Signals and System Project 1

目录

Introduction

1. 语音合成器的基本模型
    1. 首先根据耳蜗长度与频率的对应关系分解出N段频域。
    2. 对每段频域进行取包络。
    3. 对所取出的包络加上该频域中位数频域的载波还原回原来频域位置。
2. 巴特沃斯低通/带通滤波器原理
    1. [b,a]=butter(n,Wn,'ftype') 能够返回一个n阶的巴特沃斯滤波器，Wn 范围为0-1, 1.0 对应截止频率为采样率的一半。返回的b和a为n+1 长度的滤波器系数, ftype 为该滤波器的类型， 低通还是带通还是高通。
3. 滤波器函数设计、验证、使用步骤
    1. 设计：通常为直接调用特定滤波器的函数，如[b,a]=butter(n,Wn,'ftype')，就得到了滤波器系数b和a。
    2. 验证：我们可以直接调用[h,f]=freqz(b,a,512,fs)获取该滤波器的frequency response，并且画图观察是否符合我们的要求。
    3. 使用：filter(b,a,x) 来进行对于信号x的滤波。
4. 包络提取原理
    1. 先进行全波整流
    2. 再通过低通滤波器
5. 言语谱噪声产生原理
    1. 首先生成随机在-1到1分布的随机信号（初始噪音）
    2. 根据言语能量在频域上分布生成相应的滤波器
    3. 将初始噪音通过语谱滤波器
6. 能量归一化原理
    1. norm 可以计算出$\sqrt{E}$ 能量的开方
    2. $x * norm(y)/norm(x)$ 即可使得x 与 y 能量归一化

Task 1

（一）、问题重述

1、 将低通滤波器截止频率设置为50Hz

2、 通过将频段数更改为 N = 1、2、4、6 和 8 来进行滤波

3、 保存这些条件的波形文件，并描述波段数如何影响合成语音的可理解性（即可以理解多少个单词）

（二）、处理方案：

1、设置函数将语音信号经过带通滤波器，全波整流过低通滤波器，与载波信号相乘最后叠加得到语音信号

2、通过计算处理信号与原信号的RMSE（均方根误差），信噪比SNR，波形相似参数NCC

$$NCC = \frac{\sum_{n=1}^{N} As(n)Ad(n)}{\sqrt{(\sum_{n=1}^{N} As^2(n))(\sum_{n=1}^{N} Ad^2(n))}}$$

$Figure.01\ NCC$(波形相似参数)公式

$$SNR = \frac{P_{signal}}{P_{noise}} = \frac{A_{signal}^2}{A_{noise}^2}$$

$Figure.02\ SNR$(信噪比)公式

$$X_{rms} = \sqrt{\frac{\sum_{i=1}^{N} X_i^2}{N}} = \sqrt{\frac{X_1^2 + X_2^2 + \cdots + X_N^2}{N}}$$

$Figure.03\ RMSE$(均方根误差)公式

3、将人体耳蜗每段纤毛处理成相同长度，再通过f与d转化公式，将对应频率转化成一定长度的线段投射到耳蜗上去处理；目前采用的方法是将这段线段等分，每段等分的线段取其首尾频率再输入滤波器当中处理

4、相对应而在 stone-vocoder 方法里只是简单地将要处理的频率范围等分，直接输入滤波器当中处理，我们通过改变 N 的取值来探求3,4两种模拟方法下最佳 N 的范围以及其恢复信号的质量：

（三）、具体代码实现

1、函数dist_freq

N为带通滤波器数量，[output]为频率分段

```
function [output]=dist_freq(N)%output为频率
%人体耳蜗的处理频率在200~7000Hz，根据公式计算得到人体耳蜗的长度min与max
distmin=(log10(200/165.4+1))/0.06;
distmax=(log10(7000/165.4+1))/0.06;
dist_divd=(distmax-distmin)/N;
output(1)=200;
output(N+1)=7000;
for x=1:N
    output(x)=165.4*(10^(0.06*(distmin+dist_divd*(x-1)))-1);
end
end
```

2、函数tone_vocoder

N为带通滤波器数量，y为采样时域信号，fq为采样率，fcut为低通滤波器截止频率；设置巴特沃斯滤波器为4阶，将dist_freq函数的输出输入带通滤波器，全波整流，再过低通滤波器；设置载波信号，最后得到语音信号

```
function op=tone_vocoder(N,y,fq,fcut)%输出语音信号
%处理整个语音信号，首先需要设置带通滤波器，对滤波后的信号做整流和过低通，再与载波信号相乘，最后叠加
[a0,b0]=butter(4,fcut/(fq/2));
%设置巴特沃斯滤波器为4阶
[c]=dist_freq(N);
t=1:size(y,1);
op=0;
for i=1:N
    [a1,b1]=butter(4,[c(i) c(i+1)]/(fq/2));%设置一个带通滤波器
    output=filter(a1,b1,y);%过带通
    output_rect=abs(output);%全波整流
    %而后过低通
    output_rect_lpf=filter(a0,b0,output_rect);
    %设置一个正弦载波信号，频率等于带通的中间频率
    mid_freq=(c(i)+c(i+1))/2;
    sinewave=sin(2*pi*t*mid_freq*1/fq);
    op=op+output_rect_lpf.*sinewave';
end
%能量归一化
op=op*norm(y)/norm(op);
end
```

3、函数freq_tone_vocoder

```
function op=freq_tone_vocoder(N,y,fq,fcut)%输出语音信号
%处理整个语音信号，首先需要设置带通滤波器，对滤波后的信号做整流和过低通，再与载波信号相乘，最后叠加
[a0,b0]=butter(4,fcut/(fq/2));
%设置巴特沃斯滤波器为4阶
```

```
[c]=linspace(200,7000,N+1);%与tone_vocoder函数的区别
t=1:size(y,1);
op=0;
for i=1:N
    [a1,b1]=butter(4,[c(i) c(i+1)]/(fq/2));%设置一个带通滤波器
    output=filter(a1,b1,y);%过带通
    output_rect=abs(output);%全波整流
    %而后过低通
    output_rect_lpf=filter(a0,b0,output_rect);
    %设置一个正弦载波信号，频率等于带通的中间频率
    mid_freq=(c(i)+c(i+1))/2;
    sinewave=sin(2*pi*t*mid_freq*1/fq);
    op=op+output_rect_lpf.*sinewave';
end
%能量归一化
op=op*norm(y)/norm(op);
end
```

4、主函数

画图得到时域与频域上信号的可视化

```
clc;
clear;
%现在设置带阻滤波器的个数为1,2,4,6,8
%200Hz到7000Hz，distance to place公式为
%f=165.4*(10^(0.06d)-1)
%先尝试将distance等分的情况
[y,fs]=audioread("C_01_02(2).wav");
fcut=50;

out1=tone_vocoder(1,y,fs,fcut);
out2=tone_vocoder(2,y,fs,fcut);
out3=tone_vocoder(4,y,fs,fcut);
out4=tone_vocoder(6,y,fs,fcut);
out5=tone_vocoder(8,y,fs,fcut);
out6=y;

subplot(6,1,1);
plot(out1);title("N=1时域图");xlabel('t(s)');ylabel('value');legend('out1,N=1');
subplot(6,1,2);
plot(out2);title("N=2时域图");xlabel('t(s)');ylabel('value');legend('out2,N=2');
subplot(6,1,3);
plot(out3);title("N=4时域图");xlabel('t(s)');ylabel('value');legend('out3,N=4');
subplot(6,1,4);
plot(out4);title("N=6时域图");xlabel('t(s)');ylabel('value');legend('out4,N=6');
subplot(6,1,5);
plot(out5);title("N=8时域图");xlabel('t(s)');ylabel('value');legend('out5,N=8');
subplot(6,1,6);
plot(out6);title("Origin");xlabel('t(s)');ylabel('value');legend('origin');
%out1~out6都是时域图，接下来对比频域图
out6=out6';
out6_freq=fftshift(fft(out6,length(out6)));
w=linspace(-fs/2,fs/2,length(y));
figure;
```

```
subplot(6,1,6);
plot(w,abs(out6_freq));title("origin频域
图");xlabel('w(Hz)');ylabel('value');legend('origin');
out1=out1';
out1_freq=fftshift(fft(out1,length(out1)));
subplot(6,1,1);
plot(w,abs(out1_freq));title("out1_freq频域
图");xlabel('w(Hz)');ylabel('value');legend('out1_freq');
out2=out2';
out2_freq=fftshift(fft(out2));
subplot(6,1,2);
plot(w,abs(out2_freq));title("out2_freq频域
图");xlabel('w(Hz)');ylabel('value');legend('out2_freq');
out3=out3';
out3_freq=fftshift(fft(out3));
subplot(6,1,3);
plot(w,abs(out3_freq);title("out3_freq频域
图");xlabel('w(Hz)');ylabel('value');legend('out3_freq');
out4=out4';
out4_freq=fftshift(fft(out4));
subplot(6,1,4);
plot(w,abs(out4_freq));title("out4_freq频域
图");xlabel('w(Hz)');ylabel('value');legend('out4_freq');
out5=out5';
out5_freq=fftshift(fft(out5));
subplot(6,1,5);
plot(w,abs(out5_freq));title("out5_freq频域
图");xlabel('w(Hz)');ylabel('value');legend('out5_freq');

filename1="OUT1_c1.wav";
filename2="OUT2_c1.wav";
filename3="OUT3_c1.wav";
filename4="OUT4_c1.wav";
filename5="OUT5_c1.wav";

audiowrite(filename1,out1,fs);
audiowrite(filename2,out2,fs);
audiowrite(filename3,out3,fs);
audiowrite(filename4,out4,fs);
audiowrite(filename5,out5,fs);
```

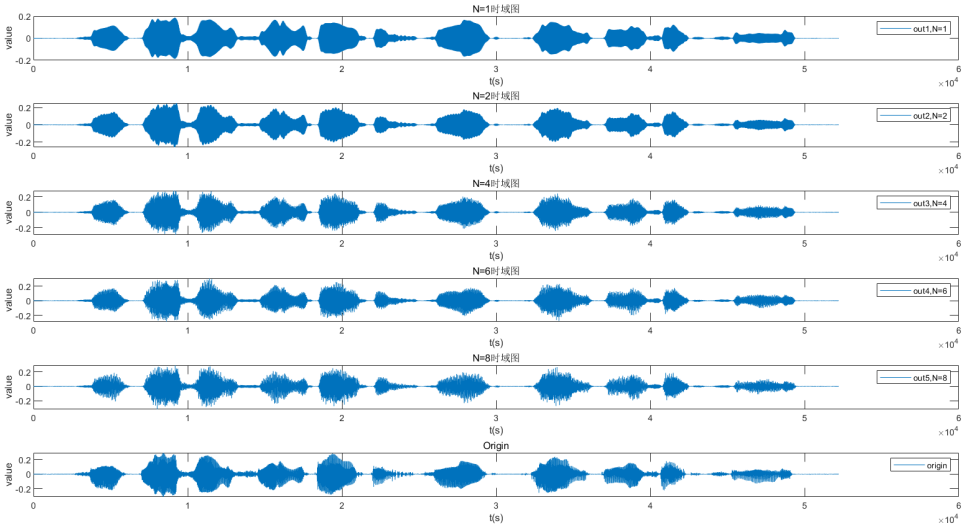


Figure.04 第一个音频的时域图（使用$tone-vocoder$）

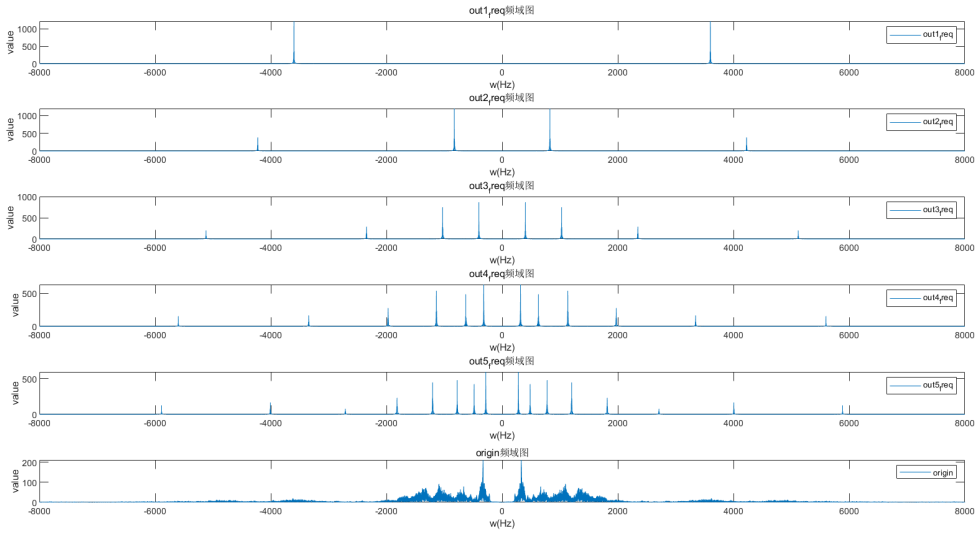


Figure.05 第一个音频的频域图

同理，对第二个音频做同样的操作，得到时域与频域的图

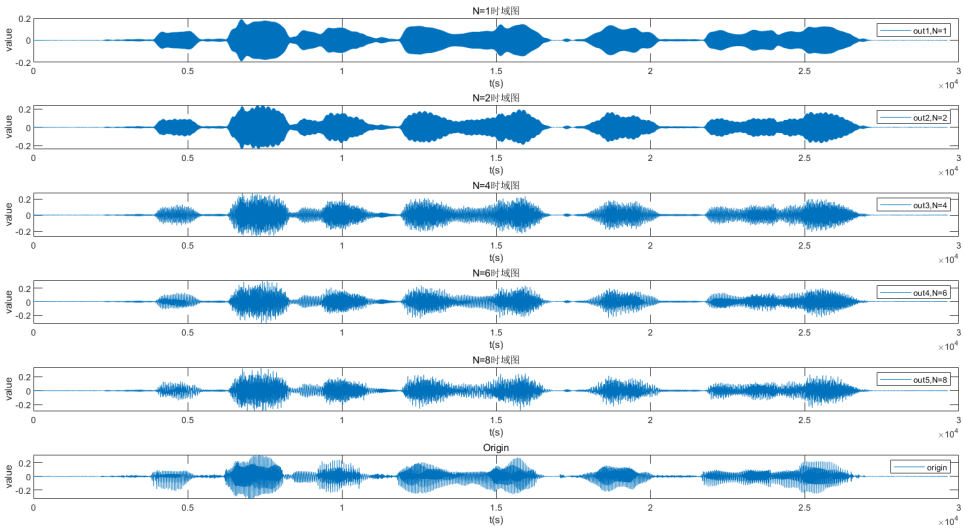


Figure.06 第二个音频的时域图

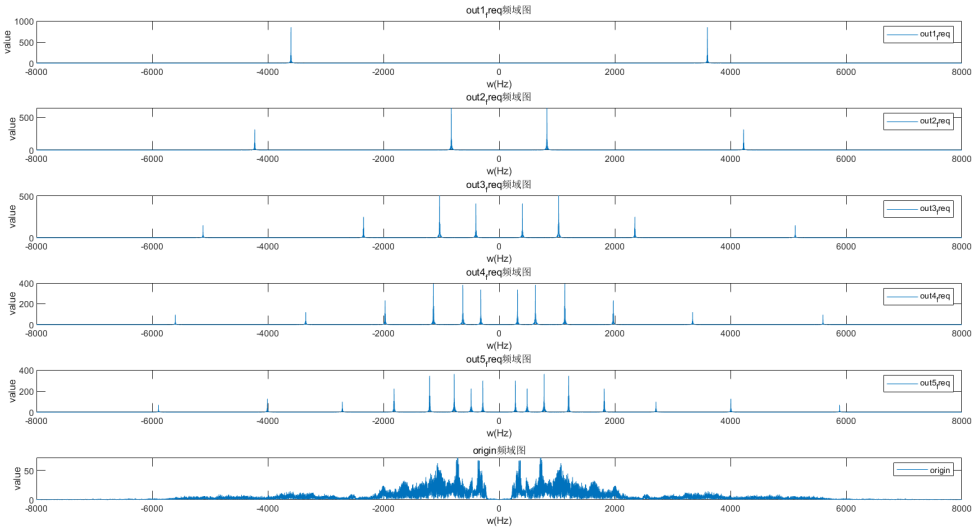


Figure.07 第二个音频的频域图

当使用freq_tone_vocoder时，同样画出时域图与频域图

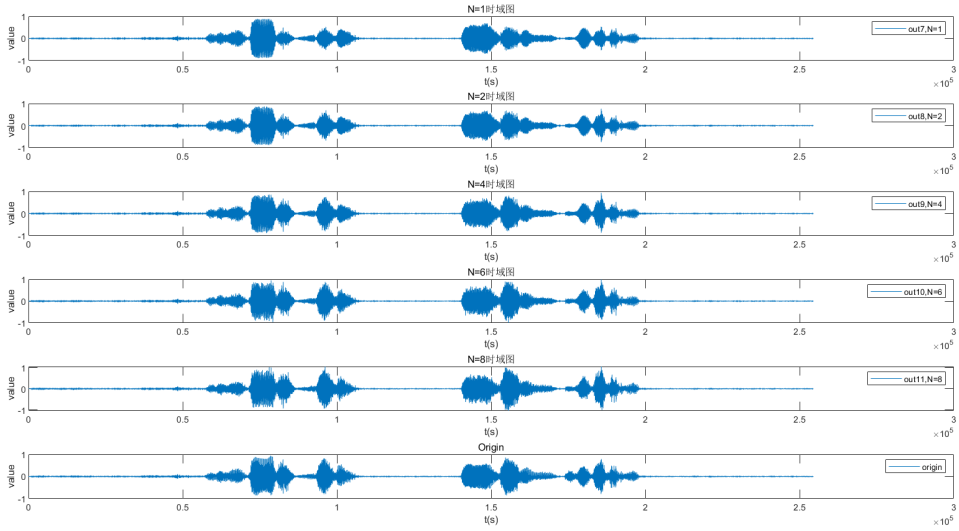


Figure.08 将频率范围等分后处理信号的时域图

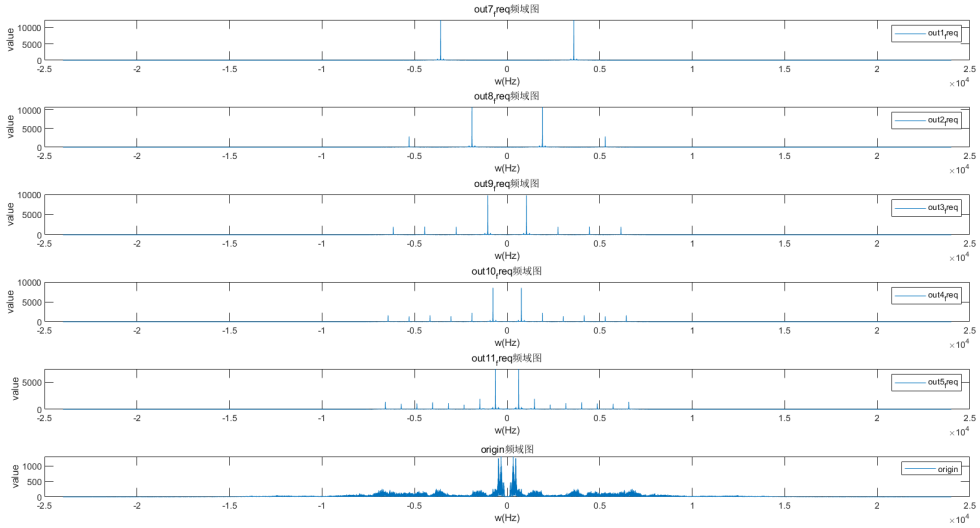


Figure.08 将频率范围等分后处理信号的频域图

5、分析

通过人耳听音可知，恢复音频效果随着 N 的增加逐渐变得清晰，在 N=1,2,4 时，恢复的音效完全无法被识别，当 N=6,8时恢复的音效大致可以被理解。

6、结论

随着 N 的增加，频域成分变多，处理信号逐渐愈来愈像原声；声音合成效果变好，音频可理解性增加。

（四）Task 1 拓展

1、继续增加N的大小，是否能够得到更清晰，能够辨认的信号?

画出N=10,25,50,100,200,350,500与原信号的频谱图和语谱图

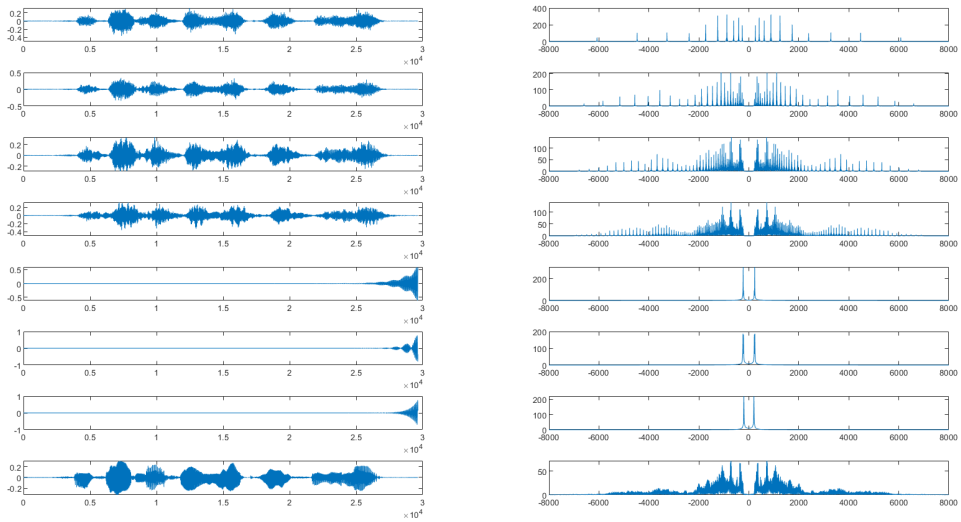


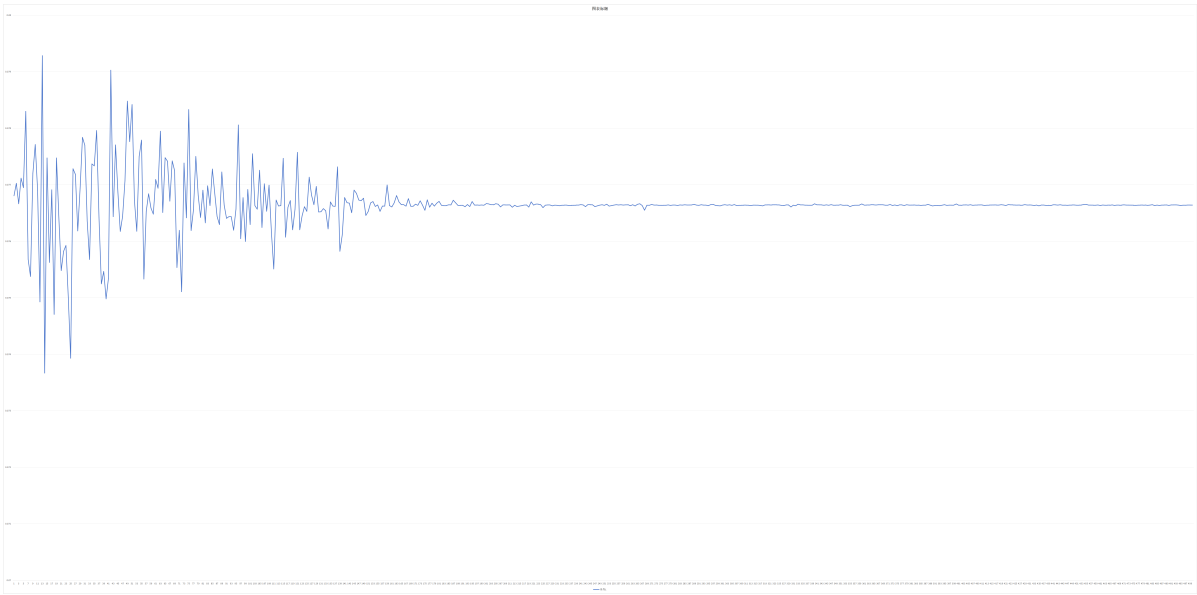
Figure.07 时域图（左）, 频域图（右）

分析：当N刚开始升高，越来越接近原信号；而当增加到一定程度由于滤波器变形导致处理信号完全与原信号不一致

数据（一）：RMSE均方根误差

根据定义，RMSE越接近0，说明处理信号与原信号越接近

```
function output=rmse(N,y,fq,fcut)
out=tone_vocoder(N,y,fq,fcut);
similarity=0;
for i=1:29636
    similarity=similarity+(out(i)-y(i)).*(out(i)-y(i));
end
similarity=similarity/29636;
similarity=sqrt(similarity);
similarity=abs(similarity);
output=similarity;
end
```



$Figure.08$ $RMSE$在不同带通滤波器数量下的大小

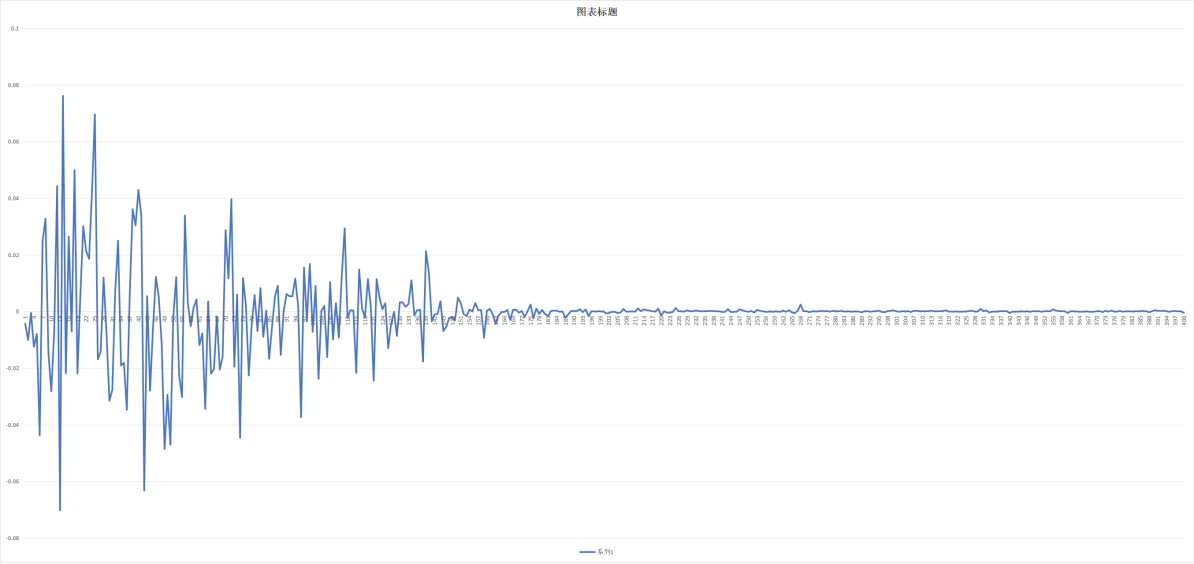
分析：可以看到RMSE的值在0.7上下震荡

结论：RMSE对于处理信号与原信号比较方面作用不大

数据（二）： NCC波形相似参数

```
function output=ncc(y1,y2)
As=0;
Ad=0;Aa=0;
for i=1:29636
   As=As+y1(i).*y1(i);
   Ad=Ad+y2(i).*y2(i);
   Aa=Aa+y1(i).*y2(i);
end
output=abs(Aa/sqrt(As*Ad));
end
```

根据定义，NCC越解决1，处理信号越接近原信号）



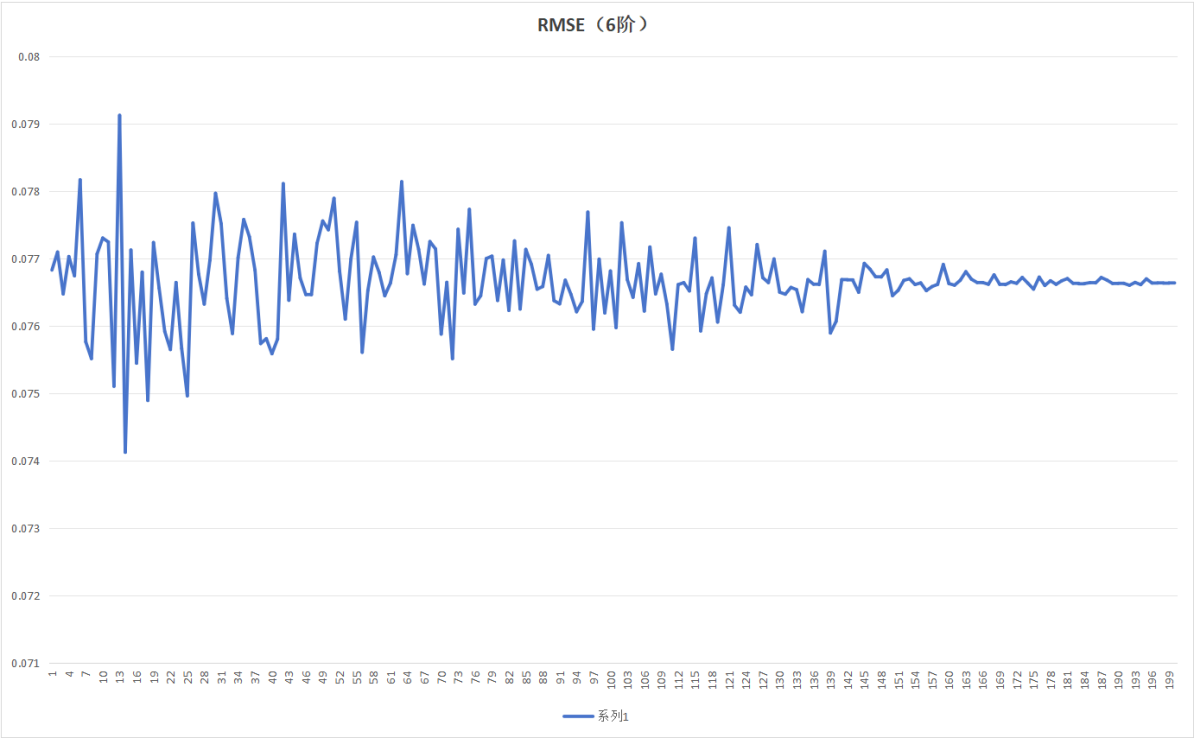
$Figure.09$ NCC在不同带通滤波器数量下的大小

分析：可以明显得看出，当N越来越大时，波形与原信号完全不相似；还原出来的信号也难以识别

结论：当相应的带通数量N增加时，恢复信号会更加清晰，但同时体积会减小，噪声也会变得越来越明显；但随着N越来越到，滤波器会随着 N 的增加而变形，效果。所以理想的通带数量范围为 N ≈100。

2、巴特沃斯滤波器阶数的改变

设置巴特沃斯滤波器阶数变为6



$Figure.11$ $RMSE$在不同低通滤波器阶数下的大小

分析：与4阶变化不大，而由于阶数增大导致滤波器所需资源增大，电脑已无法计算出8阶的数据

结论：巴特沃斯滤波器阶数对语音信号识别作用不大

3、自主录制音频，验证人工耳蜗可行性

录制单声道音频“exp.wav”（内容为“明德求是，日新自强”），采样率为48000，设置截止频率为500Hz，同样设置设置带通滤波器数量为N=1,2,4,6,8

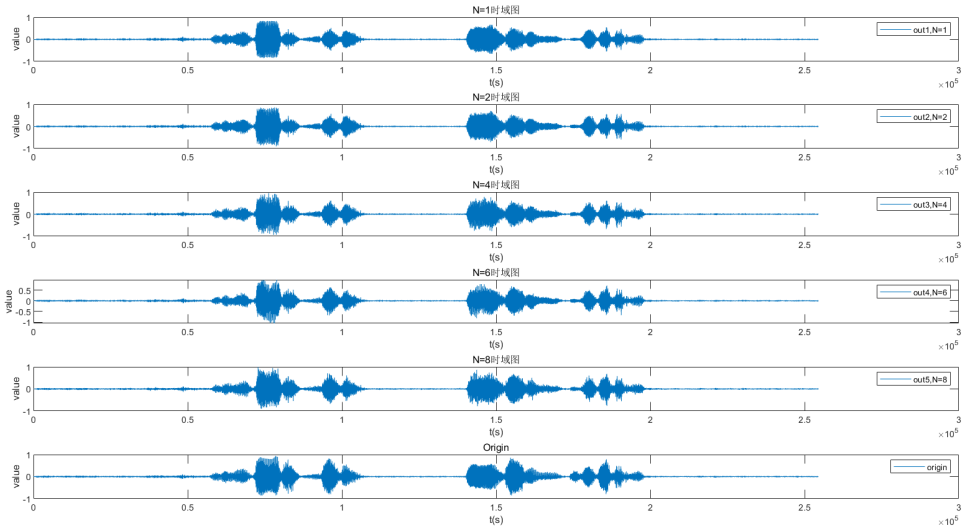


Figure.12 自主录制音频经过处理后的时域图

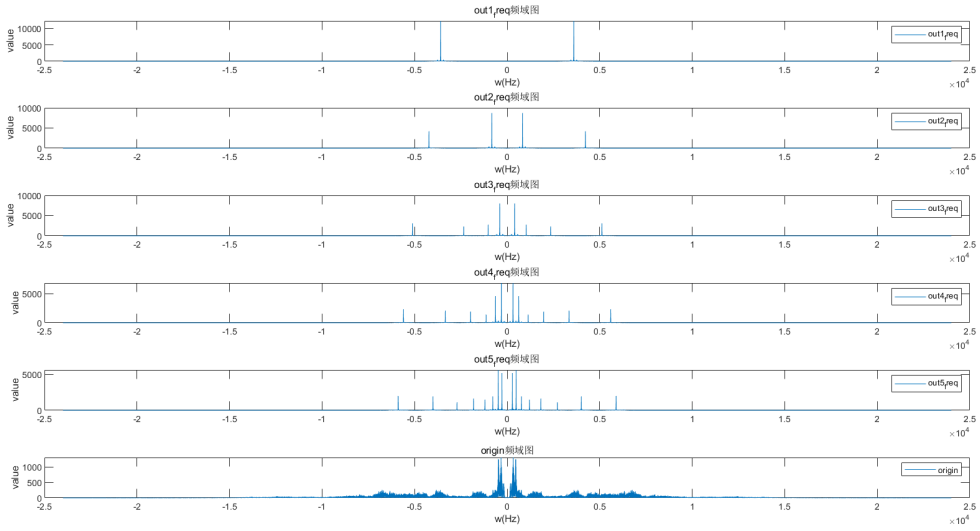


Figure.13 自主录制音频经过处理后的频域图

分析：从时域图上来看，处理后的信号与原信号相差不多；频域上缺少较多；通过人耳识别，当带通滤波器数量N=2,4,6,8时可以辨别，而N=1根本无法识别

结论：与project原本提供的音频相比，exp.wav采样率比其高出3倍，可以弥补一些因带通滤波器滤过的高频与低频信号的缺失。

Task2

题目要求：

1.将带数设置为4

2.调整截止频率分别为20 赫兹，50 赫兹，100 赫兹 和 400 赫兹

3.探究滤波器的截止频率怎样影响还原效果

处理方案

依据要求分别将低通滤波器设置为20 赫兹，50 赫兹，100 赫兹 和 400 赫兹。运用函数tone_vocoder（同task1），滤波得到包络，生成一个正弦波，并将二者乘起来，得到还原信号。

函数tone_vocoder

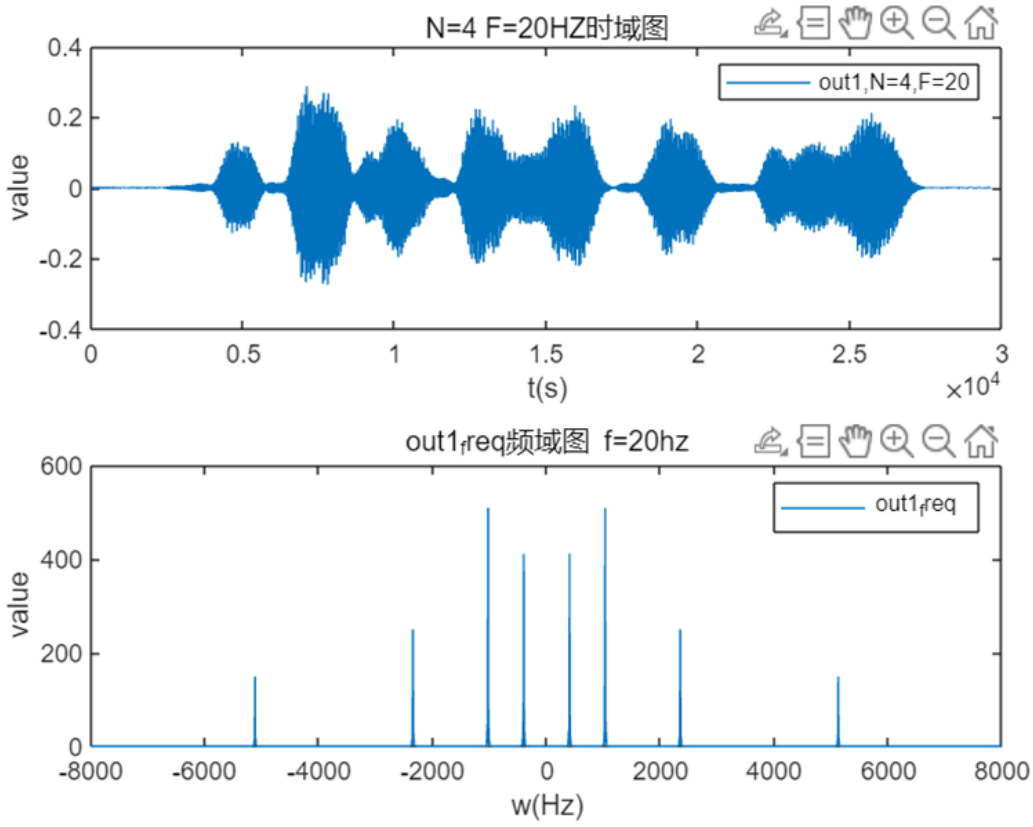
```
function op=tone_vocoder(N,y,fq,fcut)%输出语音信号
%处理整个语音信号，首先需要设置带通滤波器，对滤波后的信号做整流和过低通，再与载波信号相乘，最后叠加
[a0,b0]=butter(4,fcut/(fq/2));
%设置巴特沃斯滤波器为4阶
[c]=dist_freq(N);
t=1:size(y,1);
op=0;
for i=1:N
[a1,b1]=butter(4,[c(i) c(i+1)]/(fq/2));%设置一个带通滤波器
output=filter(a1,b1,y);%过带通
output_rect=abs(output);%全波整流
%而后过低通
output_rect_lpf=filter(a0,b0,output_rect);
%设置一个正弦载波信号，频率等于带通的中间频率
mid_freq=(c(i)+c(i+1))/2;
sinewave=sin(2*pi*t*mid_freq*1/fq);
op=op+output_rect_lpf.*sinewave';
end
%能量归一化
op=op*norm(y)/norm(op);
end
```

主函数

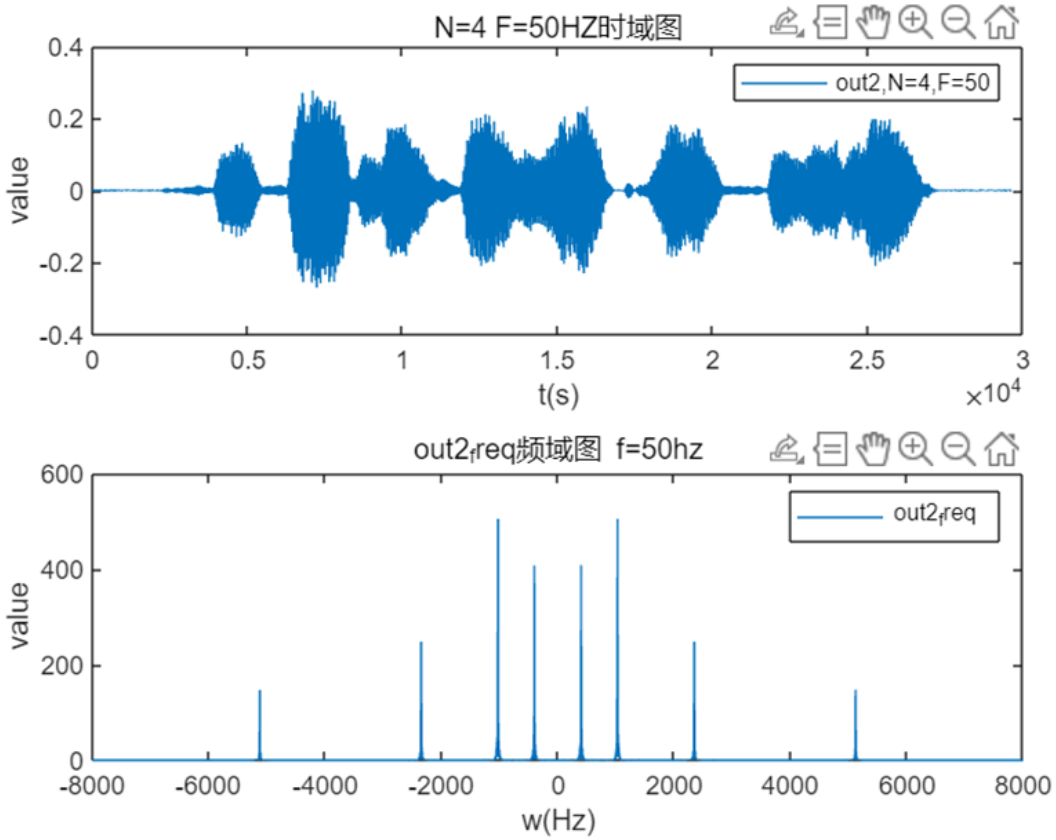
```
clc;
clear;
[y,fs]=audioread("C_01_02(2).wav");

out1=tone_vocoder(4,y,fs,20);
out2=tone_vocoder(4,y,fs,50);
out3=tone_vocoder(4,y,fs,100);
out4=tone_vocoder(4,y,fs,400);
out5=y;
w=linspace(-fs/2,fs/2,length(y));

figure
```

```
subplot(2,1,1),plot(out1);title("N=4 F=20HZ时域
图");xlabel('t(s)');ylabel('value');legend('out1,N=4,F=20');
out1=out1';
out1_freq=fftshift(fft(out1,length(out1)));
subplot(2,1,2),plot(w,abs(out1_freq));title("out1_freq频域图
f=20hz");xlabel('w(Hz)');ylabel('value');legend('out1_freq');

figure
subplot(2,1,1),plot(out2);title("N=4 F=50HZ时域
图");xlabel('t(s)');ylabel('value');legend('out2,N=4,F=50');
out2=out2';
out2_freq=fftshift(fft(out2,length(out2)));
subplot(2,1,2),plot(w,abs(out2_freq));title("out2_freq频域图
f=50hz");xlabel('w(Hz)');ylabel('value');legend('out2_freq');

figure
subplot(2,1,1),plot(out3);title("N=4 F=100HZ时域
图");xlabel('t(s)');ylabel('value');legend('out3,N=4,F=100');
out3=out3';
out3_freq=fftshift(fft(out3,length(out1)));
subplot(2,1,2),plot(w,abs(out3_freq));title("out3_freq频域图
f=100hz");xlabel('w(Hz)');ylabel('value');legend('out3_freq');

figure
subplot(2,1,1),plot(out4);title("N=4 F=400HZ时域
图");xlabel('t(s)');ylabel('value');legend('out4,N=4,F=400');
out4=out4';
out4_freq=fftshift(fft(out4,length(out4)));
subplot(2,1,2),plot(w,abs(out4_freq));title("out4_freq频域图
f=400hz");xlabel('w(Hz)');ylabel('value');legend('out4_freq');

figure
subplot(2,1,1),plot(out5);title("origin时域
图");xlabel('t(s)');ylabel('value');legend('origin');
out5=out5';
out5_freq=fftshift(fft(out5,length(out5)));
subplot(2,1,2),plot(w,abs(out5_freq));title("origin频域
图");xlabel('w(Hz)');ylabel('value');legend('origin');
%由图可知截止频率越高还原效果越好
```

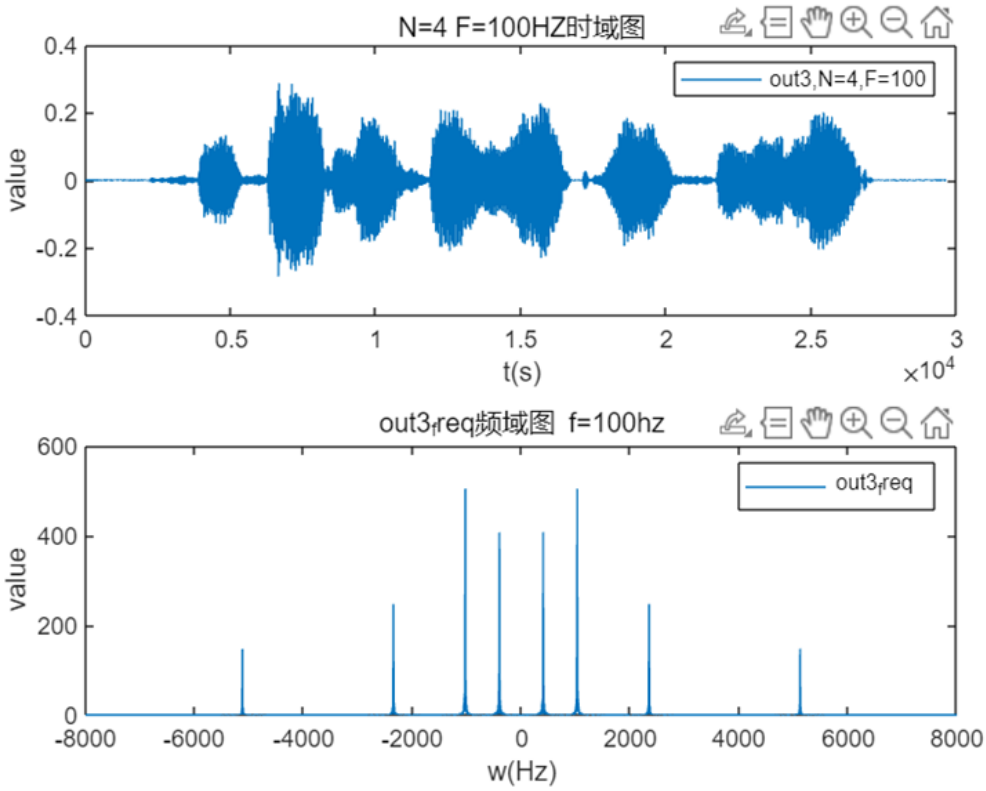
时域与频域图对比如下：



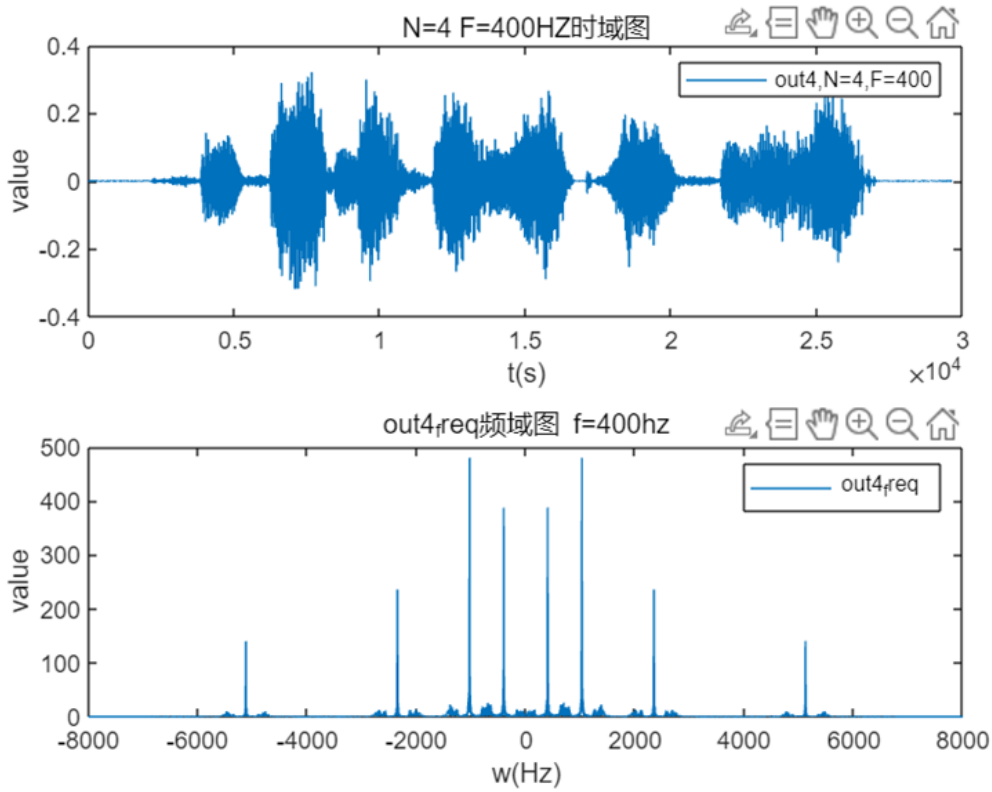
$Figure01.\ N=4\ f=20Hz$时域&频域图



$Figure02.\ N=4\ f=50Hz$时域&频域图



$Figure03.\ N=4\ f=100Hz$时域&频域图



$Figure04.\ N=4\ f=400Hz$时域&频域图

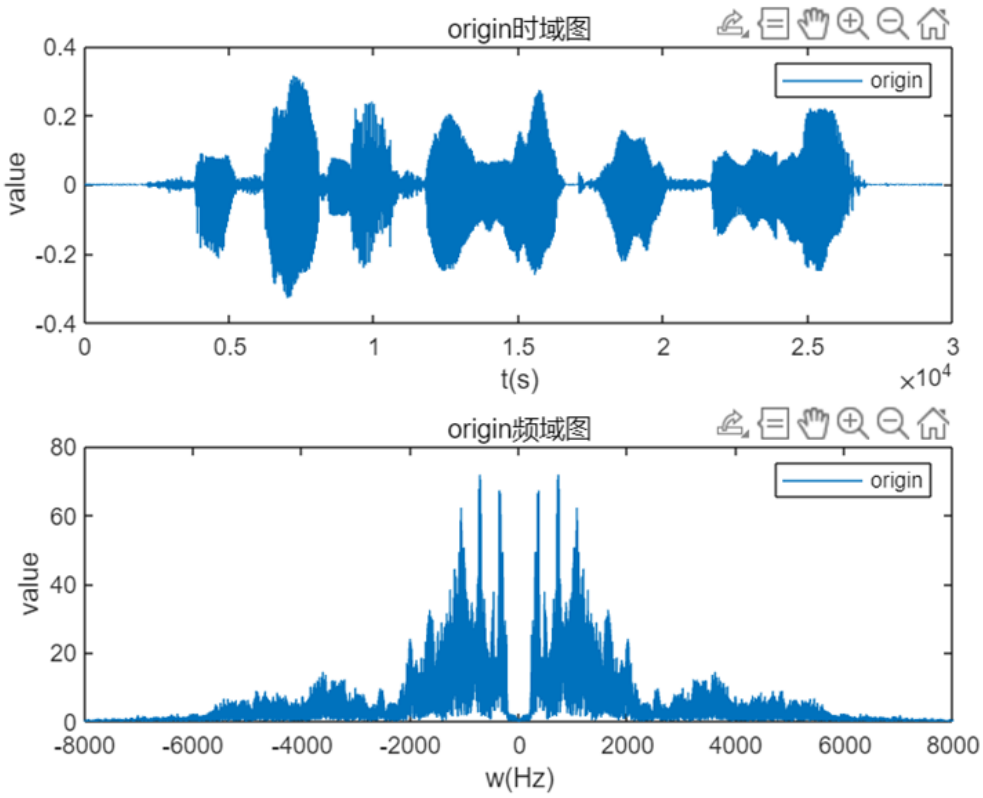


Figure05.Origin时域&频域图

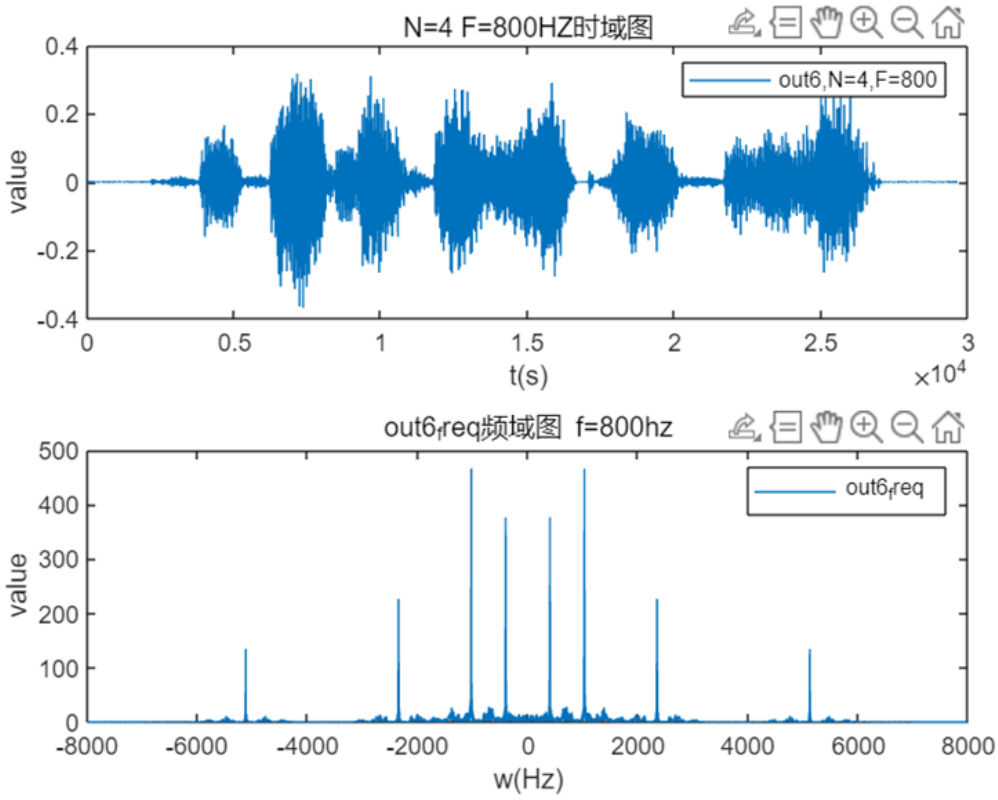
当截止频率变高时，包络会增加，这会提高语音的还原效果，听音可知只有400Hz时大概能听清原始信号。

结论：截止频率越高还原效果越好

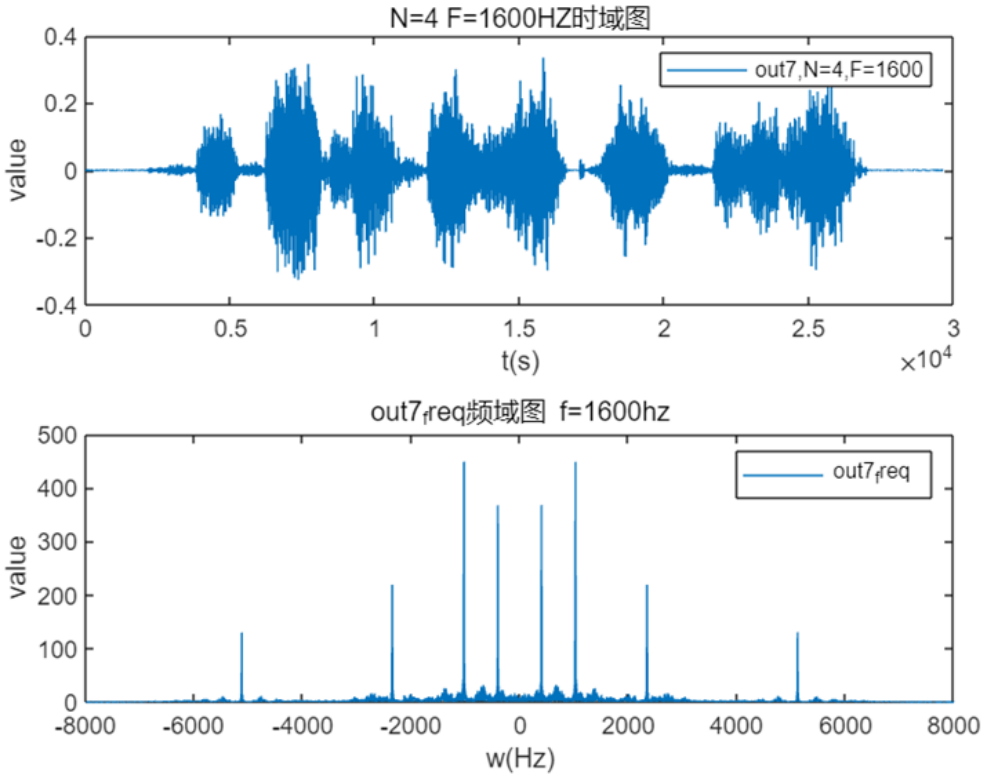
拓展

提高截止频率，测试是否能提高还原效果

```
out6=tone_vocoder(4,y,fs,800);
out7=tone_vocoder(4,y,fs,1600);
figure
subplot(2,1,1),plot(out6);title("N=4 F=800HZ时域
图");xlabel('t(s)');ylabel('value');legend('out6,N=4,F=800');
out6=out6';
out6_freq=fftshift(fft(out6,length(out6)));
subplot(2,1,2),plot(w,abs(out6_freq));title("out6_freq频域图
f=800hz");xlabel('w(Hz)');ylabel('value');legend('out6_freq');

figure
subplot(2,1,1),plot(out7);title("N=4 F=1600HZ时域
图");xlabel('t(s)');ylabel('value');legend('out7,N=4,F=1600');
out7=out7';
out7_freq=fftshift(fft(out7,length(out7)));
subplot(2,1,2),plot(w,abs(out7_freq));title("out7_freq频域图
f=800hz");xlabel('w(Hz)');ylabel('value');legend('out7_freq');
```



$Figure06.\ N=4\ f=800Hz$时域&频域图



$Figure07.\ N=4\ f=1600Hz$时域&频域图

播放声音可知，效果并不如意，截至频率过高会引入杂音，对比来说400hz的声音最清楚。

Task 3

（一）、问题重述

1、产生信噪比为-5dB的含噪信号

2、 将低通滤波器截止频率设置为50Hz

3、 通过将频段数更改为 N = 2、4、6、8 和 16 来进行滤波

4、 保存这些条件的波形文件，并描述波段数如何影响合成语音的可理解性（即可以理解多少个单词）

（二）、处理方案：

1、设置函数产生含噪信号，将语音信号经过带通滤波器，全波整流过低通滤波器，与载波信号相乘最后叠加得到语音信号

2、通过计算处理信号与原信号的RMSE（均方根误差），信噪比SNR，波形相似参数NCC

SNR:

$$SNR = 10\log_{10}(E_{signal}/E_{noise}) = 20\log_{10}(\sqrt{E_{signal}}/\sqrt{E_{noise}})$$

NCC: (As指滤波前信号的幅值，Ad指滤波后信号的幅值)

$$NCC = \frac{\sum_{n=1}^{N} As(n)Ad(n)}{\sqrt{(\sum_{n=1}^{N} As^2(n))(\sum_{n=1}^{N} Ad^2(n))}}$$

RMSE:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} |A_{signal} - A_{denoised}|^2}{N}}$$

3、将人体耳蜗每段纤毛处理成相同长度，再通过f与d转化公式，将对应频率转化成一定长度的线段投射到耳蜗上去处理；目前采用的方法是将这段线段等分，每段等分的线段取其首尾频率再输入滤波器当中处理

（三）、具体代码实现

1、函数GenerateSSN

SSN为语谱噪声，signal为语音信号，fs为语音信号的采样频率，nfft为fft数据点个数，noverlap为每个窗口的重叠长度

```
function SSN = GenerateSSN(signal, fs, nfft, noverlap)
    %generate white noise
    Len = size(signal,1);
    noise = 1-2*rand(1,Len);
    %using hamming windows
    windows = hamming(nfft);
    %PSD estimate
    [Pxx,w] = pwelch(signal,windows,noverlap,nfft,fs);
    %get the filter
    b = fir2(3000,w/(fs/2),sqrt(Pxx/max(Pxx)));
    %generate SSN
    SSN = filter(b,1,noise);
end
```

2、函数GenerateNoisySignal

noisySignal为含噪信号，signal为语音信号，ssn为语谱噪声，snr为信噪比

```
function noisySignal = GenerateNoisySignal(signal, ssn, snr)
    %calculate variable
    syms E;
    E = double(solve(20*log10(norm(signal)/E)==snr,E));
    %get the required ssn
    ssn = ssn/norm(ssn)*E;
    %add ssn into origin signal
    noisySignal = signal.' + ssn;
end
```

3、函数Split_N_freq

split_freq为模拟耳蜗的等距长度分割对应N个频率划分，freL为划分的最低频率，freH为划分的最高频率，N为划分的个数

```
function split_freq = Split_N_freq(freL, freH, N)
    syms dL dH;
    %solve the d for low frequency and high frequency
    dL = double(solve(165.4*(10^(0.06*dL)-1)==freL));
    dH = double(solve(165.4*(10^(0.06*dH)-1)==freH));
    %divide distance into N segments
    d = linspace(dL,dH,N+1);
    %divide frequency from freL to freH into N segments according to dist
    split_freq = 165.4.*(10.^(0.06.*d)-1);
end
```

4、函数toneVocoder

syn_signal为模拟耳蜗处理后的语音信号，noisySignal为含噪语音信号，fs为采样频率，N为带通滤波器的的个数，freCut为低通滤波器的截止频率

```
function syn_signal = toneVocoder(noisySignal, fs, N, freCut)
    %configuration
    freL = 200;
    freH = 7000;
```

```
    Len = length(noisySignal);
    BPorder = 4;
    LPorder = 6;

    s = zeros(1,length(noisySignal));

    split_freq = Split_N_freq(freL, freH, N);


    for i = 1 : N
        %design band-pass filter at band i
        [b, a] = butter(BPorder, [split_freq(i),split_freq(i+1)]/(fs/2));
        %do band-pass filtering at band i
        y = filter(b, a, noisySignal);
        %extract envelope
        envelope = extractEnvelope(y, fs, freCut, LPorder);
        %generate a sinewave as carrier
        freMidBand = (split_freq(i) + split_freq(i+1))/2;
        t = 0:1/fs:(Len-1)/fs;
        carrier = sin(2*pi*freMidBand.*t);
        %modulation
        modulatedSignal = envelope .* carrier;
        s = s + modulatedSignal;
    end

    %energy normalization
    syn_signal = s / norm(s) * norm(noisySignal);
end
```

5.函数Task3

```
clc, clear all;
%load signal
[signal,fs] = audioread('C_01_01.wav');%以C_01_01.wav为例
%configuration
freCut = 50;
snr = -5;
nfft = 512;
noverlap = nfft/2;
Len = length(signal);
t = 0:1/fs:(Len-1)/fs;
L = (1-Len)/Len/2*fs;
R = (Len-1)/Len/2*fs;
range = [2 4 6 8 16];

%generate noisySignal
ssn = GenerateSSN(signal, fs, nfft, noverlap);
noisySignal = GenerateNoisySignal(signal, ssn, snr);

%change number of bands from 2 to 16, step=2
fig=figure();
    subplot(length(range)+1,2,1);plot(t,noisySignal),xlabel('t
(s)'),ylabel('time-domain Wave'),xlim([0,(Len-1)/fs]);
    title("Original signal with ssn");
```

```
 subplot(length(range)+1,2,2);plot(linspace(L,R,Len),abs(fftshift(fft(noisySignal
))),xlabel('freq (Hz)'),ylabel('Magtitude');
    title("Original signal with ssn");
%save fig
for i = 1:1:length(range)
    N = range(i);
    syn_signal = toneVocoder(noisySignal, fs, N, freCut);
    %plot time-domain wave
    subplot(length(range)+1,2,2*i+1);plot(t,syn_signal),xlabel('t
(s)'),ylabel('time-domain Wave'),xlim([0,(Len-1)/fs]);
    title("N = " + num2str(N));
    %plot frequency-domain wave

 subplot(length(range)+1,2,2*i+2);plot(linspace(L,R,Len),abs(fftshift(fft(syn_sig
nal))),xlabel('freq (Hz)'),ylabel('Magtitude');
    title("N = " + num2str(N));
end
%save fig
saveas(fig, 'results','jpg')
```

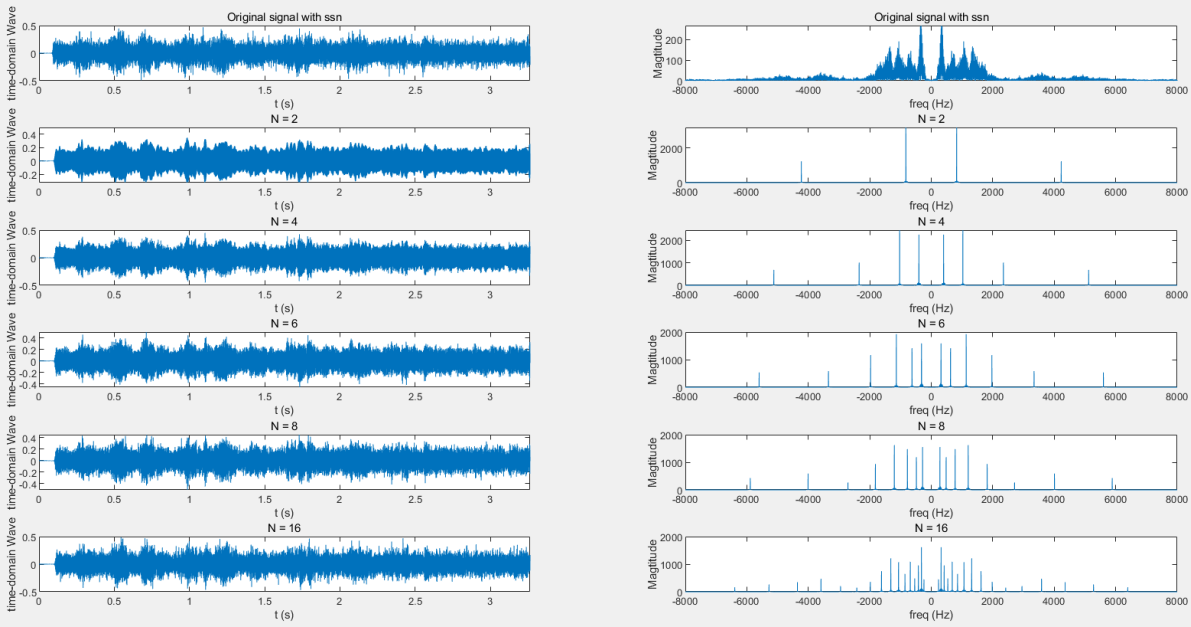


fig.1实验结果图(时域 + 频域)

5、分析

随着 N 的增大，复原信号在时域图上与原始含噪信号没有明显差别。但在频域图上，可以明显观察到复原信号保留的频率分量增多，与原始含噪信号的相似度增加

6、结论

与 task1 相对比，可以看出：带有语谱噪声的混合语音信号复原清晰度明显降低，语音信号复原效果明显变差。随着通频带数 N 的增大，复原信号的质量仅有微弱提升，即使当 N=16 时，也无法听清语音信号的内容。

（四）Task 3 拓展

1、继续增加N的大小，是否能够得到更清晰，能够辨认的信号?

图像

画出N=16,32,64,128,144,175,200,256,300,512与原信号的时域图和频域图

```
range1 = [16 32 64 128 144];
range2 = [175 200 256 300 512];
```

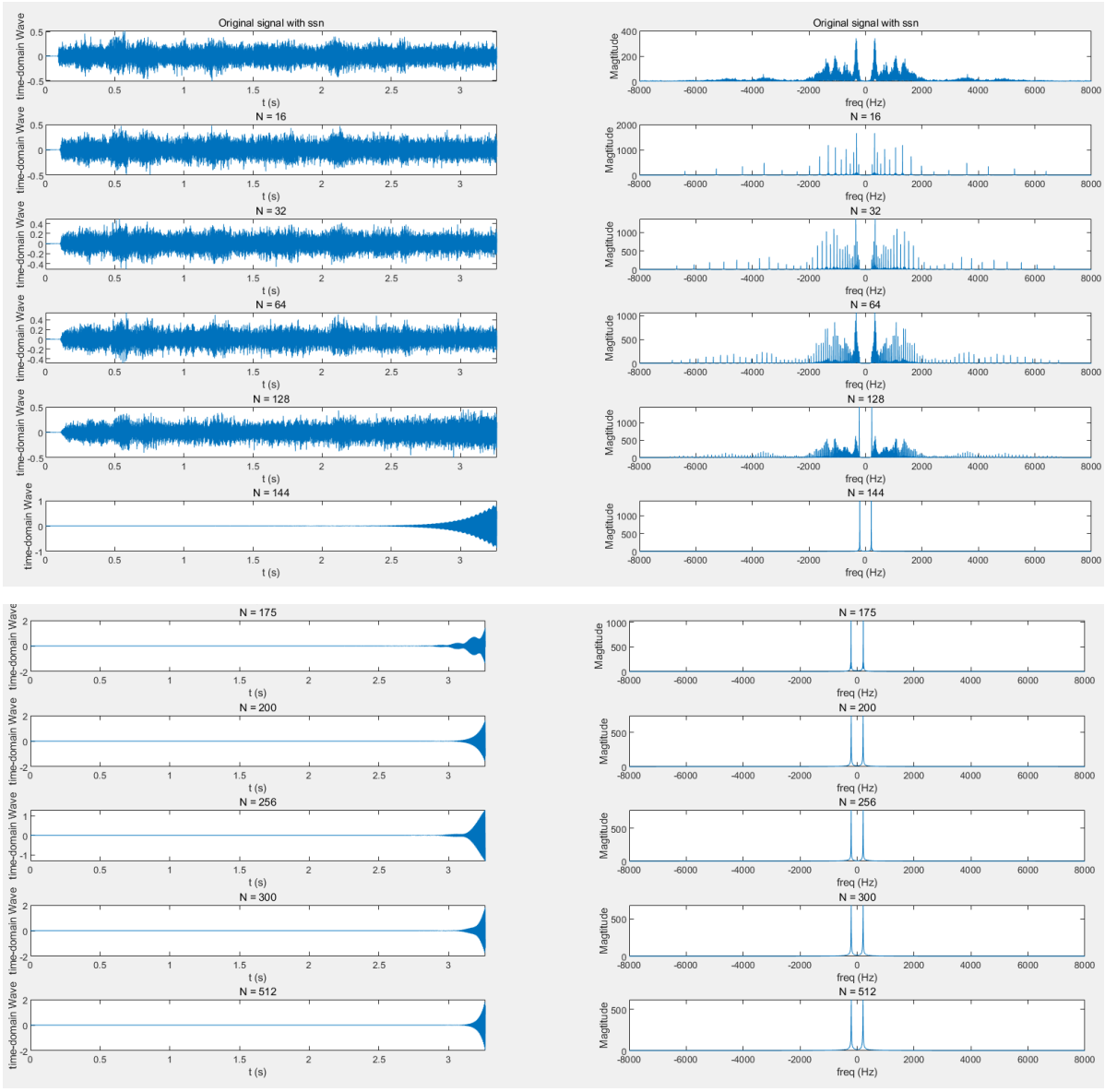


*fig.2*时域图（左），频域图（右）

分析：当N刚开始升高，越来越接近原信号；而当增加到一定程度由于滤波器变形导致处理信号完全与原信号不一致

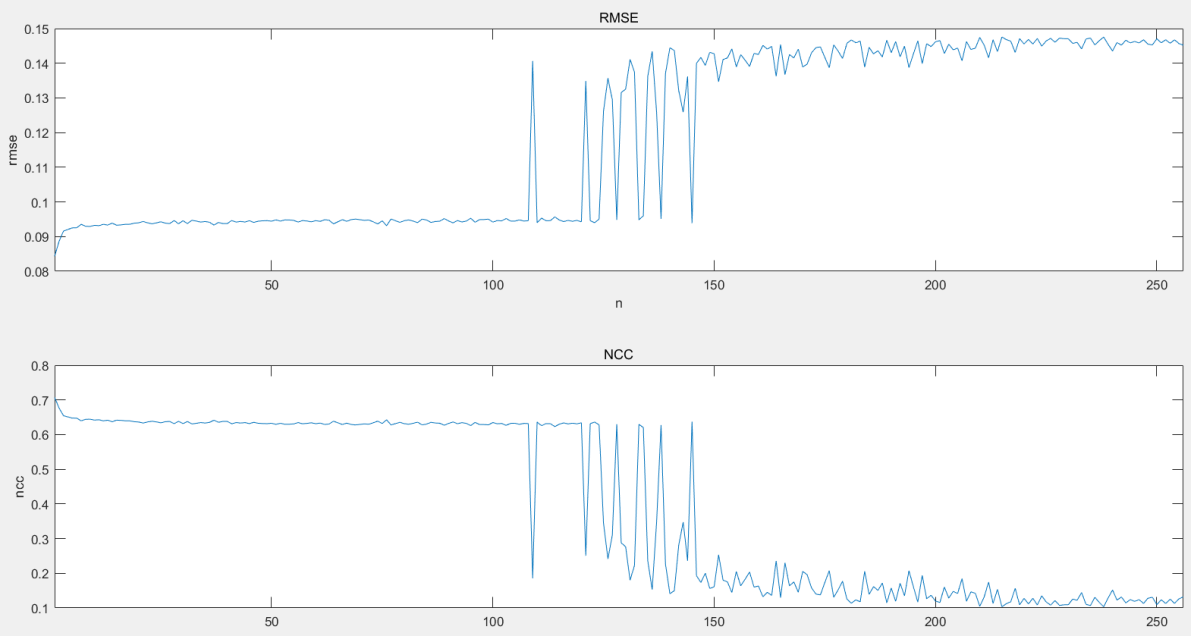
数据：RMSE均方根误差 和 NCC波形相似参数

根据定义，RMSE越接近0，说明处理信号与原信号越接近。NCC越接近1，说明去噪前后信号波形的整体相似度越高。

```
function rmse=RMSE(y1,y2)
    N = length(y1);
    rmse = norm(abs(y1)-abs(y2))/sqrt(N);
end

function ncc=NCC(y1,y2)
    ncc = sum(abs(y1).*abs(y2))/norm(y1)*norm(y2);
end
```

```
nx = 1:1:256;
rmse = zeros(1,length(nx));
ncc = zeros(1,length(nx));
for i = nx
    N=i;
    syn_signal = toneVocoder(noisySignal, fs, N, freCut);
    rmse(i) = RMSE(syn_signal,noisySignal);
    ncc(i) = NCC(syn_signal,noisySignal);
end
figure
subplot(2,1,1);plot(nx,rmse),xlabel('n'),ylabel('rmse'),xlim([1,256]),title('RMSE
');
subplot(2,1,2);plot(nx,ncc),xlabel('n'),ylabel('ncc'),xlim([1,256]),title('NCC');
```

结果：



$fig3. N \in [1,256]$的$RMSE$和NCC数据图

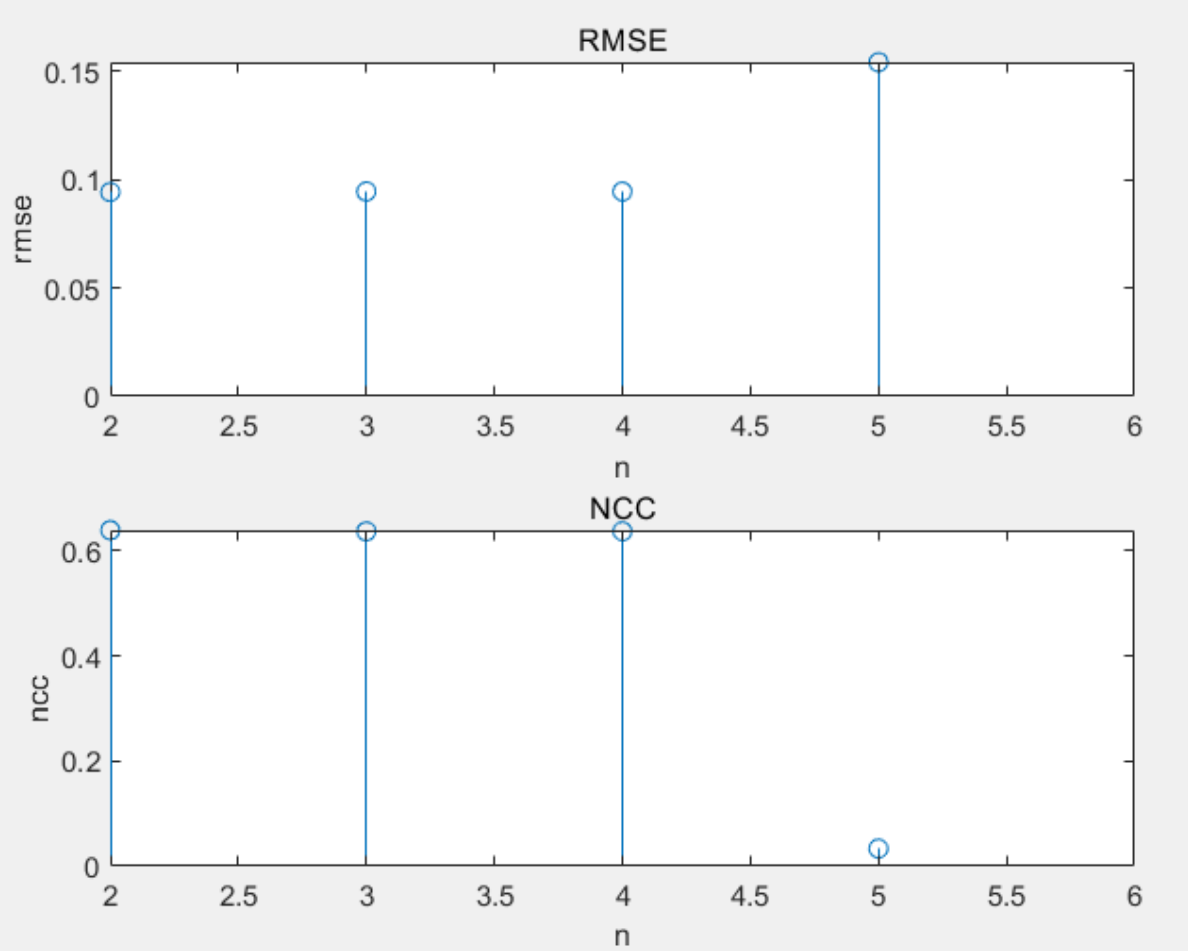
分析： 当N在1~120时，rmse和ncc指标表现较好，当N在120~150时，rmse和ncc指标上下振荡，而当N大于150后，明显rmse和ncc指标表现随N的增大变差，这与图像的验证表现也是吻合的。

结论：当相应的带通数量N增加时，恢复信号会更加清晰，但同时体积会减小，噪声也会变得越来越明显；但随着N越来越大，滤波器会随着 N 的增加而变形。结合声音播放，我们最后采取N=118作为理想带通个数，此时能辨识原句且明显听出有噪音干扰。

2、巴特沃斯滤波器阶数的改变

设置带通滤波器的阶数

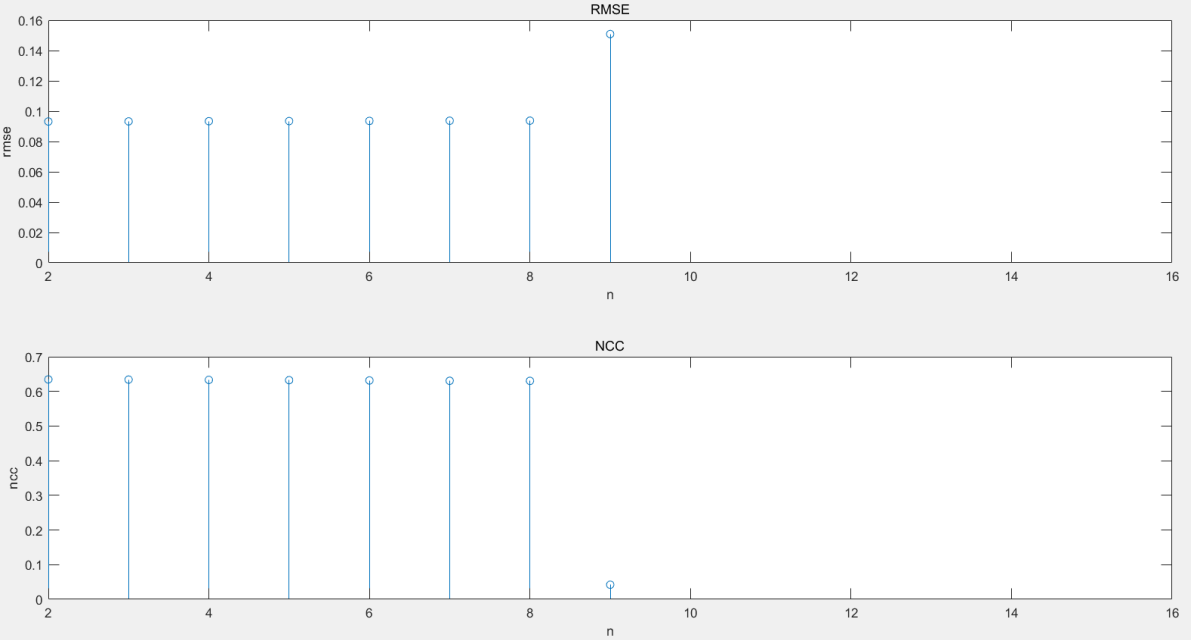
```
no = [2 3 4 5 6];
nx = 1:1:length(no);
rmse = zeros(1,length(no));
ncc = zeros(1,length(no));
for x = nx
    N=118;
    o = no(x);
    syn_signal = toneVocoder(noisySignal, fs, N, freCut,o);%toenVocoder中添加
BPorder参数，LPorer=6
    rmse(x) = RMSE(syn_signal,noisySignal);
    ncc(x) = NCC(syn_signal,noisySignal);
    disp(x)
end
figure
subplot(2,1,1);stem(no,rmse),xlabel('n'),ylabel('rmse'),xlim([2,6]),title('RMSE')
;
subplot(2,1,2);stem(no,ncc),xlabel('n'),ylabel('ncc'),xlim([2,6]),title('NCC');
```



$fig4.RMSE$和NCC在不同带通滤波器阶数下的大小

设置低通滤波器阶数

```
no = [2 3 4 5 6 7 8 9 10 12 14 16];
nx = 1:1:length(no);
rmse = zeros(1,length(no));
ncc = zeros(1,length(no));
for x = nx
    N=118;
    o = no(x);
    syn_signal = toneVocoder(noisySignal, fs, N, freCut,o);%toenVocoder中添加
LPorder参数，BPorder=4
    rmse(x) = RMSE(syn_signal,noisySignal);
    ncc(x) = NCC(syn_signal,noisySignal);
    disp(x)
end
figure
subplot(2,1,1);stem(no,rmse),xlabel('n'),ylabel('rmse'),xlim([2,16]),title('RMSE'
);
subplot(2,1,2);stem(no,ncc),xlabel('n'),ylabel('ncc'),xlim([2,16]),title('NCC');
```



$fig5.RMSE$和NCC在不同低通滤波器阶数下的大小

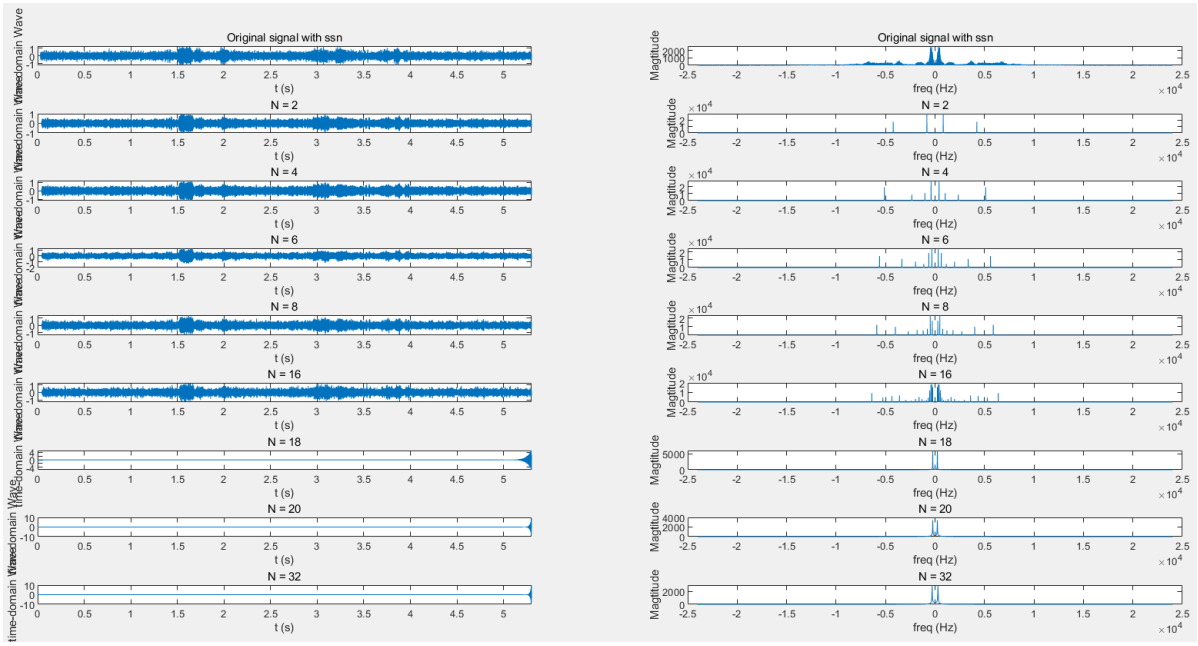
分析：带通滤波器在阶数小于等于4时表现相同，等于5时表现明显变差，大于6之后就无法计算了，低通滤波器在阶数小于等于8时表现相同，等于9时表现明显变差，大于10之后就无法计算了。

结论：带通滤波器的阶数取小于等于4的值，低通滤波器的阶数取小于等于8的值即可，有效阶数的的变化对恢复信号没有影响。我们采取BPorder=4，LPorder=6作为实验最优参数。

3、自主录制音频，验证人工耳蜗可行性

录制单声道音频“exp.wav”（内容为“明德求是，日新自强”），采样率为48000，设置截止频率为50Hz，同样设置设置带通滤波器数量为

```
range = [2 4 6 8 16 18 20 32];
```



$fig5.$自主录制音频经过处理后的时域频域图

分析：从时域图上来看，处理后的信号与原信号相差不多；频域上缺少较多；通过人耳识别，当带通滤波器数量N=16时可以大致辨别原句且有明显噪音，而当N大于16之后信号失真

结论：与project原本提供的音频相比，exp.wav采样率比其高出2倍，可以弥补一些因带通滤波器滤过的高频与低频信号的缺失，仅在N=16时即可较为明显辨别原句。

Task 4

（一）、问题重述

1、产生信噪比为-5dB的含噪信号

2、 通过将频段数更改为 N = 6

3、 通过将截止频率更改为 N = 20、50、100、400Hz 来进行滤波

4、 保存这些条件的波形文件，并描述截止频率如何影响合成语音的可理解性（即可以理解多少个单词）

（二）、处理方案：

1、设置函数产生含噪信号，将语音信号经过带通滤波器，全波整流过低通滤波器，与载波信号相乘最后叠加得到语音信号

2、通过计算处理信号与原信号的RMSE（均方根误差），信噪比SNR，波形相似参数NCC

SNR:

$$SNR = 10\log_{10}(E_{signal}/E_{noise}) = 20\log_{10}(\sqrt{E_{signal}}/\sqrt{E_{noise}})$$

NCC: (As指滤波前信号的幅值，Ad指滤波后信号的幅值)

$$NCC = \frac{\sum_{n=1}^{N} As(n)Ad(n)}{\sqrt{(\sum_{n=1}^{N} As^2(n))(\sum_{n=1}^{N} Ad^2(n))}}$$

RMSE:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} |A_{signal} - A_{denoised}|^2}{N}}$$

3、将人体耳蜗每段纤毛处理成相同长度，再通过f与d转化公式，将对应频率转化成一定长度的线段投射到耳蜗上去处理；目前采用的方法是将这段线段等分，每段等分的线段取其首尾频率再输入滤波器当中处理

4、相对应而在 stone-vocoder 方法里只是简单地将要处理的频率范围等分，直接输入滤波器当中处理，我们通过改变 N 的取值来探求3,4两种模拟方法下最佳 N 的范围以及其恢复信号的质量：

（三）、具体代码实现

1、函数GenerateSSN

SSN为语谱噪声，signal为语音信号，fs为语音信号的采样频率，nfft为fft数据点个数，noverlap为每个窗口的重叠长度

```
function SSN = GenerateSSN(signal, fs, nfft, noverlap)
    %generate white noise
    Len = size(signal,1);
    noise = 1-2*rand(1,Len);
    %using hamming windows
    windows = hamming(nfft);
    %PSD estimate
    [Pxx,w] = pwelch(signal,windows,noverlap,nfft,fs);
    %get the filter
    b = fir2(3000,w/(fs/2),sqrt(Pxx/max(Pxx)));
    %generate SSN
    SSN = filter(b,1,noise);
end
```

2、函数GenerateNoisySignal

noisySignal为含噪信号，signal为语音信号，ssn为语谱噪声，snr为信噪比

```
function noisySignal = GenerateNoisySignal(signal, ssn, snr)
    %calculate variable
    syms E;
    E = double(solve(20*log10(norm(signal)/E)==snr,E));
    %get the required ssn
    ssn = ssn/norm(ssn)*E;
    %add ssn into origin signal
    noisySignal = signal.' + ssn;
end
```

3、函数Split_N_freq

split_freq为模拟耳蜗的等距长度分割对应N个频率划分，freL为划分的最低频率，freH为划分的最高频率，N为划分的个数

```
function split_freq = Split_N_freq(freL, freH, N)
    syms dL dH;
    %solve the d for low frequency and high frequency
    dL = double(solve(165.4*(10^(0.06*dL)-1)==freL));
    dH = double(solve(165.4*(10^(0.06*dH)-1)==freH));
    %divide distance into N segments
    d = linspace(dL,dH,N+1);
    %divide frequency from freL to freH into N segments according to dist
    split_freq = 165.4.*(10.^(0.06.*d)-1);
end
```

4、函数toneVocoder

syn_signal为模拟耳蜗处理后的语音信号，noisySignal为含噪语音信号，fs为采样频率，N为带通滤波器的的个数，freCut为低通滤波器的截止频率

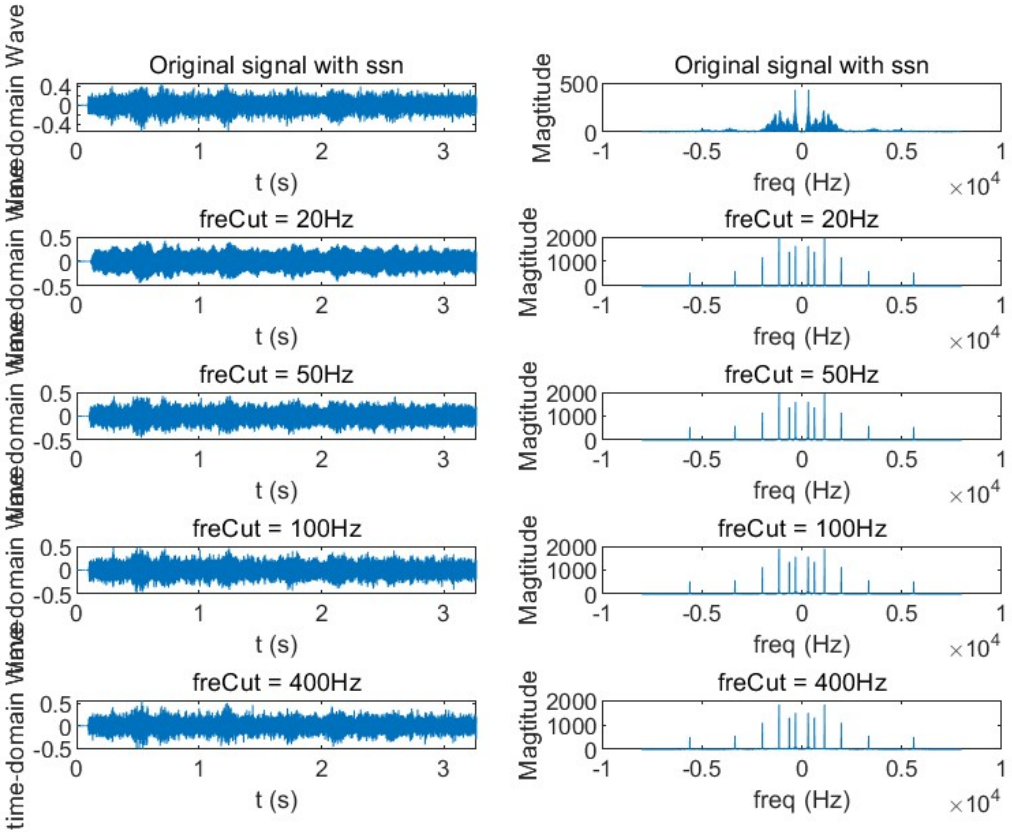
```
function syn_signal = toneVocoder(noisySignal, fs, N, freCut)
    %configuration
    freL = 200;
    freH = 7000;
    Len = length(noisySignal);
    BPorder = 4;
    LPorder = 6;

    s = zeros(1,length(noisySignal));

    split_freq = Split_N_freq(freL, freH, N);


    for i = 1 : N
        %design band-pass filter at band i
        [b, a] = butter(BPorder, [split_freq(i),split_freq(i+1)]/(fs/2));
        %do band-pass filtering at band i
        y = filter(b, a, noisySignal);
        %extract envelope
        envelope = extractEnvelope(y, fs, freCut, LPorder);
        %generate a sinewave as carrier
        freMidBand = (split_freq(i) + split_freq(i+1))/2;
        t = 0:1/fs:(Len-1)/fs;
        carrier = sin(2*pi*freMidBand.*t);
        %modulation
        modulatedSignal = envelope .* carrier;
        s = s + modulatedSignal;
    end

    %energy normalization
    syn_signal = s / norm(s) * norm(noisySignal);
end
```

5.函数Task4

```
clc, clear all;
%load signal
[signal,fs] = audioread('C_01_01.wav');%以C_01_01.wav为例
%configuration
N = 6;
snr = -5;
nfft = 512;
noverlap = nfft/2;
Len = length(signal);
t = 0:1/fs:(Len-1)/fs;
L = (1-Len)/Len/2*fs;
R = (Len-1)/Len/2*fs;
range = [20, 50, 100, 200];

%generate noisySignal
ssn = GenerateSSN(signal, fs, nfft, noverlap);
noisySignal = GenerateNoisySignal(signal, ssn, snr);
%sound(noisySignal, fs);
%change freCut from 20Hz to 100Hz
fig=figure();
    subplot(length(range)+1,2,1);plot(t,noisySignal),xlabel('t
(s)'),ylabel('time-domain Wave'),xlim([0,(Len-1)/fs]);
    title("Original signal with ssn");

 subplot(length(range)+1,2,2);plot(linspace(L,R,Len),abs(fftshift(fft(noisySignal
))),xlabel('freq (Hz)'),ylabel('Magtitude');
    title("Original signal with ssn");
%save fig
for i = 1:1:length(range)
    freCut = range(i);
    syn_signal = toneVocoder(noisySignal, fs, N, freCut);
    %plot time-domain wave
    subplot(length(range)+1,2,2*i+1);plot(t,syn_signal),xlabel('t
(s)'),ylabel('time-domain Wave'),xlim([0,(Len-1)/fs]);
    title("freCut = " + num2str(freCut) + "Hz");
    %plot frequency-domain wave

 subplot(length(range)+1,2,2*i+2);plot(linspace(L,R,Len),abs(fftshift(fft(syn_sig
nal))),xlabel('freq (Hz)'),ylabel('Magtitude');
    title("freCut = " + num2str(freCut)+ "Hz");
%    sound(syn_signal, fs);
end
%change freCut from 20Hz to 100Hz
for i = 1:1:length(range)
    freCut = range(i);
    syn_signal = toneVocoder(noisySignal, fs, N, freCut);
    %plot time-domain wave
    figure;plot(t,extractEnvelope(syn_signal, fs, 100, 6) ),xlabel('t
(s)'),ylabel('time-domain Wave'),xlim([0,(Len-1)/fs]);
    title("freCut = " + num2str(freCut) + "Hz");
    saveas(fig, "envelop" + num2str(freCut), 'jpg');
    %plot frequency-domain wave
```

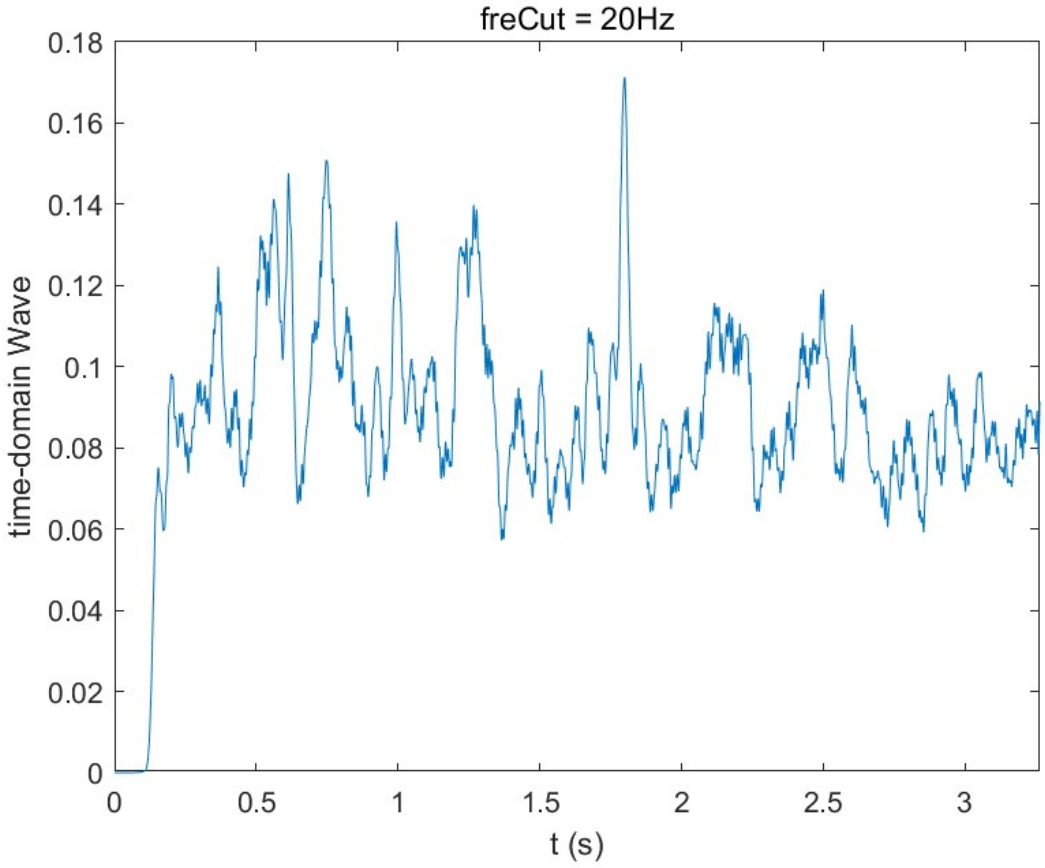
```
    figure;plot(linspace(L,R,Len),abs(fftshift(fft(syn_signal)))),xlabel('freq
(Hz)'),ylabel('Magtitude');
    title("freCut = " + num2str(freCut)+ "Hz");
    saveas(fig, "frequency" + num2str(freCut), 'jpg');
%    sound(syn_signal, fs);
end
%save fig
saveas(fig, 'results','jpg')
```

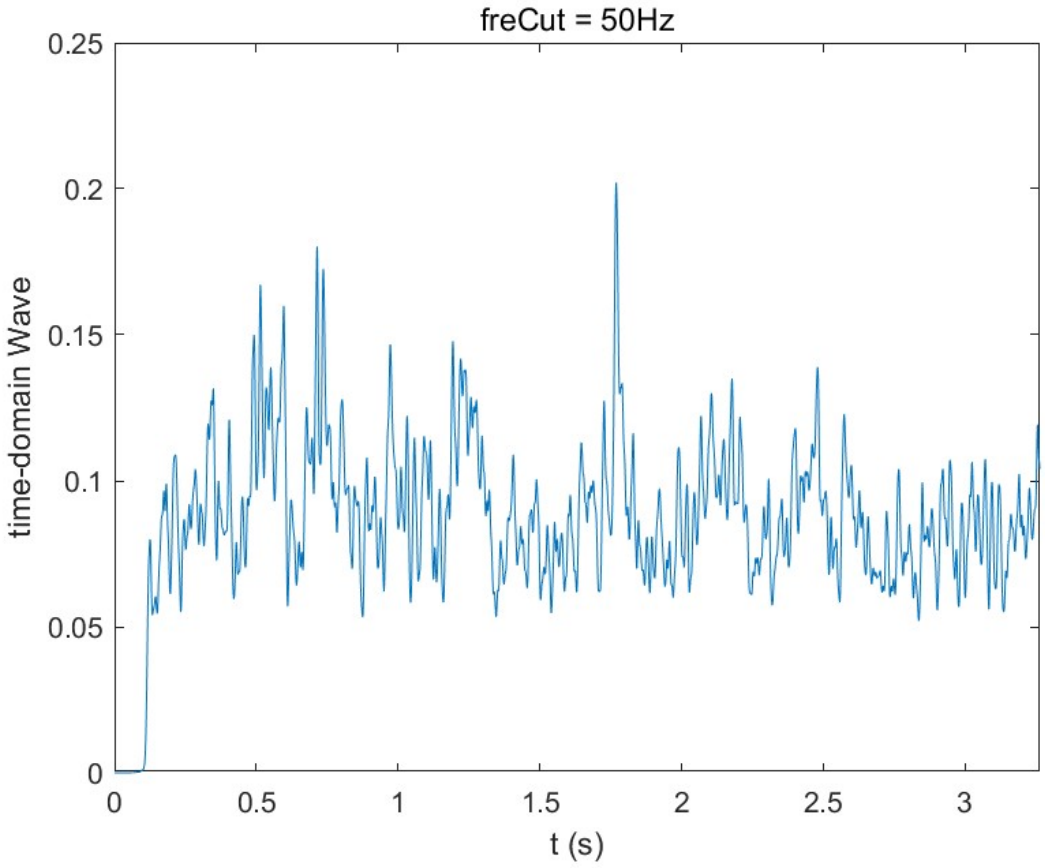
总体对比

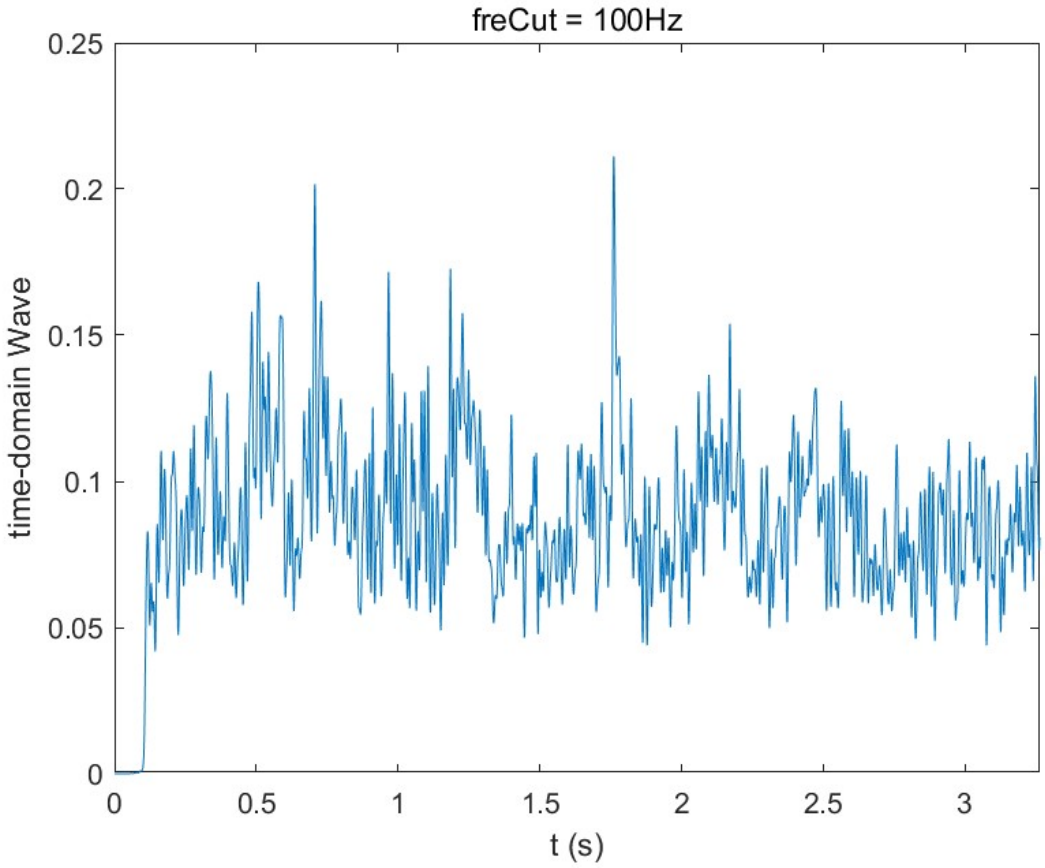


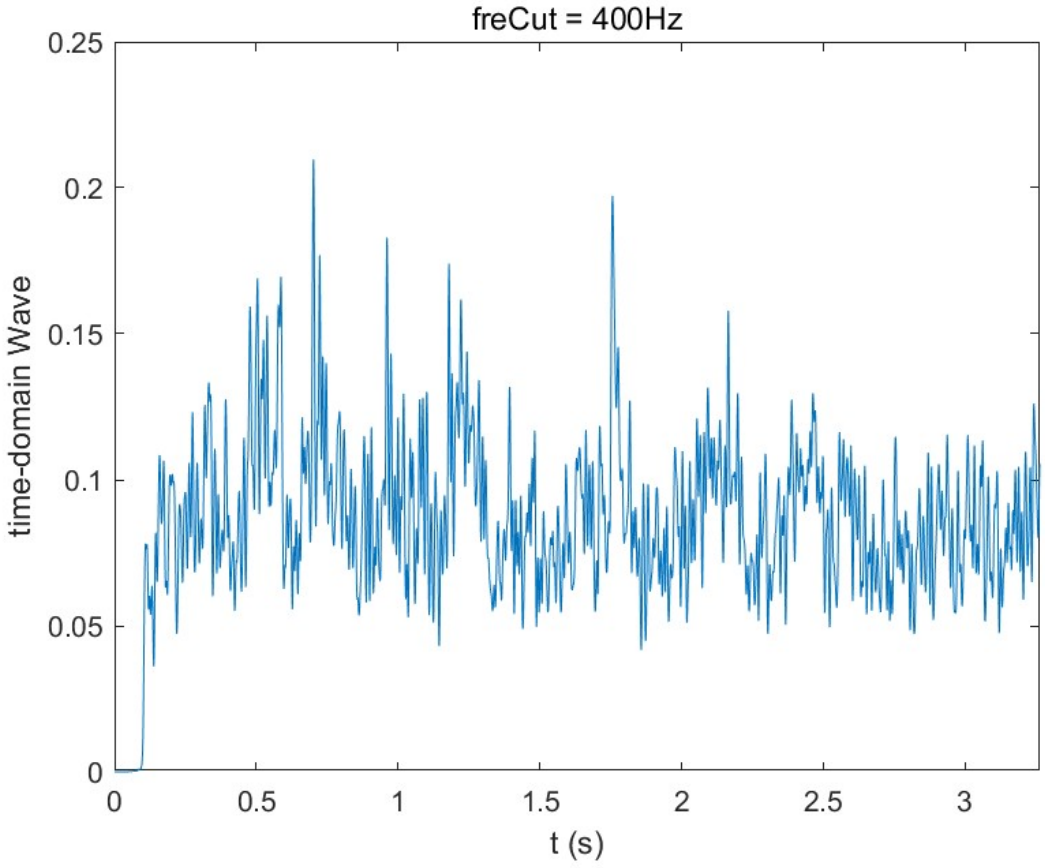
$fig.1$实验结果图(时域 + 频域)

时域包络图

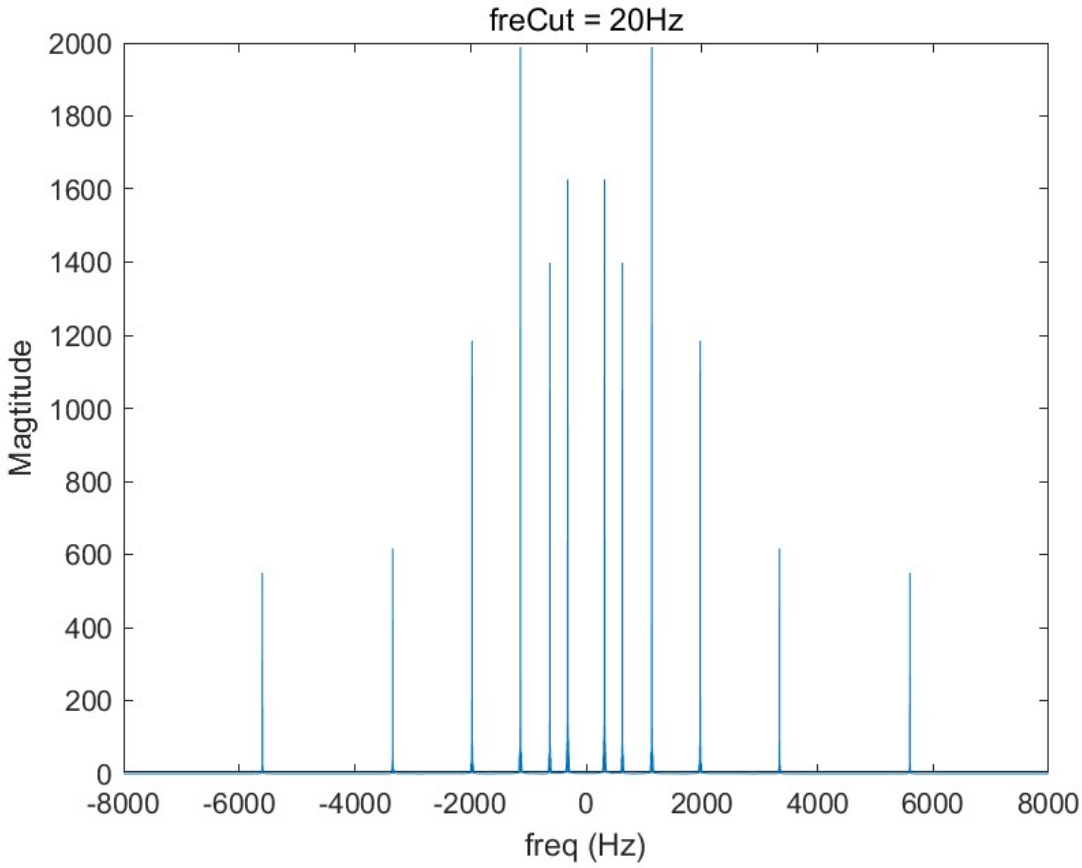

freCut = 20Hz
time-domain Wave
t (s)
0
0.02
0.04
0.06
0.08
0.1
0.12
0.14
0.16
0.18
0.5
1
1.5
2
2.5
3

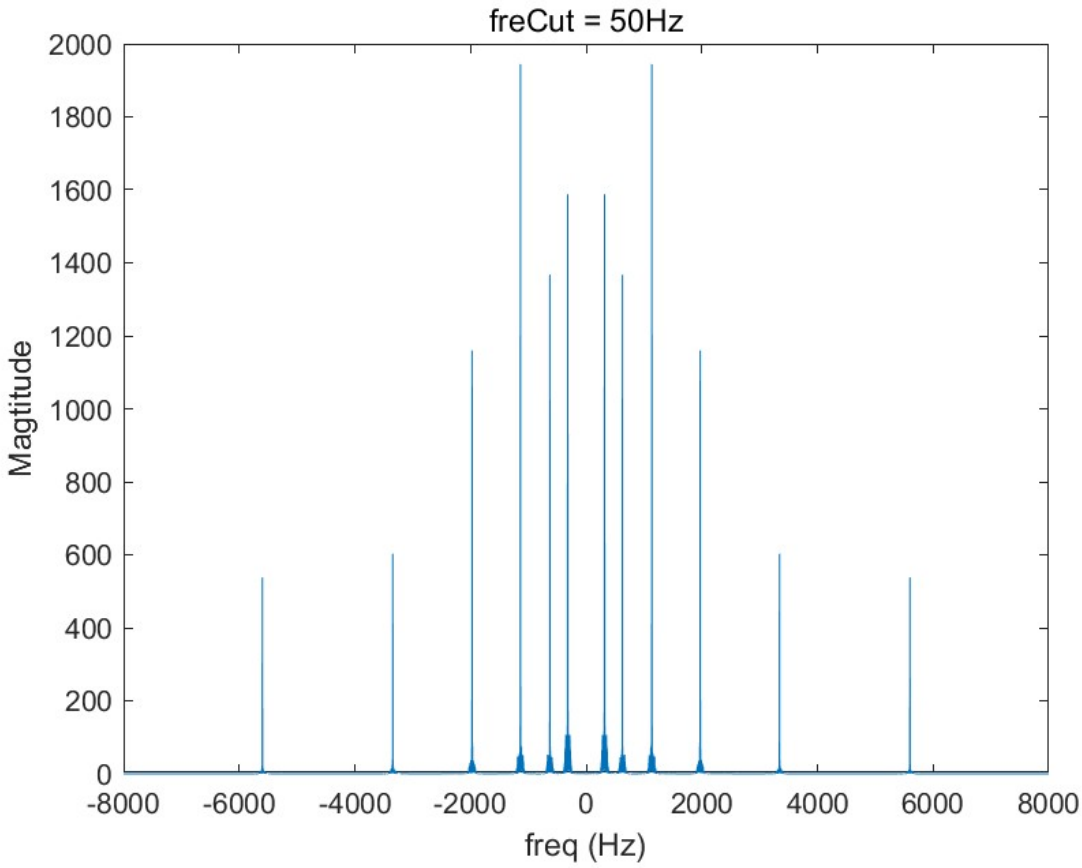

freCut = 50Hz
time-domain Wave
t (s)
0
0.05
0.1
0.15
0.2
0.25
0.5
1
1.5
2
2.5
3

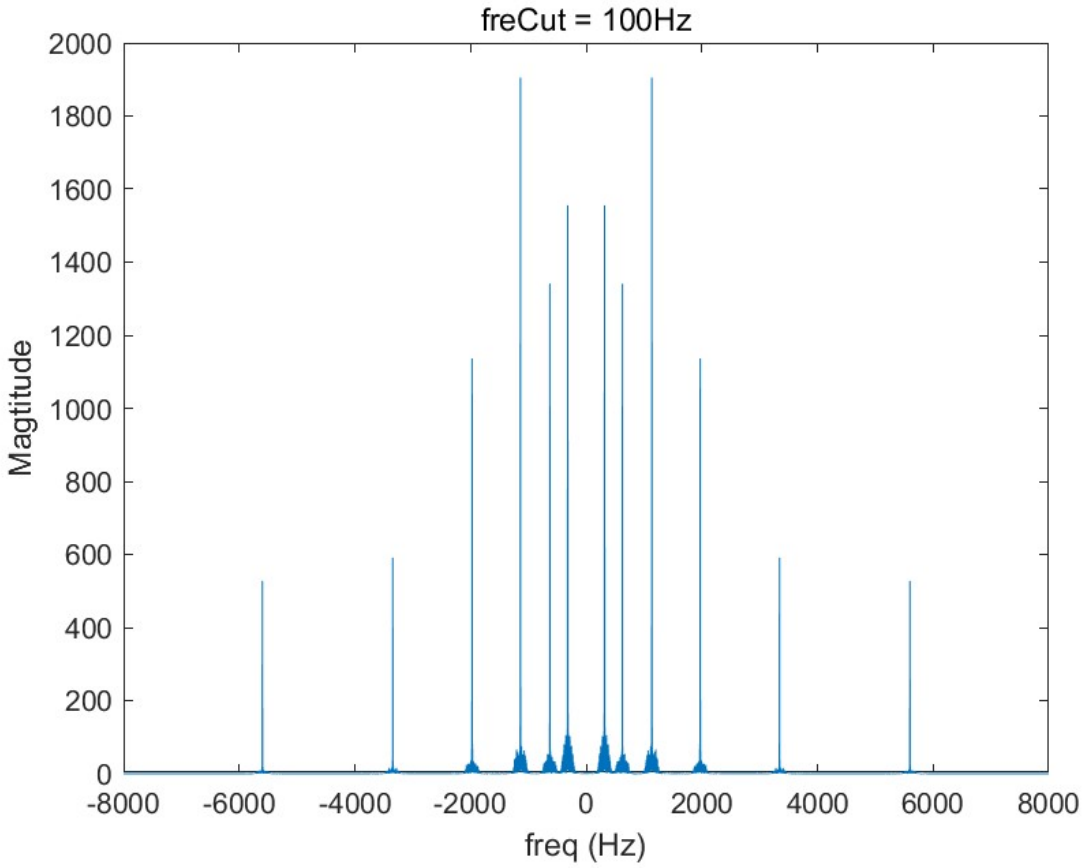

freCut = 100Hz
time-domain Wave
t (s)
0
0.05
0.1
0.15
0.2
0.25
0.5
1
1.5
2
2.5
3

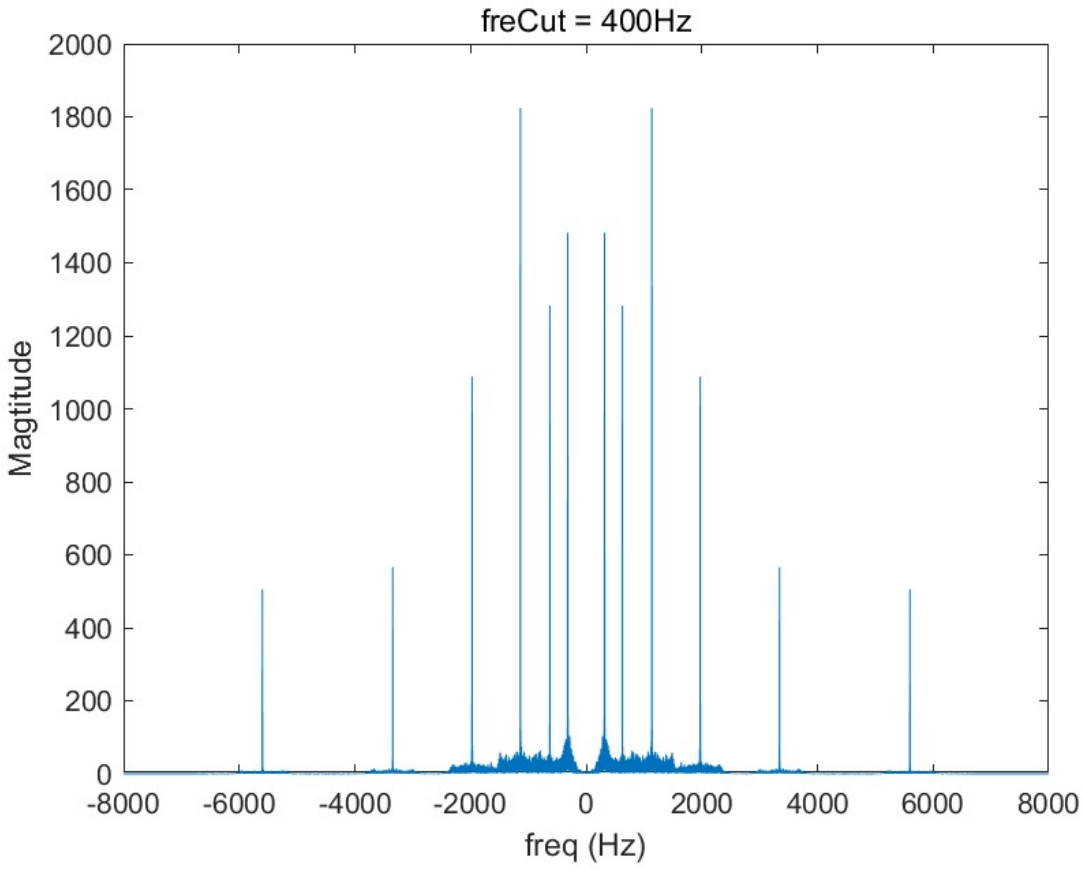

freCut = 400Hz
time-domain Wave
t (s)
0
0.05
0.1
0.15
0.2
0.25
0.5
1
1.5
2
2.5
3

频域图


freCut = 20Hz
Magtitude
freq (Hz)
2000
1800
1600
1400
1200
1000
800
600
400
200
0
-8000
-6000
-4000
-2000
0
2000
4000
6000
8000


freCut = 50Hz
Magtitude
freq (Hz)
2000
1800
1600
1400
1200
1000
800
600
400
200
0
-8000
-6000
-4000
-2000
0
2000
4000
6000
8000





5、分析

随着截止频率的增大，复原信号在时域图上与原始含噪信号没有明显区别，包络线逐渐变得锐利。在频域图上，处理过的信号明显为相对离散的分量。但是其高度和宽度并没有随着截止频率增大有明显区别，仅仅是每个峰的底部逐渐变宽，但是即使是400Hz也听不清楚。

6、结论

与 task2 相对比，可以看出：带有语谱噪声的混合语音信号复原清晰度明显降低，语音信号复原效果明显变差。随着freCut的增大，复原信号的质量仅有微弱提升，即使当freCut =400Hz 时，也无法听清语音信号的内容。

（四）Task 4 拓展

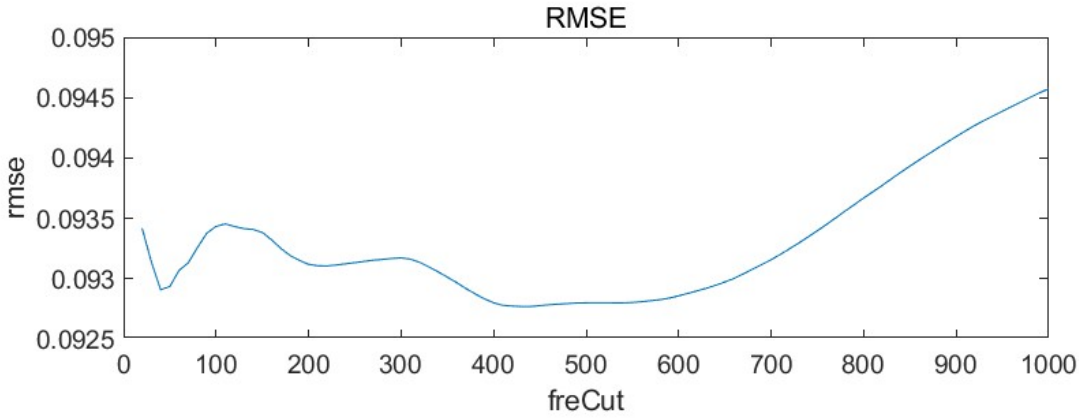
继续增加freCut的大小，是否能够得到更清晰，能够辨认的信号?

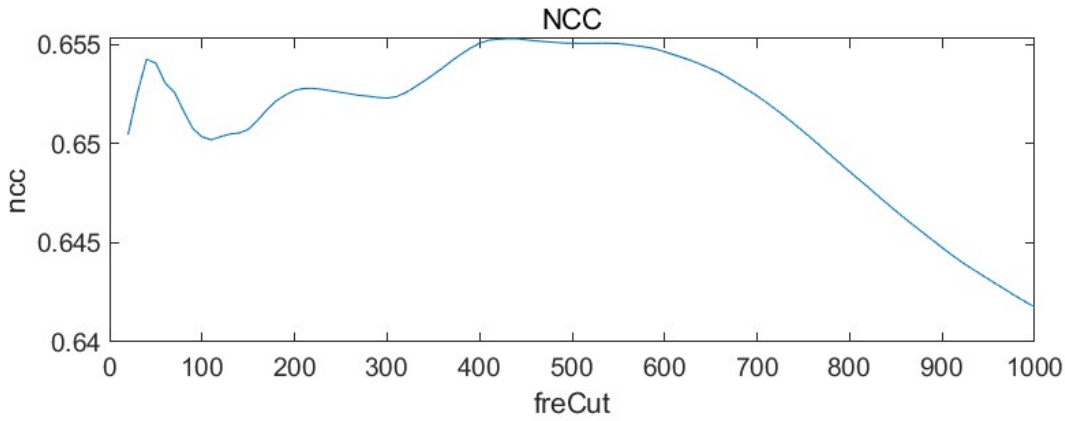
数据：RMSE均方根误差 和 NCC波形相似参数

根据定义，RMSE越接近0，说明处理信号与原信号越接近。NCC越接近1，说明去噪前后信号波形的整体相似度越高。

```
function rmse=RMSE(y1,y2)
    N = length(y1);
    rmse = norm(abs(y1)-abs(y2))/sqrt(N);
end

function ncc=NCC(y1,y2)
    ncc = sum(abs(y1).*abs(y2))/norm(y1)*norm(y2);
end
```

```
%RMSE NCC
nx = 20:10:1000;
rmse = zeros(1,length(nx));
ncc = zeros(1,length(nx));
for i = 1:length(nx)
    freCut=nx(i);
    syn_signal = toneVocoder(noisySignal, fs, N, freCut);
    rmse(i) = RMSE(syn_signal,noisySignal);
    ncc(i) = NCC(syn_signal,noisySignal);
end
fig = figure();
subplot(2,1,1);plot(nx,rmse),xlabel('freCut'),ylabel('rmse'),title('RMSE');
subplot(2,1,2);plot(nx,ncc),xlabel('freCut'),ylabel('ncc'),title('NCC');
saveas(fig, 'RMSENCC','jpg')
```





分析：

可以看到随着截止频率的上升，RMSE和NCC都呈现一个先变好再变差的情况。分析原因可能是当截止频率低时，有效的声音信息被低通滤波器过滤掉了，而当截止频率过高时又将不需要的高频杂音信息采样进来，因此效果变差。

结论：

随着截止频率的升高，效果先升高再下降，当截止频率约为500Hz时最优。

总结

在本次project中，我们完成了Task1,2,3,4，并且在每个Task的基础上进行高度的扩展。所有的项目要求都完美完成，并且进行了较为完善全面的扩展。

自评分

朱佳颖:100 (Task1 及其扩展)

潘耿炜:100 (Task2 及其扩展)

谢嘉楠:100 (Task3 及其扩展)

刘一郴:100 (Task4 及其扩展, 实验报告整理)

贡献比：各 25%