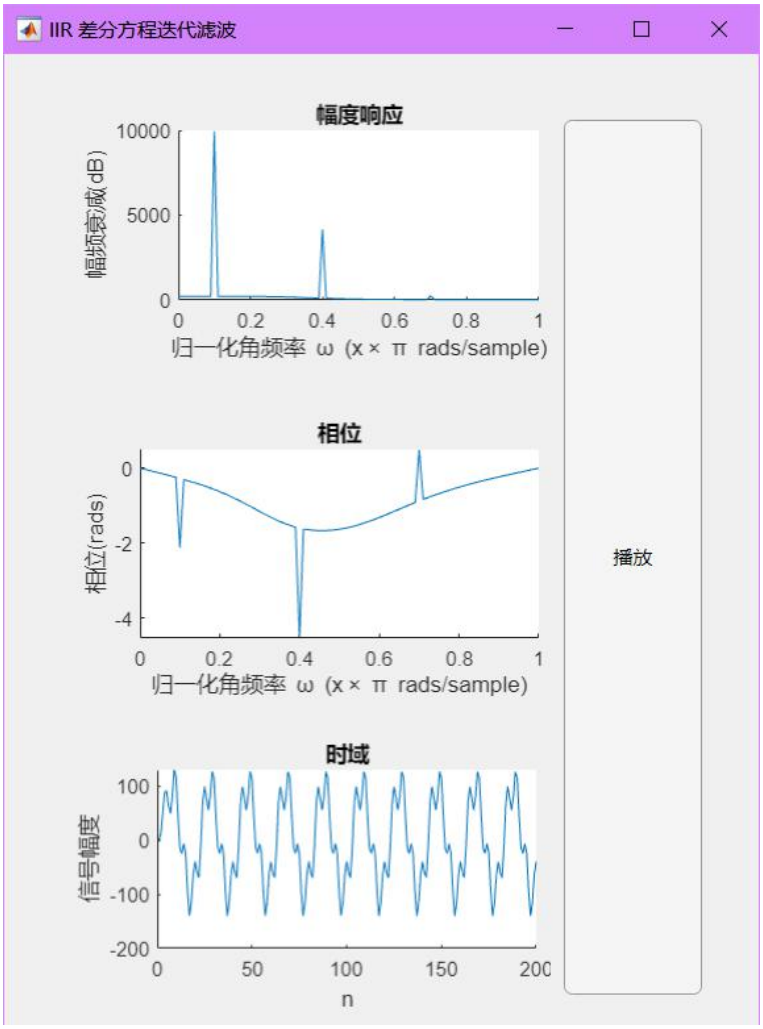

图 3.4-7 IIR 差分迭代滤波结果

3.5 FIR 滤波器的设计与滤波

FIR(Finite Impulse Response)滤波器：有限长单位冲激响应滤波器，又称为非递归型滤波器，是数字信号处理系统中最基本的元件，它可以在保证任意幅频特性的同时具有严格的线性相频特性，同时其单位抽样响应是有限长,，因而滤波器是稳定的系统。**这里采用窗函数法进行 FIR 滤波器的设计，使用重叠保留法进行快速卷积实现滤波。**

3.5.1 窗函数法设计 FIR 滤波器

对于窗函数的选择，应考虑被分析信号的性质与处理要求。如果仅要求精确读出主瓣频率，而不考虑幅值精度，则可选用主瓣宽度比较窄而便于分辨的矩形窗，例如测量物体的自振频率等；如果分析窄带信号，且有较强的干扰噪声，则应选用旁瓣幅度小的窗函数，如汉宁窗、三角窗等；对于随时间按指数衰减的函数，可采用指数窗来提高信噪比,这里共实现了矩形窗,三角窗,汉宁窗,海明窗等窗

口，主要窗口介绍如下,**窗函数的生成，通过 matlab 对应的窗函数生成(如 hanning 窗即为 hanning()函数)**。

窗函数法的设计与滤波器的绘制均通过 updateFIRFilter(lw，rw)函数实现,函数输入输出如下

输入

lw：左边截止频率

rw：右边截止频率

输出

h：幅频响应

w：幅频对应的角频率位置

流程图

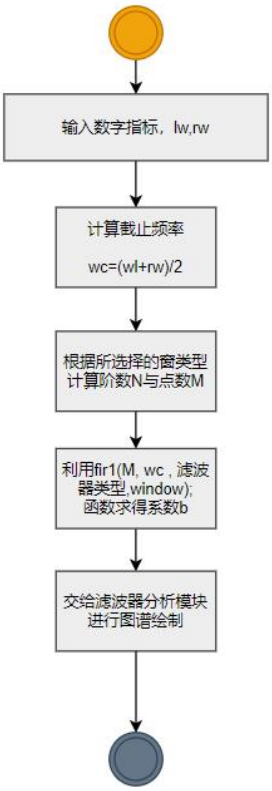

图 3.5-1 窗函数法设计函数流程图

如图 3.5-2 所示，为使用窗函数法的布莱克曼窗设计的一带通滤波器。

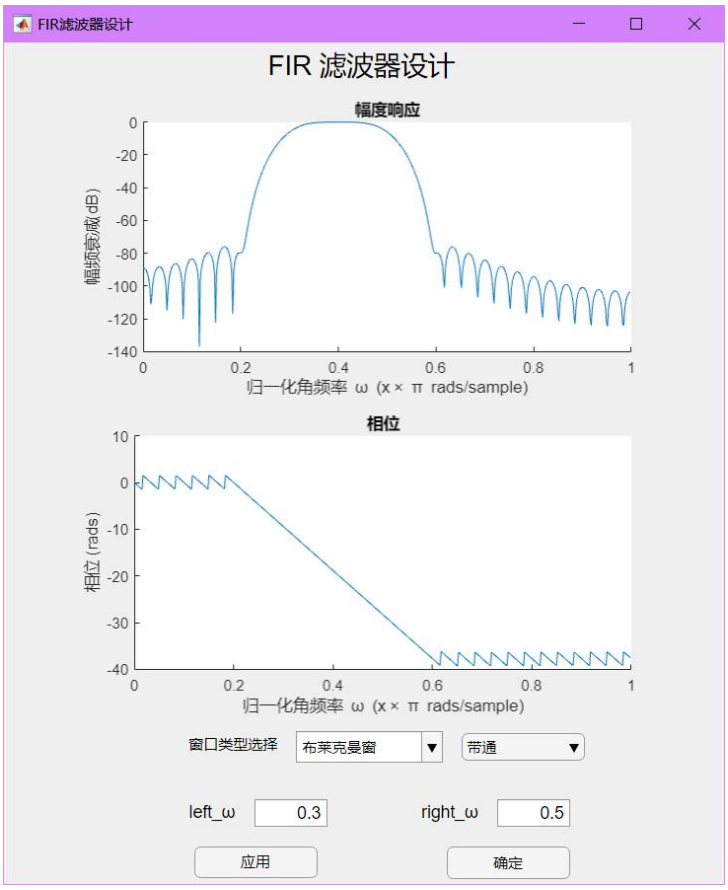

图 3.5-2 窗函数法 FIR 布莱克曼窗带通滤波器设计

3.5.2 重叠保留法计算线性卷积实现滤波与动态展示

对于实际的数字信号处理系统，设计有限长的系统序列是容易的，但是待处理的信号往往是近似无限长的，受计算机处理器字长限制，必须将信号序列分段进行快速卷积处理。设信号被等分为列长 N 的分段（N 的取值最好与 L 取值的数量级相等），系统序列长为 M，圆周卷积结果序列长为 L。为使周期卷积周期 L≥N+M-1，必须通过特定的方法延长信号序列和系统序列。按照延长序列的方法，可将卷积计算方法分为两种：重叠相加法，重叠保留法。

这里，我使用重叠保留法进行快速卷积，并且添加了动态展示，计算过程如下，卷积之前通过保留分段信号序列前端 M-1 位原输入序列（第一段前 M-1 为置零），使分段信号序列延长直至满足 L≥N+M-1（系统序列长保持不变）；卷积之后作求和运算时，舍弃每段卷积结果前 M-1 位的错误取值序列后按位相加，相加时则不存在重叠的点。该方法在卷积前保留了冗余位输入数据，卷积后又为避免无效数据重叠相加而舍去，故称为重叠保留法。

算法描述如下

输入

x: 离散时域信号

h: 系统冲击响应

L: 分块大小，不能大于 x 或 h

pauseTime: 动态展示时的停顿间隔

输出

sig 重叠保留法得到的卷积信号

算法流程图如下

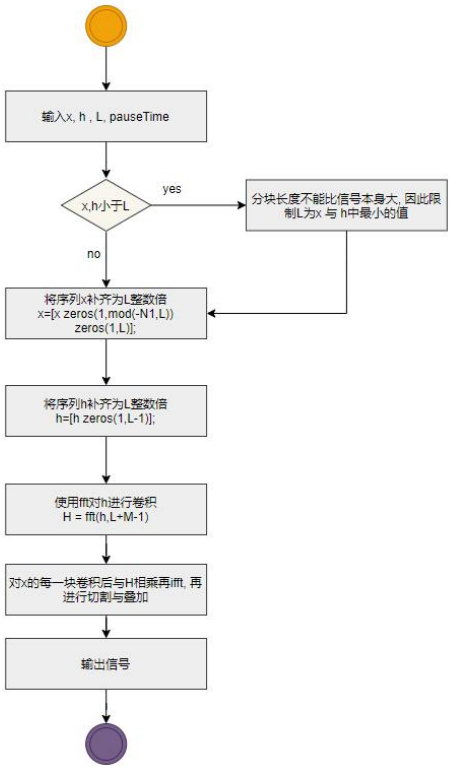

图 3.5-3 重叠保留法算法流程图

为了实现动态展示，在循环过程中添加每一块与每一块叠加结果的绘制代码

```
% 当前计算的Y
subplot(2,1,1)
hold off;
stem(Y);
xlabel(['第',num2str(stage),'块ifft结果']);

% 绘制当前X状态
subplot(2,1,2)
hold off;
stem(X);
xlabel(['前',num2str(stage),'块ifft叠加结果']);
```

图 3.5-4 动态展示重叠保留法过程

具体实现见：附录九 重叠保留法及动态展示

同样利用多频正弦信号(图 3.4-6 某多频正弦信号)， 利用图 3.5-5 所示滤波器进行 FIR 滤波， 通过重叠保留法进行滤波之后的结果如图 3.5-7 所示，动态展示部分如图 3.5-6 所示。

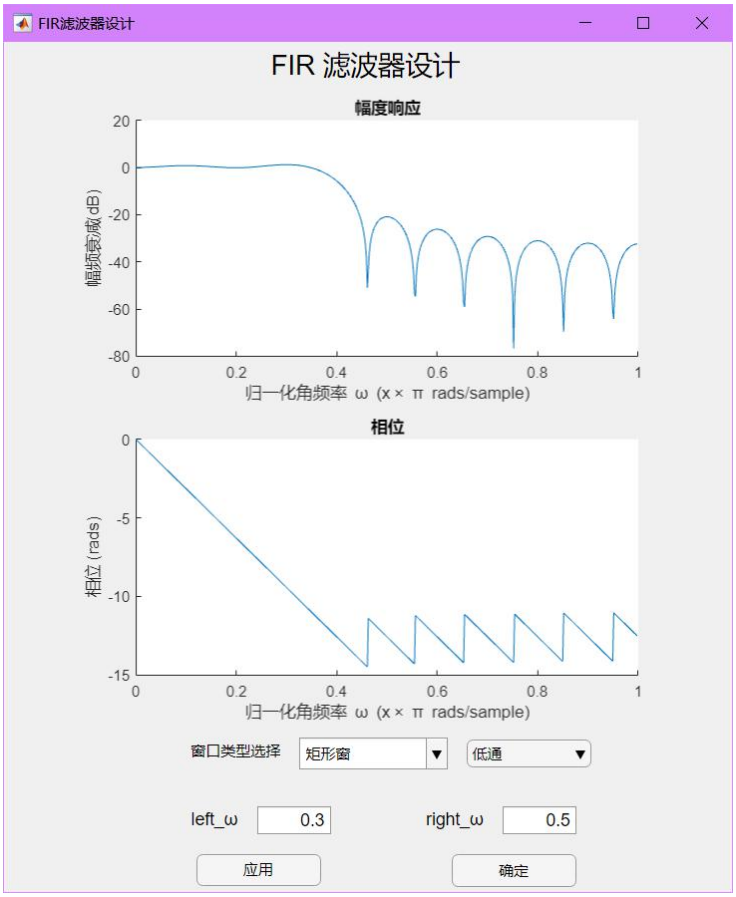

图 3.5-5 FIR 矩形窗低通滤波器

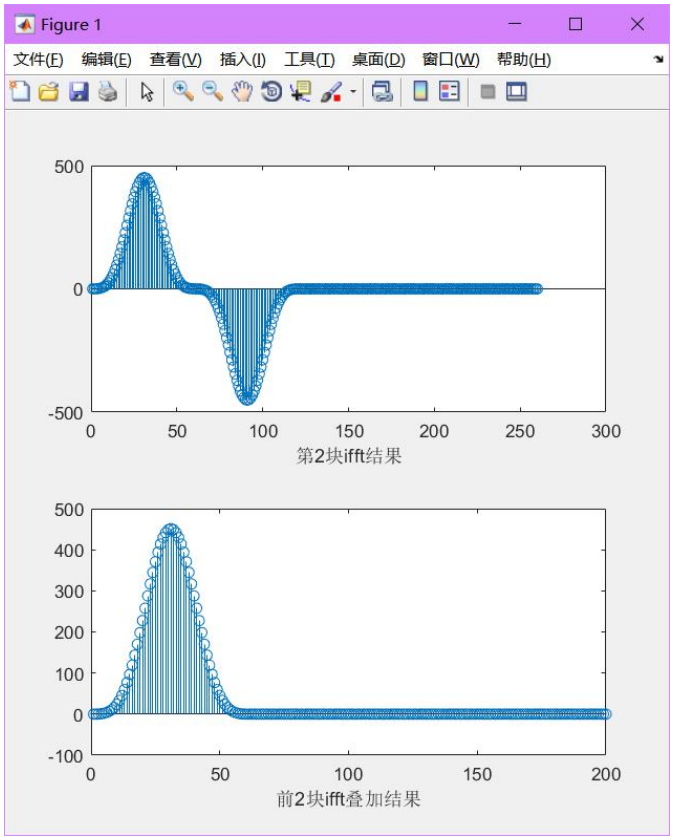

图 3.5-6 重叠保留法动态展示

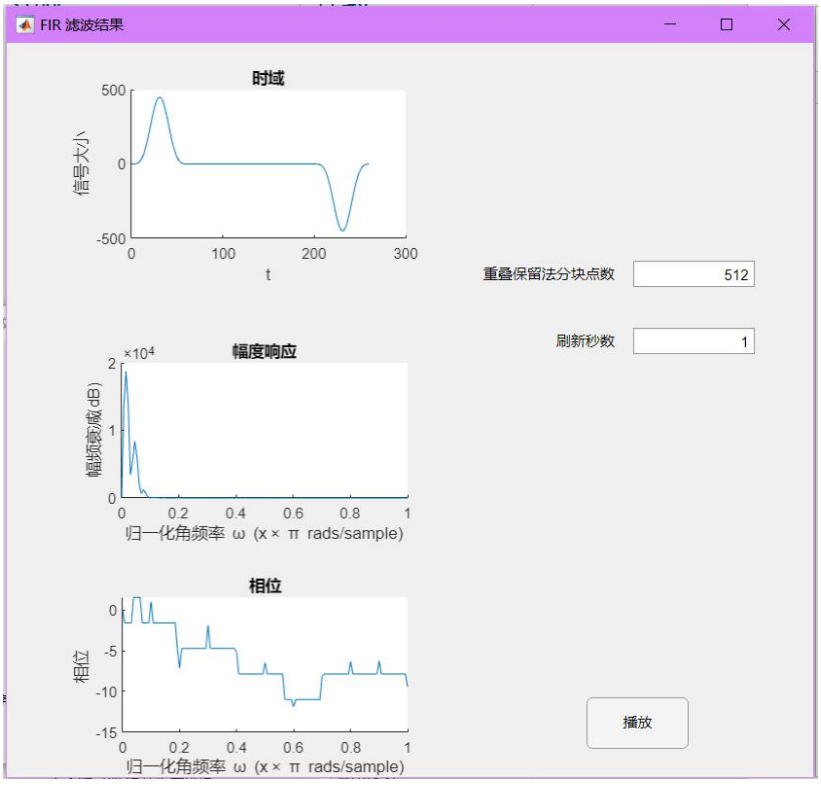

图 3.5-7 滤波结果与动态展示设置

4 总结

4.1 心得体会

虽然在实验之前我对 matlab GUI 的设计有些了解， 但是在做 GUI 的过程中还是学到了很多 GUI 与 matlab 提供的许多有用的 API 相关的内容，例如怎么使用 audioread()函数读取音频文件， 怎么使用 audiorecorder 对象录制音频等, 并且怎么实现各个窗口与模块间最大化的解耦，经过了前期查阅资料及函数的具体运用方法并且理出每道题的编程思路后中期便做到了较轻松自如的解决每一道问题，实现了音乐信号的各种窗函数的生成和比较、生成各种音乐信号并画出时域、频域、相位谱等谱线并加以分析，实现了按钮组和坐标轴的组合。并且还能看到有一些现象验证了理论，由于反正切函数的非线性会引起频率压缩现象，在使用了双线性变换法进行低通滤波器设计的图中, 可以看到幅度在靠近Π的地方迅速下降, 这是因为模拟角频率倍映射到了数字角频率的区间$[-\pi,\pi]$上, 这种映射的非线性的。

实现时，最难的逻辑部分就是到底如何实现 N 个多频正弦信号， 而不是固定的三五个，一开始甚至怀疑 matlab 中到底能否实现，不过最后经过严谨的设计与思考还是做出来了。

总体而言，实验指导书上所要求的内容我都完整的实现了，包括 n 个正弦信号的生成，动态展示重叠保留法都进行了实现，但是系统明显还存在一些比较明显的问题， 如没有根据特定的噪音进行滤波，例如高斯白噪声滤除效果非常差，图谱的分析还没办法做定量分析等。

4.2 系统仍存在的问题与解决方案

1. 高斯白噪声滤波不明显

当前使用低通,高通等简单的 IIR 与 FIR 滤波器无法较为明显的滤除高斯白噪声, 查阅了一些资料, 可以通过,均值滤波器,维纳滤波器来进行滤除。

2. 音频播放不可暂停及选择播放时间位置

利用 matlab 的 audioplay 对象对音频信号进一步封装, 实现对音频信号暂停, 恢复播放及时间位置选择的控制。

3. 没办法进定量分析图谱

在 matlab2018a 中没有实现的方法， 由于 matlab2018a 才引进 app designer 不久， 一些功能还在实现以及测试当中， 在 matlab2019a 中就开始引进了坐标轴鼠标移上去直接显示横纵坐标数值的功能。

5 参考文献

[1] Digital Signal Processing: A Computer-Based Approach，Sanjit K. K. Mitra，2000

[2] Understanding Digital Signal Processing，Richard G. Lyons，科学出版社

[3] Dual-tone multi-frequency [EB/OL] .http://en.wikipedia.org/wiki/Dual-tone_multi-frequency

[4] 数字信号处理.丁玉美等 西安电子科技大学出版社

[5] 数字信号处理教程（第二版），程佩青，清华大学出版社，2001

[6] 数字信号处理，赵树杰等，西电出版社，1997.10

[7] 数字信号处理—时域离散随机信号处理，丁玉美等，西电出版社，2002.11

[8] MATLAB 及在电子信息课程中的应用，陈怀琛等，电子工业出版社出版，2002.4

6 源代码附录

6.1 附录一 数字信号处理演示系统主模块

```
classdef mainApp < matlab.apps.AppBase
    % Properties that correspond to app components
    properties (Access = public)
        UIFigure     matlab.ui.Figure
        Label        matlab.ui.control.Label
        Label_2      matlab.ui.control.Label
        Lamp         matlab.ui.control.Lamp
        Lamp_2Label  matlab.ui.control.Label
        Lamp_2       matlab.ui.control.Lamp
        Label_3      matlab.ui.control.Label
        Lamp_3       matlab.ui.control.Lamp
        Button       matlab.ui.control.Button
        Button_2     matlab.ui.control.Button
        Button_3     matlab.ui.control.Button
        ButtonGroup  matlab.ui.container.ButtonGroup
        IIRButton    matlab.ui.control.RadioButton
        FIRButton    matlab.ui.control.RadioButton
    end
    properties (Access = public)
        % 原信号，不包括噪音
        originSig

        % 测试信号，包括噪音
        testSig

        % 噪音信号
        noiseSig

        % 输出信号
        outputSig

        % 信号类型
        sigType % 1: 多频正弦 2: 数字音频

        % 滤波器类型
        filterType % 1: IIR 滤波器 2: FIR 滤波器

        % FIR 滤波器时域
        filterSig

        % IIR 滤波器 H(z)分子与分母
        b % 分子
        a % 分母

        % 当前状态
        setTestSig % 已经设置测试信号
```

```
    setFilterDisign % 已经进行滤波器设计
    setDigitalFilter % 已经进行数字滤波
end
methods (Access = private)
    % Code that executes after component creation
    function startupFcn(app)
        app.filterType = 1;
    end
    % Button pushed function: Button
    function testSIgGenModBtn(app, event)
       testSigGenApp(app); % 打开测试信号生成模块
    end
    % Button pushed function: Button_2
    function openFilterDesinApp(app, event)

        if app.filterType == 1
            IIRfilterDesignApp(app);
        elseif app.filterType == 2
            FIRfilterDesignApp(app);
        end

    end
    % Selection changed function: ButtonGroup
    function filterTypeChangeCallback(app, event)
        selectedButton = app.ButtonGroup.SelectedObject;
        if app.IIRButton == selectedButton
            app.filterType = 1;
        elseif app.FIRButton == selectedButton
            app.filterType = 2;
        end
    end
    % Button pushed function: Button_3
    function digitalFilterApp(app, event)
        if app.filterType == 1
            % IIR 滤波器类型
            IIRFilter(app);
        elseif app.filterType == 2
            % FIR 滤波器类型
            FIRFilter(app);
        end
        app.Lamp_3.Color = 'g';
    end
end
% App initialization and construction
methods (Access = private)
    % Create UIFigure and components
    function createComponents(app)
        % Create UIFigure
        app.UIFigure = uifigure;
        app.UIFigure.Position = [100 100 632 347];
        app.UIFigure.Name = '数字信号处理演示系统';
        % Create Label
        app.Label = uilabel(app.UIFigure);
```

```
            app.Label.FontSize = 48;
            app.Label.Position = [72 241 492 69];
            app.Label.Text = '数字信号处理演示系统';
            % Create Label_2
            app.Label_2 = uilabel(app.UIFigure);
            app.Label_2.HorizontalAlignment = 'right';
            app.Label_2.Position = [72 149 77 22];
            app.Label_2.Text = '生成测试信号';
            % Create Lamp
            app.Lamp = uilamp(app.UIFigure);
            app.Lamp.Position = [164 149 20 20];
            app.Lamp.Color = [1 0 0];
            % Create Lamp_2Label
            app.Lamp_2Label = uilabel(app.UIFigure);
            app.Lamp_2Label.HorizontalAlignment = 'right';
            app.Lamp_2Label.Position = [84 97 65 22];
            app.Lamp_2Label.Text = '滤波器设计';
            % Create Lamp_2
            app.Lamp_2 = uilamp(app.UIFigure);
            app.Lamp_2.Position = [164 97 20 20];
            app.Lamp_2.Color = [1 0 0];
            % Create Label_3
            app.Label_3 = uilabel(app.UIFigure);
            app.Label_3.HorizontalAlignment = 'right';
            app.Label_3.Position = [12 43 137 22];
            app.Label_3.Text = '数字滤波与输出信号分析';
            % Create Lamp_3
            app.Lamp_3 = uilamp(app.UIFigure);
            app.Lamp_3.Position = [164 43 20 20];
            app.Lamp_3.Color = [1 0 0];
            % Create Button
            app.Button = uibutton(app.UIFigure, 'push');
            app.Button.ButtonPushedFcn = createCallbackFcn(app,
@testSIgGenModBtn, true);
            app.Button.BackgroundColor = [0.902 0.902 0.902];
            app.Button.Position = [196 140 312 40];
            app.Button.Text = '测试信号生成模块';
            % Create Button_2
            app.Button_2 = uibutton(app.UIFigure, 'push');
            app.Button_2.ButtonPushedFcn = createCallbackFcn(app,
@openFilterDesinApp, true);
            app.Button_2.BackgroundColor = [0.902 0.902 0.902];
            app.Button_2.Position = [196 87 312 40];
            app.Button_2.Text = '滤波器设计模块';
            % Create Button_3
            app.Button_3 = uibutton(app.UIFigure, 'push');
            app.Button_3.ButtonPushedFcn = createCallbackFcn(app,
@digitalFilterApp, true);
            app.Button_3.BackgroundColor = [0.902 0.902 0.902];
            app.Button_3.Position = [196 33 312 40];
            app.Button_3.Text = '数字滤波与输出信号分析模块';
            % Create ButtonGroup
            app.ButtonGroup = uibuttongroup(app.UIFigure);
```

```
            app.ButtonGroup.SelectionChangedFcn = createCallbackFcn(app,
@filterTypeChangeCallback, true);
            app.ButtonGroup.BorderType = 'none';
            app.ButtonGroup.FontSize = 10;
            app.ButtonGroup.Position = [515 87 100 40];
            % Create IIRButton
            app.IIRButton = uiradiobutton(app.ButtonGroup);
            app.IIRButton.Text = 'IIR';
            app.IIRButton.Position = [8 20 58 22];
            app.IIRButton.Value = true;
            % Create FIRButton
            app.FIRButton = uiradiobutton(app.ButtonGroup);
            app.FIRButton.Text = 'FIR';
            app.FIRButton.Position = [8 -1 65 22];
        end
    end
    methods (Access = public)
        % Construct app
        function app = mainApp
            % Create and configure components
            createComponents(app)
            % Register the app with App Designer
            registerApp(app, app.UIFigure)
            % Execute the startup function
            runStartupFcn(app, @startupFcn)
            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
```

6.2 附录二 测试信号生成模块

```
classdef testSigGenApp < matlab.apps.AppBase
    % Properties that correspond to app components
    properties (Access = public)
        UIFigure            matlab.ui.Figure
        UIAxes              matlab.ui.control.UIAxes
        readAudioFilebtn     matlab.ui.control.Button
        sigGenPanelBtn       matlab.ui.control.Button
        Label               matlab.ui.control.Label
        Label_3             matlab.ui.control.Label
        recordBtn           matlab.ui.control.StateButton
        originSIgSpectrum    matlab.ui.control.UIAxes
        originPlayBtn       matlab.ui.control.Button
        originSIgPhaseFig    matlab.ui.control.UIAxes
```

```
    testSigUIAxes        matlab.ui.control.UIAxes
    testSigSpectrum      matlab.ui.control.UIAxes
    testPlayBtn          matlab.ui.control.Button
    ButtonGroup          matlab.ui.container.ButtonGroup
    multiSineTypeBtn     matlab.ui.control.RadioButton
    digitalAudioTypeBtn  matlab.ui.control.RadioButton
    noiseGenBtn          matlab.ui.control.Button
    Button_9             matlab.ui.control.Button
    Button_10            matlab.ui.control.Button
    testSIgPhaseFig      matlab.ui.control.UIAxes
end
properties (Access = private)
    audioRecorderObj  % 音频录制对象, 用来录制,保存音频数据
end
properties (Access = public)
    originSig % 原信号 vector : 用来保存原信号数据

    testSig % 最终测试信号 vector : 用来保存测试信号数据

    sigType % 信号类型  1: 多频正弦 2: 数字音频

    noiseSig % 噪音信号 vector 保存噪音数据

    % callingApp 调用的 app mainApp
    callingApp

    isSetOrigin % 原信号是否设置
    isSetNoise % 信号噪音是否设置
    isSetFilter % 滤波器是否设置
end
methods (Access = public)

    function results = updateOrigin(app)
        % 更新原信号模块

        % app.noiseSig = zeros(size(app.originSig));

        app.isSetOrigin = 1; % 设置已经设置

        % 绘制原信号音频--时域图
        plot(app.UIAxes,app.originSig);

        % 更新数据
        N = length(app.originSig); % 原信号点数
        fft_y = fft(app.originSig); % 计算 fft

        n = linspace(0 , N-1,N); % 第 n 点
        w = 2*pi*n./N; % 角频率(未归一化)

        % 绘制原信号音频--频谱图
        plot(app.originSIgSpectrum,w/pi,abs(fft_y));  % w 归一化
```

```
    % 绘制原信号音频--相频图
    plot(app.originSIgPhaseFig,w/pi,unwrap(angle(fft_y)));

end

function results = updateTest(app)
    % 更新测试信号模块

    % 根据信号类型生成测试信号
    if app.sigType == 1
        app.testSig = app.originSig;
    elseif app.sigType==2
        app.testSig = app.originSig + app.noiseSig;
    end
    %app.isSetOrigin = 1; % 设置已经设置

    % 绘制原信号音频--时域图
    plot(app.testSigUIAxes,app.testSig);

    % 更新数据
    N = length(app.testSig); % 原信号点数
    fft_y = fft(app.testSig); % 计算 fft

    n = linspace(0 , N-1,N); % 第 n 点
    w = 2*pi*n./N; % 角频率(未归一化)

    % 绘制原信号音频--频谱图
    plot(app.testSigSpectrum,w/pi,abs(fft_y));  % w 归一化

    % 绘制原信号音频--相频图
    plot(app.testSIgPhaseFig,w/pi,unwrap(angle(fft_y)));
end


function results = redrawPanel(app)
    % 重新绘制 Panel 数据


end

function results = updateNoise(app)
    % 更新噪音信号模块

    % 代表已经设置噪音信号
    app.isSetNoise = 1;

    % 绘制噪音信号音频--时域图
    plot(app.testSigUIAxes,app.noiseSig);

    % 更新数据
    N = length(app.noiseSig); % 噪音信号点数
    fft_y = fft(app.noiseSig); % 计算 fft
```

```
        n = linspace(0 , N-1,N); % 第 n 点
        w = 2*pi*n./N; % 角频率(未归一化)

        % 绘制噪音信号--频谱图
        plot(app.testSigSpectrum,w/pi,abs(fft_y));  % w 归一化

        % 赋值最终测试信号
        app.callingApp.testSig = app.originSig+app.noiseSig;
    end



    function [] = updateSigPanel(app, panelType)
        % 更新面板显示
        if panelType==1
            app.readAudioFilebtn.Visible = 'off';
            app.recordBtn.Visible='off';
            app.sigGenPanelBtn.Visible='on';

            app.noiseGenBtn.Visible = 'off'; % 不显示噪音模块

        elseif panelType==2
            app.readAudioFilebtn.Visible = 'on';
            app.recordBtn.Visible='on';
            app.sigGenPanelBtn.Visible='off';

            if app.sigType ==2
                app.noiseGenBtn.Visible='on';
            end
        end

    end
end
methods (Access = private)
    % Code that executes after component creation
    function startupFcn(app, main)
        % 隐藏正弦函数叠加模块
        %set(app.SineSigGeneratePanel,'visible','off');
        % 初始化音频录制对象
        app.audioRecorderObj = audiorecorder(44100,16,1);

        app.isSetOrigin = 0;
        app.isSetNoise = 0;
        app.isSetFilter = 0;
        app.sigType=1; % 默认为 1:多频正弦

        updateSigPanel(app,1);

        if nargin >=2
            % 更新呼叫的 app
            app.callingApp=main;
            app.sigType = main.sigType;
```

```
                % 如果设置了
                main.setTestSig
                if main.setTestSig == 1
                    app.originSig = main.originSig;
                    app.noiseSig = main.noiseSig;
                    updateOrigin(app); % 更新原信号
                    updateTest(app); % 更新测试信号
                end

            end


        end
        % Button pushed function: readAudioFilebtn
        function readAudioFilebtnPushed(app, event)

            % 打开文件选择框
            [file,path] =
uigetfile({'*.wav';'*.ogg';'*.flac';'*.au';'*.aiff';'*.aif';'*.aifc';'*.
mp3';'*.m4a';'*.mp4';'*.*'},...
                        'Surpported Audio File');

            % 如果选择了文件那么打开音频文件
            if path ~= 0
                % 获得文件地址
                audio_file_path = [path,file];
                % 打开音频文件
                [app.originSig,Fs] = audioread(audio_file_path);
                app.originSig = app.originSig(:,1);
                app.sigType = 2; % 数字音频
                app.noiseSig = zeros(size(app.originSig)); % 设置噪音为0
                updateOrigin(app); % 更新原信号
                updateTest(app); % 更新测试信号
                updateSigPanel(app,2); % 显示噪音模块
            end
        end
        % Value changed function: recordBtn
        function recordBtnValueChanged(app, event)
            value = app.recordBtn.Value;

            % 按下录制按钮
            if value== 1

                record(app.audioRecorderObj); % 开始录制
            % 结束录制
            else
                if ~isempty(app.audioRecorderObj)
                    app.originSig = getaudiodata(app.audioRecorderObj); %
获取录制数据
                    stop(app.audioRecorderObj); % 停止录制(貌似会销毁
audioRecorder对象)
```

```
                    app.audioRecorderObj = audiorecorder; % 重新赋值避免空异
常
                    app.sigType = 2; % 数字音频
                    app.noiseSig = zeros(size(app.originSig)); % 设置噪音为
0
                    updateOrigin(app); % 更新
                    updateTest(app); % 更新测试信号
                    updateSigPanel(app,2); % 显示噪音模块
                end
            end
        end
        % Button pushed function: originPlayBtn
        function originPlayBtnPushed(app, event)
            % 播放原信号(生成的信号)
            sound(app.originSig, 44100);
        end
        % Button pushed function: sigGenPanelBtn
        function FreeGenerateSignal(app, event)


            multiSineGenApp(app);  % 打开正弦信号生成模块


        end
        % Callback function
        function closePanelBtnButtonPushed(app, event)
            set(app.SineSigGeneratePanel,'visible','off');
        end
        % Callback function
        function singQuitButtonPushed(app, event)
            % 退出控制面板
            set(app.SineSigGeneratePanel,'visible','off');
        end
        % Callback function
        function sineConfirmButtonPushed(app, event)
            set(app.SineSigGeneratePanel,'visible','on');
        end
        % Callback function
        function noiseFileRead(app, event)

            % 打开文件选择框
            [file,path] =
uigetfile({'*.wav';'*.ogg';'*.flac';'*.au';'*.aiff';'*.aif';'*.aifc';'*.
mp3';'*.m4a';'*.mp4';'*.*'},...
                        'Surpported Audio File');

            % 如果选择了文件那么打开音频文件
            if path ~= 0
                % 获得文件地址
                audio_file_path = [path,file];
                % 打开音频文件
                [app.noiseSig,Fs] = audioread(audio_file_path);
```

```
                updateNoise(app); % 更新噪音
            end
        end
        % Button pushed function: testPlayBtn
        function testPlayBtnCallback(app, event)
            sound(app.testSig,44100);
        end
        % Callback function
        function noiseGenerateCallback(app, event)
            noiseGenApp(app);
        end
        % Selection changed function: ButtonGroup
        function sigTypeChangeCallback(app, event)
            if app.multiSineTypeBtn == app.ButtonGroup.SelectedObject
                panelType= 1; % 多频正弦
            elseif app.digitalAudioTypeBtn ==
app.ButtonGroup.SelectedObject
                panelType= 2; % 数字音频
            end
            updateSigPanel(app,panelType);
        end
        % Button pushed function: noiseGenBtn
        function noiseGenBtnCallback(app, event)
            noiseGenApp(app);
        end
        % Button pushed function: Button_9
        function sigGenApply(app, event)
            app.sigType
            app.callingApp.originSig = app.originSig;
            app.callingApp.setTestSig = 1;
            if app.sigType==1
                app.callingApp.sigType=1;
                app.callingApp.testSig=app.originSig;
            elseif app.sigType==2


                app.callingApp.sigType=2;
                app.callingApp.originSig = app.originSig;
                app.callingApp.testSig = app.originSig + app.noiseSig; % 包
括噪音信号
                app.callingApp.noiseSig = app.noiseSig;
            end
            app.callingApp.Lamp.Color = 'g';
        end
        % Button pushed function: Button_10
        function sigGenConfirm(app, event)
            sigGenApply(app);

            delete(app);
        end
    end
    % App initialization and construction
    methods (Access = private)
```

```
        % Create UIFigure and components
        function createComponents(app)
            % Create UIFigure
            app.UIFigure = uifigure;
            app.UIFigure.Position = [100 100 922 751];
            app.UIFigure.Name = '测试信号生成模块';
            % Create UIAxes
            app.UIAxes = uiaxes(app.UIFigure);
            title(app.UIAxes, '时域图')
            xlabel(app.UIAxes, 'n')
            ylabel(app.UIAxes, '信号幅值')
            app.UIAxes.PlotBoxAspectRatio = [1 0.290672451193059
0.290672451193059];
            app.UIAxes.Box = 'on';
            app.UIAxes.TitleFontWeight = 'bold';
            app.UIAxes.Position = [23 535 420 165];
            % Create readAudioFilebtn
            app.readAudioFilebtn = uibutton(app.UIFigure, 'push');
            app.readAudioFilebtn.ButtonPushedFcn = createCallbackFcn(app,
@readAudioFilebtnPushed, true);
            app.readAudioFilebtn.Position = [66 52 180 28];
            app.readAudioFilebtn.Text = '从文件中读取';
            % Create sigGenPanelBtn
            app.sigGenPanelBtn = uibutton(app.UIFigure, 'push');
            app.sigGenPanelBtn.ButtonPushedFcn = createCallbackFcn(app,
@FreeGenerateSignal, true);
            app.sigGenPanelBtn.Position = [66 53 375 26];
            app.sigGenPanelBtn.Text = '正弦信号生成';
            % Create Label
            app.Label = uilabel(app.UIFigure);
            app.Label.Position = [208 719 101 22];
            app.Label.Text = '测试信号生成模块';
            % Create Label_3
            app.Label_3 = uilabel(app.UIFigure);
            app.Label_3.Position = [677 719 77 22];
            app.Label_3.Text = '最终测试信号';
            % Create recordBtn
            app.recordBtn = uibutton(app.UIFigure, 'state');
            app.recordBtn.ValueChangedFcn = createCallbackFcn(app,
@recordBtnValueChanged, true);
            app.recordBtn.Text = '语音输入';
            app.recordBtn.Position = [255 53 186 26];
            % Create originSIgSpectrum
            app.originSIgSpectrum = uiaxes(app.UIFigure);
            title(app.originSIgSpectrum, '幅频图')
            xlabel(app.originSIgSpectrum, 'w/\pi')
            ylabel(app.originSIgSpectrum, '信号幅值')
            app.originSIgSpectrum.PlotBoxAspectRatio = [1
0.290672451193059 0.290672451193059];
            app.originSIgSpectrum.Box = 'on';
            app.originSIgSpectrum.TitleFontWeight = 'bold';
            app.originSIgSpectrum.Position = [21 348 420 165];
            % Create originPlayBtn
```

```
            app.originPlayBtn = uibutton(app.UIFigure, 'push');
            app.originPlayBtn.ButtonPushedFcn = createCallbackFcn(app,
@originPlayBtnPushed, true);
            app.originPlayBtn.Position = [66 9 375 30];
            app.originPlayBtn.Text = '播放';
            % Create originSIgPhaseFig
            app.originSIgPhaseFig = uiaxes(app.UIFigure);
            title(app.originSIgPhaseFig, '相频图')
            xlabel(app.originSIgPhaseFig, 'w/\pi')
            ylabel(app.originSIgPhaseFig, '信号相位')
            app.originSIgPhaseFig.PlotBoxAspectRatio = [1
0.290672451193059 0.290672451193059];
            app.originSIgPhaseFig.YGrid = 'on';
            app.originSIgPhaseFig.TitleFontWeight = 'bold';
            app.originSIgPhaseFig.Position = [23 161 420 165];
            % Create testSigUIAxes
            app.testSigUIAxes = uiaxes(app.UIFigure);
            title(app.testSigUIAxes, '时域图')
            xlabel(app.testSigUIAxes, 'n')
            ylabel(app.testSigUIAxes, '信号幅值')
            app.testSigUIAxes.PlotBoxAspectRatio = [1 0.290672451193059
0.290672451193059];
            app.testSigUIAxes.TitleFontWeight = 'bold';
            app.testSigUIAxes.Position = [471 529 432 177];
            % Create testSigSpectrum
            app.testSigSpectrum = uiaxes(app.UIFigure);
            title(app.testSigSpectrum, '幅频图')
            xlabel(app.testSigSpectrum, 'w/\pi')
            ylabel(app.testSigSpectrum, '信号幅值')
            app.testSigSpectrum.PlotBoxAspectRatio = [1 0.290672451193059
0.290672451193059];
            app.testSigSpectrum.TitleFontWeight = 'bold';
            app.testSigSpectrum.Position = [471 339 432 197];
            % Create testPlayBtn
            app.testPlayBtn = uibutton(app.UIFigure, 'push');
            app.testPlayBtn.ButtonPushedFcn = createCallbackFcn(app,
@testPlayBtnCallback, true);
            app.testPlayBtn.Position = [517 52 386 28];
            app.testPlayBtn.Text = '播放';
            % Create ButtonGroup
            app.ButtonGroup = uibuttongroup(app.UIFigure);
            app.ButtonGroup.SelectionChangedFcn = createCallbackFcn(app,
@sigTypeChangeCallback, true);
            app.ButtonGroup.Title = '信号类型选择';
            app.ButtonGroup.Position = [66 100 241 44];
            % Create multiSineTypeBtn
            app.multiSineTypeBtn = uiradiobutton(app.ButtonGroup);
            app.multiSineTypeBtn.Text = '多频正弦';
            app.multiSineTypeBtn.Position = [11 1 70 22];
            app.multiSineTypeBtn.Value = true;
            % Create digitalAudioTypeBtn
            app.digitalAudioTypeBtn = uiradiobutton(app.ButtonGroup);
            app.digitalAudioTypeBtn.Text = '数字音频';
```

```
            app.digitalAudioTypeBtn.Position = [99 1 70 22];
            % Create noiseGenBtn
            app.noiseGenBtn = uibutton(app.UIFigure, 'push');
            app.noiseGenBtn.ButtonPushedFcn = createCallbackFcn(app,
@noiseGenBtnCallback, true);
            app.noiseGenBtn.Position = [517 91 387 27];
            app.noiseGenBtn.Text = '噪音生成';
            % Create Button_9
            app.Button_9 = uibutton(app.UIFigure, 'push');
            app.Button_9.ButtonPushedFcn = createCallbackFcn(app,
@sigGenApply, true);
            app.Button_9.Position = [517 14 186 25];
            app.Button_9.Text = '应用';
            % Create Button_10
            app.Button_10 = uibutton(app.UIFigure, 'push');
            app.Button_10.ButtonPushedFcn = createCallbackFcn(app,
@sigGenConfirm, true);
            app.Button_10.Position = [719 13 185 26];
            app.Button_10.Text = '确定';
            % Create testSIgPhaseFig
            app.testSIgPhaseFig = uiaxes(app.UIFigure);
            title(app.testSIgPhaseFig, '相频图')
            xlabel(app.testSIgPhaseFig, 'w/\pi')
            ylabel(app.testSIgPhaseFig, '信号相位')
            app.testSIgPhaseFig.PlotBoxAspectRatio = [1 0.290672451193059
0.290672451193059];
            app.testSIgPhaseFig.YGrid = 'on';
            app.testSIgPhaseFig.TitleFontWeight = 'bold';
            app.testSIgPhaseFig.Position = [471 161 432 165];
        end
    end
    methods (Access = public)
        % Construct app
        function app = testSigGenApp(varargin)
            % Create and configure components
            createComponents(app)
            % Register the app with App Designer
            registerApp(app, app.UIFigure)
            % Execute the startup function
            runStartupFcn(app, @(app)startupFcn(app, varargin{:}))
            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
```

6.3 附录三 多频正弦信号生成模块

```
classdef multiSineGenApp < matlab.apps.AppBase
    % Properties that correspond to app components
    properties (Access = public)
        UIFigure        matlab.ui.Figure
        Label_6         matlab.ui.control.Label
        sineConfirm     matlab.ui.control.Button
        singQuit        matlab.ui.control.Button
        sinePosSpinner  matlab.ui.control.Spinner
        HzLabel         matlab.ui.control.Label
        HzLabel_2       matlab.ui.control.Label
        Label           matlab.ui.control.Label
        Label_2         matlab.ui.control.Label
        Label_3         matlab.ui.control.Label
        sineSampleFreq  matlab.ui.control.NumericEditField
        EditFieldLabel  matlab.ui.control.Label
        sampleNum       matlab.ui.control.NumericEditField
        Label_4         matlab.ui.control.Label
        sineNum         matlab.ui.control.Spinner
        Label_5         matlab.ui.control.Label
        sinePosFreq     matlab.ui.control.NumericEditField
        sineGenFig      matlab.ui.control.UIAxes
        Button          matlab.ui.control.Button
        sineGenspectum  matlab.ui.control.UIAxes
    end
    properties (Access = public)
        % 调用该窗口的窗口实例
        callingApp
        % sine 信号
        sineSig = 0

        % 正弦信号个数
        sn

        % 频率列表 1..sn
        f

        % 采样点数
        N

        % 指定的正弦函数
        sinFunc = @(f, t) 100*sin(2*pi*f*t); %正弦函数

        % 采样频率
        fs

        % 默认频率 1kHz
        SINE_FREQ = 1000

    end
    methods (Access = public)
```

```
        function [] = updateCurrentFreq(app)
            % 获取当前设置的位置
            pos = app.sinePosSpinner.Value;

            % 设置当前设置的频率值
            app.f(pos) = app.sinePosFreq.Value;
        end

        function [] = draw(app)
        % ====根据属性数据绘制图形====

            % 初始化正弦信号值
            app.sineSig=zeros(app.N,1); % 初始化正弦数据列表

            % 计算采样时间位置
            np = app.N; % 采样点数
            n = linspace(0 , np-1,np);  % vector 采样向量 [0..N-1]
            sample_t = n' / app.fs; %  Nx1 vector 采样时间位置 = 采样点/采样
频率

            % 遍历采样共 sn 个正弦信号
            for i=1:app.sn
                sin_i = app.sinFunc(app.f(i) , sample_t); % Nx1 f(i)为第 i
个的频率, sample_t 为采样时间点
                app.sineSig =app.sineSig +sin_i;   % Nx1
            end

            %  绘制采样后的图
             stem(app.sineGenFig, sample_t, app.sineSig); % 绘制采样后的图
        end

        function [] = setAndDraw(app)
        % 设置信号属性并且将信号其绘制

            % 初始化正弦信号值
            app.sineSig=zeros(app.N,1); % 初始化正弦数据列表

            % 计算采样时间位置
            np = app.N; % 采样点数
            n = linspace(0 , np-1,np);  % vector 采样向量 [0..N-1]
            sample_t = n' ./ app.fs; %  Nx1 vector 采样时间位置 = 采样点/采
样频率

            % 遍历采样共 sn 个正弦信号
            for i=1:app.sn
                sin_i = app.sinFunc(app.f(i) , sample_t); % Nx1 f(i)为第 i
个的频率, sample_t 为采样时间点
                app.sineSig =app.sineSig +sin_i;   % Nx1
            end

            %  绘制采样后的图
             stem(app.sineGenFig, sample_t, app.sineSig); % 绘制采样后的图
```

```
            % ====绘制频谱图====
            NN = length( app.sineSig); % 原信号点数
            fft_y = fft( app.sineSig); % 计算 fft
            n = linspace(0 , NN-1,NN); % 第 n 点
            w = 2*pi*n./NN; % 角频率(未归一化)
            plot(app.sineGenspectum,w/pi,abs(fft_y));  % w 归一化

            % ====绘制相频图====

        end

    end
    methods (Access = private)

        function [] = reset(app, which)
            % 控制面板重置

            if(nargin < 2)
            % 未指定重置哪块内容,全部重置


            else
            % 重置指定模块

                if(strcmp(which, 'single'))
                % 重置单频控制模块
                    app.sinePosFreq.Value = app.SINE_FREQ;
                    app.sinePosSpinner.Value = 1; % 默认第一个是 默认频率
                end

            end

        end

        function [] = setSignal(app)
        % 根据现有数据来建立信号, 生成 sineSig 属性

            % 初始化正弦信号值
            app.sineSig=zeros(app.N,1); % 初始化正弦数据列表

            % 计算采样时间位置
            np = app.N; % 采样点数
            n = linspace(0 , np-1,np);  % vector 采样向量 [0..N-1]
            sample_t = n' / app.fs; %  Nx1 vector 采样时间位置 = 采样点/采样
频率

            % 遍历生成正弦信号
            for i=1:app.sn
                sin_i = app.sinFunc(app.f(i) , sample_t); % Nx1 f(i)为第 i
个的频率, sample_t 为采样时间点
                app.sineSig =app.sineSig +sin_i;   % Nx1
```

```
        end

    end

end
methods (Access = private)
    % Code that executes after component creation
    function starupFcn(app, mainapp)


        app.callingApp = mainapp; % 保存调用该窗口的窗口对象


        app.sn = app.sineNum.Value; % 正弦信号个数
        app.fs= app.sineSampleFreq.Value; % 更新采样频率
        app.N= app.sampleNum.Value; % 更新采样点数
        app.f = ones(app.sn,1) .* app.SINE_FREQ; % 初始化频率值列表

        % 初始绘制一个图
        setAndDraw(app);

    end
    % Callback function
    function openApp(app, event)




    end
    % Button pushed function: sineConfirm
    function sineConfirmCallback(app, event)

        % 应用
        sineApply(app, event);

        % 退出
        delete(app);

    end
    % Callback function: sineNum
    function sineNumChangeCallback(app, event)
    % 更新正弦信号的个数

        % 重置单一正弦信号控制块
        reset(app, 'single');

        % 正弦信号数量
        app.sn = app.sineNum.Value;  % scalar

        % 重置频率列表 (利用默认频率)
        app.f = ones(app.sn,1) .* app.SINE_FREQ;
```

```
    % 绘制图形
    setAndDraw(app);
end
% Value changed function: sineSampleFreq
function freqUpdateCallback(app, event)
    app.fs= app.sineSampleFreq.Value; % 更新采样频率

    % 重新绘制图形
    setAndDraw(app);
end
% Value changed function: sampleNum
function sampleNumValueChanged(app, event)

    % 更新采样点数
    app.N= app.sampleNum.Value;

    % 重新绘制
    setAndDraw(app);
end
% Value changed function: sinePosFreq
function posFreqChange(app, event)

    % 更新当前指定位置正弦信号的频率
    updateCurrentFreq(app);

    % 更新信号且绘制
    setAndDraw(app);
end
% Callback function: sinePosSpinner
function posChange(app, event)

    % 获取更新位置
    pos = app.sinePosSpinner.Value;

    if(pos <= app.sineNum.Value )
    % 如果未越界则更新
        app.sinePosFreq.Value = app.f(pos);
    else
    % 越界了就固定为最高值
        app.sinePosSpinner.Value = app.sineNum.Value + 0.0;
    end

end
% Button pushed function: Button
function sineApply(app, event)

    % 更新数据
    app.callingApp.originSig = app.sineSig;

     % 多频正弦信号
    app.callingApp.sigType = 1;
    app.callingApp.updateTest();
```

```
            % 绘制图像
            app.callingApp.updateOrigin();
        end
        % Button pushed function: singQuit
        function quit(app, event)
            delete(app);
        end
    end
    % App initialization and construction
    methods (Access = private)
        % Create UIFigure and components
        function createComponents(app)
            % Create UIFigure
            app.UIFigure = uifigure;
            app.UIFigure.Position = [100 100 656 634];
            app.UIFigure.Name = '正弦信号生成系统';
            % Create Label_6
            app.Label_6 = uilabel(app.UIFigure);
            app.Label_6.FontSize = 20;
            app.Label_6.Position = [226 601 206 26];
            app.Label_6.Text = '多频正弦信号生成模块';
            % Create sineConfirm
            app.sineConfirm = uibutton(app.UIFigure, 'push');
            app.sineConfirm.ButtonPushedFcn = createCallbackFcn(app,
@sineConfirmCallback, true);
            app.sineConfirm.Position = [421 232 100 25];
            app.sineConfirm.Text = '确认';
            % Create singQuit
            app.singQuit = uibutton(app.UIFigure, 'push');
            app.singQuit.ButtonPushedFcn = createCallbackFcn(app, @quit,
true);
            app.singQuit.Position = [421 123 100 26];
            app.singQuit.Text = '退出';
            % Create sinePosSpinner
            app.sinePosSpinner = uispinner(app.UIFigure);
            app.sinePosSpinner.ValueChangingFcn = createCallbackFcn(app,
@posChange, true);
            app.sinePosSpinner.Limits = [1 Inf];
            app.sinePosSpinner.RoundFractionalValues = 'on';
            app.sinePosSpinner.ValueChangedFcn = createCallbackFcn(app,
@posChange, true);
            app.sinePosSpinner.HorizontalAlignment = 'center';
            app.sinePosSpinner.Position = [84 40 178 65];
            app.sinePosSpinner.Value = 1;
            % Create HzLabel
            app.HzLabel = uilabel(app.UIFigure);
            app.HzLabel.Position = [601 61 25 22];
            app.HzLabel.Text = 'Hz';
            % Create HzLabel_2
            app.HzLabel_2 = uilabel(app.UIFigure);
            app.HzLabel_2.Position = [342 180 25 22];
            app.HzLabel_2.Text = 'Hz';
            % Create Label
```

```
            app.Label = uilabel(app.UIFigure);
            app.Label.Position = [342 125 25 22];
            app.Label.Text = '点';
            % Create Label_2
            app.Label_2 = uilabel(app.UIFigure);
            app.Label_2.Position = [342 233 25 22];
            app.Label_2.Text = '个';
            % Create Label_3
            app.Label_3 = uilabel(app.UIFigure);
            app.Label_3.HorizontalAlignment = 'right';
            app.Label_3.Position = [168 180 53 22];
            app.Label_3.Text = '采样频率';
            % Create sineSampleFreq
            app.sineSampleFreq = uieditfield(app.UIFigure, 'numeric');
            app.sineSampleFreq.Limits = [1 Inf];
            app.sineSampleFreq.ValueDisplayFormat = '%.1f';
            app.sineSampleFreq.ValueChangedFcn = createCallbackFcn(app,
@freqUpdateCallback, true);
            app.sineSampleFreq.Position = [236 180 100 22];
            app.sineSampleFreq.Value = 2000;
            % Create EditFieldLabel
            app.EditFieldLabel = uilabel(app.UIFigure);
            app.EditFieldLabel.HorizontalAlignment = 'right';
            app.EditFieldLabel.Position = [168 125 53 22];
            app.EditFieldLabel.Text = '采样点数';
            % Create sampleNum
            app.sampleNum = uieditfield(app.UIFigure, 'numeric');
            app.sampleNum.Limits = [1 Inf];
            app.sampleNum.ValueDisplayFormat = '%.0f';
            app.sampleNum.ValueChangedFcn = createCallbackFcn(app,
@sampleNumValueChanged, true);
            app.sampleNum.Position = [236 125 100 22];
            app.sampleNum.Value = 200;
            % Create Label_4
            app.Label_4 = uilabel(app.UIFigure);
            app.Label_4.HorizontalAlignment = 'right';
            app.Label_4.Position = [145 233 77 22];
            app.Label_4.Text = '正弦信号数量';
            % Create sineNum
            app.sineNum = uispinner(app.UIFigure);
            app.sineNum.ValueChangingFcn = createCallbackFcn(app,
@sineNumChangeCallback, true);
            app.sineNum.Limits = [1 Inf];
            app.sineNum.ValueChangedFcn = createCallbackFcn(app,
@sineNumChangeCallback, true);
            app.sineNum.Position = [236 233 100 22];
            app.sineNum.Value = 1;
            % Create Label_5
            app.Label_5 = uilabel(app.UIFigure);
            app.Label_5.HorizontalAlignment = 'right';
            app.Label_5.Position = [38 61 344 22];
            app.Label_5.Text = '第
个正弦信号频率设定';
```

```
            % Create sinePosFreq
            app.sinePosFreq = uieditfield(app.UIFigure, 'numeric');
            app.sinePosFreq.Limits = [0 Inf];
            app.sinePosFreq.ValueChangedFcn = createCallbackFcn(app,
@posFreqChange, true);
            app.sinePosFreq.Position = [390 61 199 22];
            app.sinePosFreq.Value = 1000;
            % Create sineGenFig
            app.sineGenFig = uiaxes(app.UIFigure);
            title(app.sineGenFig, '时域图')
            xlabel(app.sineGenFig, 't / s')
            ylabel(app.sineGenFig, 'y')
            app.sineGenFig.PlotBoxAspectRatio = [1 0.505285412262156
0.505285412262156];
            app.sineGenFig.TitleFontWeight = 'bold';
            app.sineGenFig.Position = [1 303 321 283];
            % Create Button
            app.Button = uibutton(app.UIFigure, 'push');
            app.Button.ButtonPushedFcn = createCallbackFcn(app, @sineApply,
true);
            app.Button.Position = [421 177 100 25];
            app.Button.Text = '应用';
            % Create sineGenspectum
            app.sineGenspectum = uiaxes(app.UIFigure);
            title(app.sineGenspectum, '幅频图')
            xlabel(app.sineGenspectum, '\omega / \pi')
            ylabel(app.sineGenspectum, '|H(\omega)|')
            app.sineGenspectum.PlotBoxAspectRatio = [1 0.505285412262156
0.505285412262156];
            app.sineGenspectum.TitleFontWeight = 'bold';
            app.sineGenspectum.Position = [321 287 323 315];
        end
    end
    methods (Access = public)
        % Construct app
        function app = multiSineGenApp(varargin)
            % Create and configure components
            createComponents(app)
            % Register the app with App Designer
            registerApp(app, app.UIFigure)
            % Execute the startup function
            runStartupFcn(app, @(app)starupFcn(app, varargin{:}))
            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
```

6.4 附录四 噪音生成模块

```
classdef noiseGenApp < matlab.apps.AppBase
    % Properties that correspond to app components
    properties (Access = public)
        UIFigure            matlab.ui.Figure
        Label_6             matlab.ui.control.Label
        sineConfirm         matlab.ui.control.Button
        sinePosSpinner      matlab.ui.control.Spinner
        HzLabel             matlab.ui.control.Label
        Label_2             matlab.ui.control.Label
        Label_4             matlab.ui.control.Label
        sineNum             matlab.ui.control.Spinner
        Label_5             matlab.ui.control.Label
        sinePosFreq         matlab.ui.control.NumericEditField
        sineGenFig          matlab.ui.control.UIAxes
        Button              matlab.ui.control.Button
        Label_7             matlab.ui.control.Label
        ListBox             matlab.ui.control.ListBox
        sineGenSpectum      matlab.ui.control.UIAxes
        noisePlayBtn        matlab.ui.control.Button
        gaussianNoiseGenBtn  matlab.ui.control.Button
    end
    properties (Access = public)
        % 调用该窗口的窗口实例
        callingApp
        % sine 信号
        sineSig = 0

        % 正弦信号个数
        sn

        % 频率列表 1..sn
        f

        % 采样点数
        N

        % 指定的正弦函数
        sinFunc = @(f, t) 0.005*sin(2*pi*f*t); %正弦函数 幅值: 0.005

        % 采样频率
        fs

        % 默认频率 1kHz
        SINE_FREQ = 1000

        % 最终噪音信号
        noiseSig

    end
```

```
    methods (Access = public)

        function [] = updateCurrentFreq(app)
            % 获取当前设置的位置
            pos = app.sinePosSpinner.Value;

            % 设置当前设置的频率值
            app.f(pos) = app.sinePosFreq.Value;
        end


        function [] = updateNoiseSig(app)


            NN = length( app.noiseSig); % 原信号点数
            n = linspace(0 , NN-1,NN); % 第 n 点

            % ====绘制时域图====
            stem(app.sineGenFig, n, app.noiseSig); % 绘制采样后的图

            % ====绘制频谱图====

            fft_y = fft( app.noiseSig); % 计算 fft
            w = 2*pi*n./NN; % 角频率(未归一化)
            plot(app.sineGenSpectum,w/pi,abs(fft_y));  % w 归一化
        end


        function [] = draw(app)
        % ====根据属性数据绘制图形====

            % 初始化正弦信号值
            app.sineSig=zeros(app.N,1); % 初始化正弦数据列表

            % 计算采样时间位置
            np = app.N; % 采样点数
            n = linspace(0 , np-1,np);  % vector 采样向量 [0..N-1]
            sample_t = n' / app.fs; %  Nx1 vector 采样时间位置 = 采样点/采样
频率

            % 遍历采样共 sn 个正弦信号
            for i=1:app.sn
                sin_i = app.sinFunc(app.f(i) , sample_t); % Nx1 f(i)为第 i
个的频率, sample_t 为采样时间点
                app.sineSig =app.sineSig +sin_i;   % Nx1
            end

            %  绘制采样后的图
             stem(app.sineGenFig, sample_t, app.sineSig); % 绘制采样后的图
        end

        function [] = setAndDraw(app)
```

```
        % 设置信号属性并且将信号其绘制

            % 初始化正弦信号值
            app.sineSig=zeros(app.N,1); % 初始化正弦数据列表

            % 计算采样时间位置
            np = app.N; % 采样点数
            n = linspace(0 , np-1,np);  % vector 采样向量 [0..N-1]
            sample_t = n' ./ app.fs; %  Nx1 vector 采样时间位置 = 采样点/采
样频率

            % 遍历采样共 sn 个正弦信号
            for i=1:app.sn
                sin_i = app.sinFunc(app.f(i) , sample_t); % Nx1 f(i)为第 i
个的频率, sample_t 为采样时间点
                app.sineSig =app.sineSig +sin_i;   % Nx1
            end

            %  绘制采样后的图
             stem(app.sineGenFig, sample_t, app.sineSig); % 绘制采样后的图

            % ====绘制频谱图====
            NN = length( app.sineSig); % 原信号点数
            fft_y = fft( app.sineSig); % 计算 fft
            n = linspace(0 , NN-1,NN); % 第 n 点
            w = 2*pi*n./NN; % 角频率(未归一化)
            plot(app.sineGenSpectum,w/pi,abs(fft_y));  % w 归一化

            % 设置噪音信号
            app.noiseSig = app.sineSig;
        end

    end
    methods (Access = private)

        function [] = reset(app, which)
            % 控制面板重置

            if(nargin < 2)
            % 未指定重置哪块内容,全部重置


            else
            % 重置指定模块

                if(strcmp(which, 'single'))
                % 重置单频控制模块
                    app.sinePosFreq.Value = app.SINE_FREQ;
                    app.sinePosSpinner.Value = 1; % 默认第一个是 默认频率
                end

            end
```

```
        end

        function [] = setSignal(app)
        % 根据现有数据来建立信号，生成 sineSig 属性

            % 初始化正弦信号值
            app.sineSig=zeros(app.N,1); % 初始化正弦数据列表

            % 计算采样时间位置
            np = app.N; % 采样点数
            n = linspace(0 , np-1,np);  % vector 采样向量 [0..N-1]
            sample_t = n' / app.fs; %  Nx1 vector 采样时间位置 = 采样点/采样
频率

            % 遍历生成正弦信号
            for i=1:app.sn
                sin_i = app.sinFunc(app.f(i) , sample_t); % Nx1 f(i)为第 i
个的频率，sample_t 为采样时间点
                app.sineSig =app.sineSig +sin_i;   % Nx1
            end

        end

    end
    methods (Access = private)
        % Code that executes after component creation
        function starupFcn(app, mainapp)

            if nargin<=1
                return
            end

            app.gaussianNoiseGenBtn.Visible = 'off';
            app.callingApp = mainapp; % 保存调用该窗口的窗口对象

            % 采样点数无法编辑


            app.sn = app.sineNum.Value; % 正弦信号个数
            app.fs = 44100 % app.sineSampleFreq.Value; % 更新采样频率
            app.N= length(mainapp.originSig) ; % 更新采样点数
            app.f = ones(app.sn,1) .* app.SINE_FREQ; % 初始化频率值列表

            % 初始绘制一个图
            setAndDraw(app);

        end
        % Callback function
        function openApp(app, event)
```

```
end
% Button pushed function: sineConfirm
function sineConfirmCallback(app, event)

    % 应用
    sineApply(app, event);

    % 退出
    delete(app);

end
% Value changed function: sineNum
function sineNumChangeCallback(app, event)
% 更新正弦信号的个数

    % 重置单一正弦信号控制块
    reset(app, 'single');

    % 正弦信号数量
    app.sn = app.sineNum.Value;  % scalar

    % 重置频率列表 (利用默认频率)
    app.f = ones(app.sn,1) .* app.SINE_FREQ;

    % 绘制图形
    setAndDraw(app);
end
% Callback function
function freqUpdateCallback(app, event)
    app.fs= app.sineSampleFreq.Value; % 更新采样频率

    % 重新绘制图形
    setAndDraw(app);
end
% Callback function
function sampleNumValueChanged(app, event)

    % 更新采样点数
    app.N= app.sampleNum.Value;

    % 重新绘制
    setAndDraw(app);
end
% Value changed function: sinePosFreq
function posFreqChange(app, event)

    % 更新当前指定位置正弦信号的频率
    updateCurrentFreq(app);

    % 更新信号且绘制
```

```
    setAndDraw(app);
end
% Callback function: sinePosSpinner
function posChange(app, event)

    % 获取更新位置
    pos = app.sinePosSpinner.Value;

    if(pos <= app.sineNum.Value )
    % 如果未越界则更新
        app.sinePosFreq.Value = app.f(pos);
    else
    % 越界了就固定为最高值
        app.sinePosSpinner.Value = app.sineNum.Value + 0.0;
    end
end
% Button pushed function: Button
function sineApply(app, event)

    % 更新数据
    app.callingApp.noiseSig = app.noiseSig;


    % 绘制图像
    app.callingApp.updateNoise();

    % 更新主面板数据
    app.callingApp.updateTest();
end
% Callback function
function quit(app, event)
    delete(app);
end
% Value changed function: ListBox
function noiseGenSelect(app, event)
    value = app.ListBox.Value;

    if strcmp(value, '正弦信号')

       % 显示正弦信号面板
       set(app.sineNum,'visible','on');
       %set(app.sineSampleFreq,'visible','off');
       %set(app.sampleNum,'visible', 'off');
       set(app.sinePosFreq,'visible', 'on');
       set(app.sinePosSpinner,'visible', 'on');

       %app.Label.Visible= 'off';
       %app.HzLabel_2.Visible='off';
       app.HzLabel.Visible='on';
       app.Label_2.Visible='on';
       %app.Label_3.Visible='off';
       %app.Label_8.Visible='off';
       app.Label_4.Visible = 'on';
```

```
            app.Label_5.Visible='on';

            % 隐藏白噪声面板
            app.gaussianNoiseGenBtn.Visible='off';


            %
        elseif strcmp(value,'白噪音' )
            % 白噪声
            set(app.sineNum,'visible','off');
            %set(app.sineSampleFreq,'visible','off');
            %set(app.sampleNum,'visible', 'off');
            set(app.sinePosFreq,'visible', 'off');
            set(app.sinePosSpinner,'visible', 'off');

            %app.Label.Visible= 'off';
            %app.HzLabel_2.Visible='off';
            app.HzLabel.Visible='off';
            app.Label_2.Visible='off';
            %app.Label_3.Visible='off';
            %app.Label_8.Visible='off';
            app.Label_4.Visible = 'off';
            app.Label_5.Visible='off';

            app.gaussianNoiseGenBtn.Visible='on';









        end


        % 设置可见

    end
    % Button pushed function: noisePlayBtn
    function noisePlay(app, event)
        sound(app.noiseSig, 44100);
    end
    % Button pushed function: gaussianNoiseGenBtn
    function gaussianNoiseGen(app, event)
         app.noiseSig =  0.01*randn(size(app.callingApp.originSig));
         updateNoiseSig(app);
    end
end
% App initialization and construction
methods (Access = private)
    % Create UIFigure and components
    function createComponents(app)
        % Create UIFigure
```

```
            app.UIFigure = uifigure;
            app.UIFigure.Position = [100 100 690 533];
            app.UIFigure.Name = '正弦信号生成系统';
            % Create Label_6
            app.Label_6 = uilabel(app.UIFigure);
            app.Label_6.FontSize = 24;
            app.Label_6.Position = [270 494 149 32];
            app.Label_6.Text = '噪音生成模块';
            % Create sineConfirm
            app.sineConfirm = uibutton(app.UIFigure, 'push');
            app.sineConfirm.ButtonPushedFcn = createCallbackFcn(app,
@sineConfirmCallback, true);
            app.sineConfirm.Position = [393 145 100 25];
            app.sineConfirm.Text = '确认';
            % Create sinePosSpinner
            app.sinePosSpinner = uispinner(app.UIFigure);
            app.sinePosSpinner.ValueChangingFcn = createCallbackFcn(app,
@posChange, true);
            app.sinePosSpinner.Limits = [1 Inf];
            app.sinePosSpinner.RoundFractionalValues = 'on';
            app.sinePosSpinner.ValueChangedFcn = createCallbackFcn(app,
@posChange, true);
            app.sinePosSpinner.HorizontalAlignment = 'center';
            app.sinePosSpinner.Position = [87 12 178 65];
            app.sinePosSpinner.Value = 1;
            % Create HzLabel
            app.HzLabel = uilabel(app.UIFigure);
            app.HzLabel.Position = [604 33 25 22];
            app.HzLabel.Text = 'Hz';
            % Create Label_2
            app.Label_2 = uilabel(app.UIFigure);
            app.Label_2.Position = [293 123 25 22];
            app.Label_2.Text = '个';
            % Create Label_4
            app.Label_4 = uilabel(app.UIFigure);
            app.Label_4.HorizontalAlignment = 'right';
            app.Label_4.Position = [80 123 77 22];
            app.Label_4.Text = '正弦信号数量';
            % Create sineNum
            app.sineNum = uispinner(app.UIFigure);
            app.sineNum.Limits = [1 Inf];
            app.sineNum.ValueChangedFcn = createCallbackFcn(app,
@sineNumChangeCallback, true);
            app.sineNum.Position = [171 123 100 22];
            app.sineNum.Value = 1;
            % Create Label_5
            app.Label_5 = uilabel(app.UIFigure);
            app.Label_5.HorizontalAlignment = 'right';
            app.Label_5.Position = [41 33 344 22];
            app.Label_5.Text = '第
个正弦信号频率设定';
            % Create sinePosFreq
            app.sinePosFreq = uieditfield(app.UIFigure, 'numeric');
```

```
            app.sinePosFreq.Limits = [0 Inf];
            app.sinePosFreq.ValueChangedFcn = createCallbackFcn(app,
@posFreqChange, true);
            app.sinePosFreq.Position = [393 33 199 22];
            app.sinePosFreq.Value = 1000;
            % Create sineGenFig
            app.sineGenFig = uiaxes(app.UIFigure);
            title(app.sineGenFig, '时域图')
            xlabel(app.sineGenFig, 't / s')
            ylabel(app.sineGenFig, 'y')
            app.sineGenFig.PlotBoxAspectRatio = [1 0.505285412262156
0.505285412262156];
            app.sineGenFig.TitleFontWeight = 'bold';
            app.sineGenFig.Position = [1 179 339 314];
            % Create Button
            app.Button = uibutton(app.UIFigure, 'push');
            app.Button.ButtonPushedFcn = createCallbackFcn(app, @sineApply,
true);
            app.Button.Position = [393 112 100 25];
            app.Button.Text = '应用';
            % Create Label_7
            app.Label_7 = uilabel(app.UIFigure);
            app.Label_7.HorizontalAlignment = 'right';
            app.Label_7.Position = [526 160 77 22];
            app.Label_7.Text = '噪音类型选择';
            % Create ListBox
            app.ListBox = uilistbox(app.UIFigure);
            app.ListBox.Items = {'正弦信号', '白噪音'};
            app.ListBox.ValueChangedFcn = createCallbackFcn(app,
@noiseGenSelect, true);
            app.ListBox.Position = [529 87 100 74];
            app.ListBox.Value = '正弦信号';
            % Create sineGenSpectum
            app.sineGenSpectum = uiaxes(app.UIFigure);
            title(app.sineGenSpectum, '幅频图')
            xlabel(app.sineGenSpectum, '\omega / \pi')
            ylabel(app.sineGenSpectum, '|H(\omega)|')
            app.sineGenSpectum.PlotBoxAspectRatio = [1 0.505285412262156
0.505285412262156];
            app.sineGenSpectum.TitleFontWeight = 'bold';
            app.sineGenSpectum.Position = [339 181 339 314];
            % Create noisePlayBtn
            app.noisePlayBtn = uibutton(app.UIFigure, 'push');
            app.noisePlayBtn.ButtonPushedFcn = createCallbackFcn(app,
@noisePlay, true);
            app.noisePlayBtn.Position = [393 76 100 26];
            app.noisePlayBtn.Text = '播放';
            % Create gaussianNoiseGenBtn
            app.gaussianNoiseGenBtn = uibutton(app.UIFigure, 'push');
            app.gaussianNoiseGenBtn.ButtonPushedFcn =
createCallbackFcn(app, @gaussianNoiseGen, true);
            app.gaussianNoiseGenBtn.Position = [148 121 100 25];
            app.gaussianNoiseGenBtn.Text = '高斯白噪声';
```

```
        end
    end
    methods (Access = public)
        % Construct app
        function app = noiseGenApp(varargin)
            % Create and configure components
            createComponents(app)
            % Register the app with App Designer
            registerApp(app, app.UIFigure)
            % Execute the startup function
            runStartupFcn(app, @(app)starupFcn(app, varargin{:}))
            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
```

6.5 附录五 IIR 滤波器生成模块

```
classdef IIRfilterDesignApp < matlab.apps.AppBase
    % Properties that correspond to app components
    properties (Access = public)
        IIRUIFigure              matlab.ui.Figure
        filterSpectrumFig        matlab.ui.control.UIAxes
        passEditFieldLabel       matlab.ui.control.Label
        passEditField            matlab.ui.control.NumericEditField
        pLabel                   matlab.ui.control.Label
        alphaPEditField          matlab.ui.control.NumericEditField
        stopEditFieldLabel       matlab.ui.control.Label
        stopEditField            matlab.ui.control.NumericEditField
        sLabel                   matlab.ui.control.Label
        alphasEditField          matlab.ui.control.NumericEditField
        IIRLabel                 matlab.ui.control.Label
        Label                    matlab.ui.control.Label
        DropDown                 matlab.ui.control.DropDown
        DropDown_2               matlab.ui.control.DropDown
        filterPhaseFig           matlab.ui.control.UIAxes
        stopupEditFieldLabel     matlab.ui.control.Label
        stopupEditField          matlab.ui.control.NumericEditField
        passupEditFieldLabel     matlab.ui.control.Label
        passupEditField          matlab.ui.control.NumericEditField
        stoplowEditFieldLabel    matlab.ui.control.Label
        stoplowEditField         matlab.ui.control.NumericEditField
        passlowEditFieldLabel    matlab.ui.control.Label
        passlowEditField         matlab.ui.control.NumericEditField
        passRippleEditFieldLabel matlab.ui.control.Label
```

```
        passRippleEditField      matlab.ui.control.NumericEditField
        Button                   matlab.ui.control.Button
        Button_2                 matlab.ui.control.Button
    end
    properties (Access = public)
        T % T

        % 调用该 app 的实例
        callingApp

        % 滤波器类型
        filterType  % 1: 巴特， 2: 切比雪夫 3: 椭圆
        filterPassType % 1: 低通 2:高通 3: 带通 4:带阻

        % 数字指标
        wp  % 通带截止角频率 0,1 之间的标量或者两个标量的向量(带通)
        ap  % 通带最大衰减
        ws  % 阻带截止频率
        as  % 阻带最大衰减

        wup
        wlp
        wus
        wls

        a % H(z) 分母
        b % H(z) 分子

        rp % 切比雪夫峰峰值波浪(dB)

        % 模拟指标
        Wp  % 通带截止角频率
        Ap  % 通带最大衰减
        Ws  % 阻带截止频率
        As  % 阻带最大衰减

        % 频率响应与角频率
        freqResp % 频率响应
        freqResp_w % 角频率

    end
    methods (Access = public)

        function [] = updateIIRFilter(app)
            % 更新数字滤波器

            % 更新数字频率
            app.ws = app.stopEditField.Value;
            app.wp = app.passEditField.Value;
            app.wls = app.stoplowEditField.Value;
            app.wlp = app.passlowEditField.Value;
            app.wus = app.stopupEditField.Value;
```

```
            app.wup = app.passupEditField.Value;

            app.ap = app.alphaPEditField.Value;
            app.as = app.alphasEditField.Value;

            app.rp = app.passRippleEditField.Value;

            % 确定模拟滤波器的技术指标 ， 现在直接设计数字滤波器
            %app.Wp = 2./app.T.*tan(1/2*app.wp);
            %%app.Ws = 2./app.T.*tan(1/2*app.ws);
            %app.Ap =app.ap;
            %app.As =app.as;

            [h,w] = filterDesign(app);

            plot(app.filterSpectrumFig, w/pi,20*log10(abs(h)));
            plot(app.filterPhaseFig, w/pi,unwrap(angle(h)));

            app.freqResp = h;
            app.freqResp_w = w;
        end

        function [h,w] = butterFilterDesign(app)
            % 巴特沃兹滤波器设计

            if app.filterPassType==1
                [n,Wn] = myButtord(app.wp,app.ws,app.ap,app.as);
                [app.b,app.a] = butter(n,Wn,'low');
            elseif app.filterPassType==2
                [n,Wn] = myButtord(app.wp,app.ws,app.ap,app.as);
                [app.b,app.a] = butter(n,Wn,'high');
            elseif app.filterPassType==3
                [n,Wn] =
myButtord([app.wlp,app.wup],[app.wls,app.wus],app.ap,app.as);
                [app.b,app.a] = butter(n,Wn,'bandpass');
            elseif app.filterPassType==4
                [n,Wn] =
myButtord([app.wlp,app.wup],[app.wls,app.wus],app.ap,app.as);
                [app.b,app.a] = butter(n,Wn,'stop');
            end
             % 不带 s 数字滤波器设计 返回 H(z)[分子,分母]


            % 获得频率响应，注意把 w 归一化!
            [h,w] = freqz(app.b,app.a);
        end

            function [h,w] = chebz1FilterDesign(app)
            % 切比雪夫滤波器设计

            % 直接设计数字滤波器： 生成阶数
            if app.filterPassType==1
                [n,Wn] = myCheby(app.wp,app.ws,app.ap,app.as);
```

```
                [app.b,app.a] = cheby1(n,app.rp,Wn,'low');
            elseif app.filterPassType==2
                [n,Wn] = myCheby(app.wp,app.ws,app.ap,app.as);
                [app.b,app.a] = cheby1(n,app.rp,Wn,'high');
            elseif app.filterPassType==3
                [n,Wn] =
myCheby([app.wlp,app.wup],[app.wls,app.wus],app.ap,app.as);
                [app.b,app.a] = cheby1(n,app.rp,Wn,'bandpass');
            elseif app.filterPassType==4
                [n,Wn] =
myCheby([app.wlp,app.wup],[app.wls,app.wus],app.ap,app.as);
                [app.b,app.a] = cheby1(n,app.rp,Wn,'stop');
            end


            % 获得频率响应，注意把 w 归一化!
            [h,w] = freqz(app.b,app.a);
        end

        function [h,w]= filterDesign(app)

            h = 0;
            w = 0;

            if app.filterType == 1
             [h,w] = butterFilterDesign(app);
            elseif app.filterType == 2
             [h,w] = chebz1FilterDesign(app);
            end


        end

        function results = resetPanel(app)
            if app.filterType == 1
                app.passRippleEditField.Visible = 'off';
                app.passRippleEditFieldLabel.Visible = 'off';
            elseif app.filterType == 2
                app.passRippleEditField.Visible = 'on';
                app.passRippleEditFieldLabel.Visible = 'on';
            end
        end


    end
    methods (Access = private)

        function results = showHighLowPanel(app)
            % 显示低通，高通模块

            app.passEditField.Visible = 'on';
            app.stopEditField.Visible = 'on';
            app.passEditFieldLabel.Visible = 'on';
```

```
            app.stopEditFieldLabel.Visible = 'on';

            app.passupEditField.Visible = 'off';
            app.passlowEditField.Visible='off';
            app.stoplowEditField.Visible ='off';
            app.stopupEditField.Visible = 'off';

            app.passupEditFieldLabel.Visible = 'off';
            app.passlowEditFieldLabel.Visible='off';
            app.stoplowEditFieldLabel.Visible ='off';
            app.stopupEditFieldLabel.Visible = 'off';
        end


        function results = showBandPanel(app)
            % 显示带通,带阻模块

            app.passEditField.Visible = 'off';
            app.stopEditField.Visible = 'off';
            app.passEditFieldLabel.Visible = 'off';
            app.stopEditFieldLabel.Visible = 'off';

            app.passupEditField.Visible = 'on';
            app.passlowEditField.Visible='on';
            app.stoplowEditField.Visible ='on';
            app.stopupEditField.Visible = 'on';

            app.passupEditFieldLabel.Visible = 'on';
            app.passlowEditFieldLabel.Visible='on';
            app.stoplowEditFieldLabel.Visible ='on';
            app.stopupEditFieldLabel.Visible = 'on';



        end

    end
    methods (Access = private)
        % Code that executes after component creation
        function startupFcn(app, mainApp)
            app.T = 2; % 设置 T=2

            showHighLowPanel(app);

            % 刚进来为巴特沃兹低通
            app.filterType = 1;
            app.filterPassType=1;

            resetPanel(app);

            app.callingApp = mainApp;

            app.filterTypeChangeCallback();
```

```
    app.filterPassTypeChangeCallback();


end
% Value changed function: alphaPEditField, alphasEditField,
% passEditField, passRippleEditField, passlowEditField,
% passupEditField, stopEditField, stoplowEditField,
% stopupEditField
function filterUpdateCallback(app, event)
    updateIIRFilter(app);
end
% Value changed function: DropDown
function filterTypeChangeCallback(app, event)
    value = app.DropDown.Value;
    if strcmp('巴特沃兹滤波器',value)
        app.filterType = 1;
    elseif strcmp('切比雪夫Ⅰ型滤波器',value)
        app.filterType = 2;
    end
    resetPanel(app);
    updateIIRFilter(app);
end
% Value changed function: DropDown_2
function filterPassTypeChangeCallback(app, event)
    value = app.DropDown_2.Value;
    if strcmp('低通',value)
        showHighLowPanel(app);
        app.filterPassType=1;

        app.stopEditField.Value = 0.5;
        app.passEditField.Value = 0.3;

    elseif strcmp('高通',value)
        showHighLowPanel(app);
        app.filterPassType=2;

        app.stopEditField.Value = 0.3;
        app.passEditField.Value = 0.5;
    elseif strcmp('带通',value)

        showBandPanel(app);
        app.filterPassType=3;


        app.stoplowEditField.Value = 0.1;
        app.stopupEditField.Value = 0.7;
        app.passlowEditField.Value = 0.3;
        app.passupEditField.Value = 0.5;


    elseif strcmp('带阻',value)

        showBandPanel(app);
```

```
                app.filterPassType=4;

                app.stoplowEditField.Value = 0.3;
                app.stopupEditField.Value = 0.5;
                app.passlowEditField.Value = 0.1;
                app.passupEditField.Value = 0.7;
            end

            updateIIRFilter(app);
        end
        % Button pushed function: Button
        function filterApply(app, event)
            app.callingApp.Lamp_2.Color = 'g';
            app.callingApp.setFilterDisign = 1;

            app.callingApp.b = app.b;
            app.callingApp.a = app.a;

        end
        % Button pushed function: Button_2
        function filterConfirm(app, event)
            filterApply(app);
            delete(app);
        end
    end
    % App initialization and construction
    methods (Access = private)
        % Create UIFigure and components
        function createComponents(app)
            % Create IIRUIFigure
            app.IIRUIFigure = uifigure;
            app.IIRUIFigure.Position = [100 100 588 682];
            app.IIRUIFigure.Name = 'IIR 滤波器设计';
            % Create filterSpectrumFig
            app.filterSpectrumFig = uiaxes(app.IIRUIFigure);
            title(app.filterSpectrumFig, '幅度响应')
            xlabel(app.filterSpectrumFig, '归一化角频率 ω (x\times \pi
rads/sample)')
            ylabel(app.filterSpectrumFig, '幅频衰减(dB)')
            app.filterSpectrumFig.PlotBoxAspectRatio = [1
0.472222222222222 0.472222222222222];
            app.filterSpectrumFig.XLim = [0 1];
            app.filterSpectrumFig.TitleFontWeight = 'bold';
            app.filterSpectrumFig.Position = [63 394 454 249];
            % Create passEditFieldLabel
            app.passEditFieldLabel = uilabel(app.IIRUIFigure);
            app.passEditFieldLabel.HorizontalAlignment = 'right';
            app.passEditFieldLabel.FontSize = 14;
            app.passEditFieldLabel.Position = [165 51 25 22];
            app.passEditFieldLabel.Text = 'wp';
            % Create passEditField
            app.passEditField = uieditfield(app.IIRUIFigure, 'numeric');
```

```
            app.passEditField.ValueChangedFcn = createCallbackFcn(app,
@filterUpdateCallback, true);
            app.passEditField.FontSize = 14;
            app.passEditField.Position = [205 50 59 23];
            app.passEditField.Value = 0.3;
            % Create pLabel
            app.pLabel = uilabel(app.IIRUIFigure);
            app.pLabel.HorizontalAlignment = 'right';
            app.pLabel.FontSize = 14;
            app.pLabel.Position = [165 11 25 22];
            app.pLabel.Text = 'αp';
            % Create alphaPEditField
            app.alphaPEditField = uieditfield(app.IIRUIFigure, 'numeric');
            app.alphaPEditField.ValueChangedFcn = createCallbackFcn(app,
@filterUpdateCallback, true);
            app.alphaPEditField.FontSize = 14;
            app.alphaPEditField.Position = [205 10 59 23];
            app.alphaPEditField.Value = 2;
            % Create stopEditFieldLabel
            app.stopEditFieldLabel = uilabel(app.IIRUIFigure);
            app.stopEditFieldLabel.HorizontalAlignment = 'right';
            app.stopEditFieldLabel.FontSize = 14;
            app.stopEditFieldLabel.Position = [302 51 25 22];
            app.stopEditFieldLabel.Text = 'ωs';
            % Create stopEditField
            app.stopEditField = uieditfield(app.IIRUIFigure, 'numeric');
            app.stopEditField.ValueChangedFcn = createCallbackFcn(app,
@filterUpdateCallback, true);
            app.stopEditField.FontSize = 14;
            app.stopEditField.Position = [342 50 59 23];
            app.stopEditField.Value = 0.5;
            % Create sLabel
            app.sLabel = uilabel(app.IIRUIFigure);
            app.sLabel.HorizontalAlignment = 'right';
            app.sLabel.FontSize = 14;
            app.sLabel.Position = [302 10 25 22];
            app.sLabel.Text = 'αs';
            % Create alphasEditField
            app.alphasEditField = uieditfield(app.IIRUIFigure, 'numeric');
            app.alphasEditField.ValueChangedFcn = createCallbackFcn(app,
@filterUpdateCallback, true);
            app.alphasEditField.FontSize = 14;
            app.alphasEditField.Position = [342 9 59 23];
            app.alphasEditField.Value = 15;
            % Create IIRLabel
            app.IIRLabel = uilabel(app.IIRUIFigure);
            app.IIRLabel.FontSize = 22;
            app.IIRLabel.Position = [215 652 150 29];
            app.IIRLabel.Text = 'IIR 滤波器设计';
            % Create Label
            app.Label = uilabel(app.IIRUIFigure);
            app.Label.HorizontalAlignment = 'right';
            app.Label.Position = [134 104 89 22];
```

```
            app.Label.Text = '滤波器类型选择';
            % Create DropDown
            app.DropDown = uidropdown(app.IIRUIFigure);
            app.DropDown.Items = {'巴特沃兹滤波器', '切比雪夫Ⅰ型滤波器'};
            app.DropDown.ValueChangedFcn = createCallbackFcn(app,
@filterTypeChangeCallback, true);
            app.DropDown.Position = [238 100 119 26];
            app.DropDown.Value = '巴特沃兹滤波器';
            % Create DropDown_2
            app.DropDown_2 = uidropdown(app.IIRUIFigure);
            app.DropDown_2.Items = {'低通', '高通', '带通', '带阻'};
            app.DropDown_2.ValueChangedFcn = createCallbackFcn(app,
@filterPassTypeChangeCallback, true);
            app.DropDown_2.Position = [372 102 100 22];
            app.DropDown_2.Value = '低通';
            % Create filterPhaseFig
            app.filterPhaseFig = uiaxes(app.IIRUIFigure);
            title(app.filterPhaseFig, '相位')
            xlabel(app.filterPhaseFig, '归一化角频率 ω (x\times \pi
rads/sample)')
            ylabel(app.filterPhaseFig, '幅频衰减(dB)')
            app.filterPhaseFig.PlotBoxAspectRatio = [1 0.472222222222222
0.472222222222222];
            app.filterPhaseFig.XLim = [0 1];
            app.filterPhaseFig.TitleFontWeight = 'bold';
            app.filterPhaseFig.Position = [63 138 454 249];
            % Create stopupEditFieldLabel
            app.stopupEditFieldLabel = uilabel(app.IIRUIFigure);
            app.stopupEditFieldLabel.HorizontalAlignment = 'right';
            app.stopupEditFieldLabel.FontSize = 14;
            app.stopupEditFieldLabel.Position = [438 51 31 22];
            app.stopupEditFieldLabel.Text = 'ωus';
            % Create stopupEditField
            app.stopupEditField = uieditfield(app.IIRUIFigure, 'numeric');
            app.stopupEditField.ValueChangedFcn = createCallbackFcn(app,
@filterUpdateCallback, true);
            app.stopupEditField.FontSize = 14;
            app.stopupEditField.Position = [484 50 70 23];
            app.stopupEditField.Value = 0.8;
            % Create passupEditFieldLabel
            app.passupEditFieldLabel = uilabel(app.IIRUIFigure);
            app.passupEditFieldLabel.HorizontalAlignment = 'right';
            app.passupEditFieldLabel.FontSize = 14;
            app.passupEditFieldLabel.Position = [31 51 32 22];
            app.passupEditFieldLabel.Text = 'ωup';
            % Create passupEditField
            app.passupEditField = uieditfield(app.IIRUIFigure, 'numeric');
            app.passupEditField.ValueChangedFcn = createCallbackFcn(app,
@filterUpdateCallback, true);
            app.passupEditField.FontSize = 14;
            app.passupEditField.Position = [78 50 70 23];
            app.passupEditField.Value = 0.1;
            % Create stoplowEditFieldLabel
```

```
            app.stoplowEditFieldLabel = uilabel(app.IIRUIFigure);
            app.stoplowEditFieldLabel.HorizontalAlignment = 'right';
            app.stoplowEditFieldLabel.FontSize = 14;
            app.stoplowEditFieldLabel.Position = [300 52 27 22];
            app.stoplowEditFieldLabel.Text = 'ωls';
            % Create stoplowEditField
            app.stoplowEditField = uieditfield(app.IIRUIFigure,
'numeric');
            app.stoplowEditField.ValueChangedFcn = createCallbackFcn(app,
@filterUpdateCallback, true);
            app.stoplowEditField.FontSize = 14;
            app.stoplowEditField.Position = [342 51 59 23];
            app.stoplowEditField.Value = 0.5;
            % Create passlowEditFieldLabel
            app.passlowEditFieldLabel = uilabel(app.IIRUIFigure);
            app.passlowEditFieldLabel.HorizontalAlignment = 'right';
            app.passlowEditFieldLabel.FontSize = 14;
            app.passlowEditFieldLabel.Position = [165 51 27 22];
            app.passlowEditFieldLabel.Text = 'ωlp';
            % Create passlowEditField
            app.passlowEditField = uieditfield(app.IIRUIFigure,
'numeric');
            app.passlowEditField.ValueChangedFcn = createCallbackFcn(app,
@filterUpdateCallback, true);
            app.passlowEditField.FontSize = 14;
            app.passlowEditField.Position = [207 50 59 23];
            app.passlowEditField.Value = 0.3;
            % Create passRippleEditFieldLabel
            app.passRippleEditFieldLabel = uilabel(app.IIRUIFigure);
            app.passRippleEditFieldLabel.HorizontalAlignment = 'right';
            app.passRippleEditFieldLabel.FontSize = 14;
            app.passRippleEditFieldLabel.Position = [39 9 25 22];
            app.passRippleEditFieldLabel.Text = 'Rp';
            % Create passRippleEditField
            app.passRippleEditField = uieditfield(app.IIRUIFigure,
'numeric');
            app.passRippleEditField.ValueChangedFcn =
createCallbackFcn(app, @filterUpdateCallback, true);
            app.passRippleEditField.FontSize = 14;
            app.passRippleEditField.Position = [79 9 69 23];
            app.passRippleEditField.Value = 2;
            % Create Button
            app.Button = uibutton(app.IIRUIFigure, 'push');
            app.Button.ButtonPushedFcn = createCallbackFcn(app,
@filterApply, true);
            app.Button.Position = [417 8 68 25];
            app.Button.Text = '应用';
            % Create Button_2
            app.Button_2 = uibutton(app.IIRUIFigure, 'push');
            app.Button_2.ButtonPushedFcn = createCallbackFcn(app,
@filterConfirm, true);
            app.Button_2.Position = [507 8 68 25];
            app.Button_2.Text = '确定';
```

```
        end
    end
    methods (Access = public)
        % Construct app
        function app = IIRfilterDesignApp(varargin)
            % Create and configure components
            createComponents(app)
            % Register the app with App Designer
            registerApp(app, app.IIRUIFigure)
            % Execute the startup function
            runStartupFcn(app, @(app)startupFcn(app, varargin{:}))
            if nargout == 0
                clear app
            end
        end
        % Code that executes before app deletion
        function delete(app)
            % Delete UIFigure when app is deleted
            delete(app.IIRUIFigure)
        end
    end
end
```

6.6 附录六 FIR 滤波器生成模块

```
classdef FIRfilterDesignApp < matlab.apps.AppBase
    % Properties that correspond to app components
    properties (Access = public)
        FIRUIFigure         matlab.ui.Figure
        filterSpectrumFig    matlab.ui.control.UIAxes
        left_Label          matlab.ui.control.Label
        leftOmegaEditField   matlab.ui.control.NumericEditField
        right_Label         matlab.ui.control.Label
        rightOmegaEditField  matlab.ui.control.NumericEditField
        FIRLabel            matlab.ui.control.Label
        Label               matlab.ui.control.Label
        DropDown            matlab.ui.control.DropDown
        DropDown_2          matlab.ui.control.DropDown
        filterPhaseFig       matlab.ui.control.UIAxes
        Button              matlab.ui.control.Button
        Button_2            matlab.ui.control.Button
    end
    properties (Access = public)
        T % T

        % 滤波器类型
        filterPassType % 滤波类型 1: 低通 2:高通 3: 带通 4:带阻
        windowType % 窗口类型 1: 矩形窗 2: 三角窗 3: 汉宁窗 4: 哈明窗


        window
        % 数字指标
```

```
    lw % 左边截止频率
    rw % 右边截止频率
    wc % 3dB 频率


    M %
    N % 滤波器阶数

    callingApp % App


    b % H(Z) 参数
    % 频率响应与角频率
    freqResp % 频率响应
    freqResp_w % 角频率
end
methods (Access = public)


    function [] = updateFIRFilter(app)
        % 更新数字滤波器

        % 更新数字频率 app.lw
        app.lw = app.leftOmegaEditField.Value;
        app.rw = app.rightOmegaEditField.Value;
        app.wc = (app.lw+app.rw)/2;

        % 窗函数法设计
        if app.windowType==1
             app.M = ceil(4.0/abs(app.rw - app.lw)); % 点数
             app.M=app.M+mod(app.M,2); % 保证奇数
            app.window = rectwin(app.M + 1);

        elseif app.windowType==2
            app.M = ceil(6.1/abs(app.rw - app.lw)); % 点数
            app.M=app.M+mod(app.M,2);
            app.window = bartlett(app.M + 1);

        elseif app.windowType==3
             app.M = ceil(6.2/abs(app.rw - app.lw)); % 点数
             app.M=app.M+mod(app.M,2);
            app.window = hann(app.M + 1);

        elseif app.windowType==4
             app.M = ceil(6.6/abs(app.rw - app.lw)); % 点数
             app.M=app.M+mod(app.M,2);
            app.window = hamming(app.M + 1);

        elseif app.windowType == 5
            app.M = ceil(12/abs(app.rw - app.lw));
            app.M=app.M+mod(app.M,2);
            app.window = blackman(app.M + 1);
         end
```

```
            app.N = app.M+1; % 阶数

            if app.filterPassType==1
                app.b = fir1(app.M, app.wc , 'low',app.window);
            elseif app.filterPassType==2
                app.b = fir1(app.M, app.wc, 'high',app.window );
            elseif app.filterPassType==3
                app.b = fir1(app.M,[app.lw, app.rw],
'bandpass' ,app.window);
            elseif app.filterPassType==4
                app.b = fir1(app.M,[app.lw, app.rw], 'stop' ,app.window);
            end

            [h,w] = freqz(app.b,1);
            % 绘制
            plot(app.filterSpectrumFig, w/pi,20*log10(abs(h)));
            plot(app.filterPhaseFig, w/pi,unwrap(angle(h)));



        end
    end
    methods (Access = private)



    end
    methods (Access = private)
        % Code that executes after component creation
        function startupFcn(app, otherApp)
            app.T = 2; % 设置 T=2

            if nargin >=2
                app.callingApp=otherApp;
            end
            % showHighLowPanel(app);

            % 刚进来为矩形窗 低通
            app.windowType = 1;
            app.filterPassType=1;
            app.lw = app.leftOmegaEditField.Value;
            app.rw = app.rightOmegaEditField.Value;


            % resetPanel(app);

            app.filterTypeChangeCallback();
            app.filterPassTypeChangeCallback();


        end
        % Value changed function: leftOmegaEditField,
        % rightOmegaEditField
```

```
function filterUpdateCallback(app, event)
    updateFIRFilter(app);
end
% Value changed function: DropDown
function filterTypeChangeCallback(app, event)
    value = app.DropDown.Value;


    if strcmp('矩形窗', value)
        app.windowType = 1;
    elseif strcmp('三角窗', value)
        app.windowType = 2;
    elseif strcmp('汉宁窗', value)
        app.windowType = 3;
    elseif strcmp('哈明窗', value)
        app.windowType = 4;
    elseif strcmp('布莱克曼窗', value)
        app.windowType = 5;
    end

    % resetPanel(app);
    updateFIRFilter(app);
end
% Value changed function: DropDown_2
function filterPassTypeChangeCallback(app, event)
    value = app.DropDown_2.Value;
    if strcmp('低通',value)
        % showHighLowPanel(app);
        app.filterPassType=1;

        app.rightOmegaEditField.Value = 0.5;
        app.leftOmegaEditField.Value = 0.3;

    elseif strcmp('高通',value)
        % showHighLowPanel(app);
        app.filterPassType=2;

        app.rightOmegaEditField.Value = 0.3;
        app.leftOmegaEditField.Value = 0.5;
    elseif strcmp('带通',value)

        % showBandPanel(app);
        app.filterPassType=3;

        app.rightOmegaEditField.Value = 0.5;
        app.leftOmegaEditField.Value = 0.3;

    elseif strcmp('带阻',value)

        % showBandPanel(app);
        app.filterPassType=4;

        app.rightOmegaEditField.Value = 0.5;
```

```
                app.leftOmegaEditField.Value = 0.3;
            end

            updateFIRFilter(app);
        end
        % Button pushed function: Button
        function filterApply(app, event)
            app.callingApp.filterSig = app.window;
            app.callingApp.setFilterDisign = 1;
            app.callingApp.Lamp_2.Color = 'g';
            app.callingApp.setFilterDisign = 1;
        end
        % Button pushed function: Button_2
        function FIRFilterConfirm(app, event)
              filterApply(app);
              delete(app);
        end
    end
    % App initialization and construction
    methods (Access = private)
        % Create UIFigure and components
        function createComponents(app)
            % Create FIRUIFigure
            app.FIRUIFigure = uifigure;
            app.FIRUIFigure.Position = [100 100 584 682];
            app.FIRUIFigure.Name = 'FIR 滤波器设计';
            % Create filterSpectrumFig
            app.filterSpectrumFig = uiaxes(app.FIRUIFigure);
            title(app.filterSpectrumFig, '幅度响应')
            xlabel(app.filterSpectrumFig, '归一化角频率 ω (x\times \pi
rads/sample)')
            ylabel(app.filterSpectrumFig, '幅频衰减(dB)')
            app.filterSpectrumFig.PlotBoxAspectRatio = [1
0.472222222222222 0.472222222222222];
            app.filterSpectrumFig.XLim = [0 1];
            app.filterSpectrumFig.TitleFontWeight = 'bold';
            app.filterSpectrumFig.Position = [63 394 454 249];
            % Create left_Label
            app.left_Label = uilabel(app.FIRUIFigure);
            app.left_Label.HorizontalAlignment = 'right';
            app.left_Label.FontSize = 14;
            app.left_Label.Position = [146 49 43 22];
            app.left_Label.Text = 'left_ω';
            % Create leftOmegaEditField
            app.leftOmegaEditField = uieditfield(app.FIRUIFigure,
'numeric');
            app.leftOmegaEditField.ValueChangedFcn = createCallbackFcn(app,
@filterUpdateCallback, true);
            app.leftOmegaEditField.FontSize = 14;
            app.leftOmegaEditField.Position = [204 48 59 23];
            app.leftOmegaEditField.Value = 0.3;
            % Create right_Label
            app.right_Label = uilabel(app.FIRUIFigure);
```

```
            app.right_Label.HorizontalAlignment = 'right';
            app.right_Label.FontSize = 14;
            app.right_Label.Position = [335 49 51 22];
            app.right_Label.Text = 'right_ω';
            % Create rightOmegaEditField
            app.rightOmegaEditField = uieditfield(app.FIRUIFigure,
'numeric');
            app.rightOmegaEditField.ValueChangedFcn =
createCallbackFcn(app, @filterUpdateCallback, true);
            app.rightOmegaEditField.FontSize = 14;
            app.rightOmegaEditField.Position = [401 48 59 23];
            app.rightOmegaEditField.Value = 0.5;
            % Create FIRLabel
            app.FIRLabel = uilabel(app.FIRUIFigure);
            app.FIRLabel.FontSize = 22;
            app.FIRLabel.Position = [215 652 157 29];
            app.FIRLabel.Text = 'FIR 滤波器设计';
            % Create Label
            app.Label = uilabel(app.FIRUIFigure);
            app.Label.HorizontalAlignment = 'right';
            app.Label.Position = [146 104 77 22];
            app.Label.Text = '窗口类型选择';
            % Create DropDown
            app.DropDown = uidropdown(app.FIRUIFigure);
            app.DropDown.Items = {'矩形窗', '三角窗', '汉宁窗', '哈明窗', '
布莱克曼窗'};
            app.DropDown.Editable = 'on';
            app.DropDown.ValueChangedFcn = createCallbackFcn(app,
@filterTypeChangeCallback, true);
            app.DropDown.BackgroundColor = [1 1 1];
            app.DropDown.Position = [238 100 119 26];
            app.DropDown.Value = '矩形窗';
            % Create DropDown_2
            app.DropDown_2 = uidropdown(app.FIRUIFigure);
            app.DropDown_2.Items = {'低通', '高通', '带通', '带阻'};
            app.DropDown_2.ValueChangedFcn = createCallbackFcn(app,
@filterPassTypeChangeCallback, true);
            app.DropDown_2.Position = [372 102 100 22];
            app.DropDown_2.Value = '低通';
            % Create filterPhaseFig
            app.filterPhaseFig = uiaxes(app.FIRUIFigure);
            title(app.filterPhaseFig, '相位')
            xlabel(app.filterPhaseFig, '归一化角频率 ω (x\times \pi
rads/sample)')
            ylabel(app.filterPhaseFig, '相位 (rads)')
            app.filterPhaseFig.PlotBoxAspectRatio = [1 0.472222222222222
0.472222222222222];
            app.filterPhaseFig.XLim = [0 1];
            app.filterPhaseFig.TitleFontWeight = 'bold';
            app.filterPhaseFig.Position = [63 138 454 249];
            % Create Button
            app.Button = uibutton(app.FIRUIFigure, 'push');
```

```
            app.Button.ButtonPushedFcn = createCallbackFcn(app,
@filterApply, true);
            app.Button.Position = [155 7 100 25];
            app.Button.Text = '应用';
            % Create Button_2
            app.Button_2 = uibutton(app.FIRUIFigure, 'push');
            app.Button_2.ButtonPushedFcn = createCallbackFcn(app,
@FIRFilterConfirm, true);
            app.Button_2.Position = [360 6 100 26];
            app.Button_2.Text = '确定';
        end
    end
    methods (Access = public)
        % Construct app
        function app = FIRfilterDesignApp(varargin)
            % Create and configure components
            createComponents(app)
            % Register the app with App Designer
            registerApp(app, app.FIRUIFigure)
            % Execute the startup function
            runStartupFcn(app, @(app)startupFcn(app, varargin{:}))
            if nargout == 0
                clear app
            end
        end
        % Code that executes before app deletion
        function delete(app)
            % Delete UIFigure when app is deleted
            delete(app.FIRUIFigure)
        end
    end
end
```

6.7 附录七 IIR 滤波器分析模块

```
classdef IIRFilter < matlab.apps.AppBase
    % Properties that correspond to app components
    properties (Access = public)
        IIRUIFigure        matlab.ui.Figure
        outputSpectrumFig  matlab.ui.control.UIAxes
        outputPhaseFig     matlab.ui.control.UIAxes
        outputTimeFig      matlab.ui.control.UIAxes
        Button             matlab.ui.control.Button
    end
    properties (Access = public)
        callingApp % 调用该 App 的 app 实例引用


        outputSig % Description
    end
    methods (Access = private)
```

```
        % Code that executes after component creation
        function startupFcn(app, mainApp)
            app.callingApp = mainApp;

            app.outputSig =
filter(app.callingApp.b,app.callingApp.a,app.callingApp.testSig);


            % 更新数据
            N = length(app.outputSig); % 原信号点数
            n  = linspace(0,N-1,N);
            fft_y = fft(app.outputSig); % 计算 fft

            % n = linspace(0 , N-1,N); % 第 n 点
            w = 2*pi*n./N; % 角频率(未归一化)
            plot(app.outputTimeFig, app.outputSig);
            plot(app.outputSpectrumFig, w/pi,(abs(fft_y)));
            plot(app.outputPhaseFig, w/pi,unwrap(angle(fft_y)));


            mainApp.outputSig = app.outputSig;
        end
        % Button pushed function: Button
        function playOutputSig(app, event)
            sound(app.outputSig,44100);
        end
    end
    % App initialization and construction
    methods (Access = private)
        % Create UIFigure and components
        function createComponents(app)
            % Create IIRUIFigure
            app.IIRUIFigure = uifigure;
            app.IIRUIFigure.Position = [100 100 458 594];
            app.IIRUIFigure.Name = 'IIR 差分方程迭代滤波';
            % Create outputSpectrumFig
            app.outputSpectrumFig = uiaxes(app.IIRUIFigure);
            title(app.outputSpectrumFig, '幅度响应')
            xlabel(app.outputSpectrumFig, '归一化角频率 ω (x\times \pi
rads/sample)')
            ylabel(app.outputSpectrumFig, '幅频衰减(dB)')
            app.outputSpectrumFig.PlotBoxAspectRatio = [1
0.472222222222222 0.472222222222222];
            app.outputSpectrumFig.XLim = [0 1];
            app.outputSpectrumFig.TitleFontWeight = 'bold';
            app.outputSpectrumFig.Position = [47 406 286 168];
            % Create outputPhaseFig
            app.outputPhaseFig = uiaxes(app.IIRUIFigure);
            title(app.outputPhaseFig, '相位')
            xlabel(app.outputPhaseFig, '归一化角频率 ω (x\times \pi
rads/sample)')
            ylabel(app.outputPhaseFig, '相位(rads)')
```

```
            app.outputPhaseFig.PlotBoxAspectRatio = [1 0.472222222222222
0.472222222222222];
            app.outputPhaseFig.XLim = [0 1];
            app.outputPhaseFig.TitleFontWeight = 'bold';
            app.outputPhaseFig.Position = [47 201 286 185];
            % Create outputTimeFig
            app.outputTimeFig = uiaxes(app.IIRUIFigure);
            title(app.outputTimeFig, '时域')
            xlabel(app.outputTimeFig, 'n')
            ylabel(app.outputTimeFig, '信号幅度')
            app.outputTimeFig.PlotBoxAspectRatio = [1 0.451505016722408
0.451505016722408];
            app.outputTimeFig.TitleFontWeight = 'bold';
            app.outputTimeFig.Position = [43 15 290 166];
            % Create Button
            app.Button = uibutton(app.IIRUIFigure, 'push');
            app.Button.ButtonPushedFcn = createCallbackFcn(app,
@playOutputSig, true);
            app.Button.Position = [341 24 84 532];
            app.Button.Text = '播放';
        end
    end
    methods (Access = public)
        % Construct app
        function app = IIRFilter(varargin)
            % Create and configure components
            createComponents(app)
            % Register the app with App Designer
            registerApp(app, app.IIRUIFigure)
            % Execute the startup function
            runStartupFcn(app, @(app)startupFcn(app, varargin{:}))
            if nargout == 0
                clear app
            end
        end
        % Code that executes before app deletion
        function delete(app)
            % Delete UIFigure when app is deleted
            delete(app.IIRUIFigure)
        end
    end
end
```

6.8 附录八 FIR 滤波器分析模块

```
classdef FIRFilter < matlab.apps.AppBase
    % Properties that correspond to app components
    properties (Access = public)
        FIRUIFigure        matlab.ui.Figure
        outputTimeFig      matlab.ui.control.UIAxes
        outputSpectrumFig  matlab.ui.control.UIAxes
        Button             matlab.ui.control.Button
        outputPhaseFig     matlab.ui.control.UIAxes
        Label              matlab.ui.control.Label
```

```
        EditField         matlab.ui.control.NumericEditField
        Label_2           matlab.ui.control.Label
        EditField_2       matlab.ui.control.NumericEditField
    end
    properties (Access = public)
        callingApp % Description

        % 输出信号
        outputSig

        % 重叠保留法点数
        pnum

        % 暂停时间
        pauseTime

    end
    properties (Access = private)
    end
    methods (Access = public)

        function [] = updatePointNum(app)
            % 更新重叠保留法点数
            app.pnum = app.EditField.Value;

            % 更新秒数
            app.pauseTime = app.EditField_2.Value;

            app.outputSig = overlapSave(app.callingApp.testSig,
app.callingApp.filterSig,app.pnum,app.pauseTime);
            % 更新数据
            N = length(app.outputSig); % 原信号点数
            n  = linspace(0,N-1,N);
            fft_y = fft(app.outputSig); % 计算 fft

            % n = linspace(0 , N-1,N); % 第 n 点
            w = 2*pi*n./N; % 角频率(未归一化)
            plot(app.outputTimeFig, app.outputSig);
            plot(app.outputSpectrumFig, w/pi,(abs(fft_y)));
            plot(app.outputPhaseFig, w/pi,unwrap(angle(fft_y)));
            app.callingApp.outputSig = app.outputSig;
        end

    end
    methods (Access = private)
        % Code that executes after component creation
        function startupFcn(app, main)
            app.callingApp = main;

            pointChangeCallback(app);


        end
```

```
        % Button pushed function: Button
        function playOutputSig(app, event)
            sound(app.outputSig,44100);
        end
        % Value changed function: EditField
        function pointChangeCallback(app, event)
            updatePointNum(app);
        end
        % Value changed function: EditField_2
        function secondChangeCallback(app, event)

             updatePointNum(app);
        end
    end
    % App initialization and construction
    methods (Access = private)
        % Create UIFigure and components
        function createComponents(app)
            % Create FIRUIFigure
            app.FIRUIFigure = uifigure;
            app.FIRUIFigure.Position = [100 100 663 604];
            app.FIRUIFigure.Name = 'FIR 滤波结果';
            % Create outputTimeFig
            app.outputTimeFig = uiaxes(app.FIRUIFigure);
            title(app.outputTimeFig, '时域')
            xlabel(app.outputTimeFig, 't')
            ylabel(app.outputTimeFig, '信号大小')
            app.outputTimeFig.PlotBoxAspectRatio = [1 0.540816326530612
0.540816326530612];
            app.outputTimeFig.TitleFontWeight = 'bold';
            app.outputTimeFig.Position = [53 404 286 185];
            % Create outputSpectrumFig
            app.outputSpectrumFig = uiaxes(app.FIRUIFigure);
            title(app.outputSpectrumFig, '幅度响应')
            xlabel(app.outputSpectrumFig, '归一化角频率 ω (x\times \pi
rads/sample)')
            ylabel(app.outputSpectrumFig, '幅频衰减(dB)')
            app.outputSpectrumFig.PlotBoxAspectRatio = [1
0.472222222222222 0.472222222222222];
            app.outputSpectrumFig.XLim = [0 1];
            app.outputSpectrumFig.TitleFontWeight = 'bold';
            app.outputSpectrumFig.Position = [53 194 286 168];
            % Create Button
            app.Button = uibutton(app.FIRUIFigure, 'push');
            app.Button.ButtonPushedFcn = createCallbackFcn(app,
@playOutputSig, true);
            app.Button.Position = [478 24 84 43];
            app.Button.Text = '播放';
            % Create outputPhaseFig
            app.outputPhaseFig = uiaxes(app.FIRUIFigure);
            title(app.outputPhaseFig, '相位')
            xlabel(app.outputPhaseFig, '归一化角频率 ω (x\times \pi
rads/sample)')
```

```
            ylabel(app.outputPhaseFig, '相位 ')
            app.outputPhaseFig.PlotBoxAspectRatio = [1 0.472222222222222
0.472222222222222];
            app.outputPhaseFig.XLim = [0 1];
            app.outputPhaseFig.TitleFontWeight = 'bold';
            app.outputPhaseFig.Position = [53 1 286 168];
            % Create Label
            app.Label = uilabel(app.FIRUIFigure);
            app.Label.HorizontalAlignment = 'right';
            app.Label.Position = [388 404 113 22];
            app.Label.Text = '重叠保留法分块点数';
            % Create EditField
            app.EditField = uieditfield(app.FIRUIFigure, 'numeric');
            app.EditField.ValueChangedFcn = createCallbackFcn(app,
@pointChangeCallback, true);
            app.EditField.Position = [516 404 100 22];
            app.EditField.Value = 512;
            % Create Label_2
            app.Label_2 = uilabel(app.FIRUIFigure);
            app.Label_2.HorizontalAlignment = 'right';
            app.Label_2.Position = [448 349 53 22];
            app.Label_2.Text = '刷新秒数';
            % Create EditField_2
            app.EditField_2 = uieditfield(app.FIRUIFigure, 'numeric');
            app.EditField_2.ValueChangedFcn = createCallbackFcn(app,
@secondChangeCallback, true);
            app.EditField_2.Position = [516 349 100 22];
            app.EditField_2.Value = 1;
        end
    end
    methods (Access = public)
        % Construct app
        function app = FIRFilter(varargin)
            % Create and configure components
            createComponents(app)
            % Register the app with App Designer
            registerApp(app, app.FIRUIFigure)
            % Execute the startup function
            runStartupFcn(app, @(app)startupFcn(app, varargin{:}))
            if nargout == 0
                clear app
            end
        end
        % Code that executes before app deletion
        function delete(app)
            % Delete UIFigure when app is deleted
            delete(app.FIRUIFigure)
        end
    end
end
```

6.9 附录九 重叠保留法及动态展示

```
function X = overlapSave(x,h,L,puaseTime)
%   快速卷积重叠保留法
%   x: 输入信号
%   h: 冲积行向量
%   L: 卷积块大小 不能比 x 长度还大, 否则自动变为他的长度

if nargin<4
    puaseTime = 1;
end

close All
clear All
clc

% 确保单一

% 确保 x, y 是行向量
if ~isrow(x)
    x = x';
end
if ~isrow(h)
    h = h';
end
% 计算元素个数， 确保小于 L
[r,c] = size(x);
nx = r*c;
if L > nx
    L = nx;
end



N1=length(x);
```

```
M=length(h);

% 序列补齐 0 便于分割
x=[x  zeros(1,mod(-N1,L))  zeros(1,L)];
N2=length(x);
h=[h  zeros(1,L-1)];

% 计算 H(w)
H=fft(h,L+M-1);

% 共有 S 块
S=N2/L;
index=1:L;
xm=x(index);          % 第一次迭代
x1=[zeros(1,M-1)  xm];  % 开始位置补零
X=[];
figure()
title("卷积动态展示");
for  stage=1:S
      % ifft
      X1=fft(x1,L+M-1);
      Y=X1.*H;
      Y=ifft(Y);

      % 当前计算的 Y
      subplot(2,1,1)
      hold  off;
      stem(Y);
      xlabel(['第',num2str(stage),'块 ifft 结果']);


      % 绘制当前 X 状态
      subplot(2,1,2)
      hold  off;
```

```
        stem(X);
        xlabel(['前',num2str(stage),'块 ifft 叠加结果']);

        % 忽略前面的序列
        index2=M:M+L-1;
        Y=Y(index2);

        X=[X Y]; % 保存结果
        index3=(((stage)*L)-M+2):((stage+1)*L);      % 继续选择信号进行处理
        if(index3(L+M-1)<=N2)
            x1=x(index3);
        end
      pause(puaseTime);
end
i=1:N1+M-1;
X=X(i);


end
```

6.10 附录十 巴特沃兹自实现

```
function [order,wn] = myButtord(wp,ws,rp,rs,opt)
% 巴特沃兹滤波器阶数与截频选择函数
% 仿照官方 buttord 实现， 默认设计数字滤波
%   [N, Wn] = BUTTORD(Wp, Ws, Rp, Rs) 返回满足条件的最低的阶数
%   wp : 通带截止频率， 低高通 1x1 带通带阻为 1x2
%   ws : 阻带截止频率， 低高通 1x1 带通带阻为 1x2
%   rp : 通带最大衰减
%   rs : 阻带最大衰减
%   opt: 默认为 'z', 设计数字滤波， 指定为's'时设计模拟滤波

% 默认设计数字滤波器(opt='z'), 若指定's' ， 则设计模拟滤波器
if nargin == 4
    opt = 'z';
```

```
elseif nargin == 5
    if ~strcmp(opt,'z') && ~strcmp(opt,'s')
        error(message('signal:buttord:InvalidParam'));
    end
end

% 根据 wp, ws 参数选择滤波器类型
ftype = 2*(length(wp) - 1);
if wp(1) < ws(1)
    ftype = ftype + 1; % 低通 (1) or 带阻(3)
else
    ftype = ftype + 2; % 高通(2) or 带通 (4)
end

% 第一步: 完成滤波器指标(Π归一化)向模拟指标的转化
if strcmp(opt,'z')     % digital
    WP=tan(pi*wp/2);
    WS=tan(pi*ws/2);
else   % 如果指定模拟则不用做别的
    WP=wp;
    WS=ws;
end


% 第二步: 根据滤波器类型，利用频率转化，完成从低通到各种类型滤波器的转变
if ftype == 1   % low
    WA=WS/WP;
elseif ftype == 2   % high
    WA=WP/WS;
elseif ftype == 3   % stop
    fo = optimset('display','none');
    %wp1 = lclfminbnd('bscost',WP(1),WS(1)-1e-12,fo,1,WP,WS,rs,rp,'butter');
```

```
        %WP(1) = wp1;
        %wp2 = lclfminbnd('bscost',WS(2)+1e-12,WP(2),fo,2,WP,WS,rs,rp,'butter
');
        %WP(2) = wp2;
        WA=(WS*(WP(1)-WP(2)))./(WS.^2 - WP(1)*WP(2));
    elseif ftype == 4   % pass
        WA=(WS.^2 - WP(1)*WP(2))./(WS*(WP(1)-WP(2)));
    end
```

```
% 第三步（开始设计）：根据求 N 的式子与要求，求得满足要求的最低的巴特沃兹滤波器的阶数
    % 这里的 WA 即为 Omega_p / Omega_s
    WA=min(abs(WA));
    order = ceil( log10( (10 .^ (0.1*abs(rs)) - 1)./ ...
        (10 .^ (0.1*abs(rp)) - 1) ) / (2*log10(WA)) );

    % 第四步: 计算 3dB 截止频率
    % to give exactly rs dB at WA.  W0 will be between 1 and WA:
    W0 = WA / ( (10^(.1*abs(rs)) - 1)^(1/(2*(abs(order)))));

    % 第五步: 将截止频率从低通转化回对应的模拟滤波器类型
    if ftype == 1   % low
        WN=W0*WP;
    elseif ftype == 2   % high
        WN=WP/W0;
    elseif ftype == 3   % stop
        WN(1) = ( (WP(2)-WP(1)) + sqrt((WP(2)-WP(1))^2 + ...
            4*W0.^2*WP(1)*WP(2)))./(2*W0);
        WN(2) = ( (WP(2)-WP(1)) - sqrt((WP(2)-WP(1))^2 + ...
            4*W0.^2*WP(1)*WP(2)))./(2*W0);
        WN=sort(abs(WN));
    elseif ftype == 4   % pass
```

```
        W0=[-W0  W0];   % 左右的 3dB 截频
        WN=  -W0*(WP(2)-WP(1))/2  +  sqrt(  W0.^2/4*(WP(2)-WP(1))^2  +  WP
(1)*WP(2)  );
        WN=sort(abs(WN));
    end

    % 最后，  将模拟频率转化回数字频率
    if strcmp(opt,'z')      % digital
        wn=(2/pi)*atan(WN);   % 双线性变换
    else
        wn=WN;
    end
    end
```

6.11 附录十一 切比雪夫Ⅰ型自实现

```
    function  [order,wn]  =  myCheby(wp,ws,rp,rs,opt)
    % 切比雪夫Ⅰ型滤波器设计.
    %          Lowpass:      Wp  =  .1,          Ws  =  .2
    %          Highpass:     Wp  =  .2,          Ws  =  .1
    %          Bandpass:     Wp  =  [.2  .7],  Ws  =  [.1  .8]
    %          Bandstop:     Wp  =  [.1  .8],  Ws  =  [.2  .7]


    wp  =  signal.internal.sigcasttofloat(wp,'double','cheb1ord','Wp',...
      'allownumeric');
    ws  =  signal.internal.sigcasttofloat(ws,'double','cheb1ord','Ws',...
      'allownumeric');
    rp  =  signal.internal.sigcasttofloat(rp,'double','cheb1ord','Rp',...
      'allownumeric');
    rs  =  signal.internal.sigcasttofloat(rs,'double','cheb1ord','Rs',...
      'allownumeric');

    if  nargin  ==  4
        opt  =  'z';
```

```
elseif nargin == 5
    if ~strcmp(opt,'z') && ~strcmp(opt,'s')
        error(message('signal:cheb1ord:InvalidParam'));
    end
end



ftype = 2*(length(wp) - 1);
if wp(1) < ws(1)
    ftype = ftype + 1; % low (1) or reject (3)
else
    ftype = ftype + 2; % high (2) or pass (4)
end

if strcmp(opt,'z')     % digital
    WPA=tan(pi*wp/2);
    WSA=tan(pi*ws/2);
else
    WPA=wp;
    WSA=ws;
end

% 第二步: 根据滤波器类型，利用频率转化，完成从低通到各种类型滤波器的转变
if ftype == 1  % low
    WA=WSA/WPA;
elseif ftype == 2  % high
    WA=WPA/WSA;
elseif ftype == 3  % stop
    fo = optimset('display','none');
    %wp1 = lclfminbnd('bscost',WPA(1),WSA(1)-1e-12,fo,1,...
        %WPA,WSA,rs,rp,'cheby');
```

```
    %WPA(1) = wp1;

    %wp2 = lclfminbnd('bscost',WSA(2)+1e-12,WPA(2),fo,2,...
        %WPA,WSA,rs,rp,'cheby');
   % WPA(2) = wp2;
    WA=(WSA*(WPA(1)-WPA(2)))./(WSA.^2 - WPA(1)*WPA(2));
elseif ftype == 4   % pass
    WA=(WSA.^2 - WPA(1)*WPA(2))./(WSA*(WPA(1)-WPA(2)));
end
```

% 第三步（开始设计）：根据求 N 的式子与要求，求得满足要求的最低的巴特沃兹滤波器的阶数

```
% 这里的 WA 即为 Omega_p / Omega_s
WA=min(abs(WA));
order=ceil(acosh(sqrt((10^(.1*abs(rs))-1)/(10^(.1*abs(rp))-1)))/acosh(WA));
```

```
% 最后， 将模拟频率转化回数字频率
if strcmp(opt,'z')    % digital
    wn = wp;
else
    wn = WPA;
end
```