浙江大学实验报告

专业：电子科学与技术
姓名：
学号：
日期：
地点：

课程名称：数字信号处理　指导老师：＿＿＿＿　成绩：＿＿＿＿＿＿＿＿

实验名称：IIR 数字滤波器设计与使用　实验类型：综合　同组学生姓名：——

一、实验目的和要求

本实验用模拟滤波器设计方法设计 IIR 数字滤波器，并利用滤波器产生双音多频 DTMF 拨号信号，完成对其恢复，演示 IIR 数字滤波器在通信系统中的应用。通过本次实验，掌握用 MATLAB 实现 IIR 数字滤波器的设计方法和实现结构；了解通信系统电话 DTMF 拨号的基本原理和 IIR 滤波器实现方法。

二、实验内容和步骤

编写 MATLAB 程序，完成以下工作。

2-1 数字滤波器设计：分别用冲激响应不变法和双线性变换法设计巴特沃斯和切比雪夫数字低通滤波器，满足如下技术指标：通带截止频率 fp=100Hz，阻带起始频率 fs=300Hz，通带衰减要小于 3dB，阻带衰减要大于 20dB，fs=1000Hz。观察该滤波器的幅频特性曲线，记录带宽和衰减量，检查是否满足要求。比较这两种滤波器以及两种数字化实现方法的优缺点。

2-2 DTMF 编解码设计：DTMF 信号是将拨号盘上的 0~D 共 16 个数字，用音频范围的 8 个频率来表示的一种编码方式。8 个频率分为高频群和低频群两组，分别作为列频和行频。每个字符的信号由来自列频和行频的两个频率的正弦信号叠加而成。根据 ITU Q.23 建议，DTMF 信号的技术指标是：传送/接收率为每秒 10 个号码，或每个号码 100ms。每个号码传送过程中，信号存在时间至少 45ms, 且不多于 55ms, 100ms 的其余时间是静音。在每个频率点上允许有不超过±1.5%的频率误差。任何超过给定频率±3.5%的信号，均被认为是无效的，拒绝承认接收。另外，在最坏的检测条件下，信噪比不得低于 15dB。根据课堂所讲或查阅相关资料，编写 MATLAB 程序，完成以下工作：

（1）DTMF 信号的编码: 请对自己的学号用 DTMF 的原理进行编码。

提示：可以使用查表方式模拟产生两个不同频率的正弦波。正弦表的制定要保证合成信号的频率误差在±1.5%以内，同时使取样点数尽量少。

（2）DTMF 信号的解码：请对自己的学号用 DTMF 原理进行解码。

提示：1）直接可以采用 FFT 计算 N 点频率处的频谱值，然后估计出所拨号码。2）采用 Goertzel 算法设计 IIR 滤波器。该算法只需要知道 8 个特定点的频谱值，可以有效地提高计算效率。它相当于一个含两个极点的 IIR 滤波器，8 个频点对应各自相匹配的滤波器。

三、主要仪器设备

自行编程。程序见附录。

四、操作方法和实验步骤　（参见“二、实验内容和步骤”）

五、实验数据记录和处理

5-1 使用冲激响应不变法和双线性变换法设计巴特沃斯和切比雪夫数字低通滤波器，结果分别如图 1~图 4 所示。

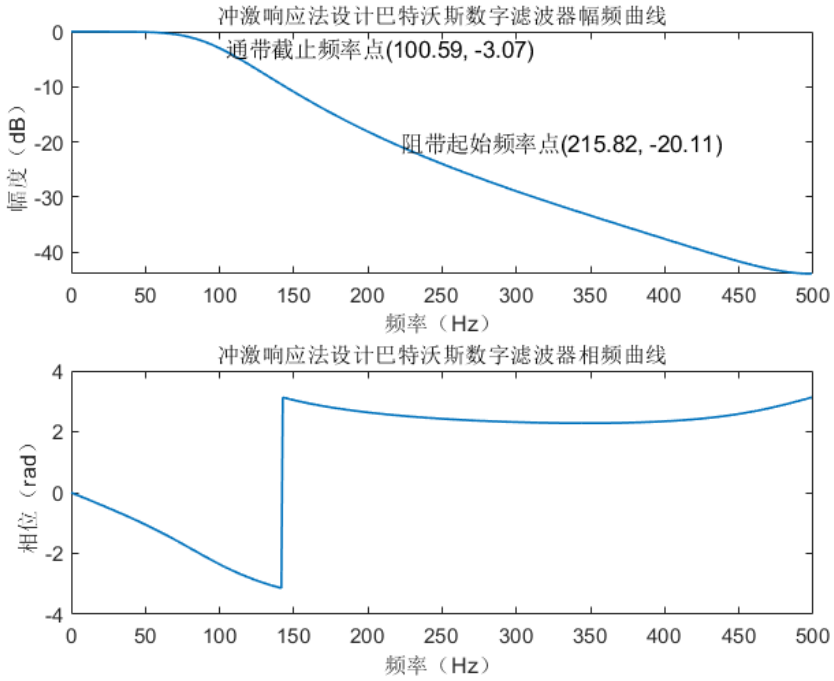

图 2 冲激响应不变法设计巴特沃斯数字低通滤波器结果

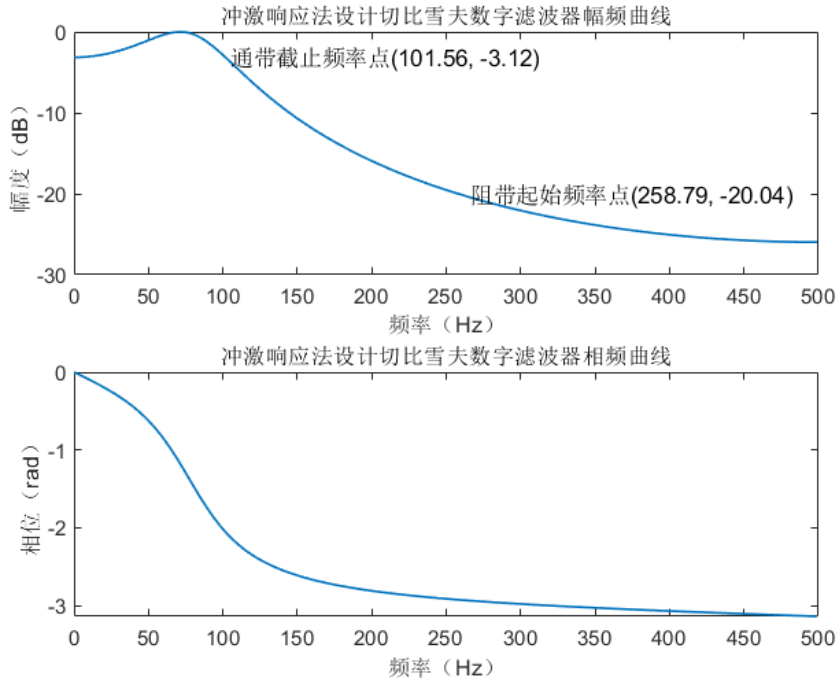

图 1 冲激响应不变法设计切比雪夫数字低通滤波器结果

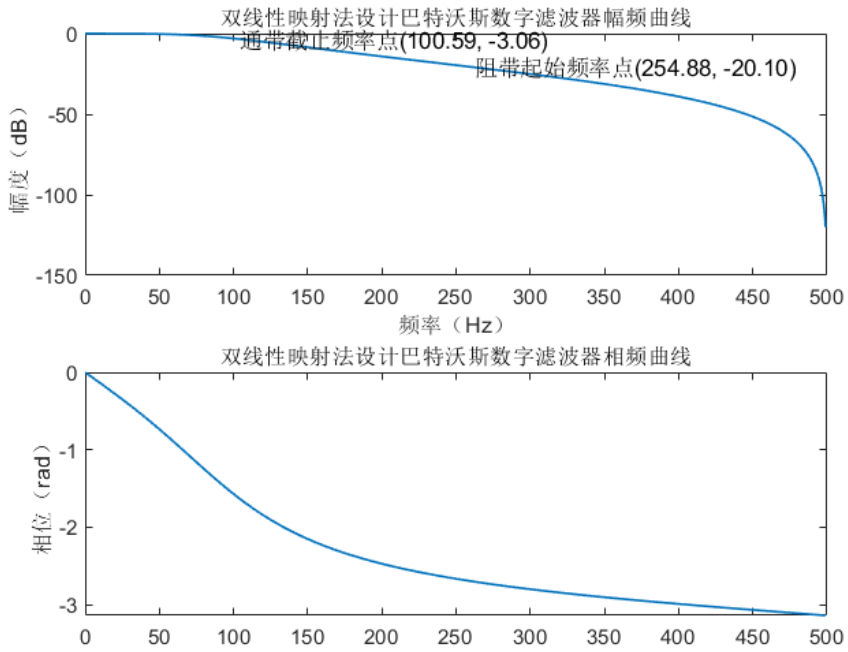

图 3 双线性变换法设计巴特沃斯数字低通滤波器结果

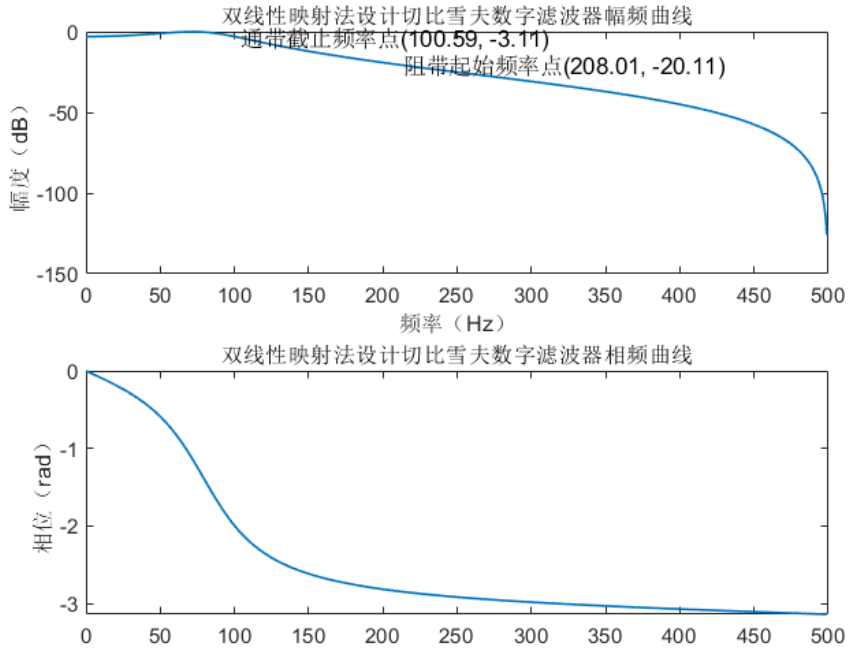

图 4 双线性变换法设计切比雪夫数字低通滤波器结果

根据结果，整理记录阶数、带宽（即通带截止频率）和衰减量，如表 1 所示。从表中可以看到，四种滤波器的各个指标均符合预期要求。

表 1 滤波器性能指标记录

类型	阶数 N	通带截止频率 /Hz	通带衰减 /dB	阻带起始频率 /Hz	阻带衰减 /dB
冲激响应不变法：巴特沃斯	3	100.59	3.07	215.82	20.11
冲激响应不变法：切比雪夫	2	101.56	3.12	258.79	20.04
双线性变换法：巴特沃斯	2	100.59	3.06	254.88	20.10
双线性变换法：切比雪夫	2	100.59	3.11	208.01	20.11

5-2-1 根据 DTMF 编码规则，使用查表法生成 DTMF 信号。

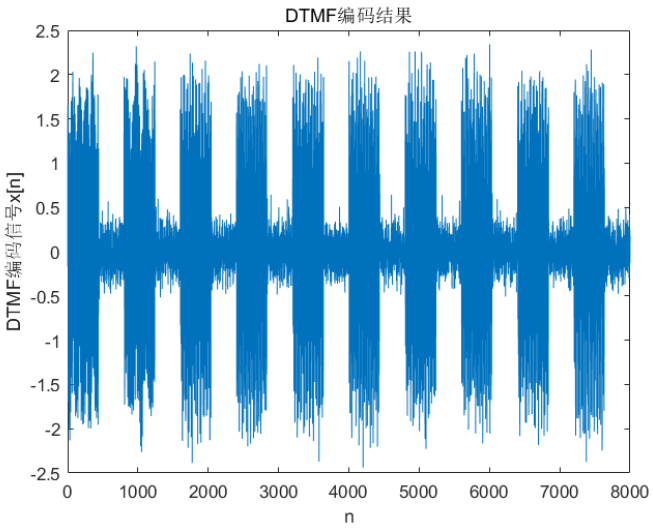

图 5 DTMF **编码结果**

当采样率为 8000Hz 时，由于信号存在时间至少 45ms，且不多于 55ms，计算得到信号存在时间内的采样点数 N 的范围为 360~440。暂取 N=440，得到 DTMF 信号如图 5。

5-2-2 直接使用 FFT 对编码信号进行分析，得到每一个号码段的信号 FFT 频谱如图 6~图 8。

对于每一个字符的 FFT 结果，高频值和低频值的误差均在相应标准基频的±3.5%以内，解码正确，终端输出正确的解码结果，如图 9 第一行。

采用 Goertzel 算法设计 IIR 滤波器进行解码。每一组低通滤波器的系统传递函数为：

$$H_k(z) = \frac{1 - W_N^k z^{-1}}{1 - 2\cos\left(\frac{2\pi k}{N}\right) z^{-1} + z^{-2}}$$

k为 DFT 的离散频率点，N为 DFT 长度，f_k为 DFT 音频频率，f_s为采样频率，它们满足：

$$k = N \cdot \frac{f_k}{f_s}$$

当采样频率为 8kHz、DFT 长度可变时，k的取值如表 2 所示。其中一次谐波的 DFT 长度为$N_1 = 205$，二次谐波的 DFT 长度为$N_2 = 201$。根据算法，输入序列$x[n]$的 DFT 结果$X(k)$和滤波器的输出之间的关系为：

$$X(k) = y_k[n]\Big|_{n = N_{1,2}} = y_k[N_{1,2}]$$

分别取每组滤波器第$N_{1,2}$个点的幅度平方值，当某频率的一次谐波滤波结果的幅度平方值很大但二次谐波滤波结果的幅度平方值较小时，该频率即为输入信号的 DTMF 编码频率，找到高、低两个频率，即可将 DTMF 信号解码。解码的结果如图 9 的第二行所示。

表 2 DTMF **解码** Goertzel **算法参数**

DTMF 频率(Hz)	离散频率点k(一次谐波)	离散频率点k(二次谐波)
697	18	35
770	20	39
852	22	43
941	24	47
1209	31	61
1336	34	67
1477	38	74
1633	42	82

逐步降低有效信号持续时间以降低N的取值（这里的N是原始信号中有效信号持续的长度，不是 DFT 长度），甚至突破 DTMF 编码规则的下限。取$N = 360,90,70$，得到结果如图 10 所示。

六、实验结果与分析

6-1-1 讨论：冲激响应不变法和双线性变换法在数字化实现方面的优缺点。

将设计过程中使用的模拟原型巴特沃斯和切比雪夫低通滤波器波特图给出，如图 11 所示。

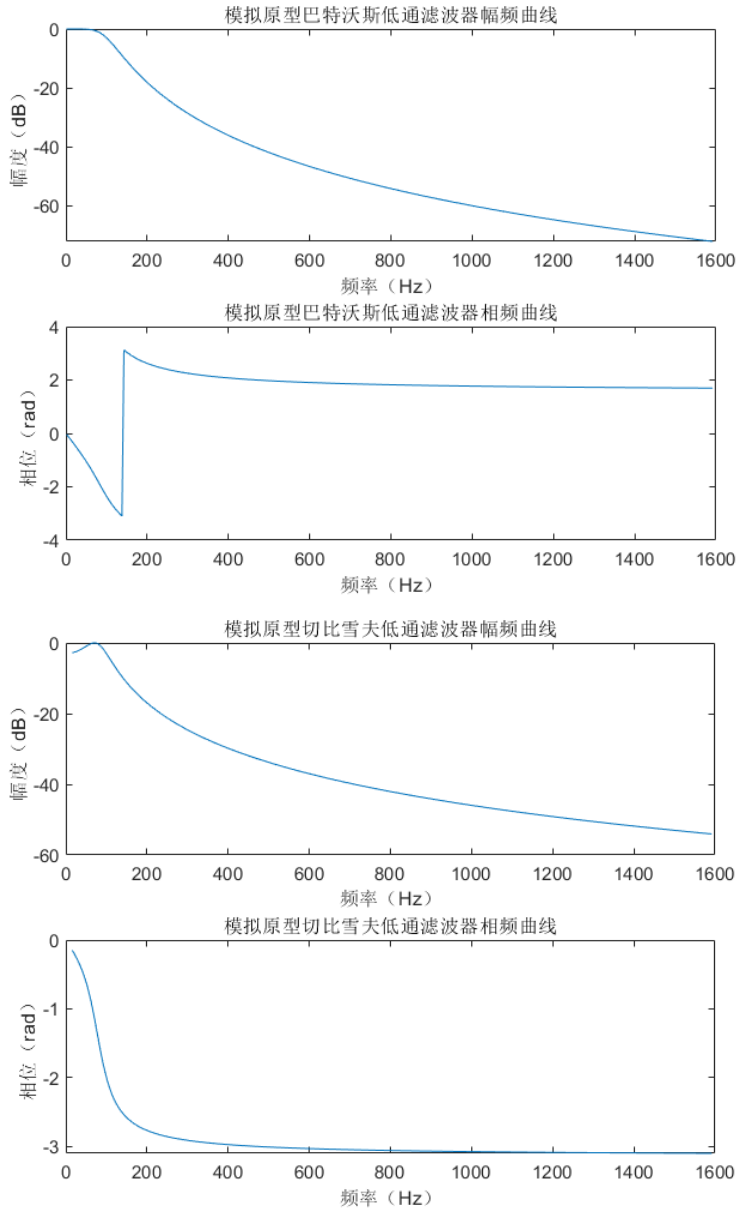

图 6 模拟原型巴特沃斯和切比雪夫低通滤波器

对比数字滤波器波特图（图 1~图 4）可得：

冲激响应不变法的优点：完全模仿模拟滤波器的单位冲激响应，即时域逼近良好，保证了模拟滤波器的时域瞬态特性。数字频率ω和模拟角频率Ω之间的关系是$\omega = \Omega T$，两者呈线性关系，没有线性失真问题。对比图 1、2 和图 11 可见，模拟滤波器和数字滤波器的波特图曲线形状基本一致，说明线性失真小。

冲激响应不变法的缺点：当ω自$[o, \pi]$和自$[-\pi, 0]$变化时，即ω在Z平面内单位圆变化一周时，Ω对应的值仅为$[0, \pi/T]$和$[-\pi/T, 0]$。如果模拟滤波器$H_a(\Omega)$不是在$[-\pi/T, \pi/T]$带限，或者取样频率不够高时，数字滤波器频谱将发生混叠失真。当$H_a(\Omega)$不严格带限，或$h_a(t)$变化不太平稳，而设计性能要求又高时，不宜

采用冲激响应不变法。另外，冲激响应不变法只适合频率响应在高频出单调递减的模拟原型滤波器，不能直接涉及高通、带阻滤波器，因为它们不是带限的。

双线性变换法的优点：在窄带内能近似保持原幅频特性，即使频带拖尾，也不产生混叠。S 平面和 Z 平面之间有简单的代数关系，可从模拟传递函数直接通过代数置换得到数字滤波器的系统函数。易于从模拟滤波器的幅频特性直接获得数字滤波器的幅频特性。对设计的滤波器类型没有限制。

双线性变换法的缺点：不再保持原有的线性相位，呈分段常数型，各个分段边缘的临界频率点产生了畸变，即相频特性为线性的滤波器经双线性变换后，线性相频特性遭到破坏。对比图 3、4 和图 11 可见，使用双线性变换法后，模拟滤波器和数字滤波器的波特图无论是幅频特性还是相频特性都发生了一定的扭曲和畸变。此外，双线性变换法不能直接得到滤波器实现所需要的传递函数或差分方程。

6-6-2 比较巴特沃斯和切比雪夫两种滤波器设计方法的优缺点。

巴特沃斯滤波器的特点是通频带内的频率响应曲线最大限度平坦，没有起伏，而在阻频带则逐渐下降为零，在过渡区下降缓慢，在阻带下降较快。巴特沃斯滤波器的频率特性曲线，无论在通带内还是阻带内都是频率的单调函数。因此，当通带的边界处满足指标要求时，通带内一定会有裕量。所以对巴特沃斯滤波器而言，更有效的设计方法应该是将精确度均匀的分布在整个通带或阻带内，或者同时分布在两者之内。这样就可用较低阶数的系统满足要求。这可通过选择具有等波纹特性的逼近函数来达到。相同阶数下，巴特沃斯滤波器实现更简单。

切比雪夫滤波器是在通带或阻带上频率响应幅度等波纹波动的滤波器，振幅特性在通带内是等波纹。在阻带内是单调的称为切比雪夫 I 型滤波器；振幅特性在通带内是单调的，在阻带内是等波纹的称为切比雪夫 II 型滤波器。当滤波器具有相同阶数时：巴特沃斯滤波器通带最平坦，阻带下降慢，如果想要提高阻带衰减就必须增加滤波器的阶数。切比雪夫滤波器通带等纹波，阻带下降较快，可以在相同的阶数下实现更快的衰减。

对比表 1 前两行可看到，实现同样指标，巴特沃斯滤波器需要 3 阶，而切比雪夫滤波器只需要 2 阶。对比表 1 后两行可以看到，同样阶数下，切比雪夫滤波器的阻带起始频率比巴特沃斯滤波器更低，说明切比雪夫滤波器衰减更快。

6-2-1 比较直接采用 FFT 和 Goertzel 两种方法的性能。

采用 FFT 时，对于每一个字符所对应的 DTMF 信号，在采样点数为N的情况下，所需的时间复杂度为$O(N\log_2 N)$，寻找两个对应谱峰所需的时间复杂度为$O(\log_2 N)$。由于 DTMF 是断续的单频串，进行 FFT 分析时需要兼顾时域分辨率和频域分辨率，而且得等待完整的数据后才能处理，存在时延问题。另外N的选择也受到限制，在$N=2^m$时 FFT 才能发挥最佳性能。

使用 Goertzel 算法时，对于每一个字符所对应的 DTMF 信号，需要使其通过 8 组 16 个 IIR 低通滤波

器，时间复杂度在$O(N)$数量级。但由于滤波器的结果是差分方程的迭代结果，而最终只需要取$n = N$时的值（即最后一个值）进行幅度平方的门限检测，包含二次谐波在内共计 16 个点的值是真正需要的，因此只需在滤波器的反馈支路的迭代结束时计算一次，只需很小的数据存储量，运算速度快，不需要等待数据输入完成时才开始进行处理，效率较高。

6-2-2 讨论算法中N的选择对实验性能的影响

根据图 10 的结果可见，当N不断减小，甚至低于 DTMF 编码规则下限时，FFT 的结果误差在逐渐加大。当$N = 90$时 FFT 结果的误差已经超过了$\pm 3.5\%$，不能被识别为有效信号，导致无法解码，甚至会分布到其他基频的区间范围内，导致解码错误，但当$N = 70$时 Goertzel 算法仍然可以实现正确解码，说明 Goertzel 算法受噪声等因素影响小，精度高，不需要很高的N值，占用资源少。此外，Goertzel 算法虽然需要考虑频率分辨率和计算时间的折中，但N值可以为任意整数。但需要注意的是。Goertzel 滤波器的极点都在单位圆上，对有限字长效应十分敏感，若需要检测的频率点数很大，FFT 可能更适合。根据实验结果也可以得知，当N取值足够大时，FFT 的精度已经足以满足要求。

七、附录

```
%IIRDesign.m
%IIR 滤波器设计
%数字滤波器参数
clc;clear;close all;
fm= 1600; %信号频率最大值，单位 Hz
Fs = 1000;    %抽样频率
fp = 100; fs = 300; %数字低通滤波器参数
Rp = 3;         %通带波动
Rs = 20;        %阻带衰减

%冲激响应不变法，无需频率预畸变
%巴特沃斯滤波器
Wp = 2*pi*fp; Ws = 2*pi*fs;
[N,~] = buttord(Wp,Ws,Rp,Rs,'s');%计算巴特沃斯模拟滤波器参数，'s'表示 Wp 和 Ws 都是模拟角频率
[b,a] = butter(N,Wp,'low','s');      %设计巴特沃斯低通滤波器
[H,W]=freqs(b,a);                   %W:模拟角频率，H:模拟滤波器的系统函数
mag=abs(H);                         %幅度
pha=angle(H);                       %相位
db=20*log10((mag+eps)/max(mag));     %转换为分贝
f=W/(2*pi);                         %将模拟角频率转为 Hz
figure(1);
subplot(2,1,1);plot(f,db);
title('模拟原型巴特沃斯低通滤波器幅频曲线');xlabel('频率（Hz）');ylabel('幅度（dB）');
subplot(2,1,2);plot(f,pha);
title('模拟原型巴特沃斯低通滤波器相频曲线');xlabel('频率（Hz）');ylabel('相位（rad）');
```

```
fprintf(['冲激响应不变法设计巴特沃斯滤波器的阶数：',num2str(N),'\n']);
%数字化
[B,A] = impinvar(b,a,Fs);           %冲激响应不变法
[H,W] = freqz(B,A);                 %数字滤波器系统函数
mag = abs(H);                       %幅度
pha = angle(H);                     %相位
db = 20*log10((mag+eps)/max(mag));  %转换为分贝 db
f = W*Fs/(2*pi);                    %将数字角频率转换为频率 Hz
figure(2);
subplot(2,1,1);plot(f,db,LineWidth=1);
title('冲激响应法设计巴特沃斯数字滤波器幅频曲线');xlabel('频率（Hz）');ylabel('幅度（dB）');
dbindex = find(db <= -Rp,1);
dbvalue = db(dbindex);
findex = find(db==dbvalue,1);
fvalue = f(findex);
format = sprintf('通带截止频率点(%.2f, %.2f)',fvalue,dbvalue);
text(findex,dbvalue,format);
dbindex = find(db <= -Rs,1);
dbvalue = db(dbindex);
findex = find(db==dbvalue,1);
fvalue = f(findex);
format = sprintf('阻带起始频率点(%.2f, %.2f)',fvalue,dbvalue);
text(findex,dbvalue,format);
subplot(2,1,2);plot(f,pha,LineWidth=1);
title('冲激响应法设计巴特沃斯数字滤波器相频曲线');xlabel('频率（Hz）');ylabel('相位（rad）
');

%切比雪夫滤波器
[N,~]=cheb1ord(Wp,Ws,Rp,Rs,'s');  %计算切比雪夫模拟滤波器参数，‘s’表示 Wp 和 Ws 都是模拟角频率
[b,a]=cheby1(N,Rp,Wp,'low','s');       %设计切比雪夫 1 低通滤波器
[H,W]=freqs(b,a);
mag=abs(H);
pha=angle(H);
db=20*log10((mag+eps)/max(mag));
f=W/(2*pi);
figure(3);
subplot(2,1,1);plot(f,db);
title('模拟原型切比雪夫低通滤波器幅频曲线');xlabel('频率（Hz）');ylabel('幅度（dB）');
subplot(2,1,2);plot(f,pha);
title('模拟原型切比雪夫低通滤波器相频曲线');xlabel('频率（Hz）');ylabel('相位（rad）');
fprintf(['冲激响应不变法设计切比雪夫滤波器的阶数：',num2str(N),'\n']);
%数字化
[B,A] = impinvar(b,a,Fs);           %冲激响应不变法
[H,W] = freqz(B,A);                 %数字滤波器系统函数
```

```
mag = abs(H);
pha = angle(H);
db = 20*log10((mag+eps)/max(mag));
f = W*Fs/(2*pi);
figure(4);
subplot(2,1,1);plot(f,db,LineWidth=1);
title('冲激响应法设计切比雪夫数字滤波器幅频曲线');xlabel('频率（Hz）');ylabel('幅度（dB）');
dbindex = find(db >= -Rp,1,"last") + 1;
dbvalue = db(dbindex);
findex = find(db==dbvalue,1);
fvalue = f(findex);
format = sprintf('通带截止频率点(%.2f, %.2f)',fvalue,dbvalue);
text(findex,dbvalue,format);
dbindex = find(db <= -Rs,1);
dbvalue = db(dbindex);
findex = find(db==dbvalue,1);
fvalue = f(findex);
format = sprintf('阻带起始频率点(%.2f, %.2f)',fvalue,dbvalue);
text(findex,dbvalue,format);
subplot(2,1,2);plot(f,pha,LineWidth=1);
title('冲激响应法设计切比雪夫数字滤波器相频曲线');xlabel('频率（Hz）');ylabel('相位（rad）
');

%双线性变换法，需要频率预畸变
wp = 2*pi*fp/Fs; ws=2*pi*fs/Fs;              %转换为数字角频率技术指标
Wp = (2*Fs)*tan(wp/2);Ws =(2*Fs)*tan(ws/2);   %将数字技术指标的反畸变为模拟指标
%巴特沃斯滤波器
[N,~] = buttord(Wp,Ws,Rp,Rs,'s'); %计算巴特沃斯模拟滤波器参数， s 表示 Wp 和 Ws 都是模拟角频率
[b,a] = butter(N,Wp,'low','s');                %设计巴特沃斯低通滤波器
fprintf(['双线性变换法设计巴特沃斯滤波器的阶数：',num2str(N),'\n']);
%数字化
[B,A] = bilinear(b,a,Fs);                      %冲激响应不变法
[H,W] = freqz(B,A);                           %数字滤波器系统函数
mag = abs(H);
pha = angle(H);
db = 20*log10((mag+eps)/max(mag));
f = W*Fs/(2*pi);
figure(5);
subplot(2,1,1);plot(f,db,LineWidth=1);
title('双线性映射法设计巴特沃斯数字滤波器幅频曲线');xlabel('频率（Hz）');ylabel('幅度（dB）
');
dbindex = find(db <= -Rp,1);
dbvalue = db(dbindex);
findex = find(db==dbvalue,1);
```

```
fvalue = f(findex);
format = sprintf('通带截止频率点(%.2f, %.2f)',fvalue,dbvalue);
text(findex,dbvalue,format);
dbindex = find(db <= -Rs,1);
dbvalue = db(dbindex);
findex = find(db==dbvalue,1);
fvalue = f(findex);
format = sprintf('阻带起始频率点(%.2f, %.2f)',fvalue,dbvalue);
text(findex,dbvalue,format);
subplot(2,1,2);plot(f,pha,LineWidth=1);
title('双线性映射法设计巴特沃斯数字滤波器相频曲线');xlabel('频率（Hz）');ylabel('相位（rad）
');
%切比雪夫滤波器
[N,~] = cheb1ord(Wp,Ws,Rp,Rs,'s');%计算切比雪夫模拟滤波器参数，s 表示 Wp 和 Ws 都是模拟角频率
[b,a] = cheby1(N,Rp,Wp,'low','s');              %设计切比雪夫 1 低通滤波器
fprintf(['双线性变换法设计切比雪夫滤波器的阶数：',num2str(N),'\n']);
%数字化
[B,A] = bilinear(b,a,Fs);                       %冲激响应不变法
[H,W] = freqz(B,A);                             %数字滤波器系统函数
mag = abs(H);
pha = angle(H);
db = 20*log10((mag+eps)/max(mag));
f = W*Fs/(2*pi);
figure(6);
subplot(2,1,1);plot(f,db,LineWidth=1);
title('双线性映射法设计切比雪夫数字滤波器幅频曲线');xlabel('频率（Hz）');ylabel('幅度（dB）
');
dbindex = find(db <= -Rp,1);
dbvalue = db(dbindex);
findex = find(db==dbvalue,1);
fvalue = f(findex);
format = sprintf('通带截止频率点(%.2f, %.2f)',fvalue,dbvalue);
text(findex,dbvalue,format);
dbindex = find(db <= -Rs,1);
dbvalue = db(dbindex);
findex = find(db==dbvalue,1);
fvalue = f(findex);
format = sprintf('阻带起始频率点(%.2f, %.2f)',fvalue,dbvalue);
text(findex,dbvalue,format);
subplot(2,1,2);plot(f,pha,LineWidth=1);
title('双线性映射法设计切比雪夫数字滤波器相频曲线');xlabel('频率（Hz）');ylabel('相位（rad）
');

%DTMF.m
```

```
%DTMF 编码
clc;clear;close all;
fl = [697 770 852 941];      %低频频率
fh = [1209 1336 1477 1633];  %高频频率
SNR = 15;                    %信噪比
Fs = 8000;                   %采样频率 8kHz
N = 440;                     %8000Hz 采样率下 100ms 内有 800 个点，信号采样点在 360~440 之间
n = 0:N-1;
x=[];
numString='123456789';%要转换的号码
%生成编码信号
for i = 1:length(numString)
    switch numString(i)
        case'1'
            freq_low=fl(1);freq_hgh=fh(1);
        case'2'
            freq_low=fl(1);freq_hgh=fh(2);
        case'3'
            freq_low=fl(1);freq_hgh=fh(3);
        case'4'
            freq_low=fl(2);freq_hgh=fh(1);
        case'5'
            freq_low=fl(2);freq_hgh=fh(2);
        case'6'
            freq_low=fl(2);freq_hgh=fh(3);
        case'7'
            freq_low=fl(3);freq_hgh=fh(1);
        case'8'
            freq_low=fl(3);freq_hgh=fh(2);
        case'9'
            freq_low=fl(3);freq_hgh=fh(3);
        case'0'
            freq_low=fl(4);freq_hgh=fh(2);
        case'*'
            freq_low=fl(4);freq_hgh=fh(1);
        case'#'
            freq_low=fl(3);freq_hgh=fh(3);
        otherwise
            error('naive!');
    end
    xi = sin(2*pi*freq_low/Fs*n) + sin(2*pi*freq_hgh/Fs*n);
    xi = [xi,zeros(1,800-N)];   %#ok<*AGROW> %留出静音
    x = [x,xi];                 %将每个按键串在一起
end
```

```
x = awgn(x,SNR);                %加入白噪声
figure(1);
plot(x);xlabel('n');ylabel('DTMF 编码信号 x[n]');title('DTMF 编码结果');
%使用 FFT 进行解码
decode_x_FFT = [];
for i = 1:length(numString)
    xi = x((i-1)*800+1:1:(i-1)*800+N);
    X = fft(xi);
    X = fftshift(X);        %转移到[-pi,pi]的对称区间内
    figure(i+1);
    plot(Fs/N*(-N/2:N/2-1),abs(X));
    xlabel('f(Hz)');ylabel('Abs(X)');
    title(['第',num2str(i),'个字符的 DTMF 信号频谱']);
    Xtmp = X(N/2+1:N);      %截取正半轴序列，寻找两处峰值对应的频率即为 DTMF 信号的两个基频
    F1value = max(abs(Xtmp));
    F1index = find(abs(Xtmp)==F1value);
    Xtmp(F1index) = 0;
    F2value = max(abs(Xtmp));
    F2index = find(abs(Xtmp)==F2value);
    freq1 = (F1index+N/2-1)*Fs/N-Fs/2;
    freq2 = (F2index+N/2-1)*Fs/N-Fs/2;
    fHigh = max(freq2,freq1);%高频信号
    fLow = min(freq2,freq1);%低频信号
%     format = '第%d 个信号 FFT 分解结果(fh,fl) = (%.2f,%.2f)\n';
%     fprintf(format,i,fHigh,fLow);
    if (abs(fHigh - fh(1))/fh(1) <= 0.035)      %判断误差并解码
        if(abs(fLow - fl(1))/fl(1) <= 0.035)
            decode_x_FFT(i) = '1';
        elseif(abs(fLow - fl(2))/fl(2) <= 0.035)
            decode_x_FFT(i) = '4';
        elseif(abs(fLow - fl(3))/fl(3) <= 0.035)
            decode_x_FFT(i) = '7';
        elseif(abs(fLow - fl(4))/fl(4) <= 0.035)
            decode_x_FFT(i) = '*'; %#ok<*SAGROW>
        else
            decode_x_FFT(i) = '?';
            fprintf(['FFT:第',num2str(i),'个字符解码错误:低频信号超出最大允许频率误差\n']);
        end
    elseif (abs(fHigh - fh(2))/fh(2) <= 0.035)
        if(abs(fLow - fl(1))/fl(1) <= 0.035)
            decode_x_FFT(i) = '2';
        elseif(abs(fLow - fl(2))/fl(2) <= 0.035)
            decode_x_FFT(i) = '5';
        elseif(abs(fLow - fl(3))/fl(3) <= 0.035)
```

```
            decode_x_FFT(i) = '8';
        elseif(abs(fLow - fl(4))/fl(4) <= 0.035)
            decode_x_FFT(i) = '0';
        else
            decode_x_FFT(i) = '?';
            fprintf(['FFT:第',num2str(i),'个字符解码错误:低频信号超出最大允许频率误差\n']);
        end
    elseif (abs(fHigh - fh(3))/fh(3) <= 0.035)
        if(abs(fLow - fl(1))/fl(1) <= 0.035)
            decode_x_FFT(i) = '3';
        elseif(abs(fLow - fl(2))/fl(2) <= 0.035)
            decode_x_FFT(i) = '6';
        elseif(abs(fLow - fl(3))/fl(3) <= 0.035)
            decode_x_FFT(i) = '9';
        elseif(abs(fLow - fl(4))/fl(4) <= 0.035)
            decode_x_FFT(i) = '#';
        else
            decode_x_FFT(i) = '?';
            fprintf(['FFT:第',num2str(i),'个字符解码错误:低频信号超出最大允许频率误差\n']);
        end
    else
        decode_x_FFT(i) = '?';
        fprintf(['FFT:第',num2str(i),'个字符解码错误:高频信号超出最大允许频率误差\n']);
    end
end
fprintf(['N = ',num2str(N),',']);
fprintf(['FFT 解码结果:',decode_x_FFT,'\n']);
%使用 Goertzel 算法进行解码
k = [18 20 22 24 31 34 38 42];      %一次谐波的离散频率点
k2 = [35 39 43 47 61 67 74 82];     %二次谐波的离散频率点
N1 = 205;                           %一次谐波的 FFT 点数 N1
N2 = 201;                           %二次谐波的 FFT 点数 N2
W_N1 = exp(-1j*2*pi/N1);
W_N2 = exp(-1j*2*pi/N2);
decode_x_Gtl = [];
for i = 1:length(numString)
    xi = x((i-1)*800+1:1:(i-1)*800+N); %分段截取原始信号
    if N1 <=N                          %x1,x2 分别为分析一次谐波和二次谐波时的输入序列
        x1 = xi(1:N1);                 %根据 N 和 N1,N2 的关系进行截断或补 0
    else
        x1 = [xi,zeros(1,N1-N)];
    end
    if N2 <=N
        x2 = xi(1:N2);
```

```
    else
        x2 = [xi,zeros(1,N2-N)];
    end
    %对每一段信号进行一次谐波和二次谐波的滤波
    yk_697_1 = filter([1 -W_N1^k(1)],[1 -2*cos(2*pi*k(1)/N1) 1],x1);
    yk_697_2 = filter([1 -W_N2^k2(1)],[1 -2*cos(2*pi*k2(1)/N2) 1],x2);
    yk_770_1 = filter([1 -W_N1^k(2)],[1 -2*cos(2*pi*k(2)/N1) 1],x1);
    yk_770_2 = filter([1 -W_N2^k2(2)],[1 -2*cos(2*pi*k2(2)/N2) 1],x2);
    yk_852_1 = filter([1 -W_N1^k(3)],[1 -2*cos(2*pi*k(3)/N1) 1],x1);
    yk_852_2 = filter([1 -W_N2^k2(3)],[1 -2*cos(2*pi*k2(3)/N2) 1],x2);
    yk_941_1 = filter([1 -W_N1^k(4)],[1 -2*cos(2*pi*k(4)/N1) 1],x1);
    yk_941_2 = filter([1 -W_N2^k2(4)],[1 -2*cos(2*pi*k2(4)/N2) 1],x2);
    yk_1209_1 = filter([1 -W_N1^k(5)],[1 -2*cos(2*pi*k(5)/N1) 1],x1);
    yk_1209_2 = filter([1 -W_N2^k2(5)],[1 -2*cos(2*pi*k2(5)/N2) 1],x2);
    yk_1336_1 = filter([1 -W_N1^k(6)],[1 -2*cos(2*pi*k(6)/N1) 1],x1);
    yk_1336_2 = filter([1 -W_N2^k2(6)],[1 -2*cos(2*pi*k2(6)/N2) 1],x2);
    yk_1477_1 = filter([1 -W_N1^k(7)],[1 -2*cos(2*pi*k(7)/N1) 1],x1);
    yk_1477_2 = filter([1 -W_N2^k2(7)],[1 -2*cos(2*pi*k2(7)/N2) 1],x2);
    %yk_1633_1 = filter([1 -W_N1^k(8)],[1 -2*cos(2*pi*k(8)/N1) 1],x1);
    %yk_1633_2 = filter([1 -W_N2^k2(8)],[1 -2*cos(2*pi*k2(8)/N2) 1],x2);
    if (abs(yk_697_1(N1))^2 >= 1000) && (abs(yk_697_2(N2))^2 <= 1000)
        if (abs(yk_1209_1(N1))^2 >= 1000) && (abs(yk_1209_2(N2))^2 <= 1000)
            decode_x_Gtl(i) = '1';
        elseif (abs(yk_1336_1(N1))^2 >= 1000) && (abs(yk_1336_2(N2))^2 <= 1000)
            decode_x_Gtl(i) = '2';
        elseif (abs(yk_1477_1(N1))^2 >= 1000) && (abs(yk_1477_2(N2))^2 <= 1000)
            decode_x_Gtl(i) = '3';
        else
            decode_x_Gtl(i) = '?';
            fprintf(['Goertzel:第',num2str(i),'个字符解码错误:当前频率下找不到对应键，高频错
误\n']);
        end
    elseif (abs(yk_770_1(N1))^2 >= 1000) && (abs(yk_770_2(N2))^2 <= 1000)
        if (abs(yk_1209_1(N1))^2 >= 1000) && (abs(yk_1209_2(N2))^2 <= 1000)
            decode_x_Gtl(i) = '4';
        elseif (abs(yk_1336_1(N1))^2 >= 1000) && (abs(yk_1336_2(N2))^2 <= 1000)
            decode_x_Gtl(i) = '5';
        elseif (abs(yk_1477_1(N1))^2 >= 1000) && (abs(yk_1477_2(N2))^2 <= 1000)
            decode_x_Gtl(i) = '6';
        else
            decode_x_Gtl(i) = '?';
            fprintf(['Goertzel:第',num2str(i),'个字符解码错误:当前频率下找不到对应键，高频错
误\n']);
        end
```

```
    elseif (abs(yk_852_1(N1))^2 >= 1000) && (abs(yk_852_2(N2))^2 <= 1000)
        if (abs(yk_1209_1(N1))^2 >= 1000) && (abs(yk_1209_2(N2))^2 <= 1000)
            decode_x_Gtl(i) = '7';
        elseif (abs(yk_1336_1(N1))^2 >= 1000) && (abs(yk_1336_2(N2))^2 <= 1000)
            decode_x_Gtl(i) = '8';
        elseif (abs(yk_1477_1(N1))^2 >= 1000) && (abs(yk_1477_2(N2))^2 <= 1000)
            decode_x_Gtl(i) = '9';
        else
            decode_x_Gtl(i) = '?';
            fprintf(['Goertzel:第',num2str(i),'个字符解码错误:当前频率下找不到对应键，高频错
误\n']);
        end
    elseif (abs(yk_941_1(N1))^2 >= 1000) && (abs(yk_941_2(N2))^2 <= 1000)
        if (abs(yk_1209_1(N1))^2 >= 1000) && (abs(yk_1209_2(N2))^2 <= 1000)
            decode_x_Gtl(i) = '*';
        elseif (abs(yk_1336_1(N1))^2 >= 1000) && (abs(yk_1336_2(N2))^2 <= 1000)
            decode_x_Gtl(i) = '0';
        elseif (abs(yk_1477_1(N1))^2 >= 1000) && (abs(yk_1477_2(N2))^2 <= 1000)
            decode_x_Gtl(i) = '#';
        else
            decode_x_Gtl(i) = '?';
            fprintf(['Goertzel:第',num2str(i),'个字符解码错误:当前频率下找不到对应键，高频错
误\n']);
        end
    else
        decode_x_Gtl(i) = '?';
        fprintf(['Goertzel:第',num2str(i),'个字符解码错误:当前频率下找不到对应键，低频错误
\n']);
    end
end
fprintf(['N = ',num2str(N),',']);
fprintf(['Goetrzel 解码结果:',decode_x_Gtl,'\n']);
```