湖南工业大学

课 程 设 计

资 料 袋

交通工程 学院（系、部） 2019 － 2020 学年第 1 学期

课程名称 信号与信息处理综合课程设计 指导教师 舒小华

学生姓名 夏渔平 专业班级 信息工程 1701 学号 17419001163

题 目 语音信号处理及软件界面设计

成 绩 起止日期 2019 年 12 月 16 日～ 2020 年 1 月 3 日

目 录 清 单

序号	材 料 名 称	资料数量	备 注
1	课程设计任务书	1	
2	课程设计说明书	1	
3			
4			
5			
6			

湖南工业大学

课程设计任务书

2019—2020 学年度　第　1 学期

交通工程　学院（系、部）　信息工程　专业　1701　班级

课程名称：信号与信息处理综合课程设计

设计题目：语音信号处理及软件界面设计

完成期限：2019　年　12　月　16　日～　2020　年　1　月　3　日　共　3　周

<table>
<tr><td>内容及任务</td><td colspan="2">在学习和掌握数字信号处理的理论基础上，综合运用数字信号处理的知识进行语音信号频谱分析和滤波器设计，利用 MATLAB 作为编程工具进行软件仿真设计，再用 VC 进行算法设计及计算机实现，本课程综合数字信号处理、语音信号处理、MATLAB 程序设计、VC 软件设计等多门课程知识与技能，进行综合设计。具体任务如下：
1）基于 MATLAB 的仿真程序设计。在 MATLAB 平台下，进行时域和频域的分析。设计不同滤波器进行滤波，绘出叠加噪声后的语音信号时域和频谱图，与原始语音信号图形对比，进行时域和频域的分析，并从听觉上感觉滤波前后的区别。
2）基于 VC 的程序设计。利用 VC 设计一 WAV 格式的语音录音和播放器。
3）利用设计好的录音器录制一段语音信号，在 VC 平台下，设计不同滤波参数及类型的滤波进行语音滤波，并从听觉上感觉滤波前后的区别。</td></tr>
<tr><td rowspan="4">进度安排</td><td>起止日期</td><td>工作内容</td></tr>
<tr><td>2019 年 12 月 16 日～12 月 20 日</td><td>基于 MATLAB 的仿真程序设计
课程设计内容讲解：FIR 滤波器设计、IIR 滤波器设计、采用设计的滤波器对语音进行滤波处理，并进行时域与频域的分析。
基于 MATLAB 的软件界面设计。</td></tr>
<tr><td>2019 年 12 月 23 日～12 月 27 日</td><td>基于 VC 的程序设计。利用 VC 设计一 WAV 格式的语音录音和播放器
VC 知识讲解及学习与训练、界面制作；
WAV 录音器设计；WAV 语音播放器设计。</td></tr>
<tr><td>2019 年 12 月 30 日～2020 年 1 月 3 日</td><td>WAV 格式的语音读入、处理及播放。
利用 c++语言编写语音滤波器程序，包含 IIR 和 FIR 滤波程序，对语音进行滤波，并根据滤波指标对比滤波前后的语音的区别
检查考核。</td></tr>
<tr><td>主要参考资料</td><td colspan="2">[1] 吴镇扬. 数字信号处理[M] （第 4 版）. 北京：高等教育出版社，2016
[2] 陈超.MATLAB 应用实例精讲[M] （第 2 版）。电子工业出版社,2011
[3] 明日科技. Visual C++从入门到精通[M]（第 4 版）。清华大学出版社,2017</td></tr>
</table>

指导教师（签字）：_____　　2019 年　　月　　日

系（教研室）主任（签字）：_____　　2019 年　　月　　日

信号与信息处理

设　计　说　明　书

语音信号处理及软件界面设计

起止日期：2019 年 12 月 16 日～ 2020 年 1 月 3 日 共 3 周

学　生　姓　名	夏渔平
班　　级	信息工程 1701 班
学　　号	17419001163
成　　绩	
指导教师(签字)	

交通工程学院

2020 年　1 月　4 日

信号与信息处理综合课程设计

一、 课程设计目的

在学习和掌握数字信号处理的理论基础上，综合运用数字信号处理的知识进行语音信号频谱分析和滤波器设计，利用 MATLAB 作为编程工具进行软件仿真设计，再用 VC 进行算法设计及计算机实现，本课程综合数字信号处理、语音信号处理、MATLAB 程序设计、VC 软件设计等多门课程知识与技能，进行综合设计。

二、 课程设计要求

根据整个设计系统要实现的功能，本次课程设计主要包含三个部分，分别是基于 MATLAB 的仿真程序设计、基于 VC 的程序设计、基于 VC++的语音滤波算法。

1、基于 MATLAB 的仿真程序设计。在 MATLAB 平台下，进行时域和频域的分析。设计不同滤波器进行滤波，绘出叠加噪声后的语音信号时域和频谱图，与原始语音信号图形对比，进行时域和频域的分析，并从听觉上感觉滤波前后的区别。

利用 MATLAB 软件，根据给定的数据，设计不同类型的滤波器。主要分为两大类，无限长单位脉冲响应滤波器(IIR 滤波器)和有限长单位脉冲响应滤波器(FIR 滤波器)，IIR 型和 FIR 型滤波器又分别分为低通、高通、带通及带阻四种类型。在进行语音信号处理前，需要由录音设备录制一段 2-3 秒的 WAV 格式音频文件，然后再由 MATLAB 内集成的语音读取函数对音频进行数据的读取，用于音频文件时域及频域的分析和滤波器对信号进行滤波处理等。

2、基于 VC 的程序设计。利用 VC 设计一 WAV 格式的语音录音和播放器。

利用 VC 下的微软基础类库(MFC)中包含的 Windows 句柄封装类和 Windows 的内建控件和组装的封装类设计一具有录制和播放 WAV 格式音频的录音和播放器。

3、利用设计好的录音器录制一段语音信号，在 VC 平台下，设计不同滤波参数及类型的滤波进行语音滤波，并从听觉上感觉滤波前后的区别。

利用上述 2 中所设计的录音器录制一 8 位，采样频率为 22050 的 WAV 格式音频文件以用于信号滤波处理，以 C++语言分别设计出不同类型的滤波器对信号进行滤波，并且对比滤波前后的变化及区别。

三、 整个项目实现的效果和功能

1、IIR 滤波器的设计

1）低通滤波器 滤波器参数为采样频率$f_s = 8820hz$，通带截止频率$f_c = 1000hz$，阻带截止频率为$f_r = 1200hz$，通带波动 $\delta = 1dB$，阻带衰减为 $At = 40dB$。MATLAB 程序代码如下：

```
fc1=1000;fr1=1200;fs1=fs/5;rs=40;ap=1;
Wc1=2*fs1*tan(2*pi*fc1/(2*fs1));
Wr1=2*fs1*tan(2*pi*fr1/(2*fs1));
[N1,Wn1]=buttord(Wc1,Wr1,ap,rs,'s');
[B1,A1]=butter(N1,Wn1,'s');
[num1,den1]=bilinear(B1,A1,fs1);
[h1,w1]=freqz(num1,den1);
f1=w1/(pi*2)*fs1;
subplot(4,2,5);
plot(f1,(abs(h1)));
title('滤波器频率响应');
xlabel('频率f');
ylabel('幅度/dB');
axis([0 4000 0 2]);
grid;
```

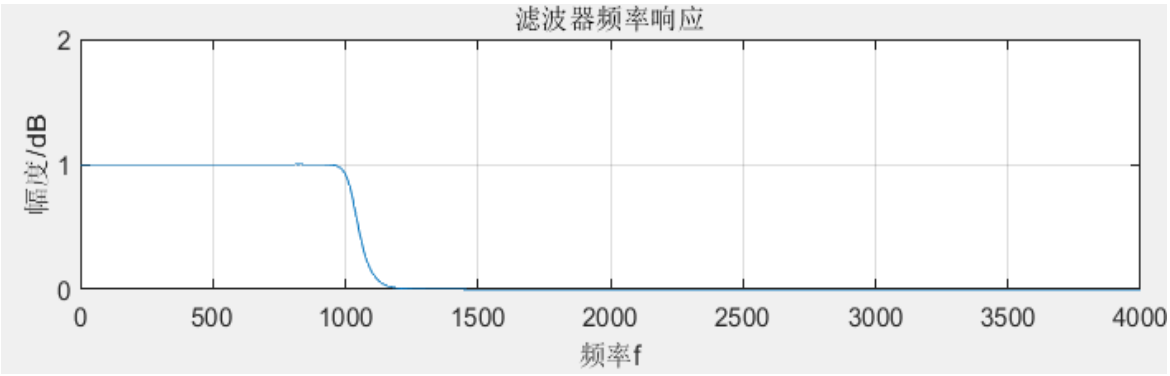


图 3.1.1 IIR 低通滤波器频率响应图

2）高通滤波器 滤波器参数为采样频率$f_s = 8820hz$，通带截止频率$f_c = 1000hz$,阻带截止频率为$f_r = 800hz$,通带波动 $\delta = 1dB$,阻带衰减为 $At = 40dB$。MATLAB 程序代码如下：

```
fc2=1000;fr2=800;
Wc2=2*fs1*tan(2*pi*fc2/(2*fs1));
Wr2=2*fs1*tan(2*pi*fr2/(2*fs1));
[N2,Wn2]=buttord(Wc2,Wr2,ap,rs,'s');
[B2,A2]=butter(N2,Wn2,'high','s');
[num2,den2]=bilinear(B2,A2,fs1);
[h2,w2]=freqz(num2,den2);
f2=w2/(pi*2)*fs1;
subplot(4,2,6);
plot(f2,(abs(h2)));
title('滤波器频率响应');
xlabel('频率f');
ylabel('幅度/dB');
axis([0 4000 0 2]);
grid;
```

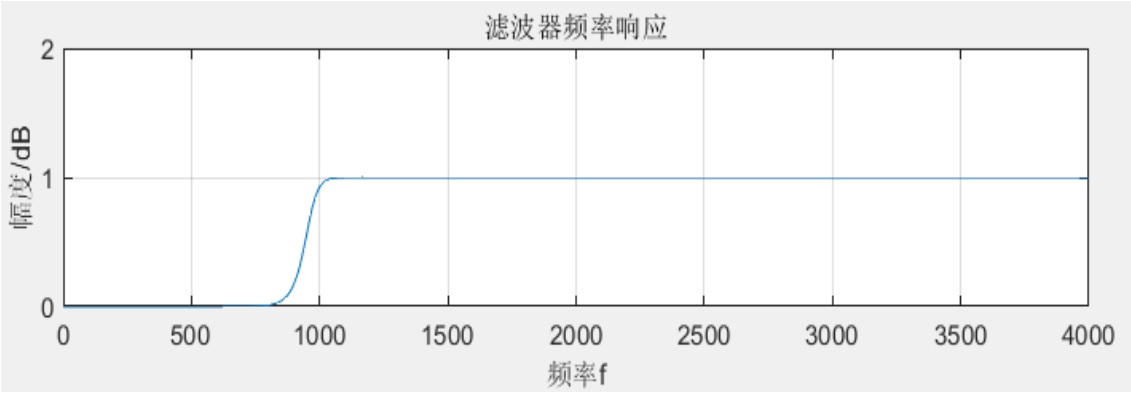


图 3.1.2 IIR 高通滤波器频率响应图

3）带通滤波器 滤波器参数为采样频率$f_s = 8820hz$，通带截止频率$f_{c1} =$

$1800hz, f_{c2} = 2200hz$ 阻带截止频率为$f_r = 2400hz$,通带波动 $\delta = 1dB$,阻带衰减为 $At = 40dB$。MATLAB 程序代码如下：

```
f1=1800;f2=2200;fr2=2400;

   W1=2*fs1*tan(2*pi*f1/(2*fs1));
   W2=2*fs1*tan(2*pi*f2/(2*fs1));
   Wr3=2*fs1*tan(2*pi*fr2/(2*fs1));
   [N3,Wn3]=buttord([W1,W2],[1,Wr3],ap,rs,'s');
   [B3,A3]=butter(N3,Wn3,'s');
   [num3,den3]=bilinear(B3,A3,fs1);
   [h3,w3]=freqz(num3,den3);
   f3=w3/(pi*2)*fs1;
   subplot(4,2,7);
   plot(f3,(abs(h3)));
   title('滤波器频率响应');
   xlabel('频率f');
   ylabel('幅度/dB');
   axis([0 4000 0 2]);
   grid;
```

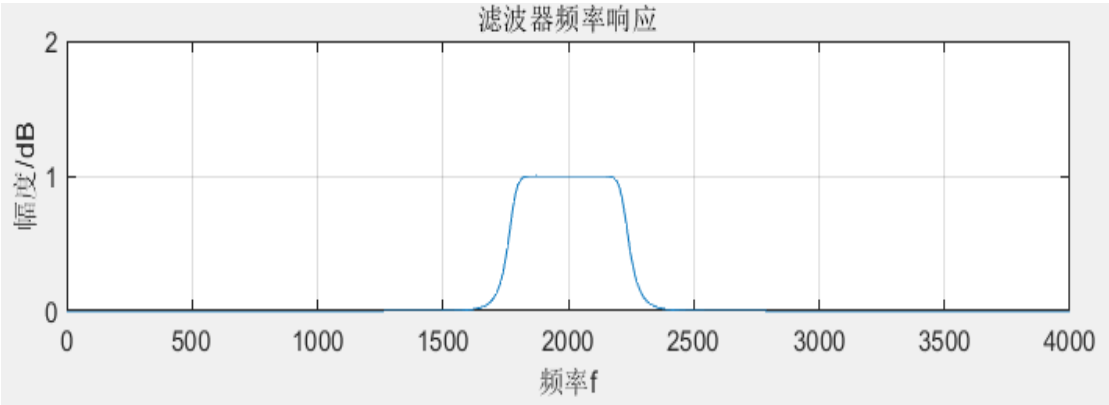


图 3.1.3 IIR 带通滤波器频率响应图

4）带阻滤波器 滤波器参数为采样频率$f_s = 8820hz$，通带截止频率$f_{c1} =$

1800hz, f_{c2} = 2200hz 阻带截止频率为f_r = 1900hz, 通带波动 δ = 1dB,

阻带衰减为 At = 40dB。MATLAB 程序代码如下：。MATLAB 程序代码如下：

```
f3=1800;f4=2200;f5=1900;f6=2100;
W3=2*fs1*tan(2*pi*f3/(2*fs1));
W4=2*fs1*tan(2*pi*f4/(2*fs1));
W5=2*fs1*tan(2*pi*f5/(2*fs1));
W6=2*fs1*tan(2*pi*f6/(2*fs1));
[N4,Wn4]=buttord([W3 W4],[W5 W6],ap,rs,'s');
[B4,A4]=butter(N4,Wn4,'stop','s');
[num4,den4]=bilinear(B4,A4,fs1);
[h4,w4]=freqz(num4,den4);
f4=w4/(pi*2)*fs1;
subplot(4,2,8);
plot(f4,(abs(h4)));
title('滤波器频率响应');
xlabel('频率f');
ylabel('幅度/dB');
axis([0 4000 0 2]);
grid;
```

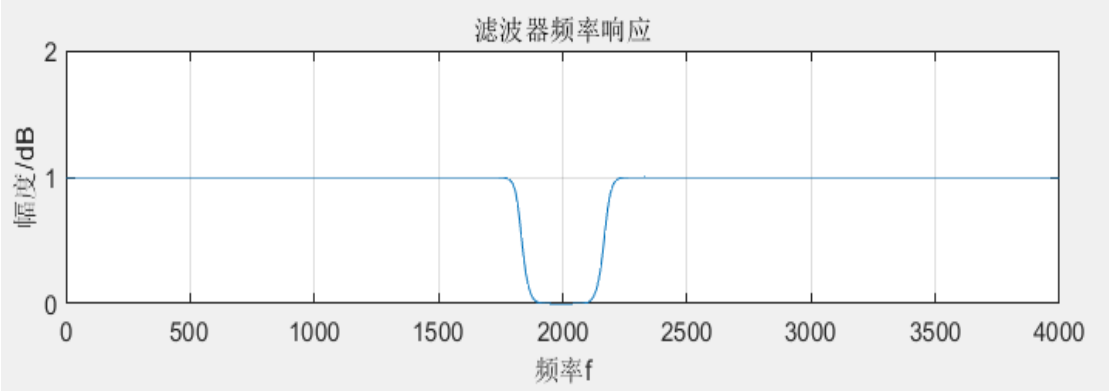


图 3.1.4 IIR 带阻滤波器频率响应图

2、FIR 滤波器的设计

1）低通滤波器 滤波器参数为采样频率$f_s = 8820hz$，通带截止频率$f_c = 1000hz$,阻带截止频率为$f_r = 1200hz$,通带波动 $\delta = 1dB$,阻带衰减为 $At = 40dB$。MATLAB 程序代码如下：

```
fc1=1000;fr1=1200;fs1=8820;
M=145;
f1=fir1(M-1,(fc1*2)/fs1,'low',hanning(M));
F1=fft(f1);
[h1,w1]=freqz(f1,1);
n1=0:M-1;
subplot(2,2,1);
stem(n1,f1);
axis([0 150 -0.1 0.3]);
grid;
xlabel('n1');
ylabel('f1');
title('FIR低通滤波器时域图');
subplot(2,2,2);
plot((fs1/2)*w1/pi,20*log10(abs(h1)));
axis([0 fs1/2 -80 10]);
grid;
xlabel('频率');
ylabel('幅度/dB');
title('FIR低通滤波器频域图');
subplot(2,2,3);
plot(w1/pi,angle(h1));
xlabel('频率\omega');
ylabel('\phi(\omega)');
```

```
title('FIR低通滤波器相位图');
```

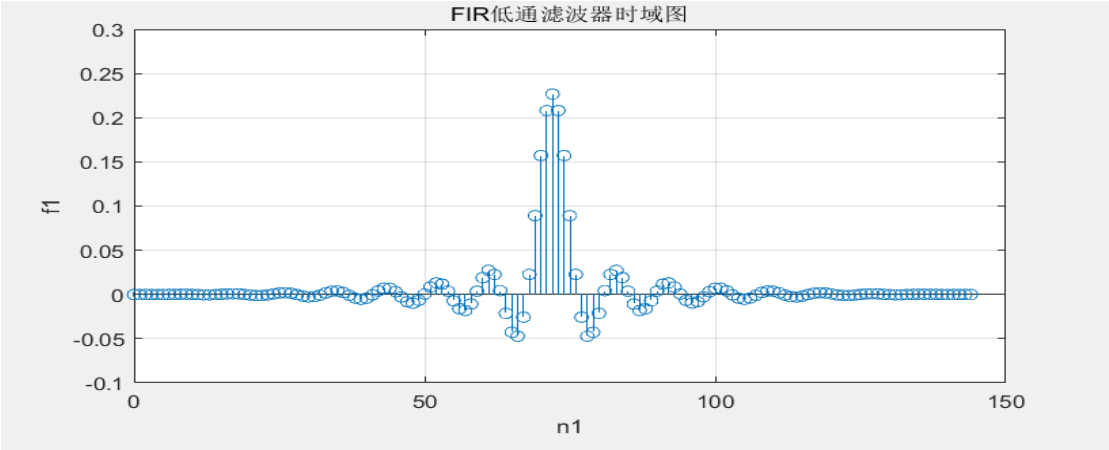


图 3.2.1 FIR 低通滤波器时域图

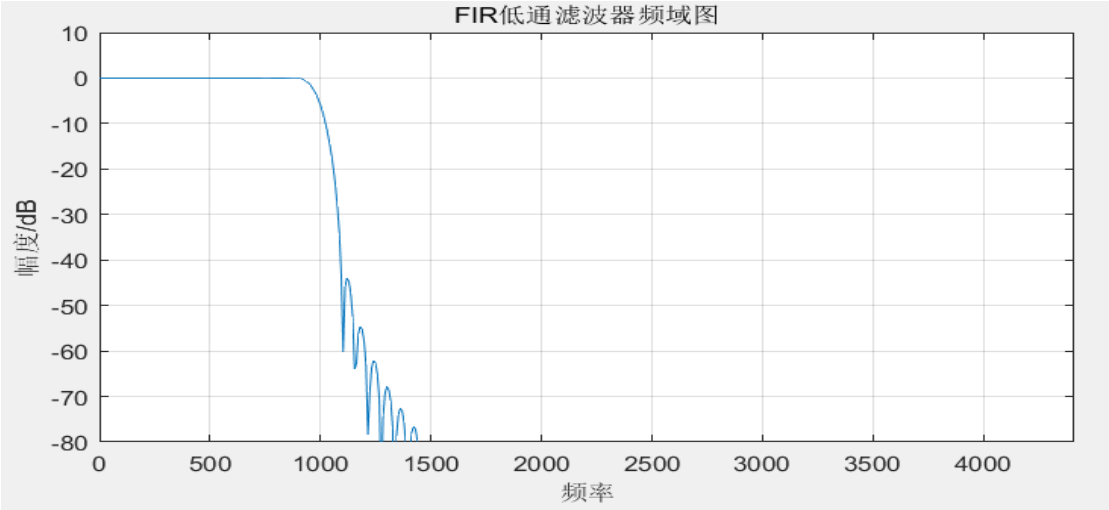


图 3.2.2 FIR 低通滤波器频域图

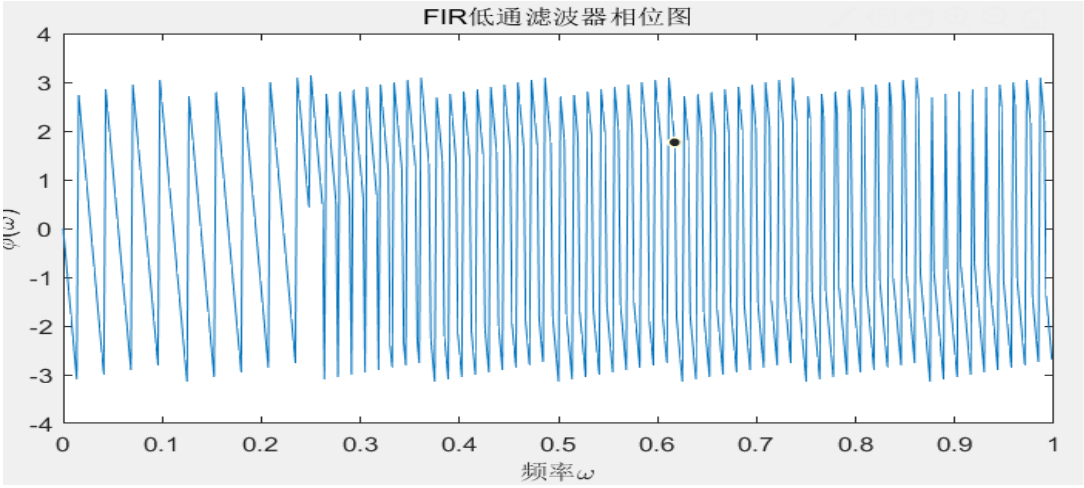


图 3.2.3 FIR 低通滤波器相位图

2）高通滤波器 滤波器参数为采样频率$f_s = 8820hz$，通带截止频率$f_c = 1000hz$,阻带截止频率为$f_r = 1200hz$,通带波动 $\delta = 1dB$,阻带衰减为 $At = 40dB$。MATLAB程序代码如下：

```
fc1=1000;fr1=800;fs1=8820;
M=145;
f1=fir1(M-1,(fc1*2)/fs1 ,'high',hanning(M));
F1=fft(f1);
[h1,w1]=freqz(f1,1);
n1=0:M-1;
subplot(2,2,1);
stem(n1,f1);
axis([0 150 -0.1 0.3]);
grid;
xlabel('n1');
ylabel('f1');
title('FIR高通滤波器时域图');
subplot(2,2,2);
plot((fs1/2)*w1/pi,20*log10(abs(h1)));
axis([0 fs1/2 -120 10]);
grid;
xlabel('频率');
ylabel('幅度/dB');
title('FIR高通滤波器频域图');
subplot(2,2,3);
plot(w1/pi,angle(h1));
xlabel('频率\omega');
ylabel('\phi(\omega)');
title(' FIR高通滤波器相位图');
```

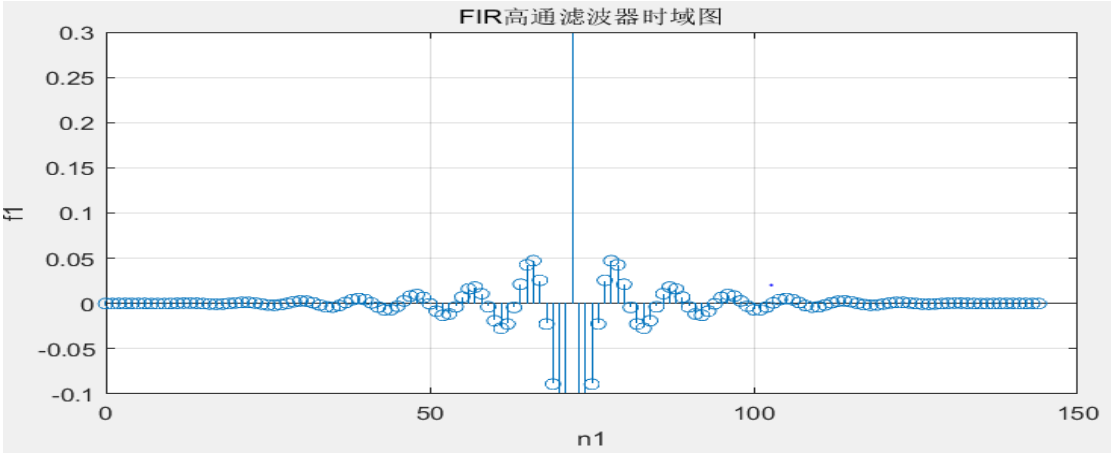


图 3.2.4 FIR 高通滤波器时域图

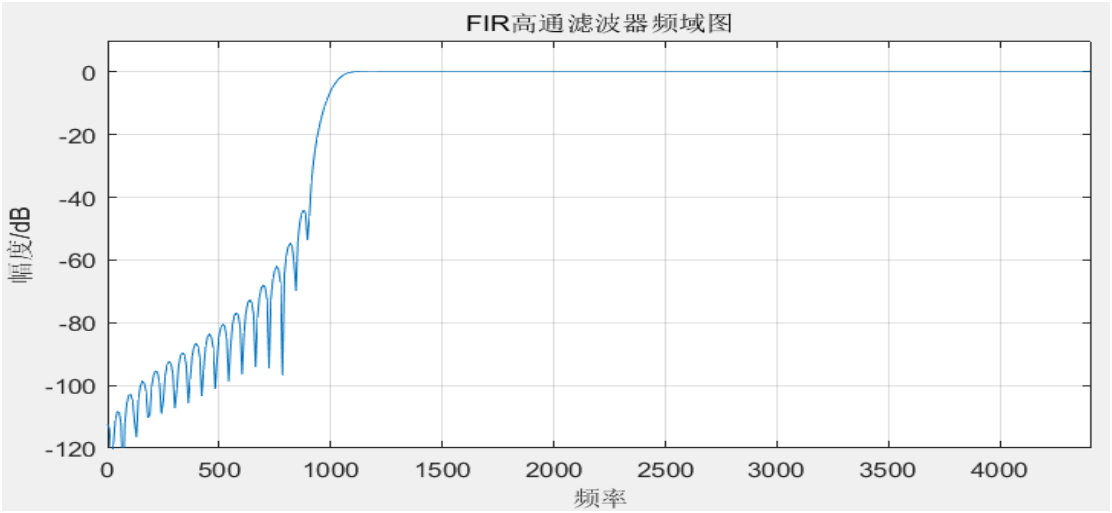


图 3.2.5 FIR 低通滤波器频域图

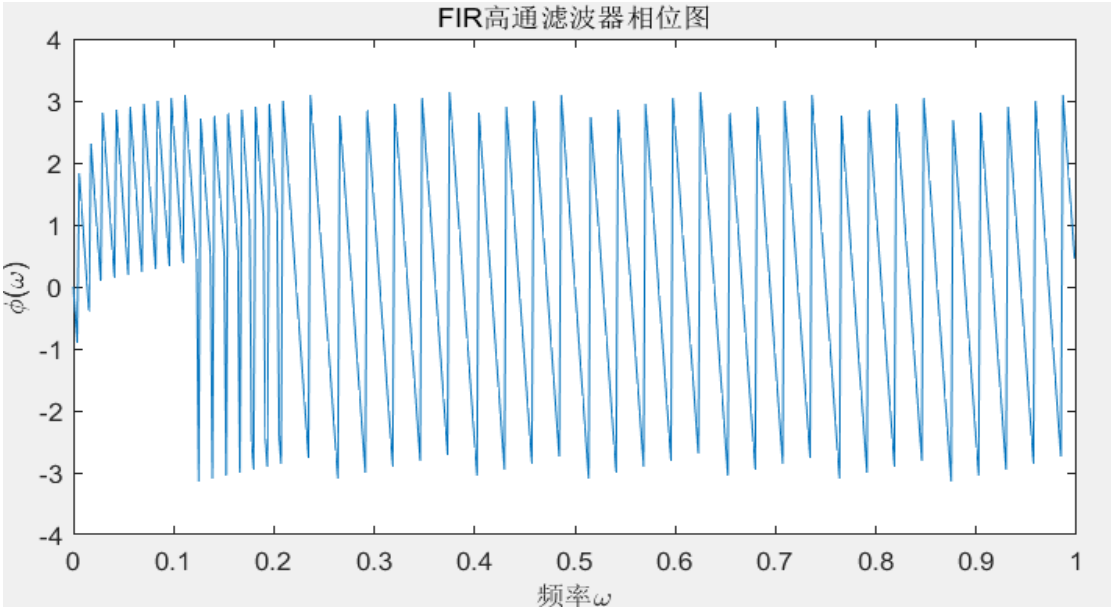


图 3.2.6 FIR 低通滤波器相位图

3）带通滤波器 滤波器参数为采样频率$f_s = 8820hz$，通带截止频率$f_c = 1000hz, f_c = 1000hz$ 阻带截止频率为$f_r = 1500hz$,通带波动 $\delta = 1dB$,

阻带衰减为 $At\ 40dB$。MATLAB 程序代码如下：

```
 fc1=1000;fr1=1500;fs1=8820;
M=145;
f1=fir1(M-1,[(fc1*2)/fs1 (fr1*2)/fs1],'bandpass',hanning(M));
F1=fft(f1);
[h1,w1]=freqz(f1,1);
n1=0:M-1;
subplot(2,2,1);
stem(n1,f1);
axis([0 150 -0.1 0.3]);
grid;
xlabel('n1');
ylabel('f1');
title('FIR带通滤波器时域图');
subplot(2,2,2);
plot((fs1/2)*w1/pi,20*log10(abs(h1)));
axis([0 fs1/2 -120 10]);
grid;
xlabel('频率');
ylabel('幅度/dB');
title('FIR高通滤波器频域图');
subplot(2,2,3);
plot(w1/pi,angle(h1));
xlabel('频率\omega');
ylabel('\phi(\omega)');
title(' FIR高通滤波器相位图');
```

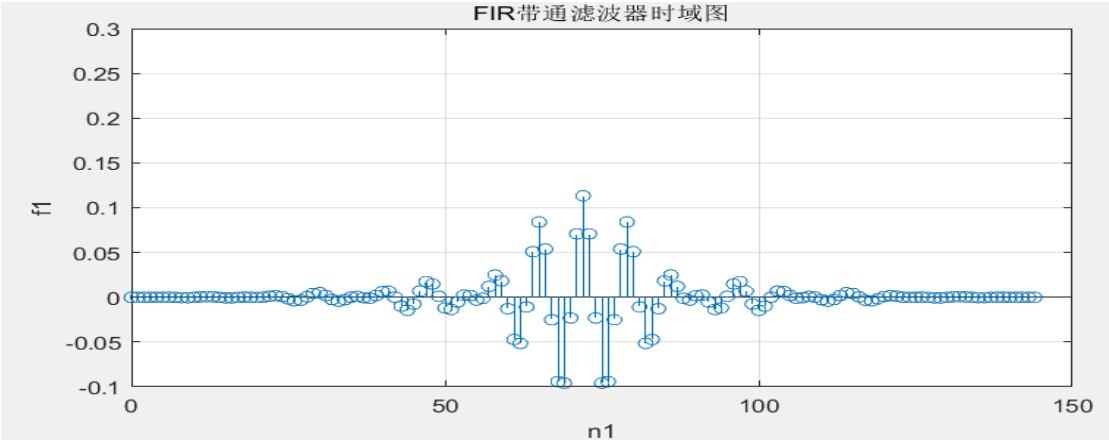


图3.2.7 FIR带通滤波器时域图

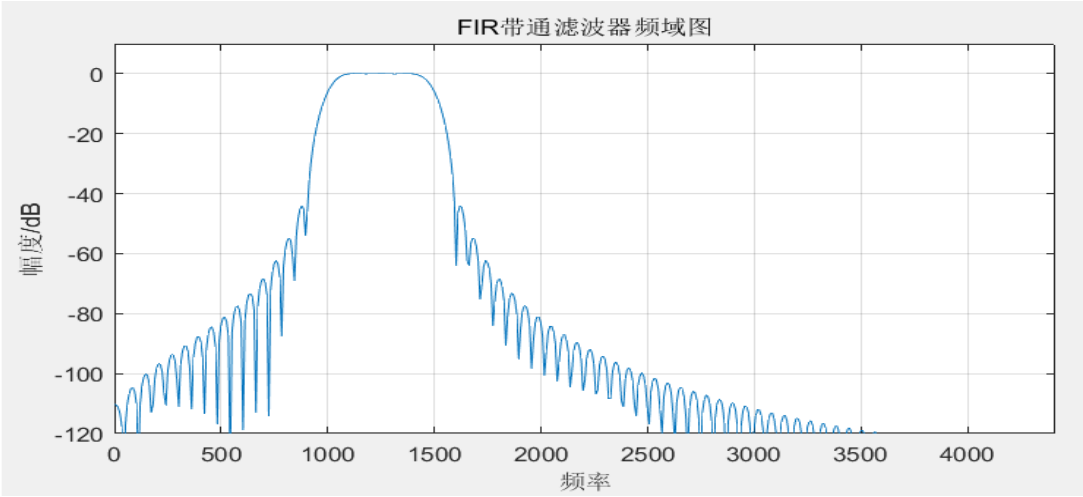


图 3.2.8 FIR 带通滤波器频域图

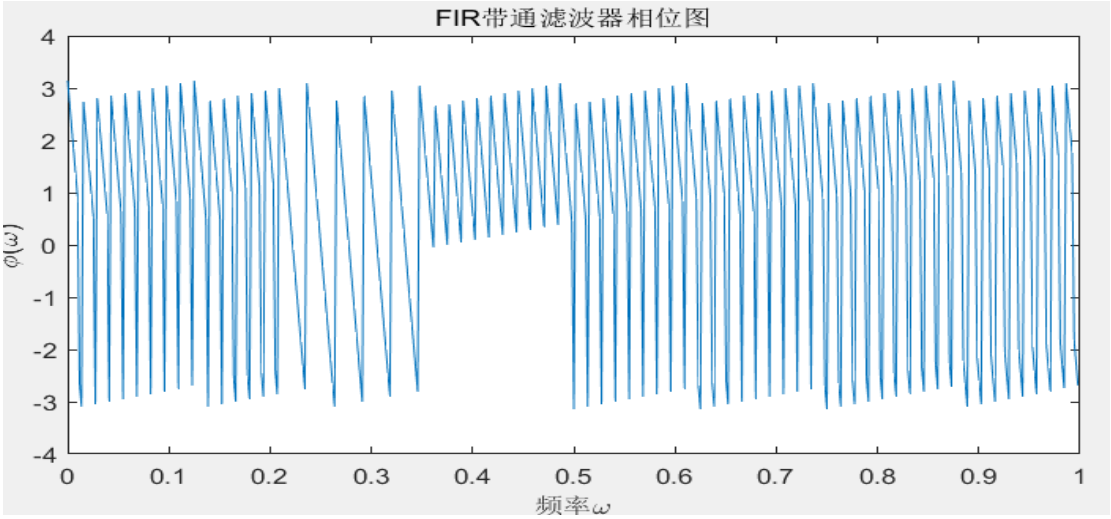


图 3.2.9 FIR 带通滤波器相位图

4）带通滤波器　滤波器参数为采样频率$f_s = 8820hz$，通带截止频率$f_c =$

1000hz, f_c = 1000hz 阻带截止频率为f_r = 1500hz, 通带波动 δ = 1dB, 阻带衰减为 At 40dB。MATLAB 程序代码如下：

```
fc1=1000;fr1=1500;fs1=8820;
M=145;
f1=fir1(M-1,[(fc1*2)/fs1 (fr1*2)/fs1],'stop',hanning(M));
F1=fft(f1);
[h1,w1]=freqz(f1,1);
n1=0:M-1;
subplot(2,2,1);
stem(n1,f1);
axis([0 150 -0.1 0.3]);
grid;
xlabel('n1');
ylabel('f1');
title('FIR带阻滤波器时域图');
subplot(2,2,2);
plot((fs1/2)*w1/pi,20*log10(abs(h1)));
axis([0 fs1/2 -120 10]);
grid;
 xlabel('频率');
ylabel('幅度/dB');
title('FIR带阻滤波器频域图');
subplot(2,2,3);
plot(w1/pi,angle(h1));
xlabel('频率\omega');
ylabel('\phi(\omega)');
title(' FIR带阻滤波器相位图');
```

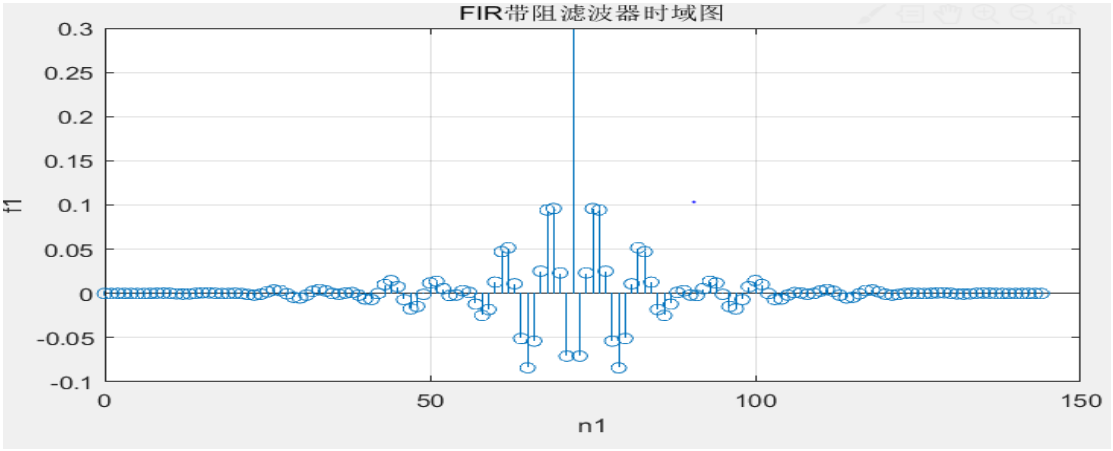


图 3.2.10 FIR 带通滤波器时域图

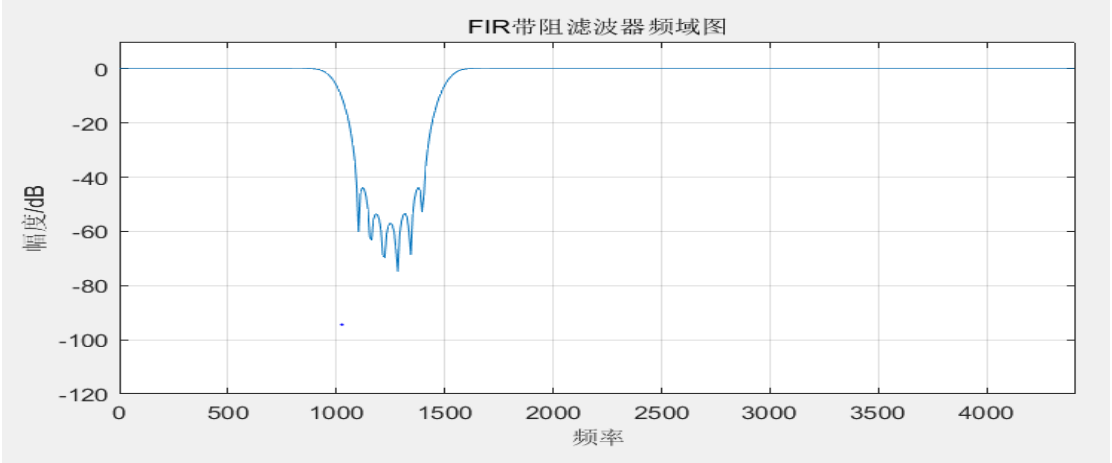


图 3.2.11 FIR 带通滤波器频域图

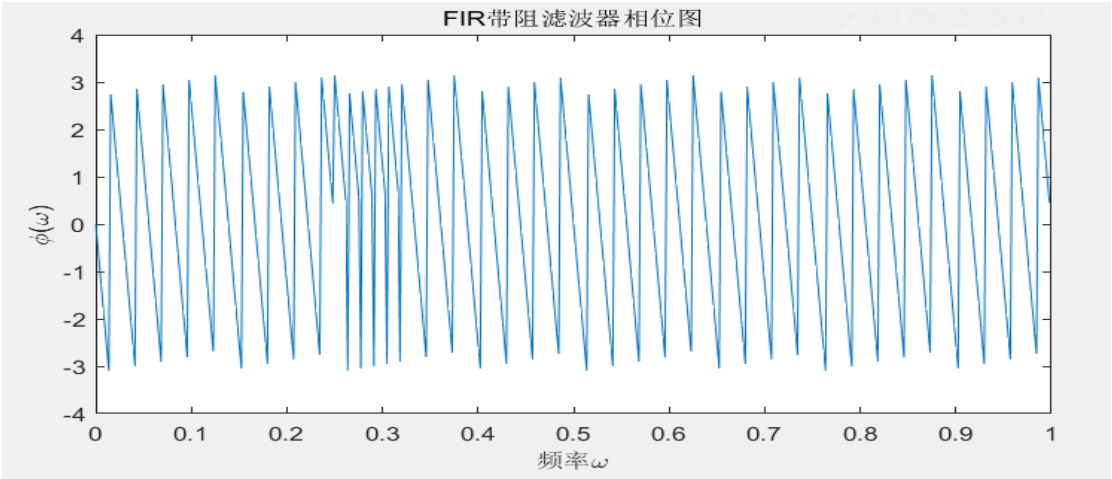


图 3.2.12 FIR 带通滤波器相位图

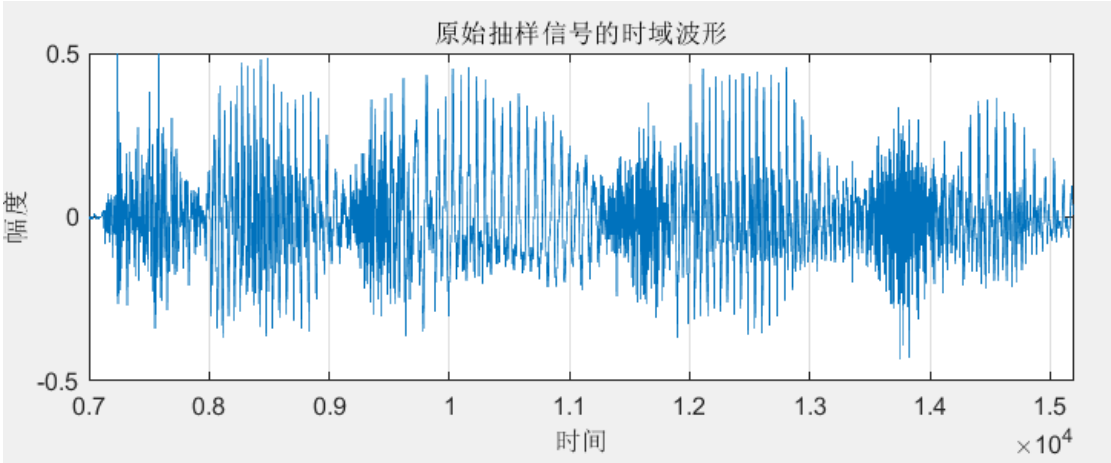
3、MATLAB 语音信号处理

使用已经录制好的 WAV 格式音频文件，通过 MATLAB 内集成的 audioread 函数读取数据，作为滤波器的处理数据，通过滤波处理，比较滤波前后语音信号的区别。MATLAB 程序如下：

```
[z1s,fs]=audioread('D:\xia.wav');
sound(z1s,fs);
N=size(z1s);
for i=1:(N)/5
    z2(i)=z1s(5*i);
end
   pause(4);
    sound(z2,fs/5);
   fc=1000;fr=1200;fs1=fs/5;rs=40;ap=1;
   Wc=2*fs1*tan(2*pi*fc/(2*fs1));
   Wr=2*fs1*tan(2*pi*fr/(2*fs1));
   [N1,Wn]=buttord(Wc,Wr,ap,rs,'s');
   [B,A]=butter(N1,Wn,'s');
   [B2,A2]=bilinear(B,A,fs1);
   [h,w]=freqz(B2,A2);
   fc1=1000;fr1=1200;fs1=fs/5;
   M=145;
   f1=fir1(M-1,(fc1*2)/fs1,'low',hanning(M));
   F1=fft(f1);
   [h1,w1]=freqz(f1,1);
   x1=filter(f1,1,z2);
   x2=filter(B2,A2,z2);
   n2=7000:15191;
   y1=z2(7000:15191);
```

```
y2=x1(7000:15191);
y3=x2(7000:15191);
Y1=fft(y1);
Y2=fft(y2);
Y3=fft(y3);
subplot(3,2,1);
plot(n2,y1);
title('原始信号的时域波形');
xlabel( '时间' );
ylabel('幅度');
axis([7000 15200 -0.5 0.5]);
grid;
subplot(3,2,2);
plot(abs(Y1));
title('原始信号的频域图');
xlabel('频率');
ylabel('幅度');
axis([0 8000 0 150]);
grid;
subplot(3,2,3);
plot(n2,y2);
title('FIR滤波后的时域图');
xlabel('时间');
ylabel('幅度');
axis([7000 15200 -0.5 0.5]);
grid;
subplot(3,2,4);
plot(abs(Y2));
title('FIR滤波后频域图');
```

```
xlabel('频率');
ylabel('幅度');
axis([0 8000 0 150]);
grid;
subplot(3,2,5);
plot(n2,y3);
title('IIR滤波后时域图');
xlabel('时间');
ylabel('幅度');
axis([7000 15200 -0.5 0.5]);
grid;
subplot(3,2,6);
plot(abs(Y3));
title('IIR滤波后频域图');
xlabel('频率');
ylabel('幅度');
axis([0 8000 0 150]);
grid;
```



3.3.1 原始信号时域图

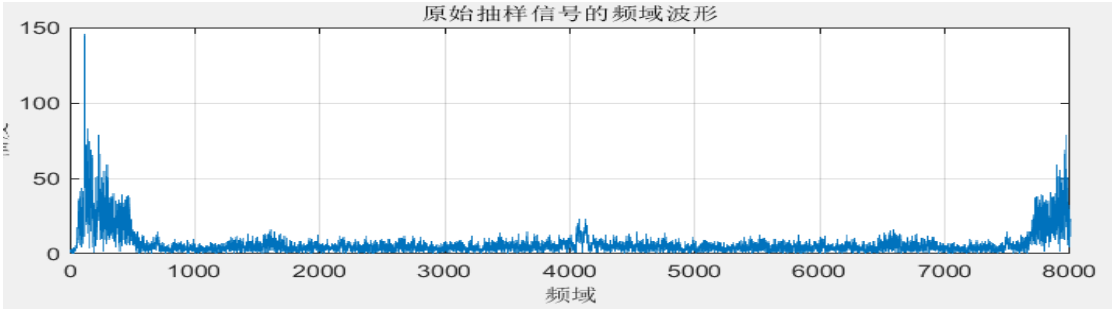


图 3.3.2 原始信号频域图

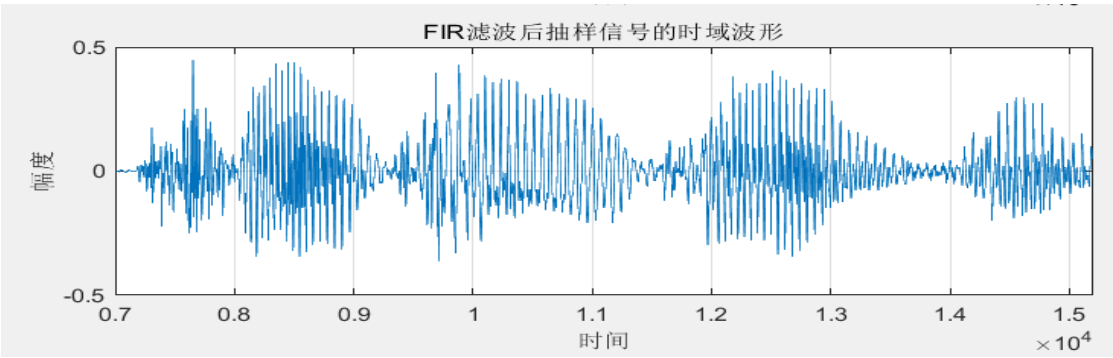


图3.3.3 FIR滤波后时域图

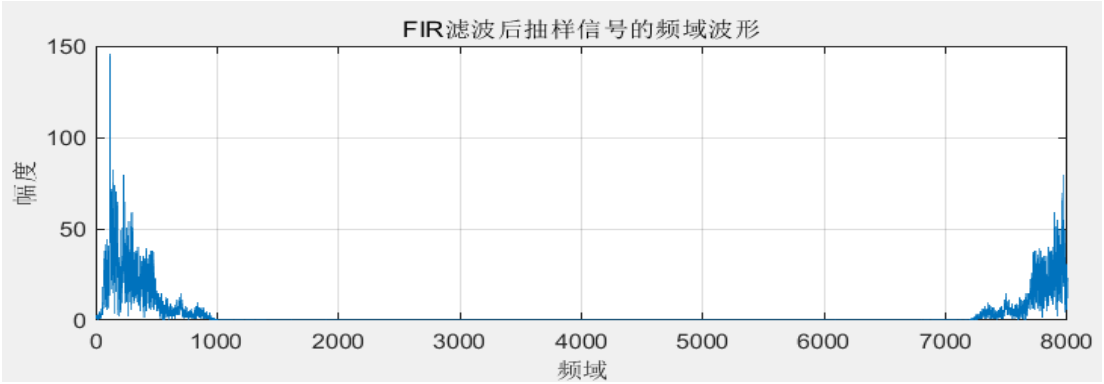


图3.3.4 FIR滤波后频域图

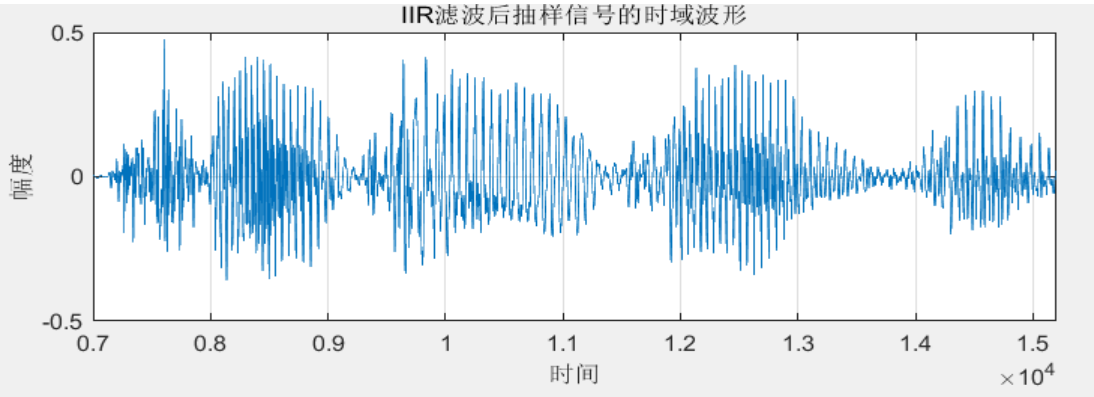


图3.3.5 IIR滤波后时域图

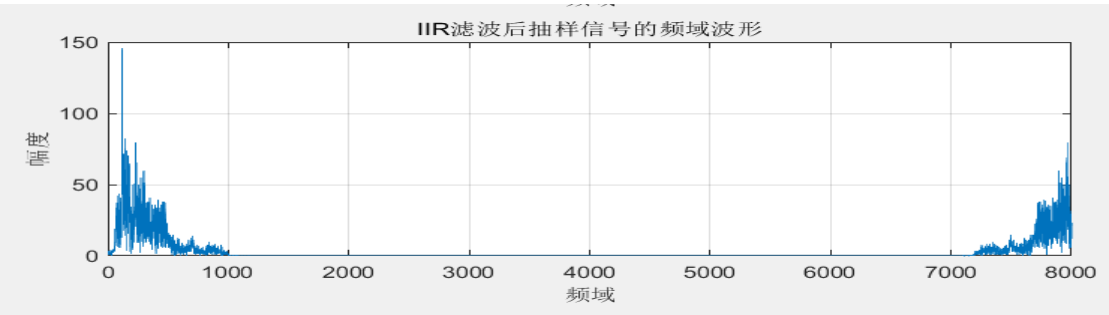


图3.3.6 IIR滤波后频域图

通过以上的滤波处理，由MATLAB内集成的播放函数sound(x,f)播放，可以发现，经过滤波处理之后的语音变的比较低沉，语音比较尖锐的地方变得缓慢，和滤波前存在着一定的区别。同时也可以从语音信号滤波前后的频域波形中看出，语音信号的高频部分被滤波器滤掉，只剩下通带以内的频率部分，所以语音信号变得比较低沉，缓慢。

4、MATLAB GUI 界面设计

通过前面的几步准备工作，已经设计好了不同类型的滤波器，现在将不同的滤波器以及滤波后的语音播放集成到同一个界面，通过输入输出参数，即可得到结果。故通过 MATLAB GUI 功能将他们集成到同一界面。

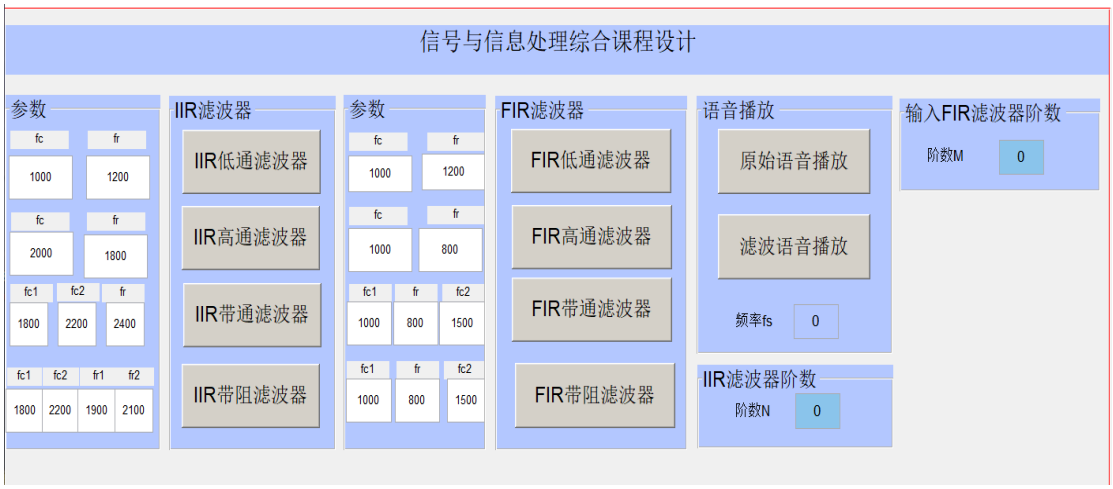


图 3.4.1 GUI 界面图

本界面实现的功能为：

1、可以通过输入滤波器参数，输出具体指标的滤波器频域图；

2、播放滤波前后的音频文件

3、显示所设计的滤波器的阶数。

可以看出，本界面的设计具有良好的交互性，可以设计出不同指标的滤波器。

部分代码如下：

IIR 低通滤波器按钮回调函数

```
function pushbutton28_Callback(hObject, eventdata, handles)
    [z1s,fs]=audioread('D:\xia.wav');
    N=size(z1s);
    fc=str2double(get(handles.edit24,'string'));
     //可以通过界面输入指标参数
    fr=str2double(get(handles.edit25,'string'));
    fs1=fs/5;rs=40;ap=1;
    Wc=2*fs1*tan(2*pi*fc/(2*fs1));
    Wr=2*fs1*tan(2*pi*fr/(2*fs1));
    [N1,Wn]=buttord(Wc,Wr,ap,rs,'s');
    set(handles.edit42,'string',num2str(N1));
     //可以显示滤波器具体参数
    [B,A]=butter(N1,Wn,'s');
    [B2,A2]=bilinear(B,A,fs1);
    [h,w]=freqz(B2,A2);
    f=w/(pi*2)*fs1;
    figure(1);
    plot(f,(abs(h)));
    title('滤波器频率响应');
    xlabel('频率f');
    ylabel('幅度/dB');
    axis([0 4000 0 2]);
    grid;
```

IIR 低通滤波器按钮回调函数

```
function pushbutton22_Callback(hObject, eventdata, handles)
    [z1s,fs]=audioread('D:\xia.wav');
    N=size(z1s);
     fc1=str2double(get(handles.edit32,'string'));
     fr1=str2double(get(handles.edit45,'string'));fs1=fs/5;
    wc1=2*pi*fc1/fs1;wr1=2*pi*fr1/fs1;
    M=floor(6.6*pi/(wr1-wc1));
    set(handles.edit43,'string',num2str(M));
    f1=fir1(M-1,(fc1*2)/fs1,'low',hanning(M));
    F1=fft(f1);
    [h1,w1]=freqz(f1,1);
    figure(2);
    plot((fs1/2)*w1/pi,20*log10(abs(h1)));
    axis([0 fs1/2 -120 10]);
    grid;
    xlabel('频率');
    ylabel('幅度/dB');
    title('FIR滤波器频率响应');
```

滤波语音播放函数按钮

```
function pushbutton34_Callback(hObject, eventdata, handles)
[z1s,fs]=audioread('D:\xia.wav');
N=size(z1s);
for i=1:(N)/5
     z2(i)=z1s(5*i);
end
    set(handles.edit41,'string',num2str(fs));
    fc=1000;fr=1200;fs1=fs/5;rs=40;ap=1;
```

```
Wc=2*fs1*tan(2*pi*fc/(2*fs1));
Wr=2*fs1*tan(2*pi*fr/(2*fs1));
[N1,Wn]=buttord(Wc,Wr,ap,rs,'s');
[B,A]=butter(N1,Wn,'s');
[B2,A2]=bilinear(B,A,fs1);
 x=filter(B2,A2,z2);
 sound(x,fs1);
```

5、基于 VC 的 WAV 格式音频录音和播放器

使用 MFC 封装的类设计一录音和播放器，部分代码如下：

1、保存文件按钮

```
void CRecordDlg::OnFile()
{
    // TODO: Add your control notification handler code here
    CFileDialog dlg(FALSE,
                    "wav",
                    "Noname",
                    OFN_HIDEREADONLY | OFN_OVERWRITEPROMPT,
                    "Wave File(*.wav)|*.wav||");
    if(dlg.DoModal() == IDOK)
    {
        m_strFile=dlg.GetPathName();
        UpdateData(FALSE);
    }
}
```

通过点击此按钮可以为录制的音频文件选择一保存的目录。

2、开始录音按钮

```
void CRecordDlg::OnRecord()
{
```

```
    UpdateData(TRUE);//将控件的值赋值给成员变量
    if(!recording)//recording=0时允许录音，等待录音
    {
        PrepareFormat();
        CreateWaveFile((LPSTR)(LPCSTR)m_strFile,&params);
        RecordStart();//开始录音  //录音正常结束时recording=TRUE，否则
       recording=FALSE
        m_lTime=0;
        UpdateData(FALSE);//将成员变量的值赋值给控件显示
        if(recording)//录音正常结束时recording=TRUE
            SetDlgItemText(IDC_RECORDBT,"Stop(&T)");//将开始录音按钮显
示为停止录音
    }
    else //以前RecordStart()录音正常结束时recording=TRUE，已经录音，此时录
音按钮为停止录音
    {
        RecordEnd();//结束录音的收尾工作，断开麦克风，释放内存
        CloseWaveFile(&params);
        if(!recording)
            SetDlgItemText(IDC_RECORDBT,"RecordStart(&R)");
    }
}
```

通过点击此按钮，开始录制音频文件，并且保存到(1)中的目录。

3、语音播放按钮

```
void CRecordDlg::OnPlay()
{
    LPSTR pData;
    LONG len;
```

```
    UpdateData(TRUE);
    PlayStart(m_strFile,&wfm,&pData,&len);
}
```

通过点击此按钮，播放(2)中已经录制好的音频文件。

4、PlayStart 播放准备函数

```
BOOL PlayStart(LPCSTR file,WAVEFORMATEX *pwf,LPSTR *pData,LONG *pLen)
{
    HMMIO hmmio=NULL;
    MMCKINFO ckiRiff,cki;
    MMRESULT result;
    LONG len;
    if(hwo != NULL) //回放设备已打开
    {
        result=waveOutReset(hwo);//停止放音
        if(result != MMSYSERR_NOERROR)
            goto END_ERROR;
        result=waveOutClose(hwo);//关闭打开的回放设备
        if(result != MMSYSERR_NOERROR)
            goto END_ERROR;
        hwo=NULL;//关闭打开的回放设备之后让设备回到空闲备用
    }
    hmmio=mmioOpen((LPSTR)file,NULL,MMIO_READ | MMIO_ALLOCBUF);
     //为输入输出打开一个多媒体文件 file
    if(hmmio == NULL)
        return FALSE;
    ckiRiff.fccType=mmioFOURCC('W','A','V','E');
    result=mmioDescend(hmmio,&ckiRiff,NULL,MMIO_FINDRIFF);
    if(result != MMSYSERR_NOERROR)
        goto END_ERROR;
```

```
    cki.ckid=mmioFOURCC('f','m','t',' ');
    result=mmioDescend(hmmio,&cki,&ckiRiff,MMIO_FINDCHUNK);
    if(result != MMSYSERR_NOERROR)
        goto END_ERROR;
    len=mmioRead(hmmio,(LPSTR)pwf,sizeof(WAVEFORMATEX));
//hmmio：文件句柄,被读取的文件的句柄。pwf：指向一个缓冲区,包含/存放从文件读取的数据。sizeof(WAVEFORMATEX)要从文件读取的字节数
    if(len == -1)
        goto END_ERROR;
    result=mmioAscend(hmmio,&cki,0);
    if(result != MMSYSERR_NOERROR)
        goto END_ERROR;
    cki.ckid=mmioFOURCC('d','a','t','a');
    result=mmioDescend(hmmio,&cki,&ckiRiff,MMIO_FINDCHUNK);
    if(result != MMSYSERR_NOERROR)
        goto END_ERROR;
    *pLen=cki.cksize;
    *pData=(LPSTR)GlobalAlloc(GMEM_FIXED | GMEM_SHARE,*pLen);
 //分配内存地址数*pLen 所指向的大小的内存，GMEM_FIXED 分配固定的内存,返回值是一个指针
//若函数调用成功,则返回一个新分配的内存对象的句柄,如果堆内没有足够的空间满足请求,函数将返回 NULL
    if(*pData == NULL)
        goto END_ERROR;
    len=mmioRead(hmmio,(LPSTR)*pData,*pLen);
//!!从打开的文件 hmmio 中读取长度为*pLen 的数据到 pData 所指向的地址中
    if(len == -1)
        goto END_ERROR;
```

```
    result=waveOutOpen(&hwo,/
                        WAVE_MAPPER,
                        pwf,
                        NULL,
                        (DWORD)AfxGetInstanceHandle(),
                        CALLBACK_NULL
                        );
//这个函数打开一个给定的波形音频输出装置来进行回放
    if(result != MMSYSERR_NOERROR)
        goto END_ERROR;
    memset(&wh2,0,sizeof(WAVEHDR));   //WAVEHDR wh2; wh2 为全局变量
     /*typedef struct {
     LPSTR    lpData; //波形缓冲数据(传入首地址)
     DWORD    dwBufferLength; //缓冲区长度
     DWORD    dwBytesRecorded; //指明录音时缓冲区容量
     DWORD    dwUser; //用户数据
     DWORD    dwFlags; //提供缓冲区标示
     DWORD    dwLoops; //循环次数
     struct wavehdr_tag * lpNext; //预留,NULL
     DWORD    reserved; //预留,0
        } WAVEHDR;*/
    wh2.lpData=*pData;
    wh2.dwBufferLength=*pLen;
    result=waveOutPrepareHeader(hwo,&wh2,sizeof(WAVEHDR));
//准备一个波形数据块用于播放
    if(result != MMSYSERR_NOERROR)
        goto END_ERROR;;
waveOutWrite(hwo,&wh2,sizeof(WAVEHDR));
//播放语音,mmsystem  函数,将一个数据块发送到一个指定的波形音频输出装置。
```

在播放语音的同时运行后面的程序

//hwo:HWAVEOUT 波形音频输出装置的柄（handle）,&wh2: 一个指向包含有数据块信息的 WAVEHDR 结构的指针

//第三个参数就是 WAVEHDR 结构的大小（用 sizeof (WAVEHDR) 就可以了）

```
    return TRUE;
END_ERROR:
    if(*pData != NULL)
    {
        GlobalFree(*pData);
        *pData=NULL;
        *pLen=0;
    }
    if(hmmio != NULL)
        mmioClose(hmmio,0);
    if(hwo != NULL)
        waveOutClose(hwo);
}
```

此函数为 WAV 格式音频文件数据提取以及准备播放函数。通过以上 4 个函数以及按钮，按图 3.5.1 选择好对应的格式，即可录制一 WAV 格式的音频文件并且播放。

图 3.5.1 VC 录音和播放器界面

6、基于 C++的 IIR 和 FIR 滤波器设计

使用 C++语言，设计不同类型的滤波器，再在(5)的基础上对录制的 WAV 格式音频文件进行滤波处理并播放。部分代码如下：

1、IIR 滤波器设计

1）IIR 滤波器类定义

```
class IIR_I
{
    private:
    double *m_pNum;
    double *m_pDen;
    double *m_px;
     double *m_py;
     int m_num_order;
     int m_den_order;
     public:
    void setPara(double num[], int num_order, double den[], int den_order);
    double filter(double data);
};
```

2）滤波器参数设置函数，通过此函数，可以设置对应滤波器的系数以及阶数。

```
void IIR_I::setPara(double num[], int num_order, double den[], int den_order)
{
    delete[] m_pNum;
    delete[] m_pDen;
    delete[] m_px;
    delete[] m_py;
    m_pNum = new double[num_order + 1];
    m_pDen = new double[den_order + 1];
    m_num_order = num_order;
    m_den_order = den_order;
```

```
    m_px = new double[num_order + 1];
    m_py = new double[den_order + 1];
    for(int i = 0; i <= m_num_order; i++)
    {
        m_pNum[i] = num[i];
        m_px[i] = 0.0;
    }
    for( i = 0; i <= m_den_order; i++)
    {
        m_pDen[i] = den[i];
        m_py[i] = 0.0;
    }
}
```

3）滤波器处理函数，在(2)中设置好参数后，通过此函数传入 WAV 格式音频文件数据，返回一 double 类型的数据，完成数据的滤波处理。

```
double IIR_I::filter(double data)
{
    m_py[0] = 0.0; // 存放滤波后的结果
    m_px[0] = data;
    for(int i = 0; i <= m_num_order; i++)
    {
      m_py[0] = m_py[0] + m_pNum[i] * m_px[i];
    }
  for( i = 1; i <= m_den_order; i++)
    {
       m_py[0] = m_py[0] - m_pDen[i] * m_py[i];
    }
    for( i = m_num_order; i >= 1; i--)
    {
```

```
        m_px[i] = m_px[i-1];
        }
        for(i = m_den_order; i >= 1; i--)
        {
              m_py[i] = m_py[i-1];
        }
        return m_py[0];
}
```

2、FIR 滤波器设计

1) IIR 滤波器类的定义

```
class CFilter_FIR
  {
       public:
      CFilter_FIR();
      virtual ~CFilter_FIR();
       private:
      double *m_pB;
      double *m_pdata_buf;
      int m_nB_order;
      int m_nFirIndex;
      void setPara(double B[], int nB_order);
      double filter(double data);
  };
```

2) 滤波器参数设置函数，通过此函数，可以设置对应滤波器的系数以及阶数。

```
void CFilter_FIR::setPara(double B[], int nB_order)
  {
       m_nB_order = nB_order;
       if (m_pB)
       {
```

```
    delete m_pB;
    }
     if (m_pdata_buf)
     {
          delete m_pdata_buf;
    }
     m_pB = new double[nB_order ];
     m_pdata_buf = new double[nB_order ];
     for(int i = 0; i < nB_order; i++)
     {
       m_pB[i] = B[i];
        m_pdata_buf[i] = 0.0;
     }
}
```

3）滤波器处理函数，在(2)中设置好参数后，通过此函数传入 WAV 格式音频文件

```
double CFilter_FIR::filter(double data)
 {
     int k;
     double fOut = 0.0;
     int i = 0;
     m_pdata_buf[m_nFirIndex%m_nB_order] = data;
     fOut = 0.0;
     for (i = 0; i < m_nB_order; ++i)
     {
          fOut += m_pdata_buf[(m_nFirIndex +i+1)%m_nB_order]*m_pB[i];
    }
     m_nFirIndex++;
     return fOut;
 }
```

7、基于 VC 的语音信号处理

通过上面几步已经设计好了语音录制和播放器以及不同类型的滤波器，现在需要将设计的滤波器加入到语音播放器中进行语音信号处理，故只需要在(5)的基础之上进行修改。部分代码如下：

1、IIR 滤波

```
doubleb[28]，double a[28]//此处为存放滤波器的系数，由 MATLAB 提供
IIR_I filter;
char *shuru_x,*shuchu_y; //定义输入输出数据
int tt;
long int i;
/***************以下为滤波处理算法***************/
filter.setPara(b, 28, a, 28); //滤波器参数设置
for(i=50;i<*pLen;i++)
{
 tt=int(filter.filter(byte(*(shuru_x+i)))); //输入数据到滤波器
 if(tt>255)
*(shuchu_y+i)=255;
else if(tt<0) *(shuchu_y+i)=0;
else *(shuchu_y+i)=byte(tt);  //存放数据到输出数据中
}
/***************以下为滤波处理算法***************/
```

2、FIR 滤波

```
double B[145]; //存放滤波器系数
CFilter_FIR filter;
 char *shuru_x,*shuchu_y;
 int tt;
long int i;
/***************以下为滤波处理算法***************/
filter.setPara(B,145);  //滤波器参数设置
```

```
for(i=50;i<*pLen;i++)
{
  tt=int(filter.filter(byte(*(shuru_x+i)))); //输入数据到滤波器
  if(tt>255)
 *(shuchu_y+i)=255;
  else if(tt<0) *(shuchu_y+i)=0;
  else *(shuchu_y+i)=byte(tt);   //滤波后数据输出
}
/***************以下为滤波处理算法***************/
```

将以上滤波函数加入已经设置好的播放器以后，即可将录制好的议信号进行滤波处理。设计好的 MFC 界面如下：

图 3．7．1　基于 VC 的语音信号处理 MFC 界面

３、部分代码

1)打开文件按钮

```
void CRecordDlg::OnFOBt()
{
    CFileDialog dlg(TRUE,
                    "wav",
                    "Noname",
```

```
                    OFN_HIDEREADONLY | OFN_OVERWRITEPROMPT,
                    "Wave File(*.wav)|*.wav||");
    if(dlg.DoModal() == IDOK)
    {
        m_strFile1=dlg.GetPathName();
        UpdateData(FALSE);
    }
}
```

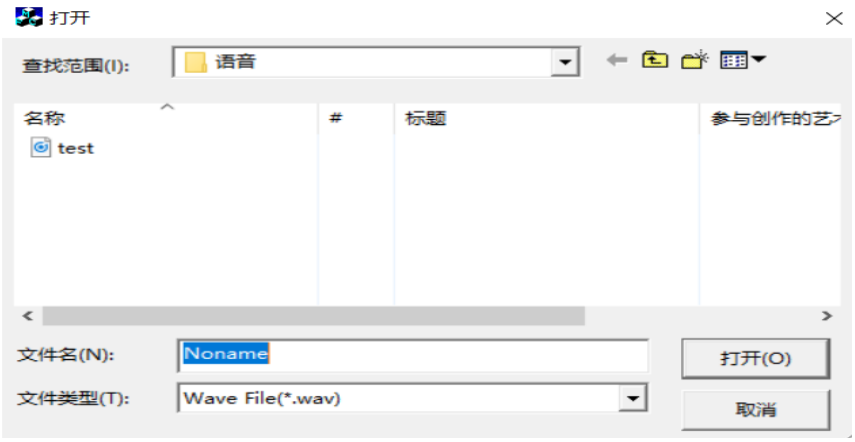


图 3.7.2 打开文件按钮结果图

通过此函数可以打开想要进行处理的语音信号。在完成以上的综合设计之后，通过运行，出现 MFC 界面，首先点击打开文件，打开要进行处理的 WAV 格式音频文件，再点击 IIR 滤波按钮，首先播放的是原始语音信号，播完完之后，等待几秒，播放的为进行滤波后的语音信号。通过对比播放前后的语音信号，滤波后的语音信号变得比较低沉，缓慢，既对应高频部分被滤掉。

四、 课程设计收获与体会

通过本次综合课程设计，我学习到了很多新的知识，真正的将所学习到的知识应用到了实际的操作当中。首先是滤波器的设计，通过数字信号处理这门课程，我们学习到了许多关于滤波器的理论知识以及如何去设计一数字滤波器，当都始终停留在纸面上，很少进行实际操作，在本次课程实际中，使用 MATLAB 软件进行了各种类型滤波器的设计，通过不断反反复复的设计，从当初的不会到逐渐的慢慢熟悉使用 MATLAB，将所学习到的理论知识应用到了实际的操作中来。然后就是 MATLAB 中 GUI 软件界面的设计，在老师细心教学以及自己的不断的摸索下，开始慢慢的熟悉各种操作，最终设计出了一交互性比较好的 GUI 软件界面，同时也收获到了许许多多的书本上没有的知识。其次就是 VC 下的录音和播放器，由于对 C++不是特别熟悉，故在设计时存在很多的问题，但我相信只要不断的坚持，一定会成功的。一开始由老师细心的给我们讲解 C++的基础知识点，然后由我们慢慢的吸收消化，查阅各种学习资料，以及老师一步步的对我们的教学，在老师的一点点帮助下，终于在最后设计出了整个 VC 语音信号处理系统。在这次课程设计中利用了数字信号处理、语音信号处理、MATLAB 程序设计、VC 软件设计等多门课程知识与技能，进一步的增加了我自己对所学专业的了解程度，提高了我的实际动手操作能力，增加了我许许多多的技能，为以后的专业课程以及今后的工作打下了坚实的基础。