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**

|  |  |  |
| --- | --- | --- |
| **内**  **容**  **及**  **任**  **务** | 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。 | |
| **进**  **度**  **安**  **排** | 起止日期 | 工作内容 |
| 2019年12月16日**～**12月20日 | 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。 |
| 2019年12月23日**～**12月27日 | 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。 |
| 2019年12月30日～2020年1月3日 | 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。 |
| **主**  **要**  **参**  **考**  **资**  **料** | [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](http://search.dangdang.com/?key3=%C7%E5%BB%AA%B4%F3%D1%A7%B3%F6%B0%E6%C9%E7&medium=01&category_path=01.00.00.00.00.00),2017 | |

**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**

1. **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。

1. **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。

1. **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。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;

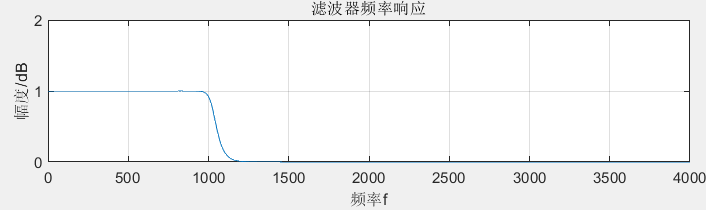


图3.1.1 IIR低通滤波器频率响应图

2）高通滤波器 滤波器参数为。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;

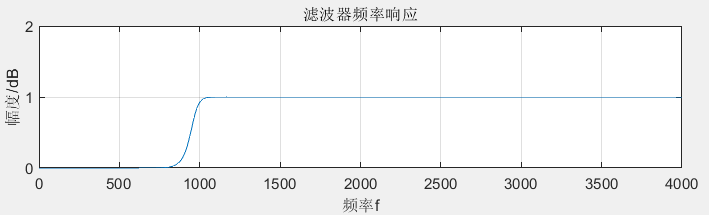


图3.1.2 IIR高通滤波器频率响应图

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;

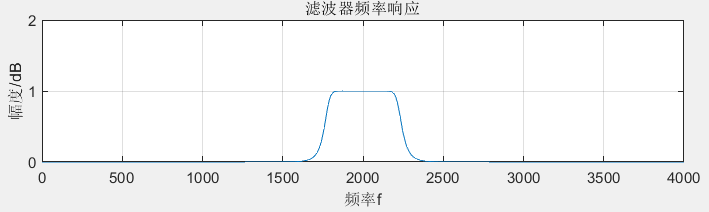


图3.1.3 IIR带通滤波器频率响应图

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;

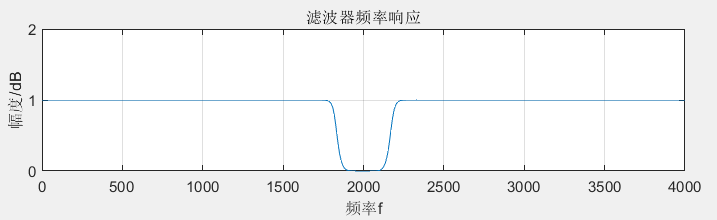


图3.1.4 IIR带阻滤波器频率响应图

**2、FIR滤波器的设计**

1）低通滤波器 滤波器参数为。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');

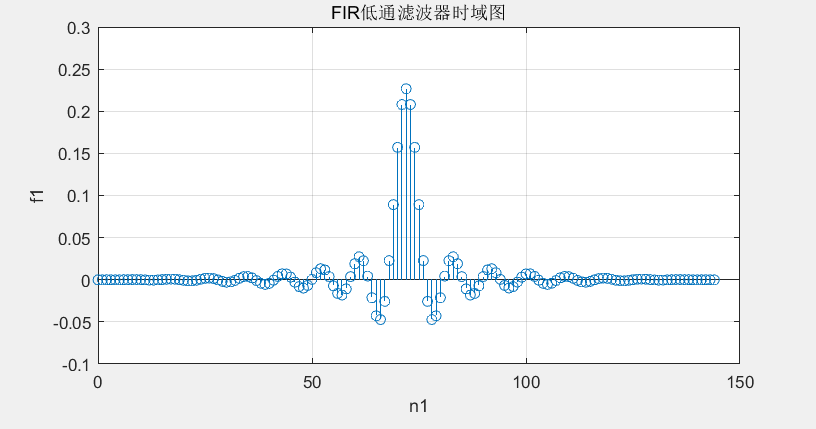


图3.2.1 FIR低通滤波器时域图

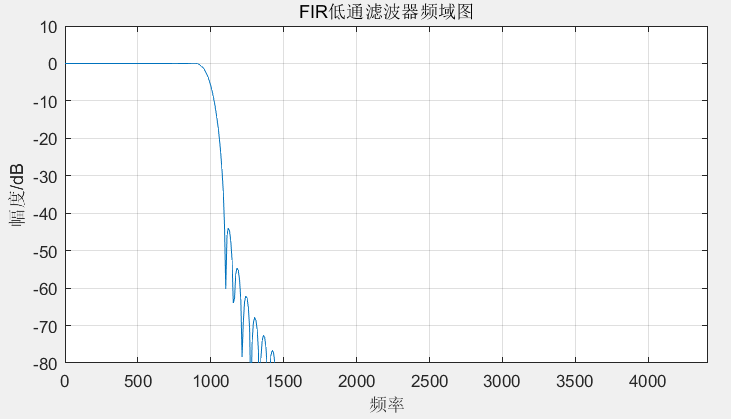


图3.2.2 FIR低通滤波器频域图

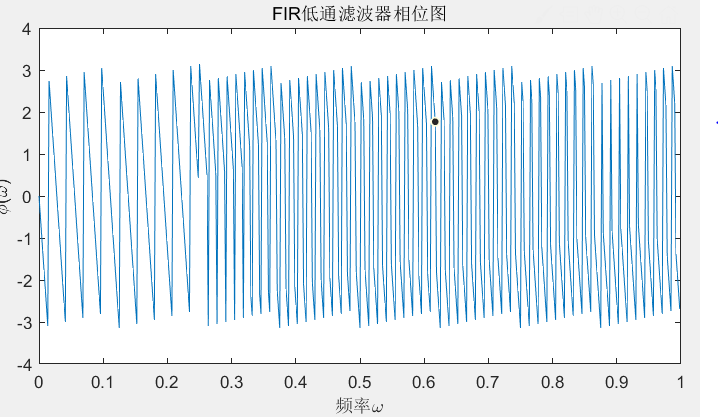


图3.2.3 FIR低通滤波器相位图

2）高通滤波器 滤波器参数为。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');

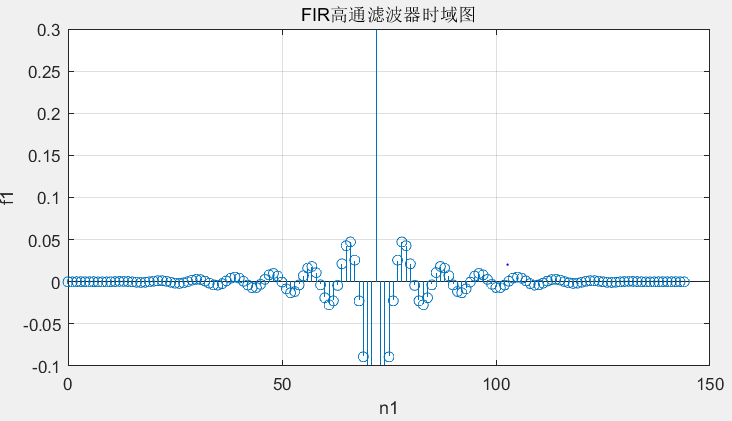


图3.2.4 FIR高通滤波器时域图

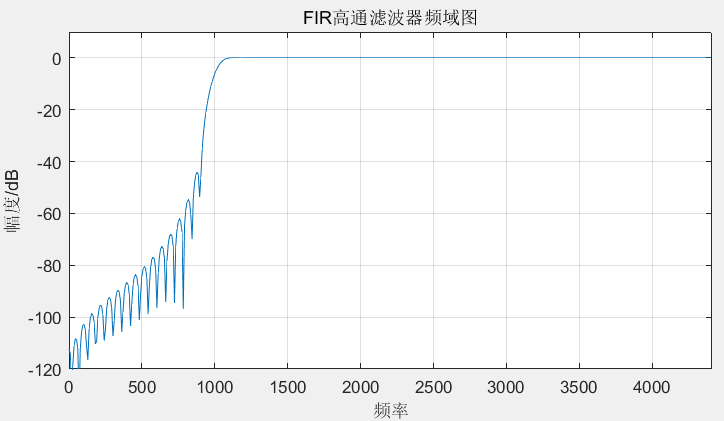


图3.2.5 FIR低通滤波器频域图

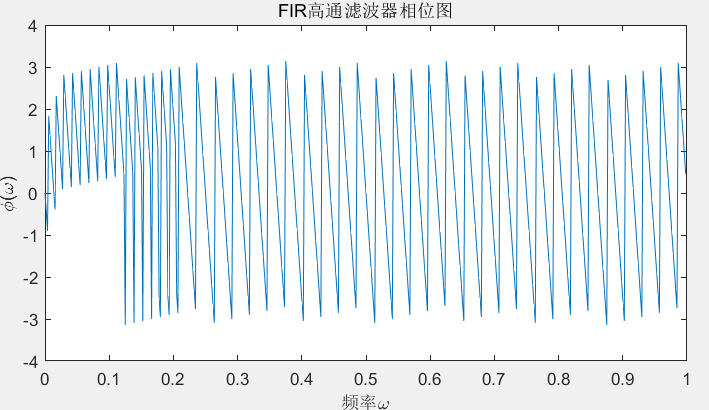


图3.2.6 FIR低通滤波器相位图

3）带通滤波器 滤波器参数为。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');

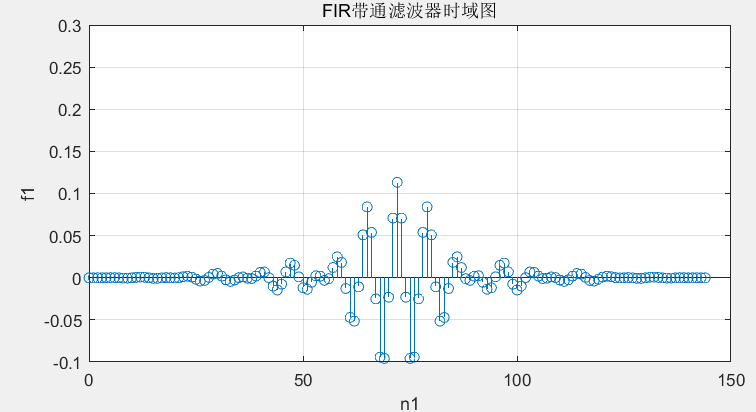
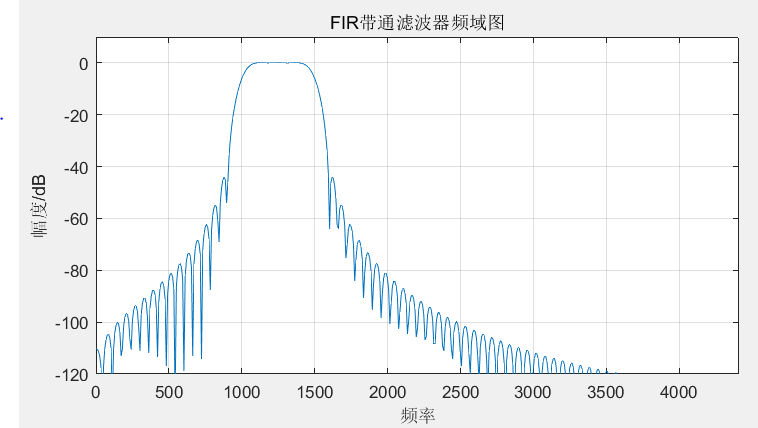
图3.2.7 FIR带通滤波器时域图

图3.2.8 FIR带通滤波器频域图

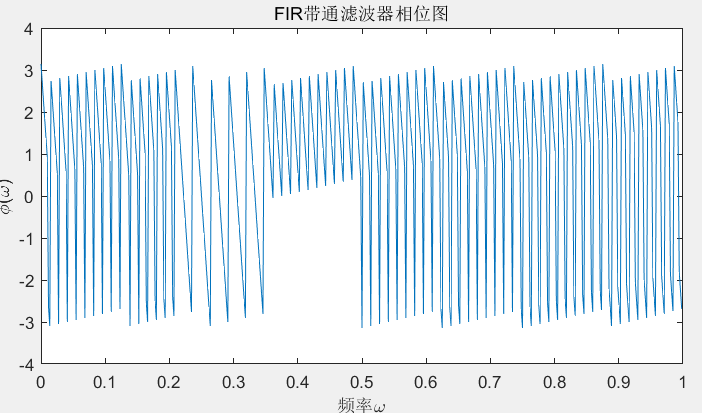


图3.2.9 FIR带通滤波器相位图

4）带通滤波器 滤波器参数为。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');

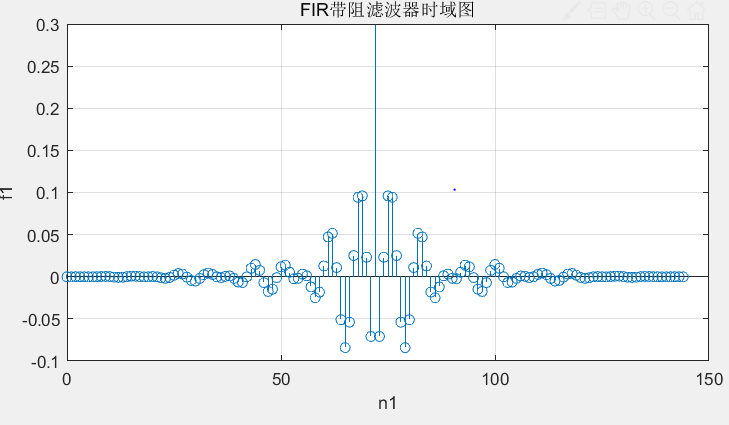
subplot(2,2,3);

plot(w1/pi,angle(h1));

xlabel('频率\omega');

ylabel('\phi(\omega)');

title(' FIR带阻滤波器相位图');



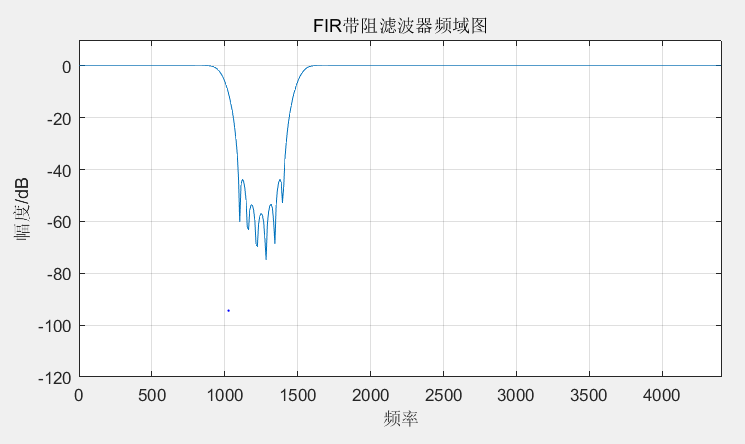
图3.2.10 FIR带通滤波器时域图

图3.2.11 FIR带通滤波器频域图

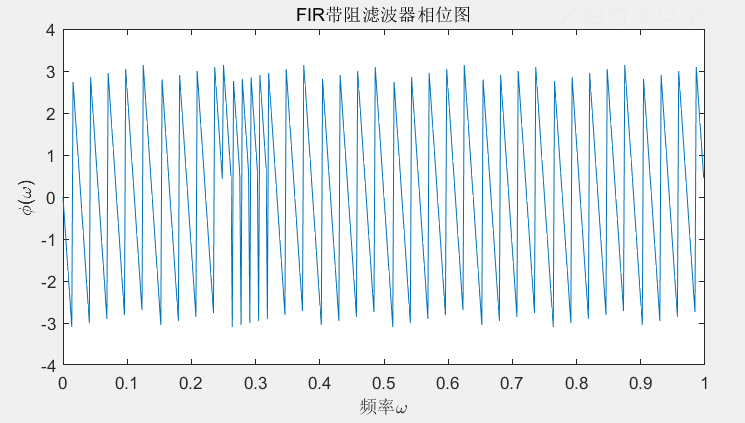


图3.2.12 FIR带通滤波器相位图

**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));

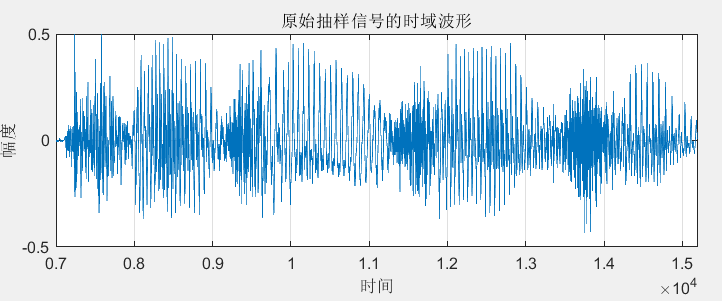
title('IIR滤波后频域图');

xlabel('频率');

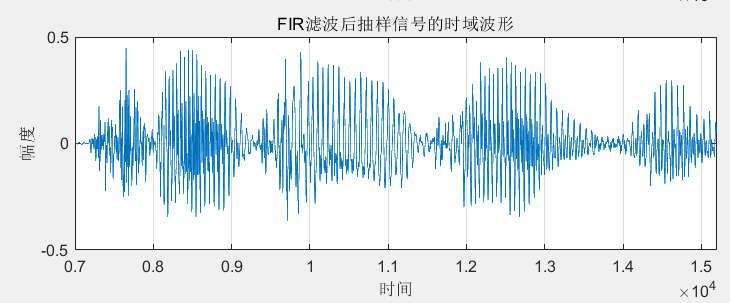
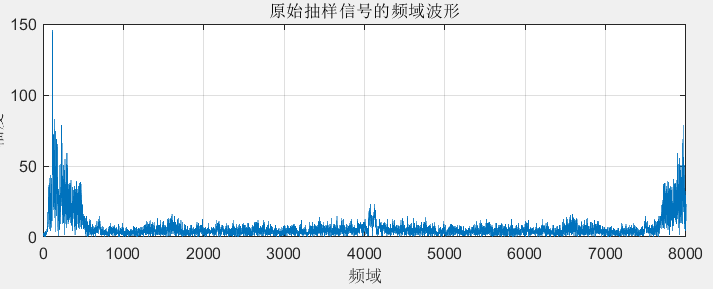
ylabel('幅度');

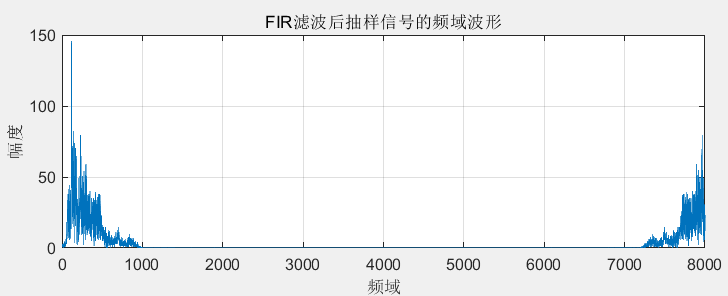
axis([0 8000 0 150]);

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

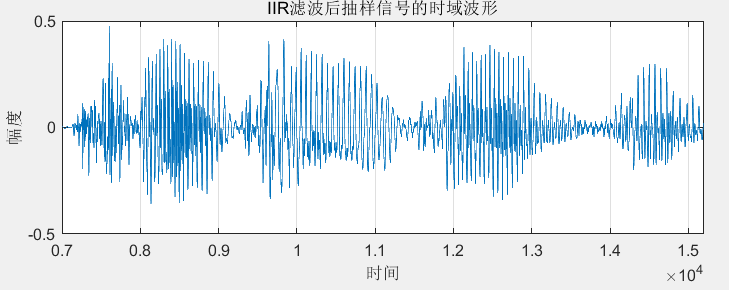
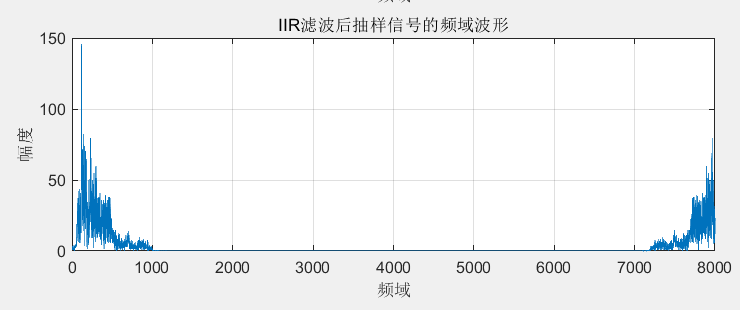
图3.3.4 FIR滤波后频域图

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);

}

}

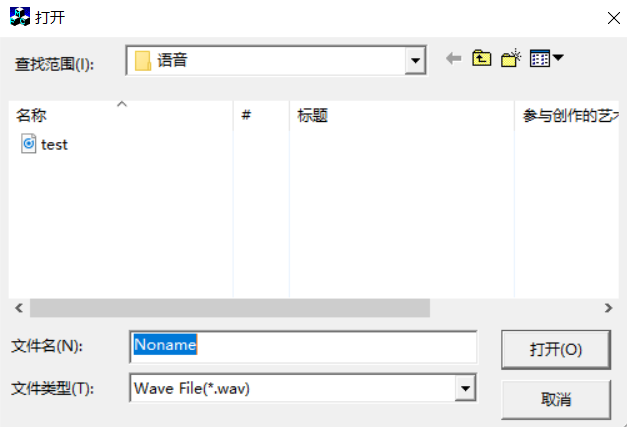


图３.7.2 打开文件按钮结果图

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。

1. **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。