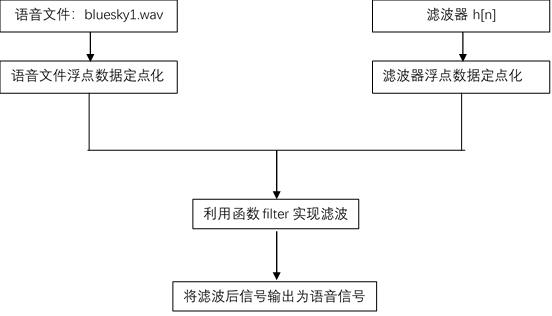
C语言具体代码实现

1、概述

该实验主要要求使用C语言将语言信号读出，并实现浮点数据定点化。而以.wav格式储存的信号都是小于1的浮点数据，通过函数fread读出数据放入十六位整型的数组indata中，电脑自动将浮点数据向左移位15位化为整型数据；而将19点滤波器化为定点数据方便滤波运算。将输入语音数据以一帧180位输出，在之后MATLAB运算中做比较，验证实验是否成功。数据处理完成之后利用自定义滤波函数filter对语音信号进行滤波并输出。

2、程序流程图

3、主程序模块

（1）、语音信号浮点数据定点化

定义接收数据数组为十六位整型数据,通过fread(buffer , size , count , stream );将语音信号读入该整型数组中，小于1的数据自动转化为整型数据，实现浮点数据定点化。

while(length==fread(indata,2,length,fp1))

（2）、滤波器整型化

而要实现语音信号的滤波处理，往往采用长序列卷积进行计算。整型数据的卷积计算要快于小数的卷积计算。故将19点滤波器都乘以2^15，实现滤波器的整型化。

for(i=0;i<filterlen;i++) temph[i]=h[i]\*32768;

其中filterlen为滤波器长度

（3）、语音信号数据处理

由于语音文件前44个字节都是该语音信号的头格式信息，故使用函数fseek(FILE \*stream, long offset, int fromwhere); 将指针指向文件内部的第44个字节位置；

fseek(fp1,44,SEEK\_SET);

而之后使用ftell(FILE \*stream);函数得到当前文件指针，便于之后的操作

fpend=ftell(fp1); //获取当前文件指针

（4）、将语音信号一帧一帧的打印出

printf("frame=%d:",frame);

for(j=0;j<length;j++) printf("%d, ",indata[j]);

4、低通滤波定点子程序模块

该滤波函数使用长序列卷积的叠接舍去法，叠接舍去法定义及操作过程可见实验原理，其中由于在主函数中对于语音信号的处理是将小于1的语音信号数据乘以2^15，使其转化为整型数据，故在该滤波程序中相应要除以2^15。使滤波后信号正确输出。

void filter(short xin[],short xout[],int n,long long int temph[])

{

int i,j;

float sum=0.0;

long long int tempsum;

for(i=0;i<length;i++)x1[n+i-1]=xin[i];

for(i=0;i<length;i++)

{

tempsum=0;

for(j=0;j<n;j++)tempsum+=temph[j]\*x1[i-j+n-1];

sum=tempsum/32768; //除以2的15次方

xout[i]=(short)sum;

}

for(i=0;i<(n-1);i++)x1[n-i-2]=xin[length-1-i];

}

5、将滤波后信号输出为.wav文件

随着对原信号的滤波实现，对滤波处理后信号利用fwrite(const void\* buffer, size\_t size, size\_t count, FILE\* stream); 函数以180项2字节写入文件，并将指向头的指针向后移动。

if(flag==fseek(fp2,fpstart,SEEK\_SET))

{

/\*判定条件：设定写入文件的方式，180项2字节数\*/

if(length==fwrite(outdata,2,length,fp2))

{

fseek(fp2,fpend,SEEK\_SET); //设置当前处理完的指针

fpstart=fpend; //后移处理指针，下一个180项

}

}

C语言完整代码及运行结果

#include<stdio.h>

#include<math.h>

#include<stdlib.h>

#define length 180 /\*语音帧长为180点=22.5ms@8kHz采样\*/

#define filterlen 19

#define flag 0

/\*19点滤波器系数\*/

static float h[filterlen]={0.01218354,-0.009012882,-0.02881839,-0.04743239,-0.04584568,

-0.008692503,0.06446265,0.1544655,0.2289794,0.257883,

0.2289794,0.1544655,0.06446265,-0.008692503,-0.04584568,

-0.04743239,-0.02881839,-0.009012882,0.01218354} ; static short x1[length+filterlen-1];

static long long int temph[filterlen];

/\*低通滤波定点子程序\*/

void filter(short xin[],short xout[],int n,long long int temph[])

{

int i,j;

float sum=0.0;

long long int tempsum;

for(i=0;i<length;i++)x1[n+i-1]=xin[i];

for(i=0;i<length;i++)

{

tempsum=0;

for(j=0;j<n;j++)tempsum+=temph[j]\*x1[i-j+n-1];

sum=tempsum/32768; //除以2的15次方

xout[i]=(short)sum;

}

for(i=0;i<(n-1);i++)x1[n-i-2]=xin[length-1-i];

}

/\*主函数\*/

int main()

{

int i,j;

int fpstart=44,fpend;

FILE \*fp1=0,\*fp2=0;

int frame=0;

short indata[length],indata1[length],indata2[length],outdata[length];

for(i=0;i<filterlen;i++) temph[i]=h[i]\*32768; //乘以2的15次方

fp1=fopen("bluesky1.wav","r+b"); //输入语音文件

fp2=fopen("outdata.wav","w+b"); //滤波后语音文件

fseek(fp1,44,SEEK\_SET);

while(length==fread(indata,2,length,fp1))

{

fpend=ftell(fp1); //获取当前文件指针

frame++;

printf("frame=%d:",frame);

for(j=0;j<length;j++) printf("%d, ",indata[j]);

printf("\n");

filter(indata,outdata,filterlen,temph); //调用低通滤波子程序

if(flag==fseek(fp2,fpstart,SEEK\_SET))

{

/\*判定条件：设定写入文件的方式，180项2字节数\*/

if(length==fwrite(outdata,2,length,fp2))

{

fseek(fp2,fpend,SEEK\_SET); //设置当前处理完的指针

fpstart=fpend; //后移处理指针，下一个180项

}

}

else

{

printf("fail\n");

break;

}

if(frame>4000)

{

printf("data overflow\n");

break;

}

}

fclose(fp1); //关闭文件

fclose(fp2);

return(0);

}

MATLAB具体代码实现

1、概述

本次实验需要使用matlab汇出滤波器的频谱，原语音信号频谱，滤波后的频谱，加入低频噪声的频谱，滤除低频噪声的频谱，加入高频噪声的频谱，滤除高频噪声的频谱；并观察上述语音的语谱图。故首先我们需要通过使用matlab读出语音信号的数据，分别做出其时域与频域特性曲线；分别对其加载高频与低频正弦噪声，并做出其时域与频域特性曲线；将滤波器分别用时域与频域特性曲线表示；然后通过fftfilt函数分别将原始信号、加高频噪声信号、加低频噪声信号进行滤波，并做其频谱图。最后通过函数spectrogram做信号的语谱图。将三种情况下的语音信号及其滤波后的语谱图进行比较。

2、程序流程图

3、对.wav文件对处理

（1）、读取语音文件

本次实验是利用调用函数[x,fs,nbits]=wavread(filename)来实现读取语音文件的功能的，其中wavread函数内容如下：

function [ x, fs, nbits ] = wavread( filename )

% 模拟封装老版本的wavread函数，读取数据、采样率、采样位数

% 读取数据和采样率

[x, fs] = audioread(filename);

% 读取采样位数

info = audioinfo(filename);

nbits = info.BitsPerSample;

end

（2）、将处理后数据输出为.wav文件

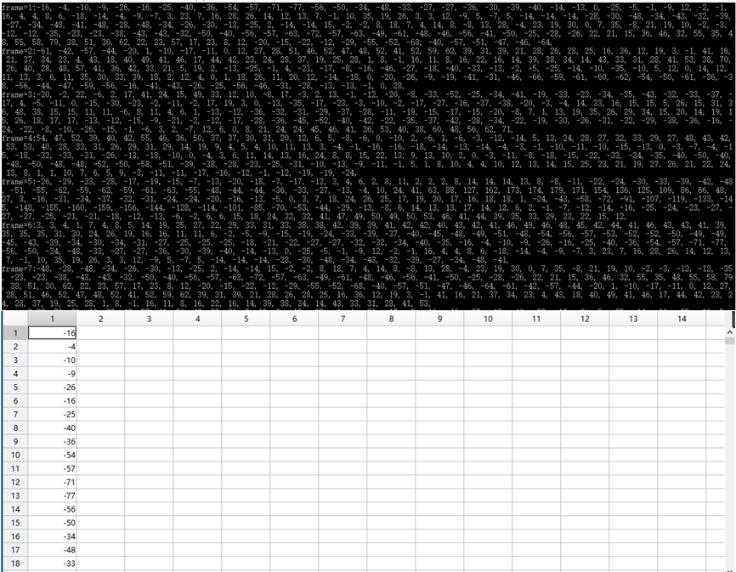
本次实验之中是通过调用**audiowrite(filename ,y,fs)**；函数将处理后的数据输出为.wav文件

（3）、和C语言读出数据进行比较

由于读出数据是幅值绝对值小于1的值，在C语言代码里面是对其转化成整数之后进行处理，故在MATLAB中是对数据放大2^15之后再进行比较的。故需执行如下一步：

[x,fs,n]=wavread('bluesky1.wav');

s1=x\*(2^15);

比较结果如下：

**4、滤波器**

本次实验滤波器是事先给定的语音信号800Hz 19点FIR低通滤波

h=[ 0.01218354,-0.009012882,-0.02881839,-0.04743239,-0.04584568,

-0.008692503,0.06446265,0.1544655,0.2289794,0.257883,

0.2289794,0.1544655,0.06446265,-0.008692503,-0.04584568,

-0.04743239,-0.02881839,-0.009012882,0.01218354];

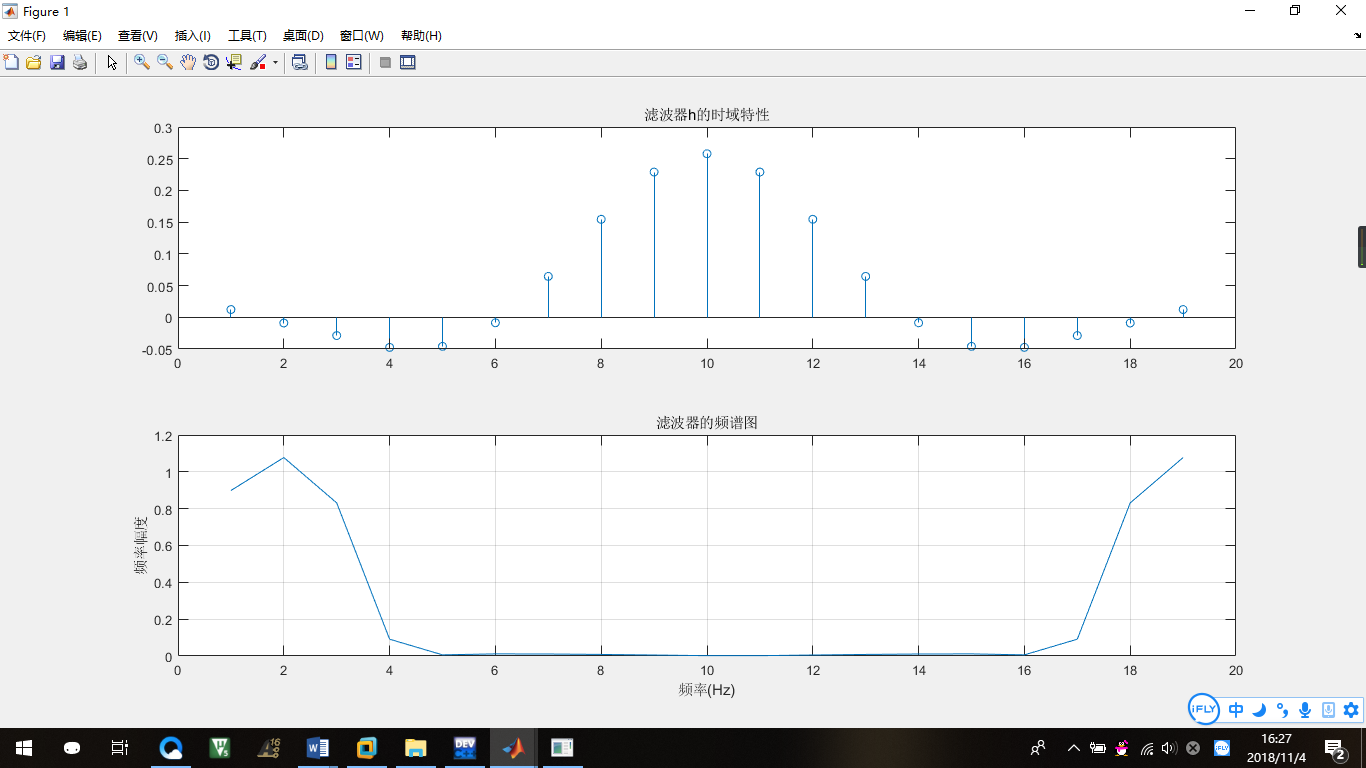
subplot(4,4,3);stem(h);title('滤波器h的时域特性');

H=fft(h,19);

subplot(4,4,4);plot(abs(H)), title('滤波器的频谱图'),grid;

xlabel('频率(Hz)'); ylabel('频率幅度');

滤波器其时域特性曲线及频域特性曲线如下：



5、原始信号的曲线及处理

（1）、原始信号的时域波形及频域波形

利用函数wavread('bluesky1.wav')读取一个WAVE文件，并返回采样数据到向量x中，fs表示采样频率,n表示采样位数，然后截取第一个声道声音进行分析

[x,fs,n]=wavread('bluesky1.wav');

ft=x(:,1);%我们分析声音文件的第一个声道

得到语音信号时域的数据之后，将其进行变换到频域进行分析

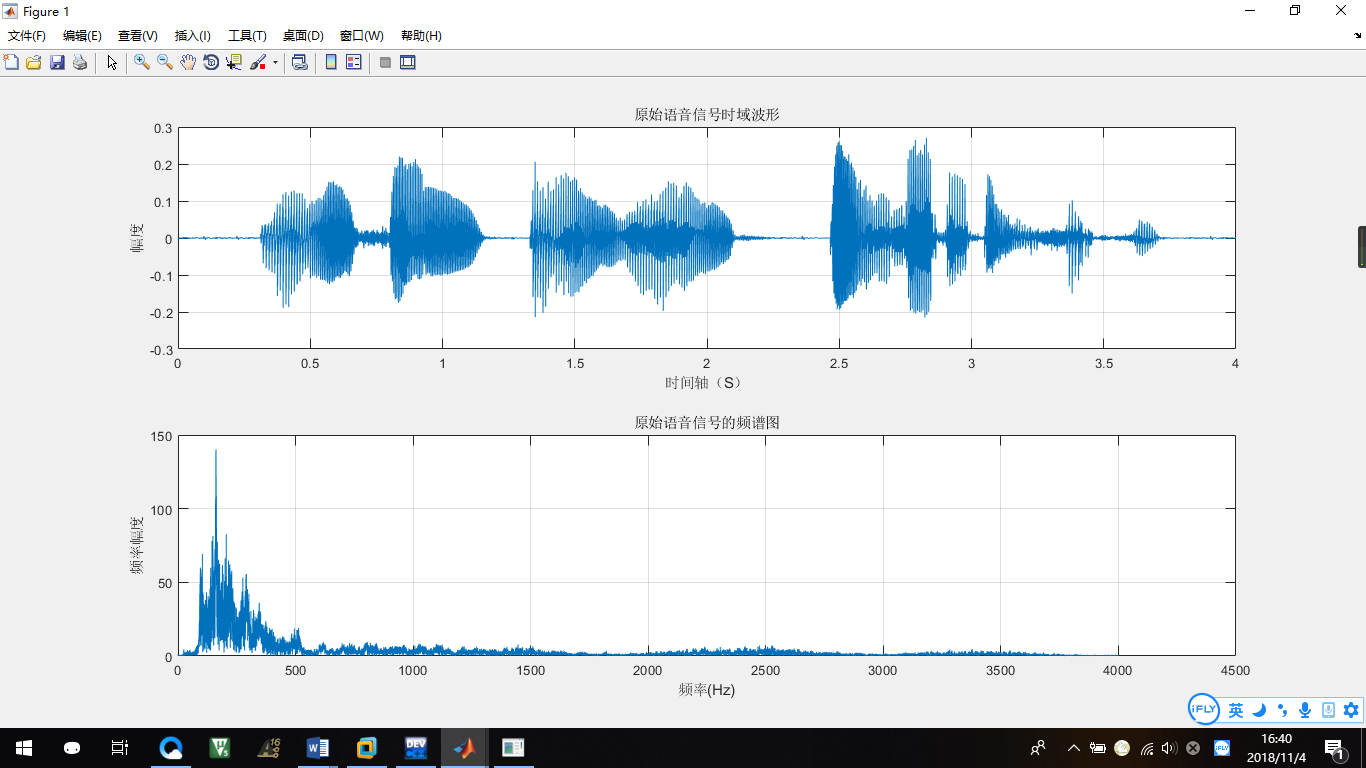
X=fft(ft,sigLength);

%使用快速傅里叶变换算法返回向量ft的离散型傅里叶变换

Fx =X(1:halfLength + 1); % 只选取前半截部分

Fx = abs(Fx);%用于计算复向量的Y的振幅

其中sigLength为语音信号的长度。

做原始语音信号的时域特性曲线及频域特性曲线：

（2）、对原始信号进行滤波

利用函数y=fftfilt(h,x);进行原始信号的滤波.并将滤波后的信号以语音的模式输出。做出原始语音信号滤波后的时域特性曲线和频域特性曲线：

y=fftfilt(h,x);

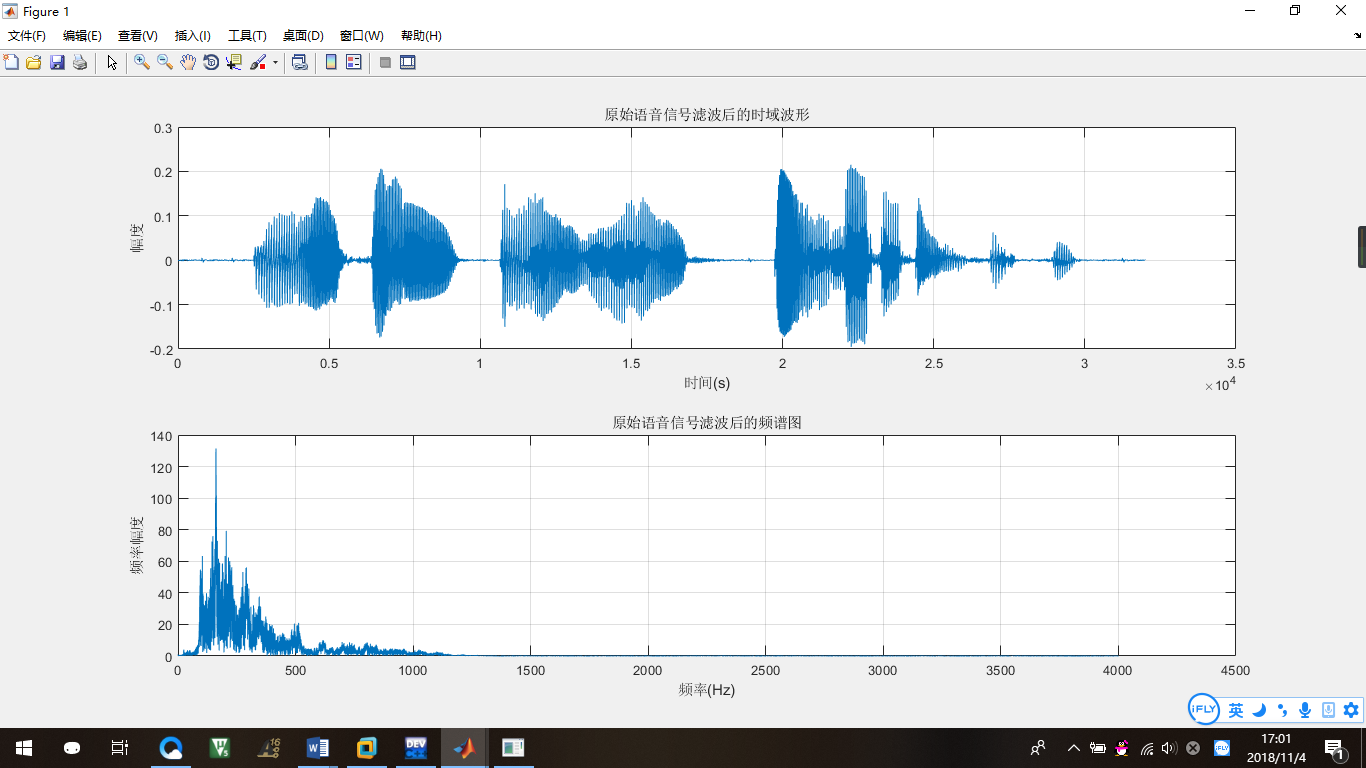
audiowrite('out\_y.wav',y,fs);

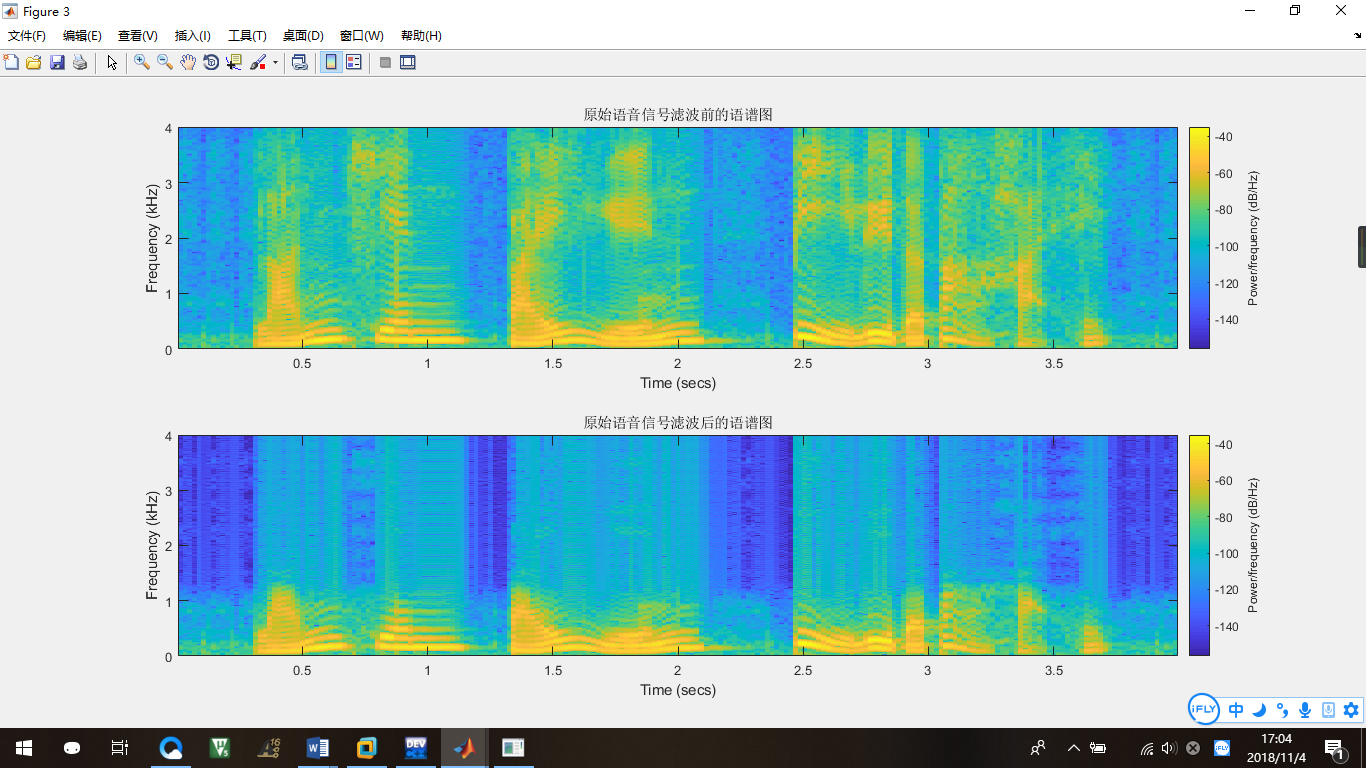
Y=fft(y,sigLength);

%使用快速傅里叶变换算法返回向量ft的离散型傅里叶变换

Fy =Y(1:halfLength + 1); % 只选取前半截部分

Fy = abs(Fy);%用于计算复向量的Y的振幅

（3）、原始信号及滤波后的语谱图

利用函数**spectrogram**做信号的语谱图

6、加高频噪声的信号的曲线及处理

（1）、加高频噪声的信号的时域波形及频域波形

对原始信号加一高频正弦噪声，由于原始信号为语音信号，其频率大约是800Hz左右；故选择高频频率为5000Hz。加载在原始信号上，并转化为语音输出

Au1=0.03;

d1=[Au1\*cos(2\*pi\*5000\*t)]'; %噪声为5kHz的余弦信号

x1=ft+d1;

audiowrite('out\_x1.wav',x1,fs);

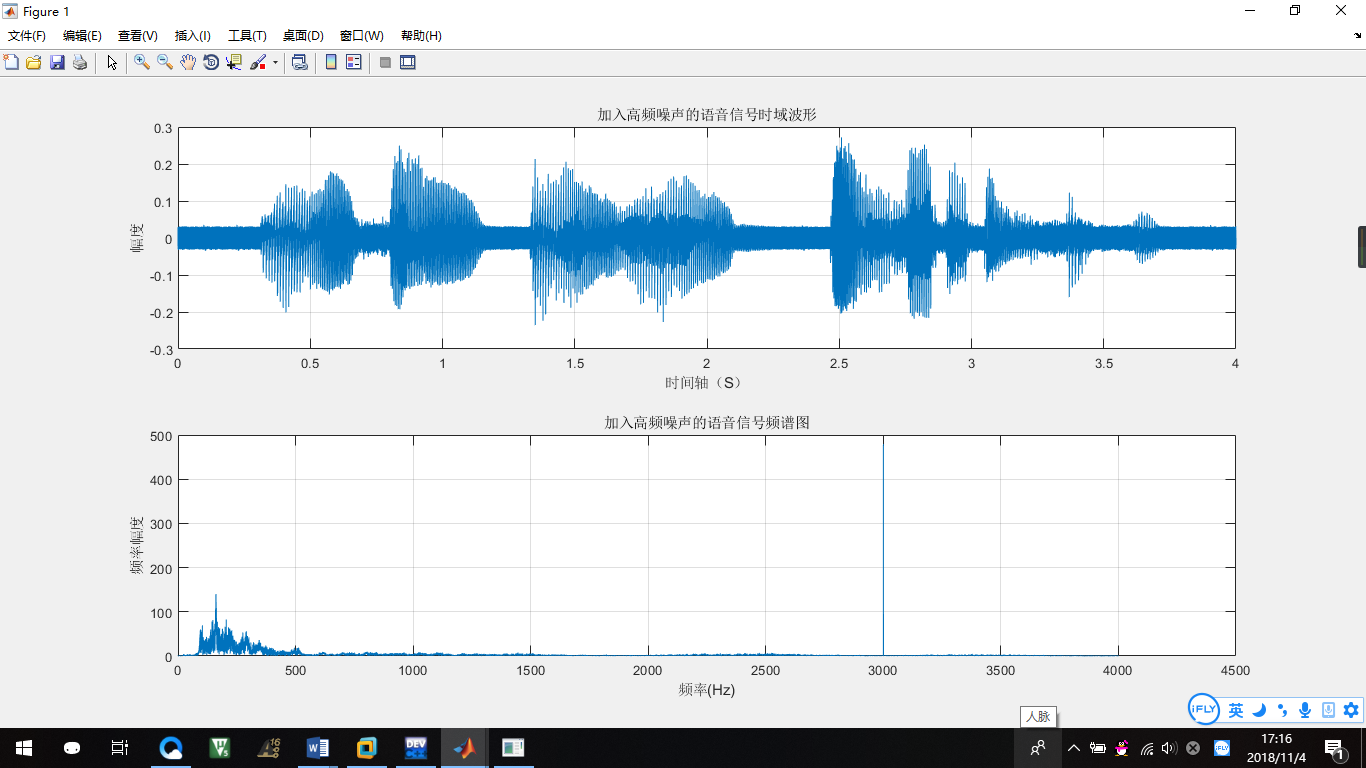
得到语音信号时域的数据之后，将其进行变换到频域进行分析

X1=fft(x1,sigLength);

Fx1 =X1(1:halfLength + 1); % 只选取前半截部分

Fx1 = abs(Fx1);%用于计算复向量的Y的振幅

其中sigLength为语音信号的长度。

做加高频噪声的信号的时域特性曲线及频域特性曲线：

（2）、对加高频噪声的信号进行滤波

利用函数y=fftfilt(h,x);进行加高频噪声的信号的滤波.并将滤波后的信号以语音的模式输出。

y1=fftfilt(h,x1);

audiowrite('out\_y1.wav',y1,fs);

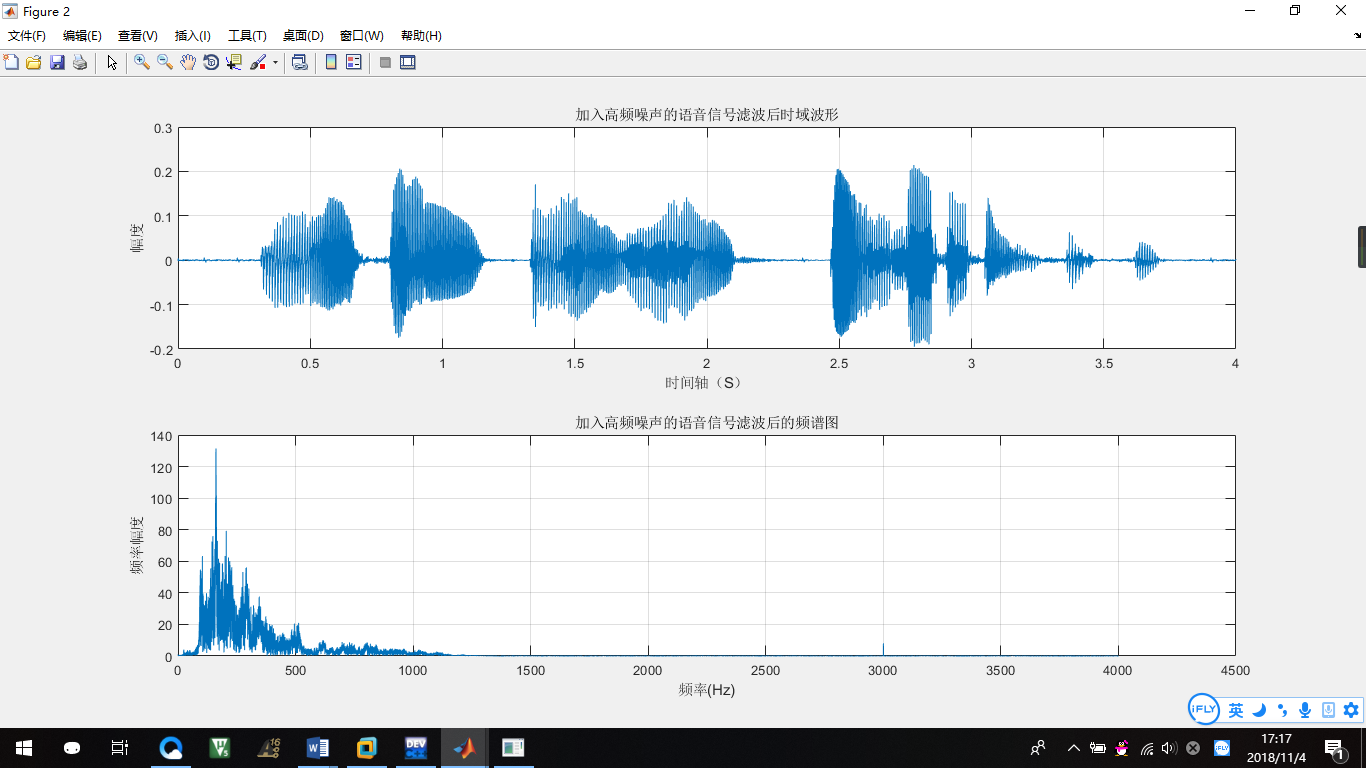
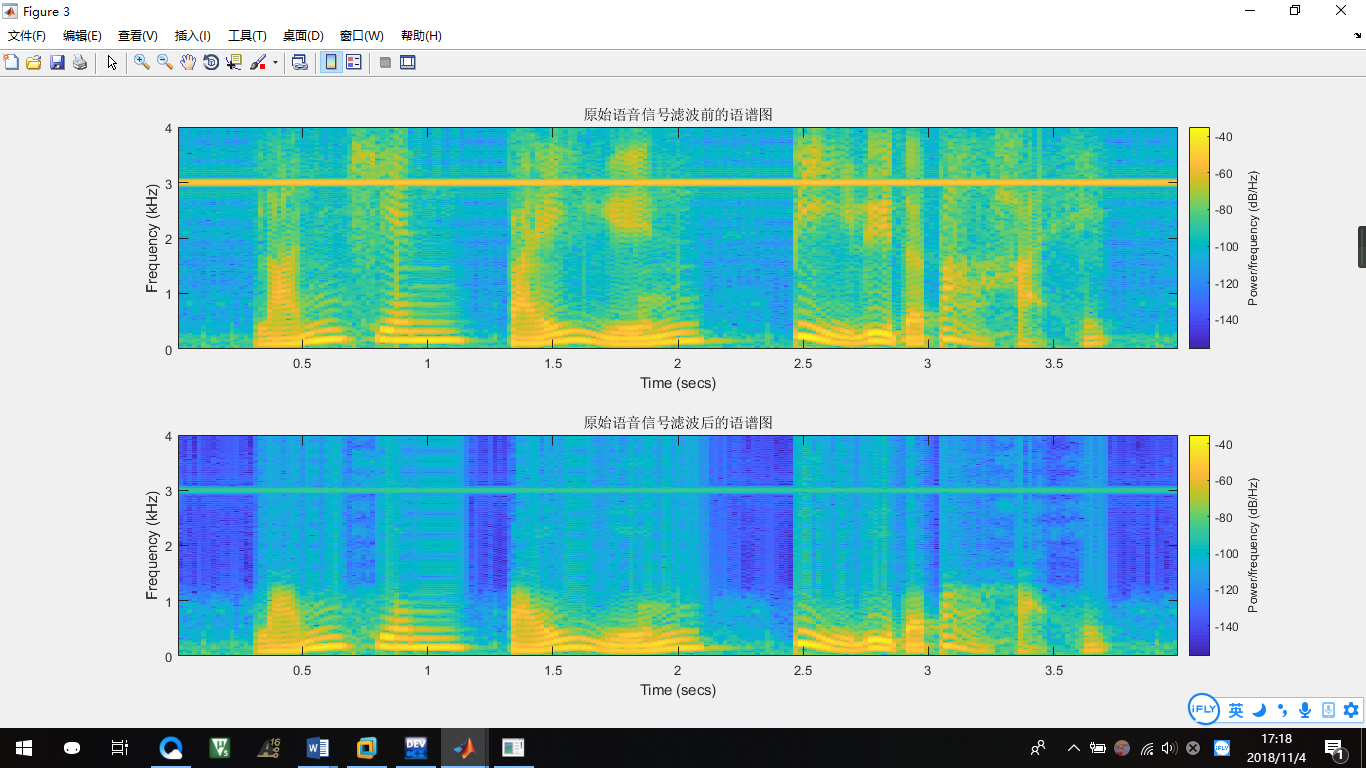
Y1=fft(y1,sigLength);

%使用快速傅里叶变换算法返回向量ft的离散型傅里叶变换

Fy1 =Y1(1:halfLength + 1); % 只选取前半截部分

Fy1=abs(Fy1);%用于计算复向量的Y的振幅

做出原始语音信号滤波后的时域特性曲线和频域特性曲线：

（3）、加高频噪声的信号及其滤波后的语谱图

利用函数**spectrogram**做信号的语谱图

7、加低频噪声的信号的曲线及处理

（1）、加低频噪声的信号的时域波形及频域波形

对原始信号加一低频正弦噪声，由于原始信号为语音信号，其频率大约是800Hz左右；故选择高频频率为500Hz。加载在原始信号上，并转化为语音输出

Au2=0.03;

d2=[Au2\*cos(2\*pi\*500\*t)]'; %噪声为500Hz的余弦信号

x2=ft+d2;

audiowrite('out\_x2.wav',x2,fs);

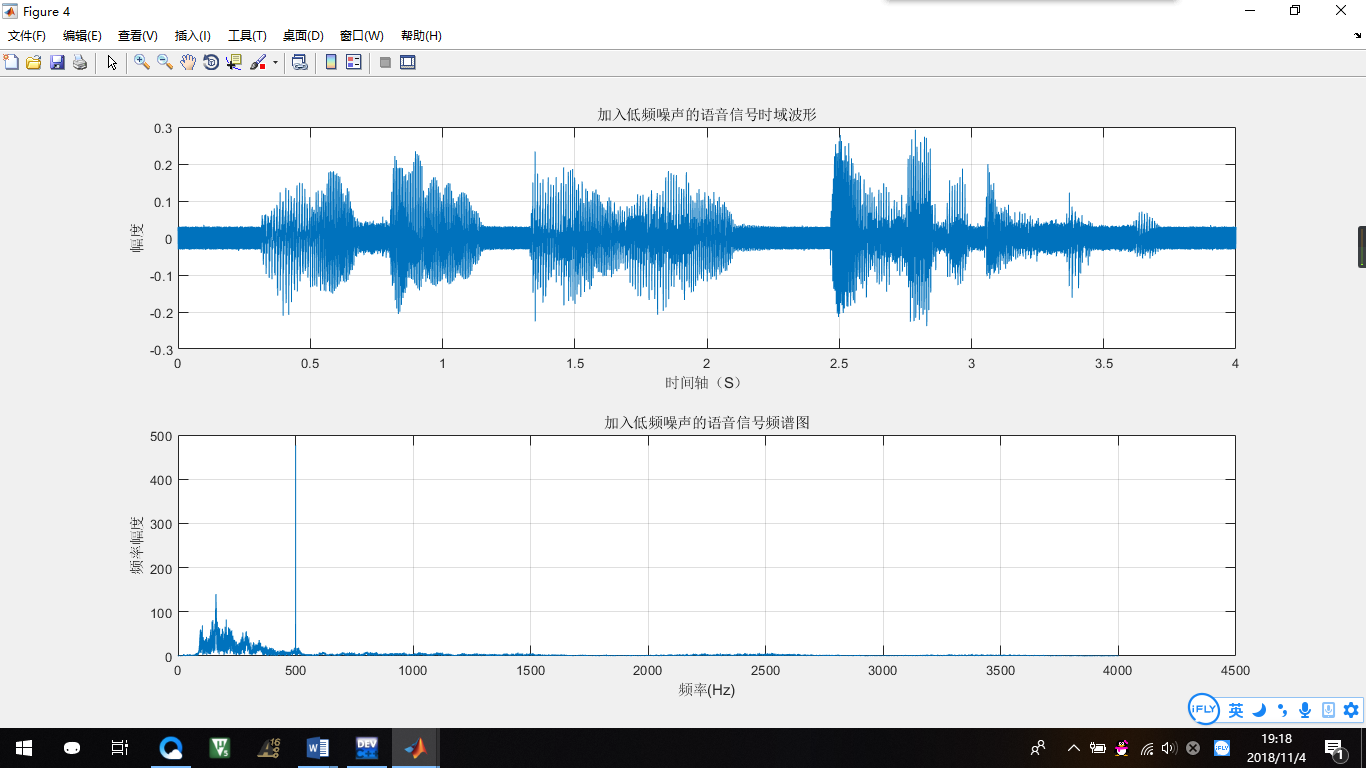
得到语音信号时域的数据之后，将其进行变换到频域进行分析

X2=fft(x2,sigLength);

Fx2 =X2(1:halfLength + 1); % 只选取前半截部分

Fx2 = abs(Fx2);%用于计算复向量的Y的振幅

其中sigLength为语音信号的长度。

做加低频噪声的信号的时域特性曲线及频域特性曲线：

（2）、对加低频噪声的信号进行滤波

利用函数y=fftfilt(h,x);进行加低频噪声的信号的滤波.并将滤波后的信号以语音的模式输出。

y2=fftfilt(h,x2);

audiowrite('out\_y2.wav',y2,fs);

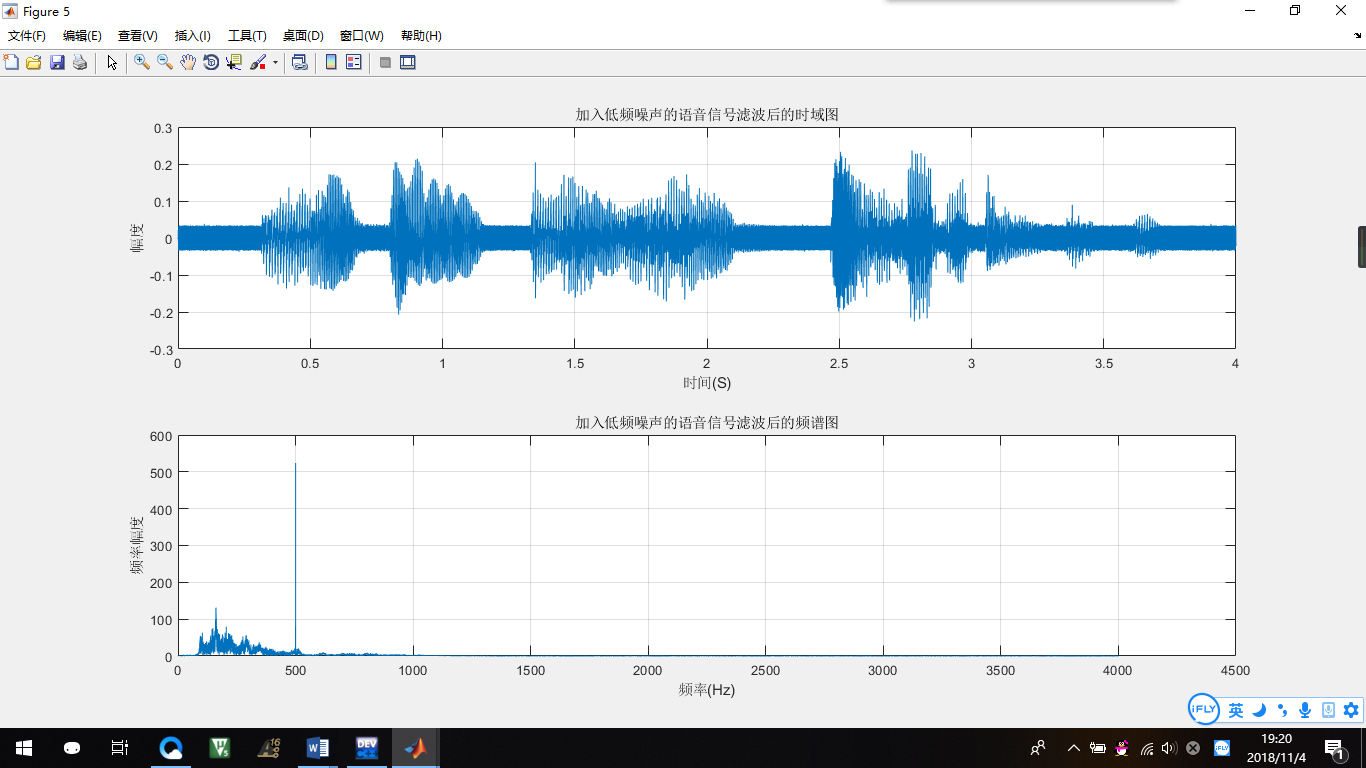
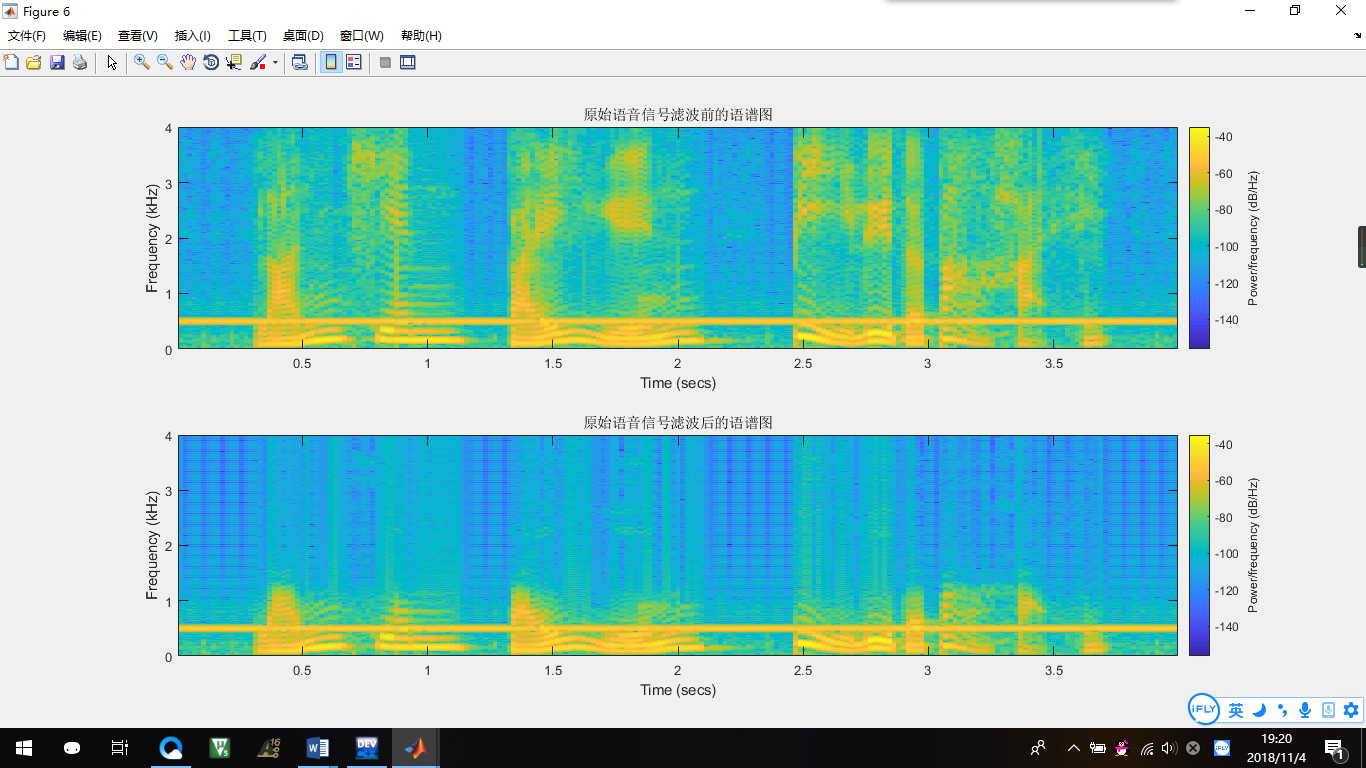
Y2=fft(y2,sigLength);

%使用快速傅里叶变换算法返回向量ft的离散型傅里叶变换

Fy2 =Y2(1:halfLength + 1); % 只选取前半截部分

Fy2=abs(Fy2);%用于计算复向量的Y的振幅

做出原始语音信号滤波后的时域特性曲线和频域特性曲线：

（3）、加低频噪声的信号及其滤波后的语谱图

利用函数**spectrogram**做信号的语谱图

8、单独取连续一帧

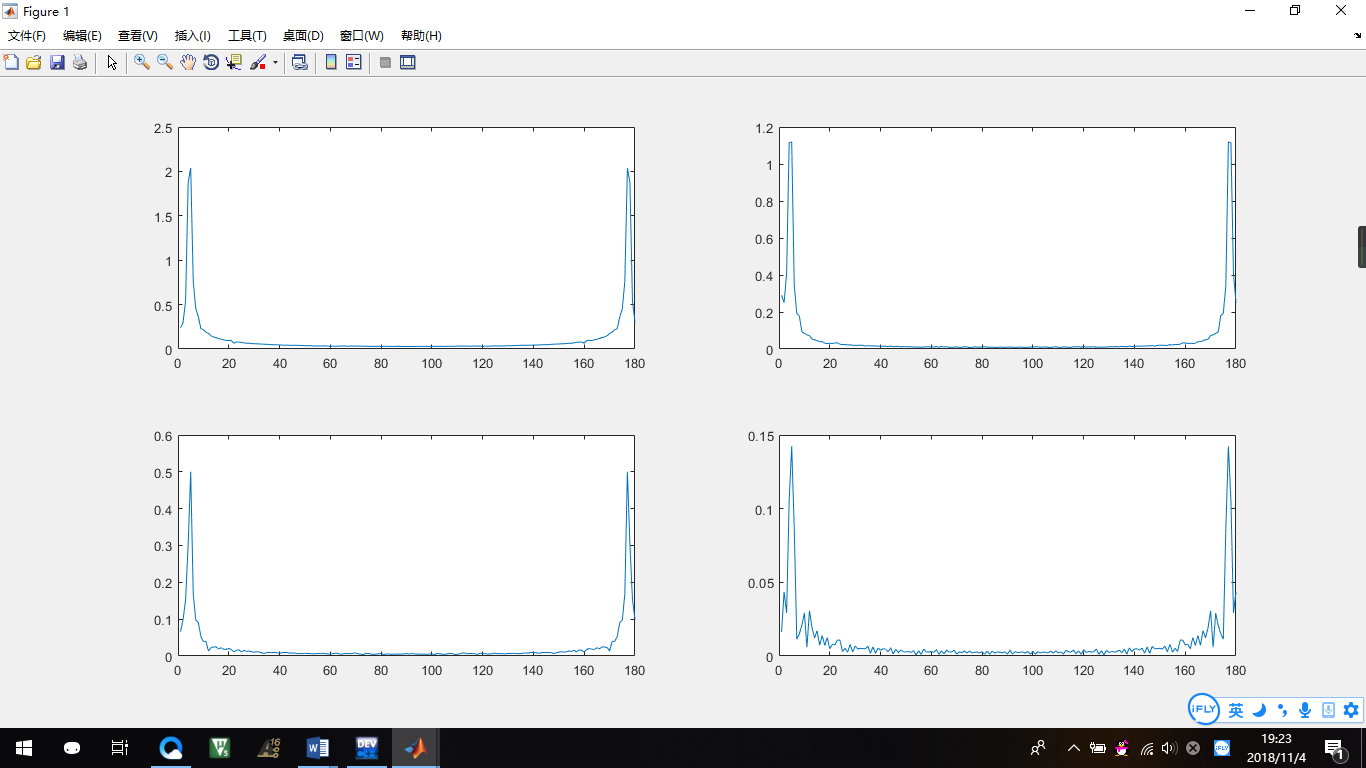
由于原数据每帧长度为180，故从第50帧开始连续取四帧，且相邻两帧之间重叠部分长度为90bits

p1=ft(9000:9179); P1=fft(p1,180); subplot(221);plot(abs(P1));

p2=ft(9090:9269); P2=fft(p2,180); subplot(222);plot(abs(P2));

p3=ft(9180:9359); P3=fft(p3,180); subplot(223);plot(abs(P3));

p4=ft(9270:9349); P4=fft(p4,180); subplot(224);plot(abs(P4));



MATLAB完整代码及运行结果

1、MATLAB完整代码

fs=8000; %采样率

n=16; %分辨率

[x,fs,n]=wavread('bluesky1.wav');%读取一个WAVE文件，并返回采样数据到向量y中，Fs表示采样频率, bits表示采样位数

%C语言与MATLAB比对数据%

s1=x\*(2^15);

%定义所需的参变量%

ft=x(:,1);%我们分析声音文件的第一个声道

sigLength=length(ft); %获取声音长度

halfLength = floor(sigLength/2);

f=((0:halfLength)+1)\* fs/sigLength;

t=(0:sigLength-1)/fs; %定义变量t

%做出语谱图%

subplot(4,4,1); plot(t,ft), title('原始语音信号时域波形'),grid; %做出语谱图

xlabel('时间轴（S）'); ylabel('幅度');

%原始信号频谱图%

X=fft(ft,sigLength); %使用快速傅里叶变换算法返回向量ft的离散型傅里叶变换

Fx =X(1:halfLength + 1); % 只选取前半截部分

Fx = abs(Fx);%用于计算复向量的Y的振幅

subplot(4,4,2);plot(f,Fx), title('原始语音信号的频谱图'),grid; %做出语音信号的频谱图

xlabel('频率(Hz)'); ylabel('频率幅度');

%滤波器的设计与频谱图%

h=[0.01218354,-0.009012882,-0.02881839,-0.04743239,-0.04584568,-0.008692503,0.06446265,0.1544655,0.2289794,0.257883,0.2289794,0.1544655,0.06446265,-0.008692503,-0.04584568,-0.04743239,-0.02881839,-0.009012882,0.01218354];

subplot(4,4,3);stem(h);title('滤波器h的时域特性');

H=fft(h,19);

subplot(4,4,4);plot(abs(H)), title('滤波器的频谱图'),grid; %做出语音信号的频谱图

xlabel('频率(Hz)'); ylabel('频率幅度');

%加入高频为5kHz的余弦噪声的语音信号%

Au1=0.03;

d1=[Au1\*cos(2\*pi\*5000\*t)]'; %噪声为5kHz的余弦信号

x1=ft+d1;

audiowrite('out\_x1.wav',x1,fs);

subplot(4,4,5); plot(t,x1), title('加入高频噪声的语音信号时域波形'),grid; %做出语谱图

xlabel('时间轴（S）'); ylabel('幅度');

X1=fft(x1,sigLength);

Fx1 =X1(1:halfLength + 1); % 只选取前半截部分

Fx1 = abs(Fx1);%用于计算复向量的Y的振幅

subplot(4,4,6);plot(f,Fx1), title('加入高频噪声的语音信号频谱图'),grid; %做出语音信号的频谱图

xlabel('频率(Hz)'); ylabel('频率幅度');

%加入低频为1000Hz的余弦噪声的语音信号%

Au2=0.03;

d2=[Au2\*cos(2\*pi\*500\*t)]'; %噪声为5kHz的余弦信号

x2=ft+d2;

audiowrite('out\_x2.wav',x2,fs);

subplot(4,4,7); plot(t,x2), title('加入低频噪声的语音信号时域波形'),grid; %做出语谱图

xlabel('时间轴（S）'); ylabel('幅度');

X2=fft(x2,sigLength);

Fx2 =X2(1:halfLength + 1); % 只选取前半截部分

Fx2 = abs(Fx2);%用于计算复向量的Y的振幅

subplot(4,4,8);plot(f,Fx2), title('加入低频噪声的语音信号频谱图'),grid; %做出语音信号的频谱图

xlabel('频率(Hz)'); ylabel('频率幅度');

%对原始信号进行滤波

y=fftfilt(h,x);

audiowrite('out\_y.wav',y,fs);

Y=fft(y,sigLength); %使用快速傅里叶变换算法返回向量ft的离散型傅里叶变换

Fy =Y(1:halfLength + 1); % 只选取前半截部分

Fy = abs(Fy);%用于计算复向量的Y的振幅

subplot(4,4,9);plot(f,Fy), title('原始语音信号滤波后的频谱图'),grid; %做出语音信号的频谱图

xlabel('频率(Hz)'); ylabel('频率幅度');

%对加入高频为5kHz的余弦噪声的语音信号进行滤波

y1=fftfilt(h,x1);

audiowrite('out\_y1.wav',y1,fs);

Y1=fft(y1,sigLength); %使用快速傅里叶变换算法返回向量ft的离散型傅里叶变换

Fy1 =Y1(1:halfLength + 1); % 只选取前半截部分

Fy1=abs(Fy1);%用于计算复向量的Y的振幅

subplot(4,4,10);plot(f,Fy1), title('加入高频噪声的语音信号滤波后的频谱图'),grid; %做出语音信号的频谱图

xlabel('频率(Hz)'); ylabel('频率幅度');

%对加入低频为50Hz的余弦噪声的语音信号进行滤波

y2=fftfilt(h,x2);

audiowrite('out\_y2.wav',y2,fs);

Y2=fft(y2,sigLength); %使用快速傅里叶变换算法返回向量ft的离散型傅里叶变换

Fy2 =Y2(1:halfLength + 1); % 只选取前半截部分

Fy2=abs(Fy2);%用于计算复向量的Y的振幅

subplot(4,4,11);plot(f,Fy2), title('加入低频噪声的语音信号滤波后的频谱图'),grid; %做出语音信号的频谱图

xlabel('频率(Hz)'); ylabel('频率幅度');

m=180;

p1=ft(9000:9179); P1=fft(p1,180); subplot(221);plot(abs(P1));

p2=ft(9090:9269); P2=fft(p2,180); subplot(222);plot(abs(P2));

p3=ft(9180:9359); P3=fft(p3,180); subplot(223);plot(abs(P3));

p4=ft(9270:9349); P4=fft(p4,180); subplot(224);plot(abs(P4));

%画语谱图

figure;

subplot(321); spectrogram(ft,200,50,32000,8000,'yaxis');

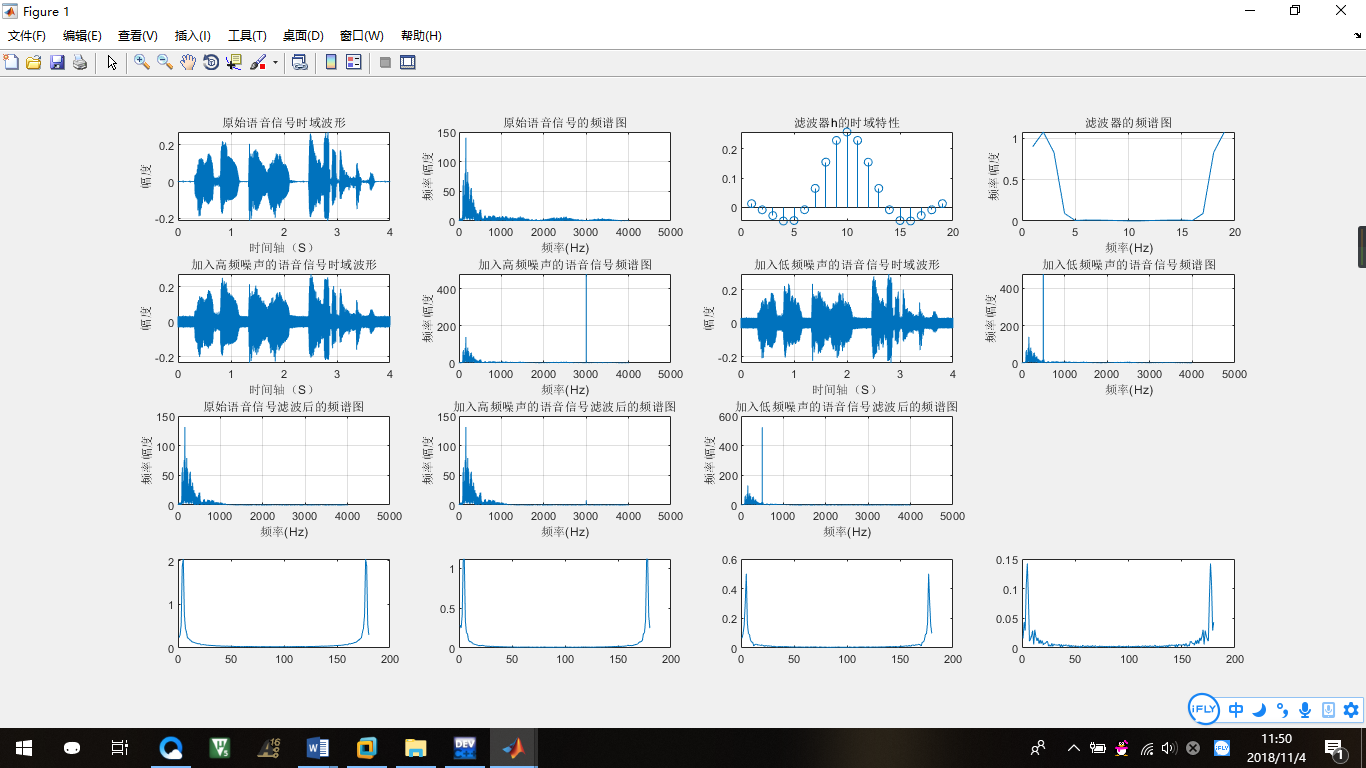
subplot(322); spectrogram(y,200,50,32000,8000,'yaxis');

subplot(323); spectrogram(x1,200,50,32000,8000,'yaxis');

subplot(324); spectrogram(y1,200,50,32000,8000,'yaxis');

subplot(325); spectrogram(x2,200,50,32000,8000,'yaxis');

subplot(326); spectrogram(y2,200,50,32000,8000,'yaxis');

2、运行结果

