Project1 : Speech synthesis and perception

涂峻绫(12213010), 欧阳莹轩(12212961), 肖星辰(12212904), 欧阳安男(12211831)

Contents

Introduction

1.语音合成的基本模型：

a.记录的语音信号s：有意义的声音信息中通常包含语音，语音由不同的时间和频谱信息组成，可以表征为几个调幅窄频带的总和

b.带通滤波器：接收记录的信号，能够通过该信号一定频率范围内的频率分量，但将其他范围内的频率分量衰减到很低的水平，实现滤波

c.整流器和低通滤波器：进行全波整流和低通滤波，以获取对应频段的包络

d.生成一个正弦波，其频率等于对应频段的中心频率，与c中包络信号相乘

e.对所有N个频段重复上述操作，对所有输出进行求和（将求和后的输出表示为s'）

f.进行能量归一化，即让s'的能量等于s的能量

g.将所有正弦波组合在一起，获得合成的语音信号。

2. 巴特沃斯低通/带通滤波器原理：

巴特沃斯滤波器是一种具有最大平坦幅度响应低通滤波器，能够通过滤波器的频率信号范围构成通带，而被衰减的频率信号则不能在输出端输出，这些被衰减的频率范围构成了滤波器的阻带。通带与阻带交界点的频率称为截止频率。理想的滤波器其频率响应于通带区没有插入损耗(insertion-loss)，而于阻带则有无限大衰减以阻绝讯号通过。

巴特沃斯低通滤波器可用如下振幅的平方对频率的公式表示：

$$|H(\omega)|^2 = \frac{1}{1+\left(\frac{\omega}{\omega_c}\right)^{2n}} = \frac{1}{1+\epsilon^2\left(\frac{\omega}{\omega_p}\right)^{2n}}$$

n= 滤波器的阶数

wc = 截止频率 = 振幅下降为 -3分贝时的 频率

wp = 通频带边缘频率

$$\frac{1}{1+\epsilon^2} = |H(\omega)|^2 =$$

在通频带边缘的数值

3、滤波器函数设计、验证、使用步骤：

A.确定滤波器参数

a.根据频率到位置的映射等式：f = 165.4*（10^0.06*d -1)，设计耳蜗上位置（单位mm）与对应频率互相转换的函数，`f = cochlea2freq (d)` 与 `f = divide_freq (n, f0, f1)`

b.通过给出的耳蜗信息确定需要滤波的带通宽度`[frequency_bottom, frequency_top]`（单位为Hz），在此频率范围内设计滤波器

c.设置频段数量为N

B.使用滤波器设计函数

a.对于第i个频段设置带通滤波器的采样频率fs

b.在频段i上通过函数`[b,a] = butter(N, [frequency_bottom, frequency_top]/(fs/2))`设置带通滤波器

C.滤波器函数验证

创建测试信号，应用滤波器并分析滤波前后的信号，确保滤波器能够正确地滤除或保留所需的频率成分，进行验证。

D.应用滤波器

在对于频段i上对于信号s使用 `filter` 函数： `y = filter(b, a, s);`

.

4、包络提取原理:

进行全波整流和低通滤波，以获取对应频段的包络，更好地分析信号的特性。

其中全波整流将信号的所有负值变为正值。这可以通过取信号的绝对值实现。全波整流的目的是消除信号中的负频率成分，使其更易于分析。

低通滤波器在全波整流之后，过滤信号中还包含的高频噪声和其他不需要的成分，只保留低频信号。

例如： `[b1,a1] = butter(2,100/(0.5*fs));` %使用Butterworth函数设计一个二阶低通滤波器。其N=2，截止频率为100 Hz，参数fs是信号的采样频率。

`yb1 = filter(b1,a1,abs(y1));` %使用filter函数将设计的低通滤波器应用于信号的绝对值，实现了全波整流和低通滤波，滤波后的输出yb1包含了信号在100 Hz以下的包络信息。

plot(t,yb1),xlabel('t'),ylabel("Envelop Wave"),grid on, % 绘制时间t和包络波形yb1之间的关系。xlabel(‘t’)和ylabel(“Envelop Wave”)分别设置了x轴和y轴的标签。grid on表示在图上添加网格线

5、语谱噪声产生原理:

1.生成语音信号的长时频谱

将10个语音波形连接成一个信号

• sig =[x,x,x,x,x,x,x,x,x,x]; % x 是一个行向量，sig 是一个长时语音 （ sig = repmat(x,1,10);)

估计语音信号的功率谱密度

• [Pxx,f] =pwelch(sig, [], [], 512, fs); % [pxx, f] = pwelch(x, window, noverlap, nfft, fs)

2.生成滤波器系数

b = fir2(3000,f/(fs/2),sqrt(Pxx/max(Pxx))) % fir2: 从频率响应到系数

[h,wh] = freqz(b,1,128); %在此处检查频率响应

3.对白噪声信号进行滤波：y = filter(b,1,whitenoise)，得到到语谱噪声

6、能量归一化原理 :

能量归一化的操作目的是调整合成语音信号的能量，使之与目标语音信号（或称为参考信号）具有相同的能量水平。

应用的函数公式：y’=y/norm(y)*norm(x);

其中y是待处理的语音信号，y’是处理后的信号，norm(y)表示y的范数（通常指欧几里得范数），它反映了信号y的能量大小，norm(x)表示参考信号x的能量。

y/norm(y)：将信号y除以其自身的能量（范数），减小了信号的能量。即如果y的能量大于x，那么这一步将使得y的能量降低到与x相同的水平。

norm(x)：计算参考信号x的能量，用它作为归一化的一个因子来确保最终的信号y’与x具有相同的能量。

y/norm(y)*norm(x)：将调整后的y信号能量与x的能量相乘，最终得到的新信号y’就有与x相同的能量。

通过这样的能量归一化处理，合成的语音信号就可以在播放时具有与原始语音相同的音量水平，使得语音听起来更加自然和舒适。这在语音识别、语音合成和其他需要语音信号处理的领域中非常重要，因为它可以改善语音的质量，提高系统的性能。

验证部分：通过norm函数表示能量，再以编写的feq函数进行浮点数对比，在结果上得到了能量归一的验证。

Results and Analysis

Task 1

- Set LPF cut-off frequency to 50 Hz.
- Implement tone-vocoder by changing the number of bands to N=4, N=8, N=16, and N=32.
- Save the wave files for these conditions and describe how the number of bands affects the intelligibility (i.e., how many words can be understood) of synthesized sentence.

Initialize and load audio.

```
clear;clc;
[sig, fs] = audioread('proj1/C_01_02.wav');
```

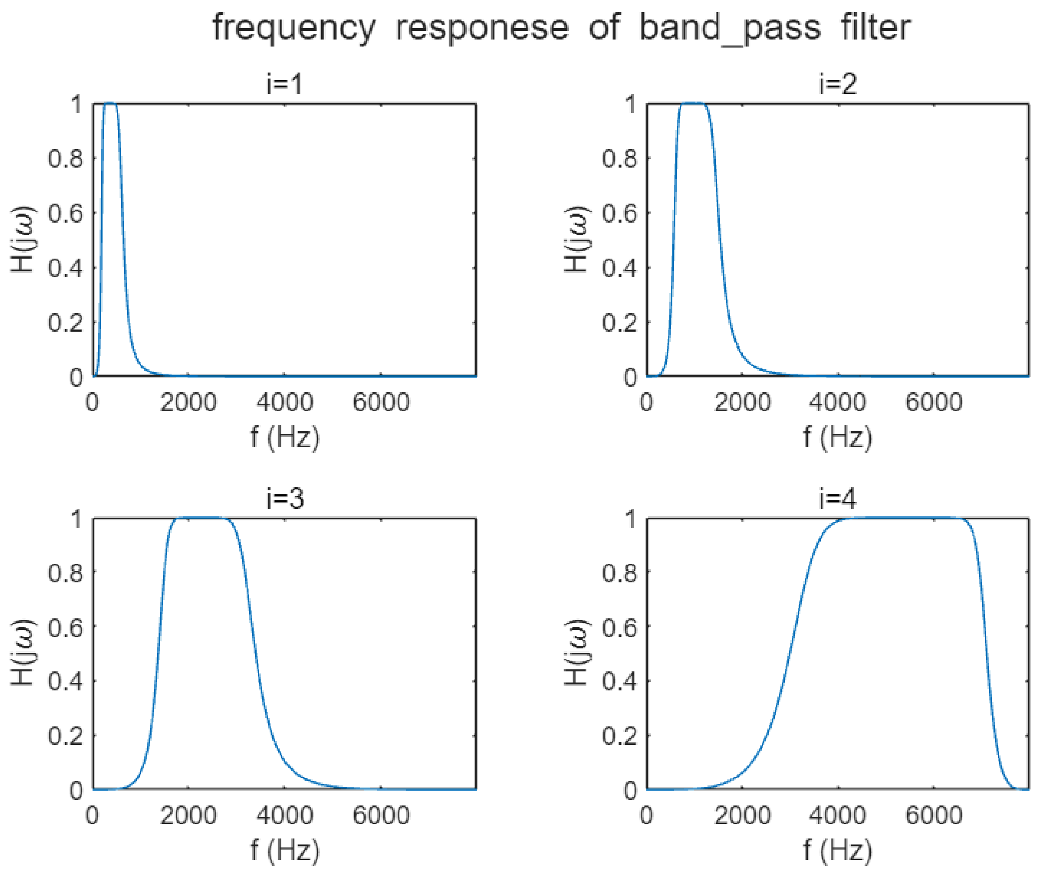
Set LPF cut-off frequency.

```
cut_off = 50;
```

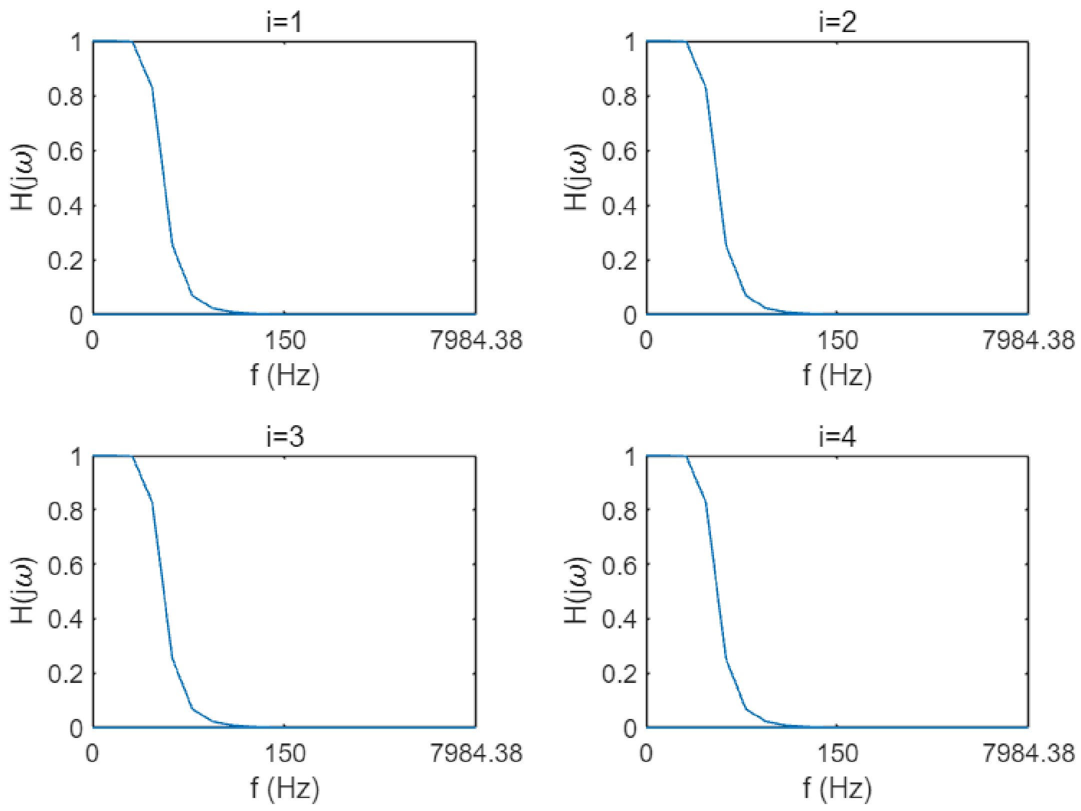
Implement tone-vocoder.

N = 4

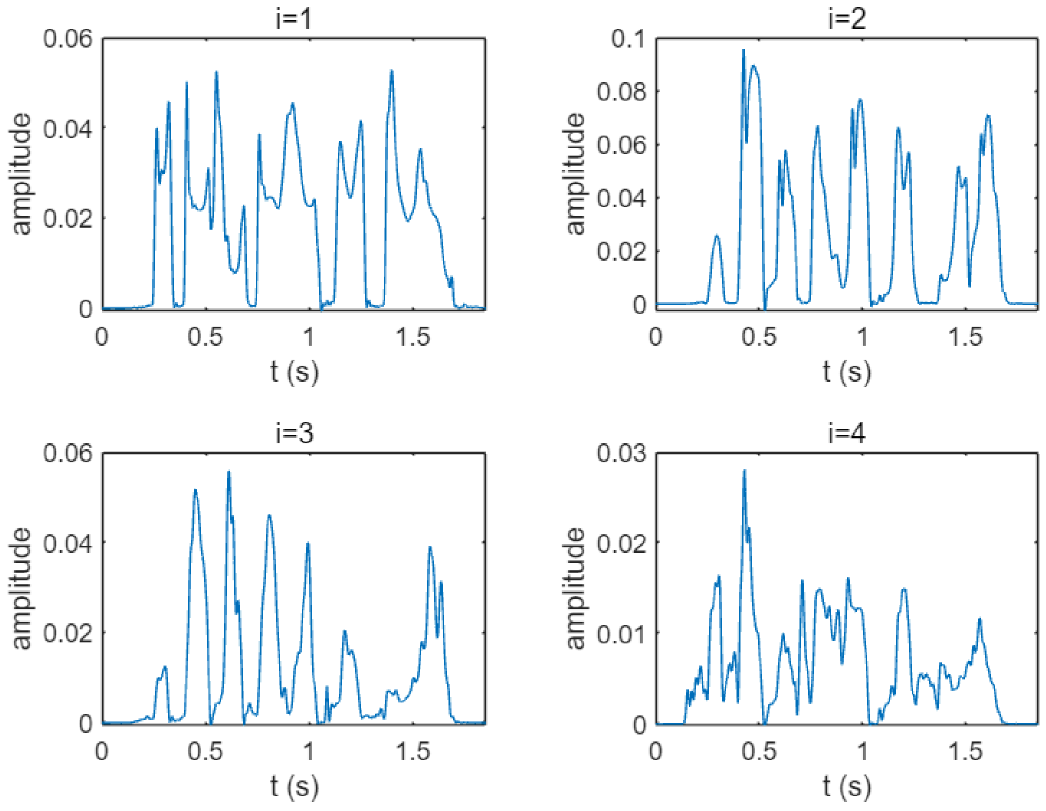
```
N_4 = speech_synthesis(sig, fs, 4, cut_off);
```



frequency responese of low_pass filter

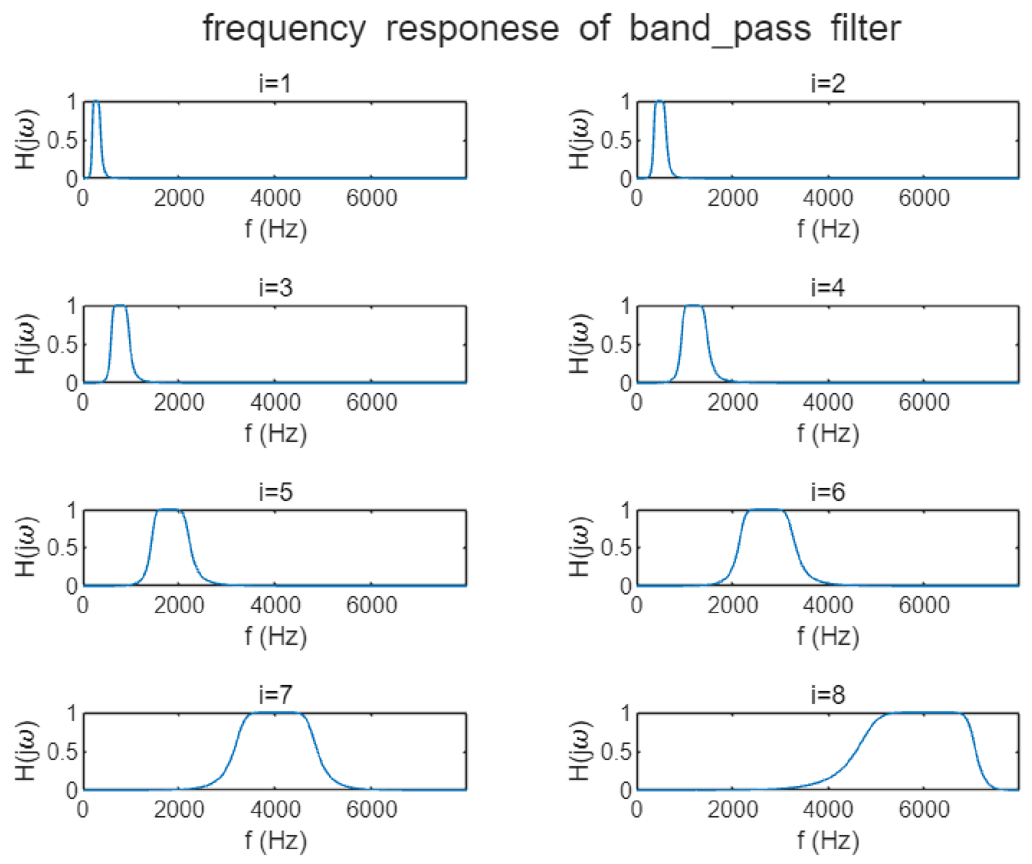


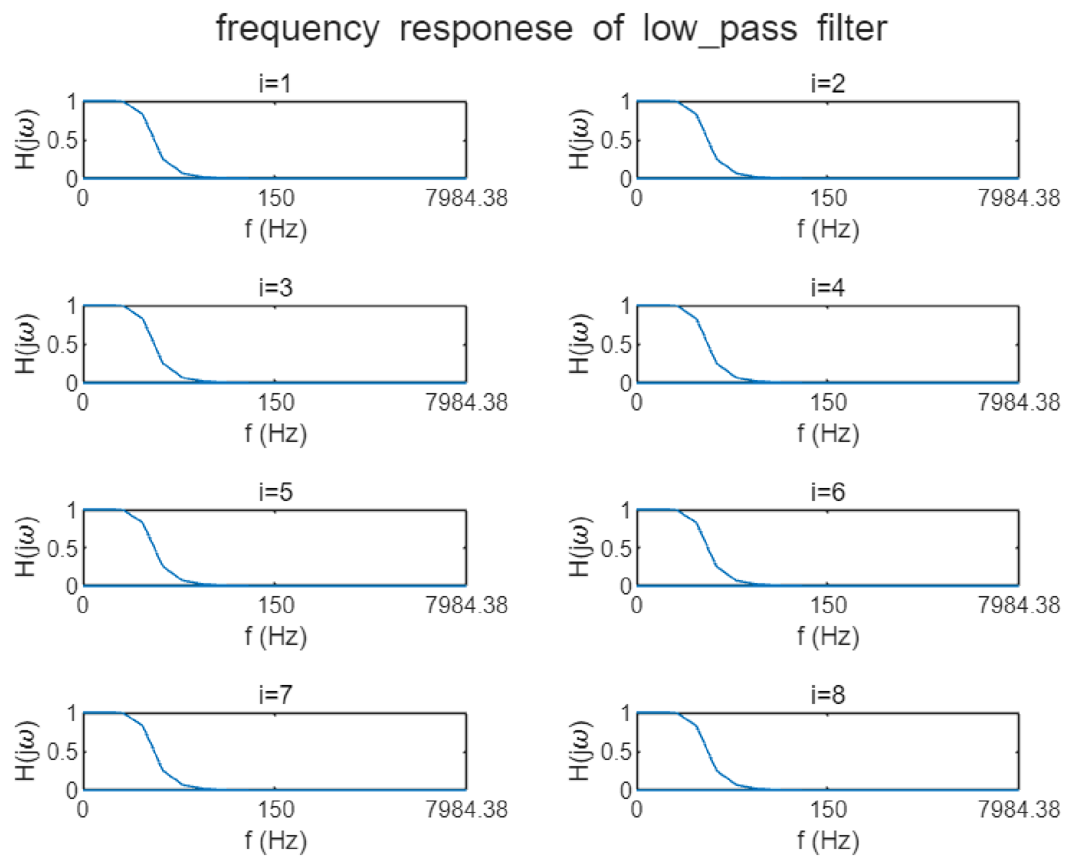
envelope

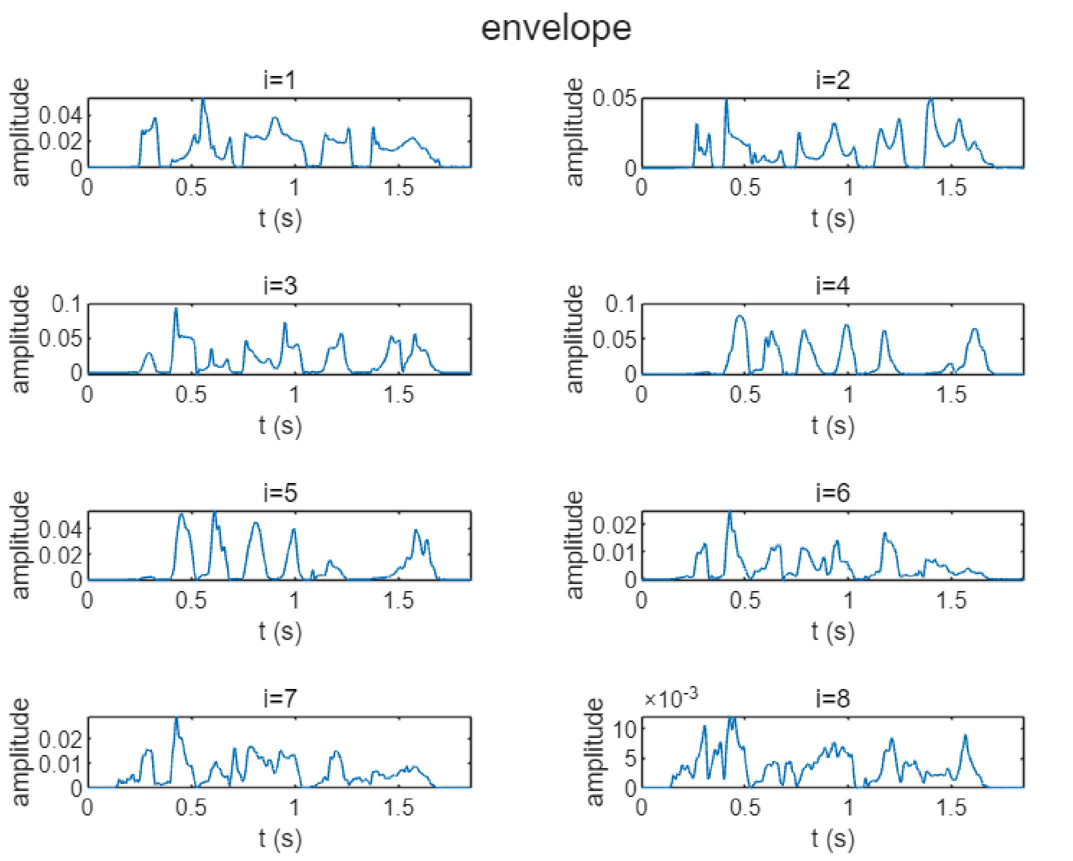


N = 8

```
N_8 = speech_synthesis(sig, fs, 8, cut_off);
```

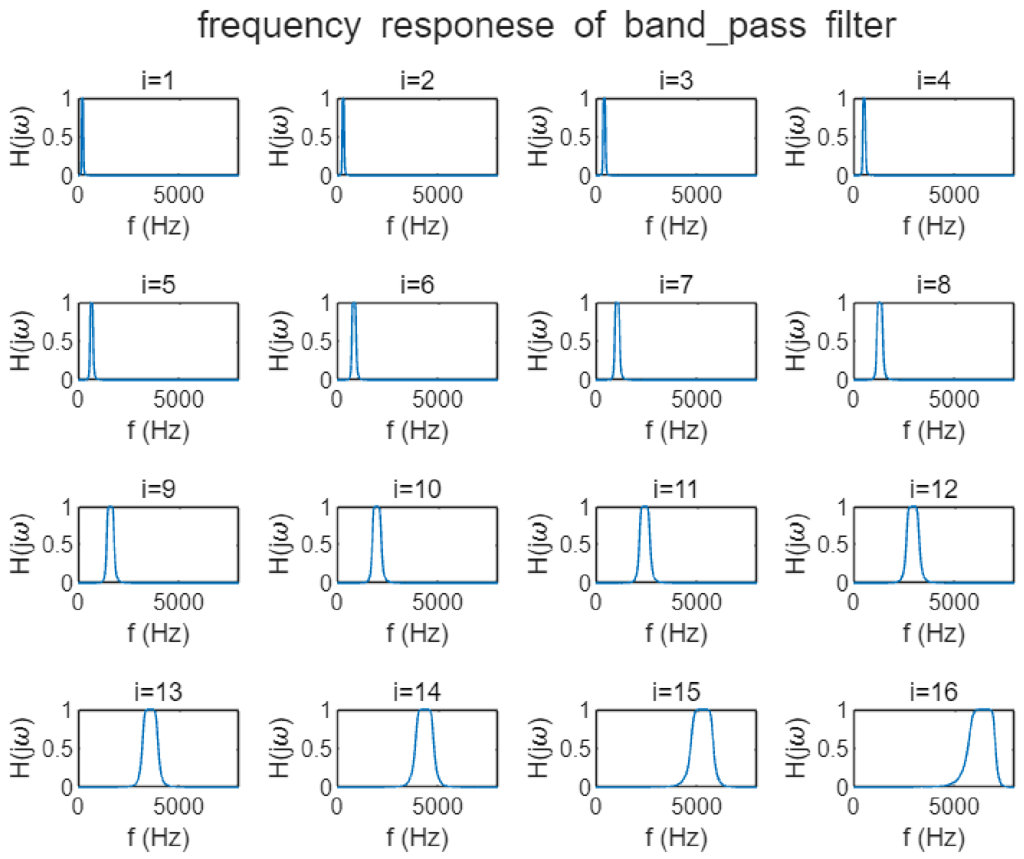


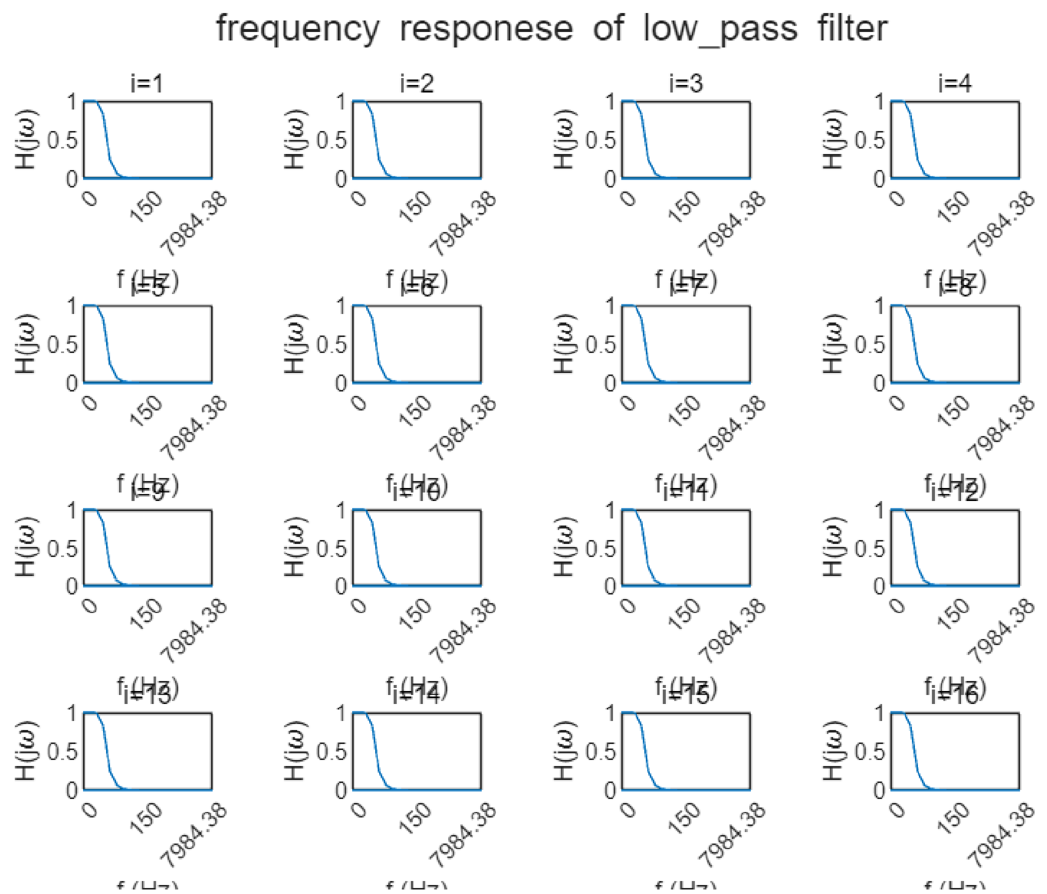

frequency responese of low_pass filter
i=1
i=2
i=3
i=4
i=5
i=6
i=7
i=8
H(jω)
f (Hz)
0
150
7984.38
0
0.5
1

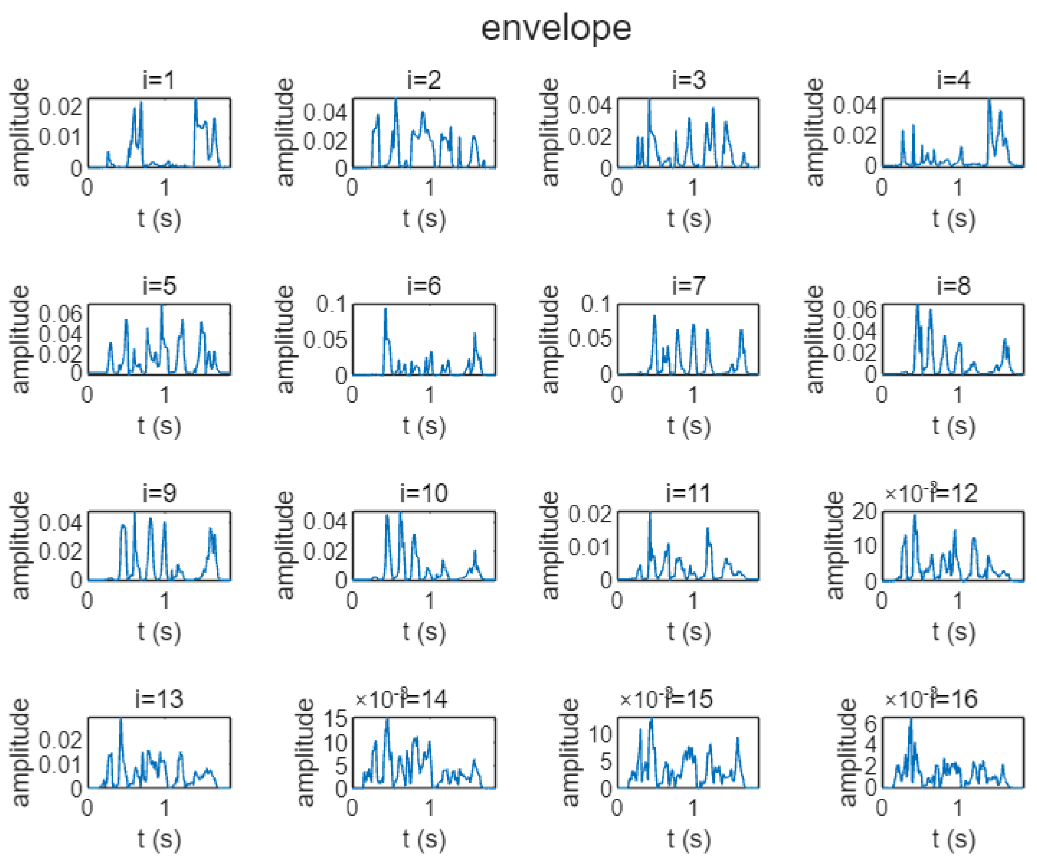

envelope
i=1
i=2
i=3
i=4
i=5
i=6
i=7
i=8
amplitude
t (s)
0
0.5
1
1.5
0.02
0.04
0.05
0.1
0.01
×10^-3
5
10

N = 16

```
N_16 = speech_synthesis(sig, fs, 16, cut_off);
```

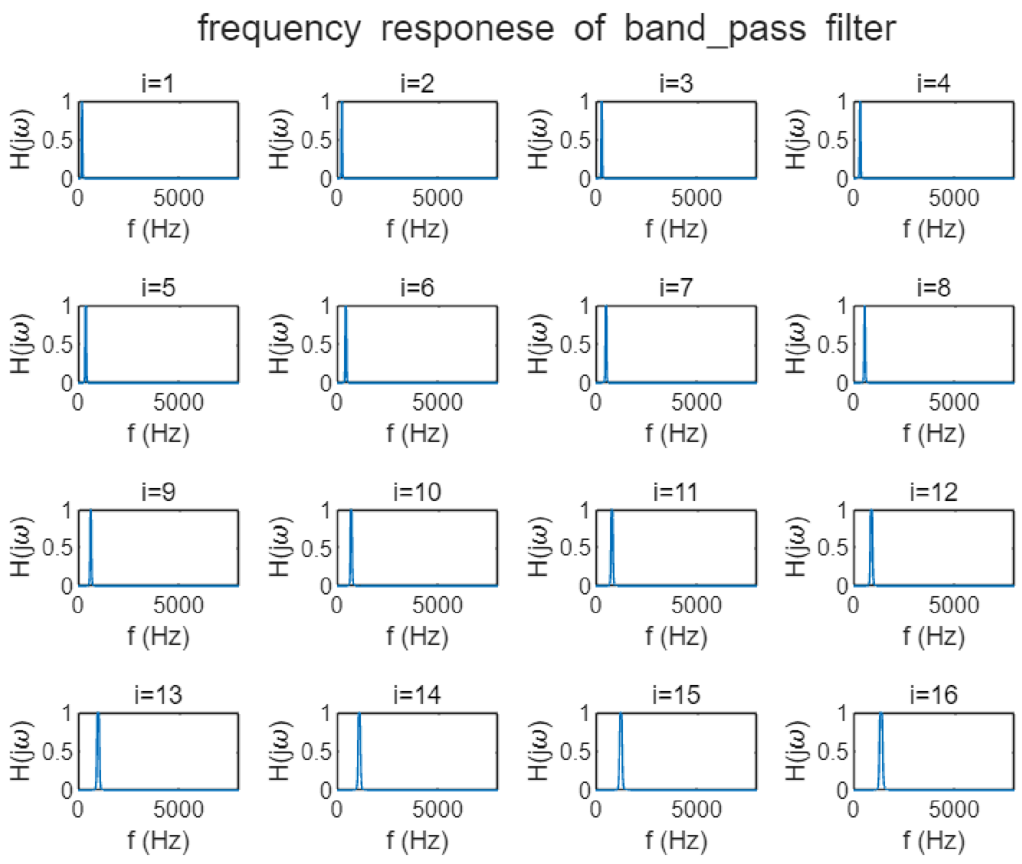


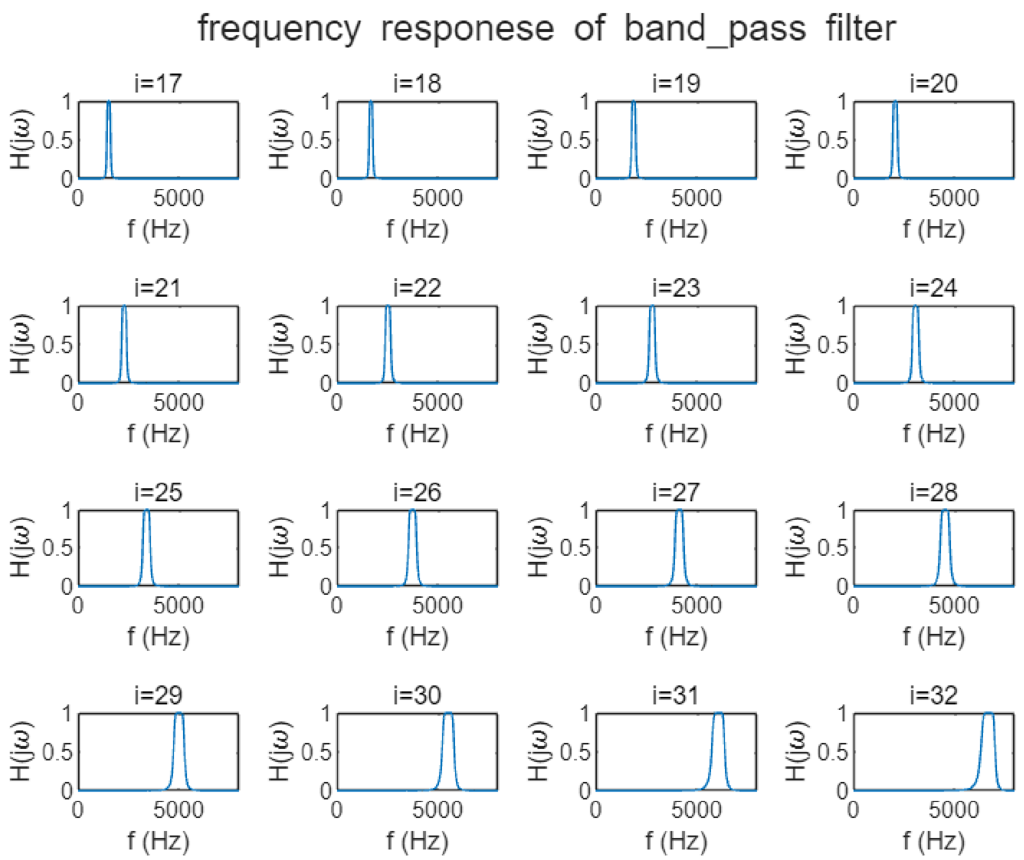

frequency responese of low_pass filter
i=1
i=2
i=3
i=4
i=5
i=6
i=7
i=8
i=9
i=10
i=11
i=12
i=13
i=14
i=15
i=16
H(jω)
f (Hz)
0
150
7984.38

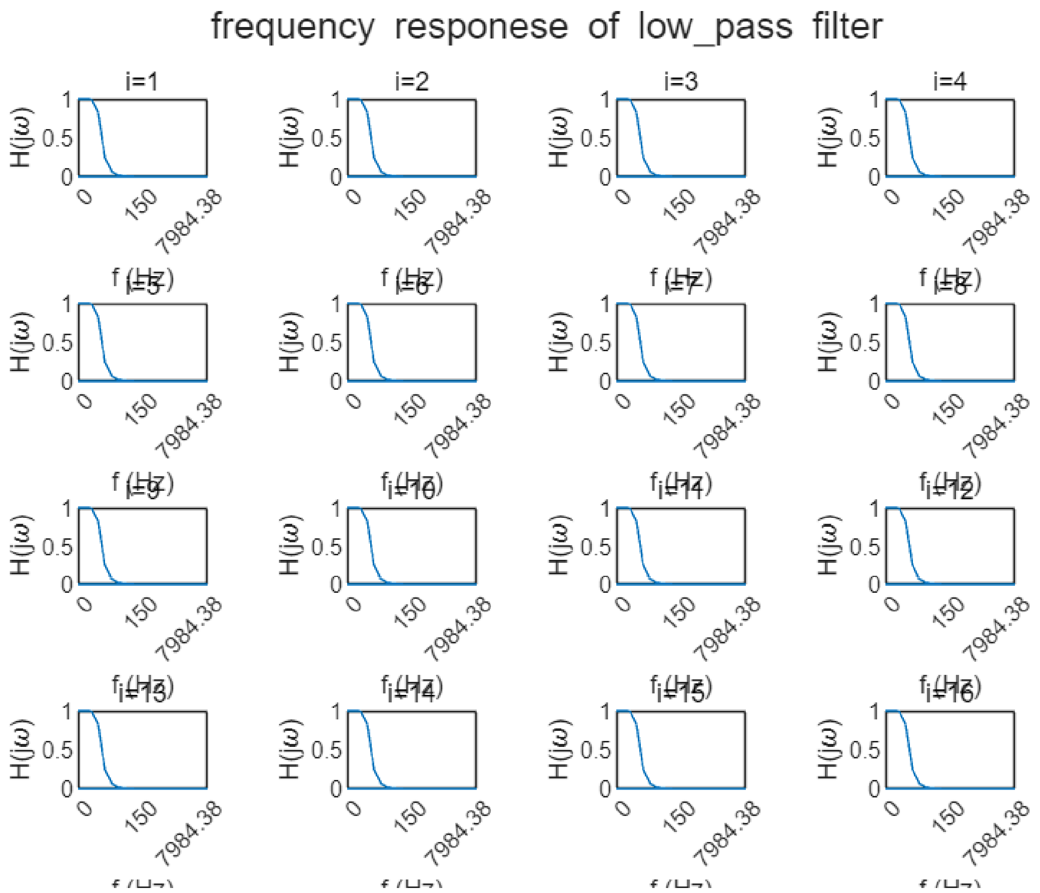

envelope
i=1
i=2
i=3
i=4
i=5
i=6
i=7
i=8
i=9
i=10
i=11
i=12
i=13
i=14
i=15
i=16
amplitude
t (s)
×10⁻³

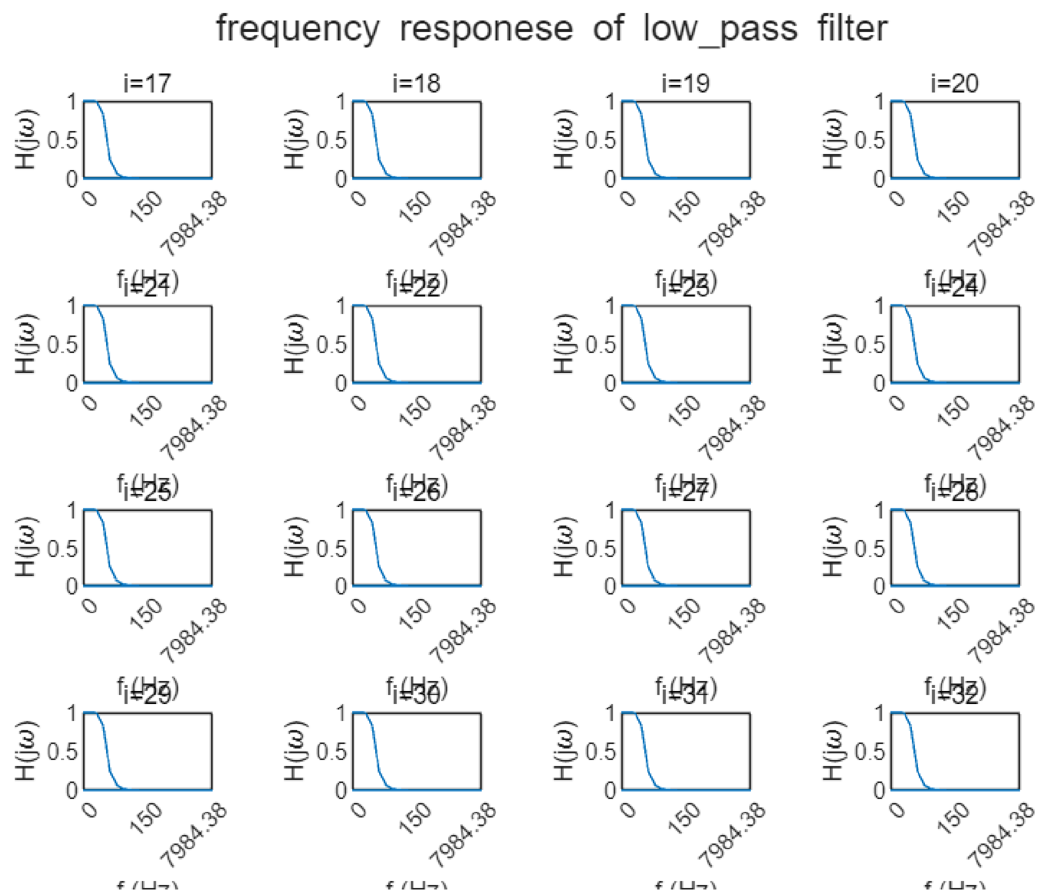
N = 32

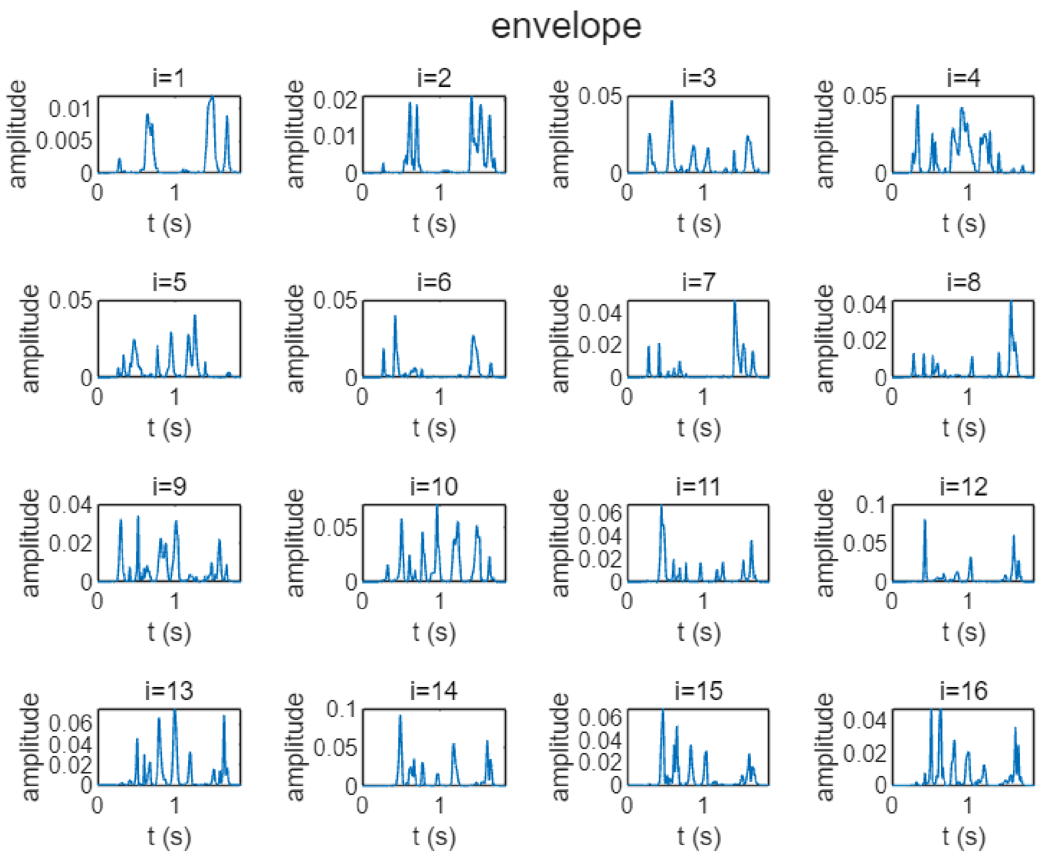
```
N_32 = speech_synthesis(sig, fs, 32, cut_off);
```

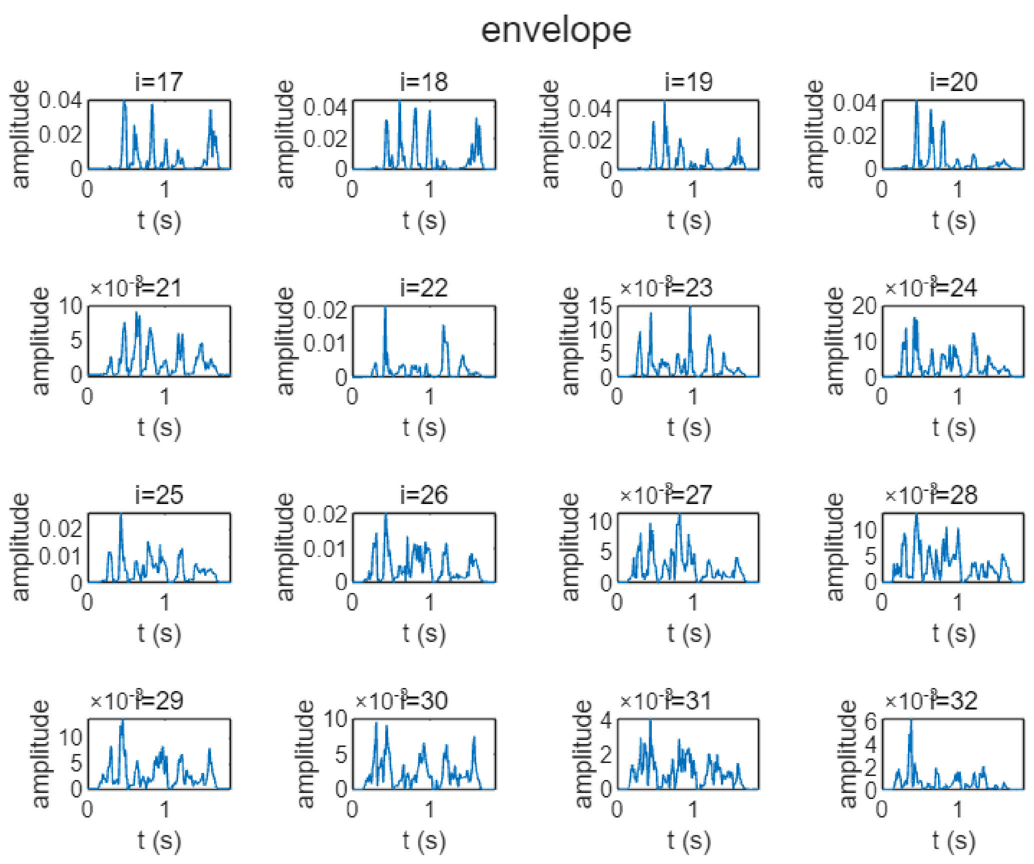



frequency responese of band_pass filter
i=17
i=18
i=19
i=20
i=21
i=22
i=23
i=24
i=25
i=26
i=27
i=28
i=29
i=30
i=31
i=32
H(jω)
f (Hz)
0
0.5
1
0
5000


frequency responese of low_pass filter
i=1
i=2
i=3
i=4
i=5
i=6
i=7
i=8
i=9
i=10
i=11
i=12
i=13
i=14
i=15
i=16
H(jω)
f (Hz)
0
0.5
1
0
150
7984.38


frequency responese of low_pass filter
i=17
i=18
i=19
i=20
i=21
i=22
i=23
i=24
i=25
i=26
i=27
i=28
i=29
i=30
i=31
i=32
H(jω)
f (Hz)
0
150
7984.38


envelope
i=1
i=2
i=3
i=4
i=5
i=6
i=7
i=8
i=9
i=10
i=11
i=12
i=13
i=14
i=15
i=16
amplitude
t (s)



Check energy normalization.

```
sig_norm = norm(sig);
feq(norm(N_4), sig_norm)
```

```
ans = logical
   1
```

```
feq(norm(N_8), sig_norm)
```

```
ans = logical
   1
```

```
feq(norm(N_16), sig_norm)
```

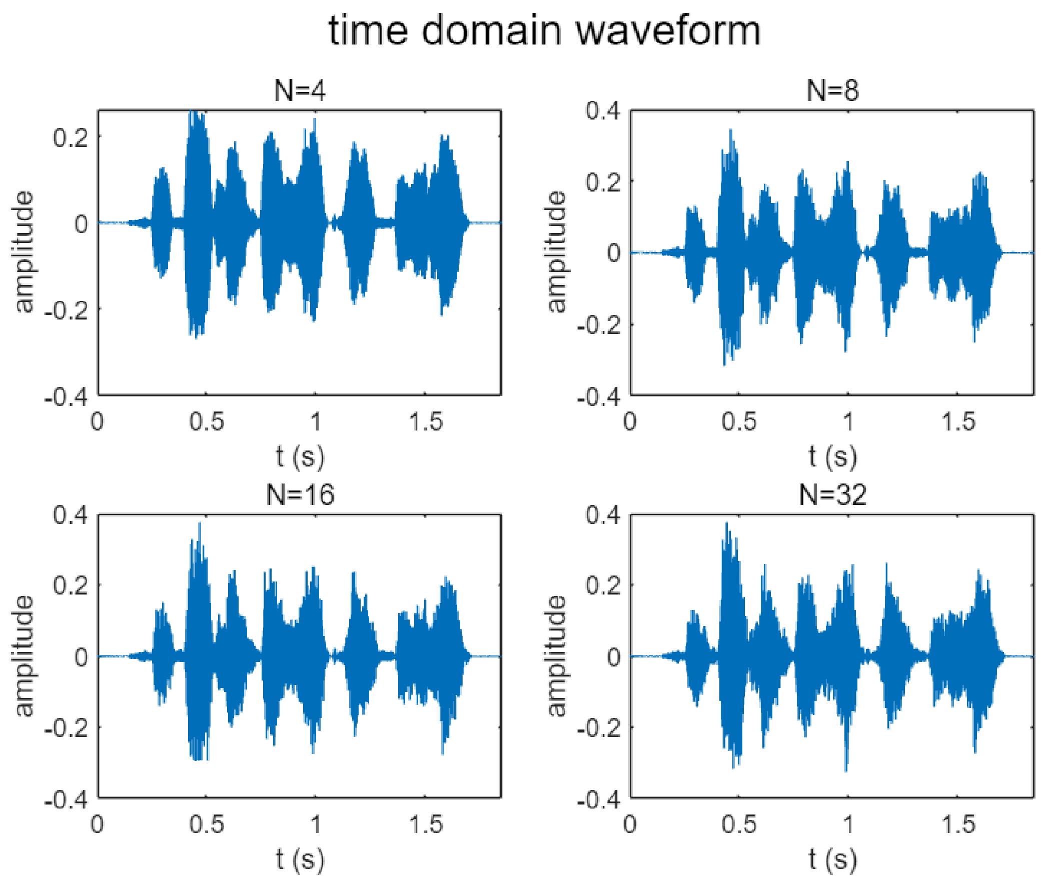
```
ans = logical
   1
```

```
feq(norm(N_32), sig_norm)
```

```
ans = logical
   1
```

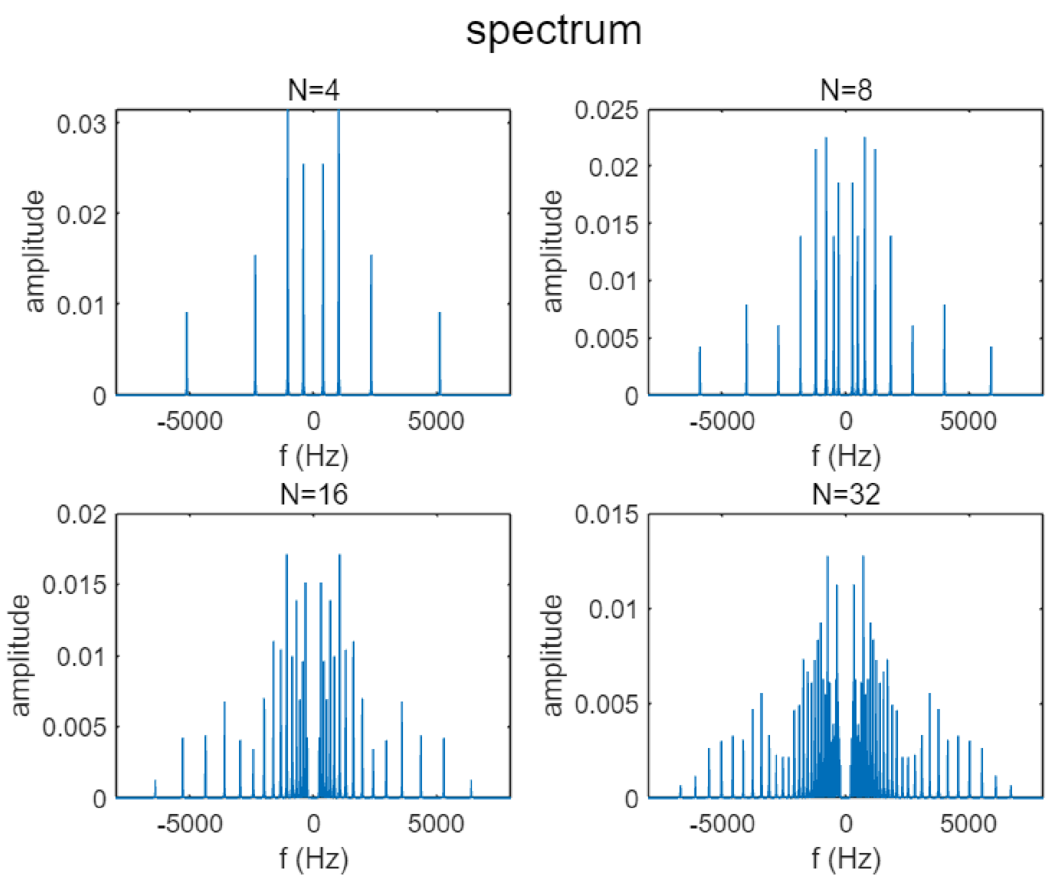
Plot each result.

```matlab
t = (0:length(sig)-1)'./fs;
figure;
sgtitle('time domain waveform');
subplot(2, 2, 1);
plot(t, N_4);
axis([t(1) t(end) ylim]);
title('N=4');
xlabel('t (s)');
ylabel('amplitude');
subplot(2, 2, 2);
plot(t, N_8);
axis([t(1) t(end) ylim]);
title('N=8');
xlabel('t (s)');
ylabel('amplitude');
subplot(2, 2, 3);
plot(t, N_16);
axis([t(1) t(end) ylim]);
title('N=16');
xlabel('t (s)');
ylabel('amplitude');
subplot(2, 2, 4);
plot(t, N_32);
axis([t(1) t(end) ylim]);
title('N=32');
xlabel('t (s)');
ylabel('amplitude');
```



```
if mod(length(sig), 2) == 1
    freqency = linspace(-fs/2, fs/2, length(sig));
else
    freqency = linspace(-fs/2, fs/2, length(sig) + 1);
    freqency = freqency(1:end-1);
end
figure;
sgtitle('spectrum');
subplot(2, 2, 1);
plot(freqency, abs(fftshift(fft(N_4))/fs));
axis([freqency(1) freqency(end) ylim]);
title('N=4');
xlabel('f (Hz)');
ylabel('amplitude');
subplot(2, 2, 2);
plot(freqency, abs(fftshift(fft(N_8))/fs));
axis([freqency(1) freqency(end) ylim]);
title('N=8');
xlabel('f (Hz)');
ylabel('amplitude');
subplot(2, 2, 3);
plot(freqency, abs(fftshift(fft(N_16))/fs));
axis([freqency(1) freqency(end) ylim]);
title('N=16');
```

```
xlabel('f (Hz)');
ylabel('amplitude');
subplot(2, 2, 4);
plot(freqency, abs(fftshift(fft(N_32))/fs));
axis([freqency(1) freqency(end) ylim]);
title('N=32');
xlabel('f (Hz)');
ylabel('amplitude');
```



Save the processed audio.

```
audiowrite('proj1/T1_4_50.wav', N_4, fs);
audiowrite('proj1/T1_8_50.wav', N_8, fs);
audiowrite('proj1/T1_16_50.wav', N_16, fs);
audiowrite('proj1/T1_32_50.wav', N_32, fs);
```

Destcription

1、题目要求

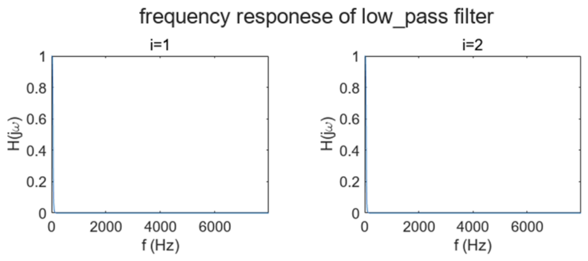
1、将 LPF 截止频率设为 50 赫兹。

2、将波段数改为 N=4、N=8、N=16 和 N=32。

3、保存这些条件下的波形文件，并描述波段数如何影响合成句子的可懂度（即能听懂多少单词）。

2、绘图及验证

（1）低通/带通滤波器系统验证



（2）结果图：时域波形【横坐标以时间秒为单位，纵坐标为幅度】，频谱【横坐标以频率Hz为单位，纵坐标为幅度谱 】，包络图均在对应的代码段落后

（3）使用norm函数与进行浮点数数值比较的feq函数进行能量归一化验证

3、对结果的分析和解释

分析：频域上阶数越高频谱越密集

阶数为4，8时难以辨认，16时基本可以辨认，32时完全可辨认

解释：阶数越高时，频域上的分段越多，合成的信号与原始信号越接近，还原性可分辨性越强

结论：语音合成信号的质量随着分段数的增加而提高

Task 2

- Set the number of bands N=8.
- Implement tone-vocoder by changing the LPF cut-off frequency to 20 Hz, 50 Hz, 100 Hz, and 400 Hz.
- Describe how the LPF cut-off frequency affects the intelligibility of synthesized sentence.

Initialize and load audio.

```
clear;clc;
[sig, fs] = audioread('proj1/C_01_02.wav');
```

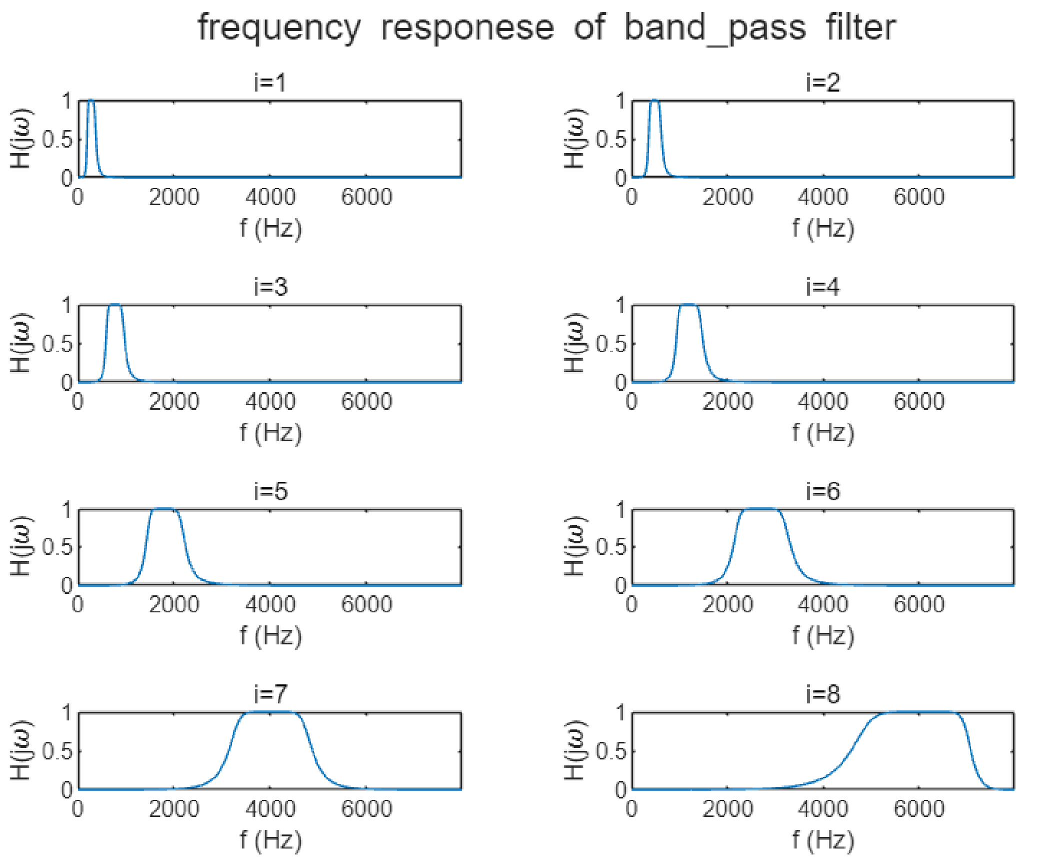
Set the number of bands.

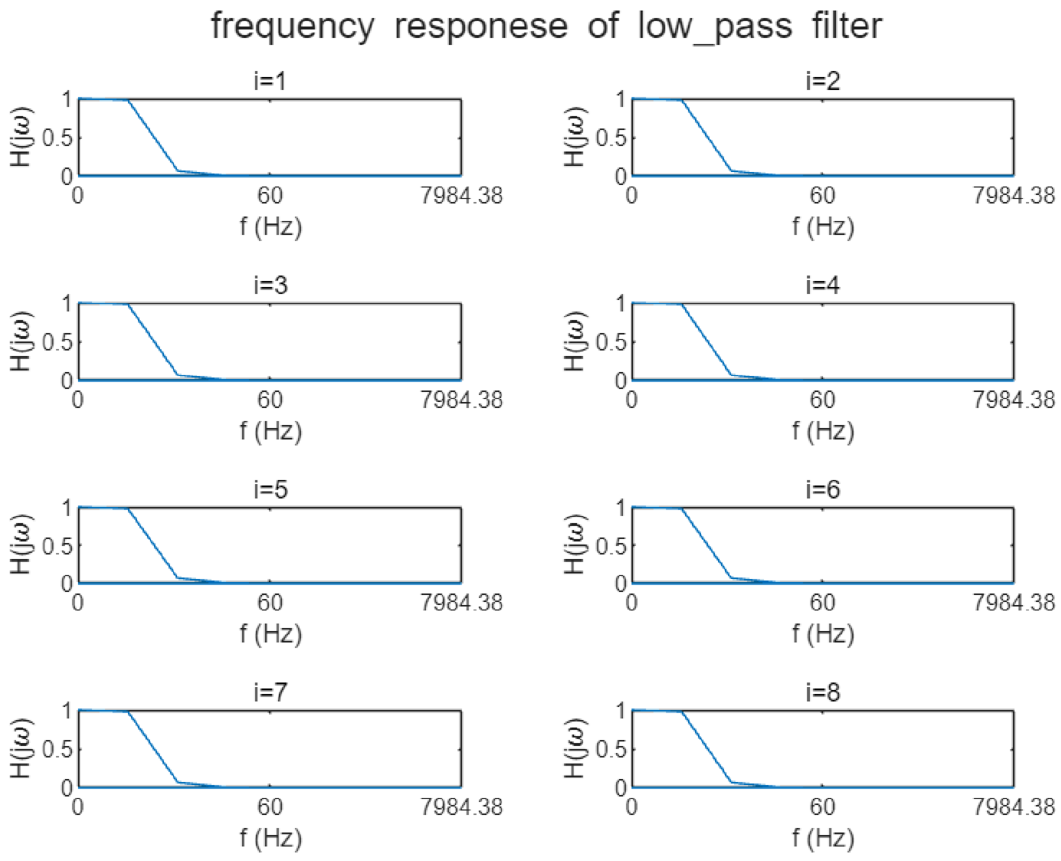
```
N = 8;
```

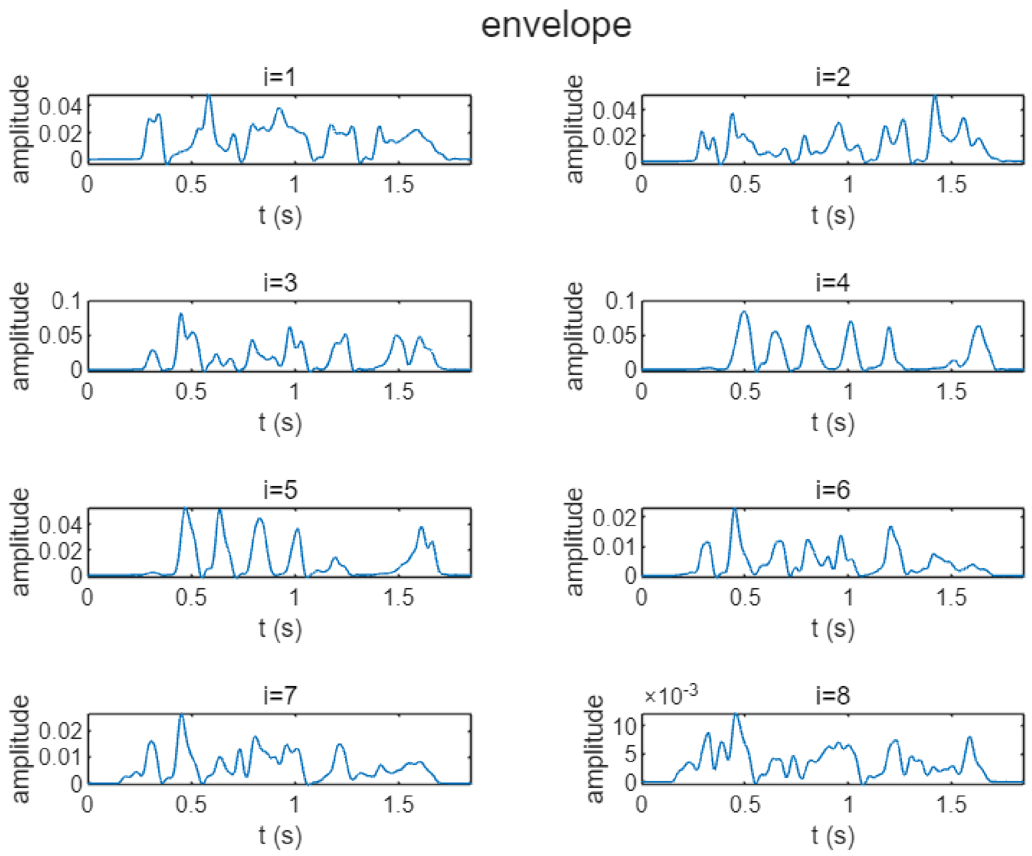
Implement tone-vocoder.

cut_off = 20 (Hz)

```
cut_off_20 = speech_synthesis(sig, fs, N, 20);
```

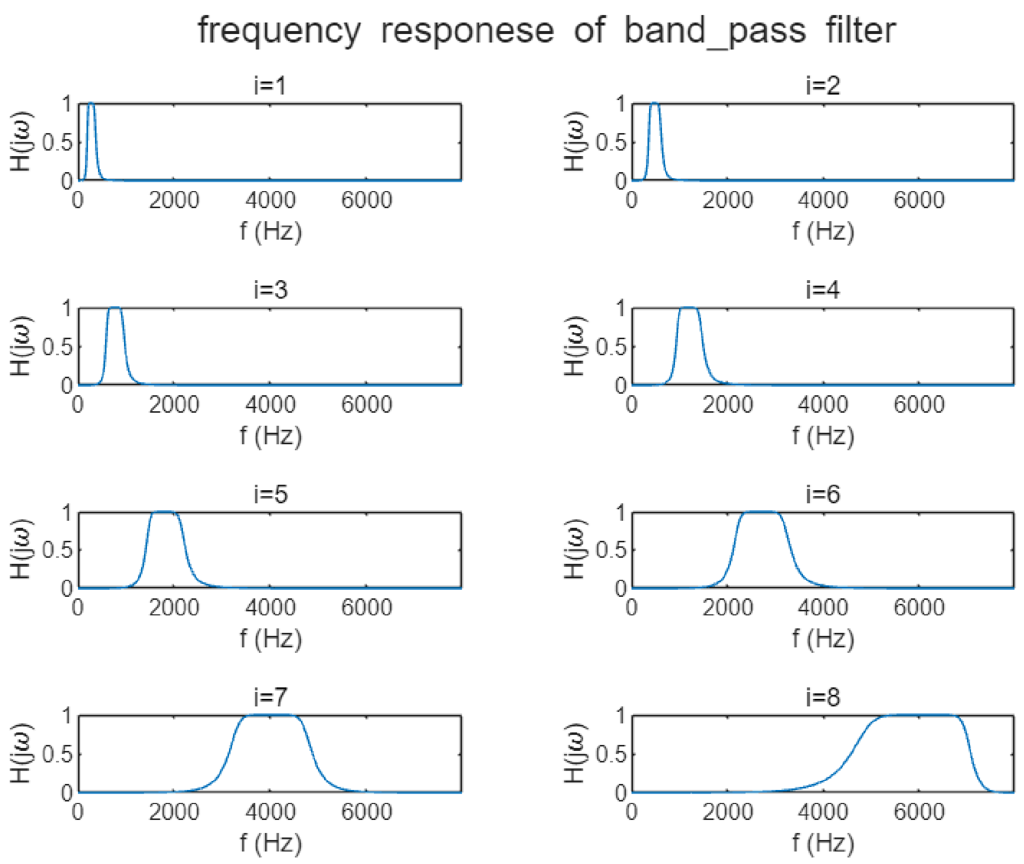

frequency responese of band_pass filter
i=1
i=2
i=3
i=4
i=5
i=6
i=7
i=8
H(jω)
f (Hz)
0
0.5
1
0
2000
4000
6000

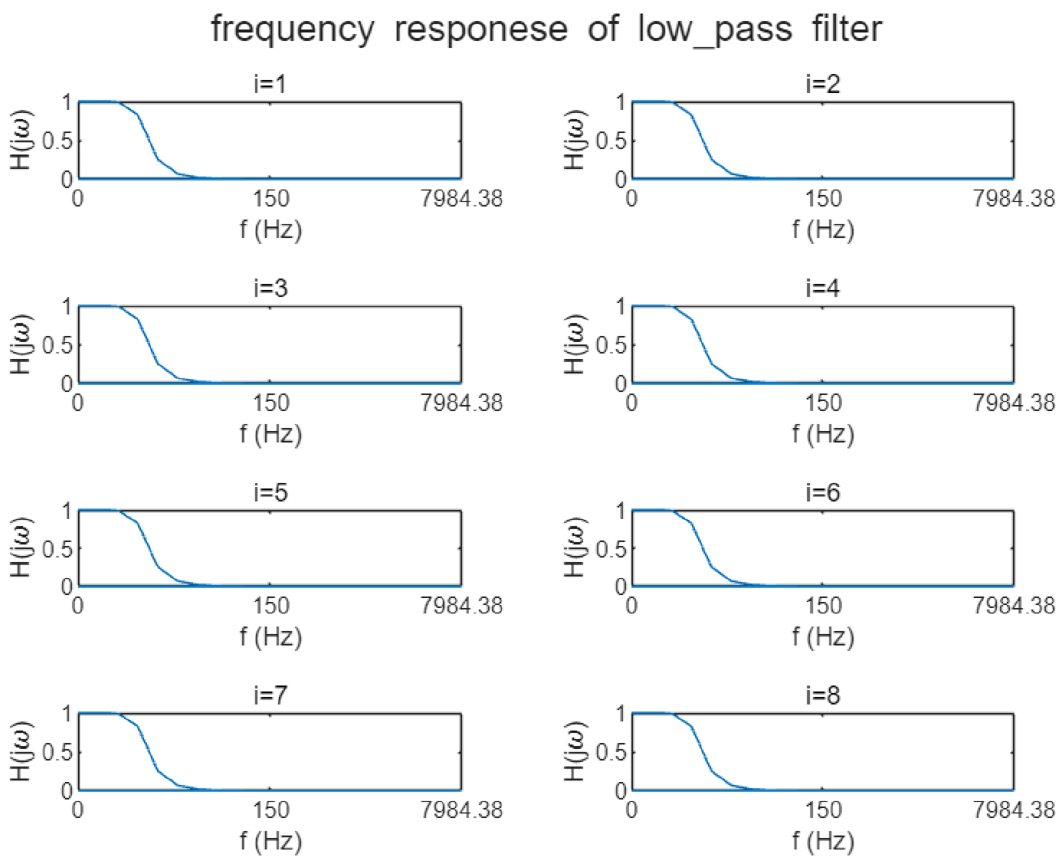

frequency responese of low_pass filter
i=1
i=2
i=3
i=4
i=5
i=6
i=7
i=8
H(jω)
f (Hz)
0
0.5
1
0
60
7984.38

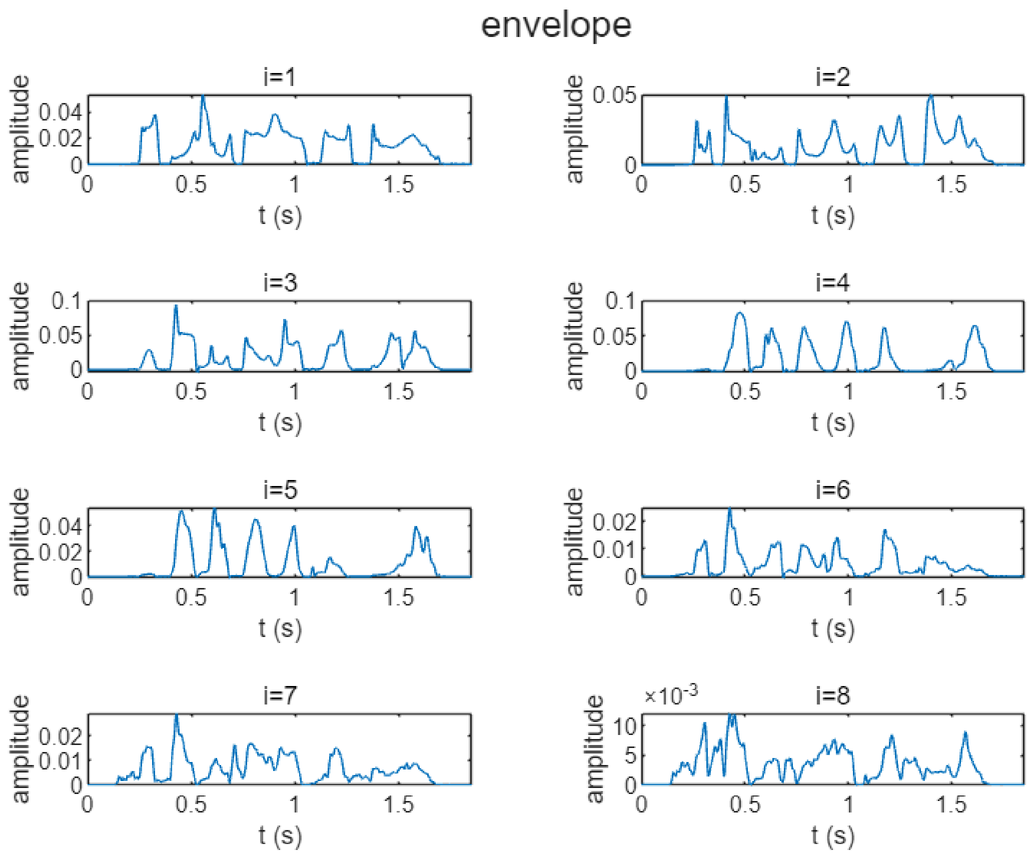


cut_off = 50 (Hz)

```
cut_off_50 = speech_synthesis(sig, fs, N, 50);
```

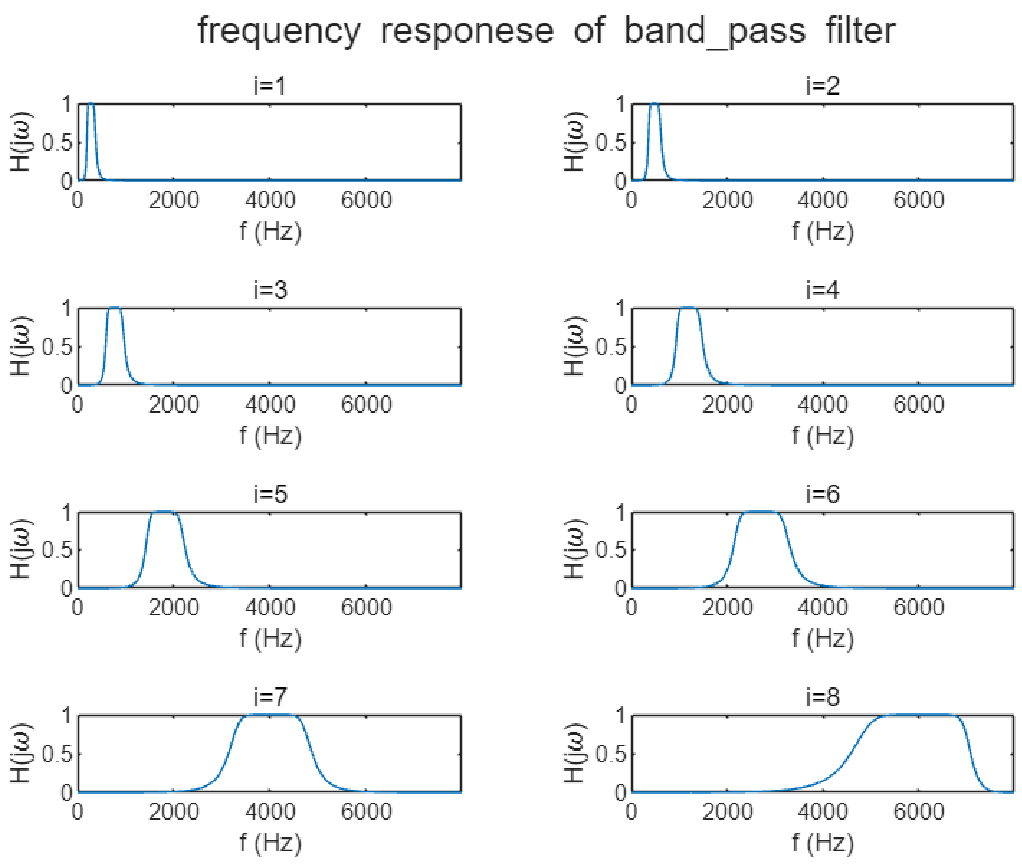

frequency responese of band_pass filter
i=1
i=2
i=3
i=4
i=5
i=6
i=7
i=8
H(jω)
f (Hz)
0
2000
4000
6000
0.5
1

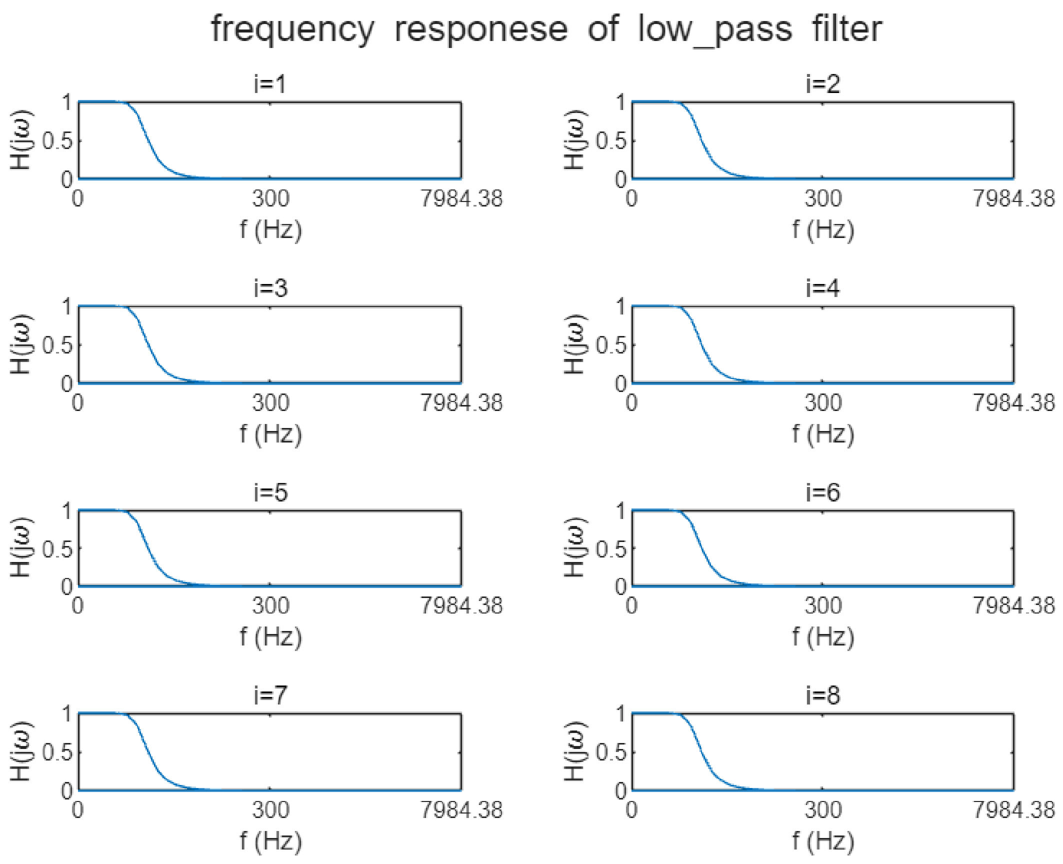

frequency responese of low_pass filter
i=1
i=2
i=3
i=4
i=5
i=6
i=7
i=8
H(jω)
f (Hz)
0
150
7984.38
0.5
1

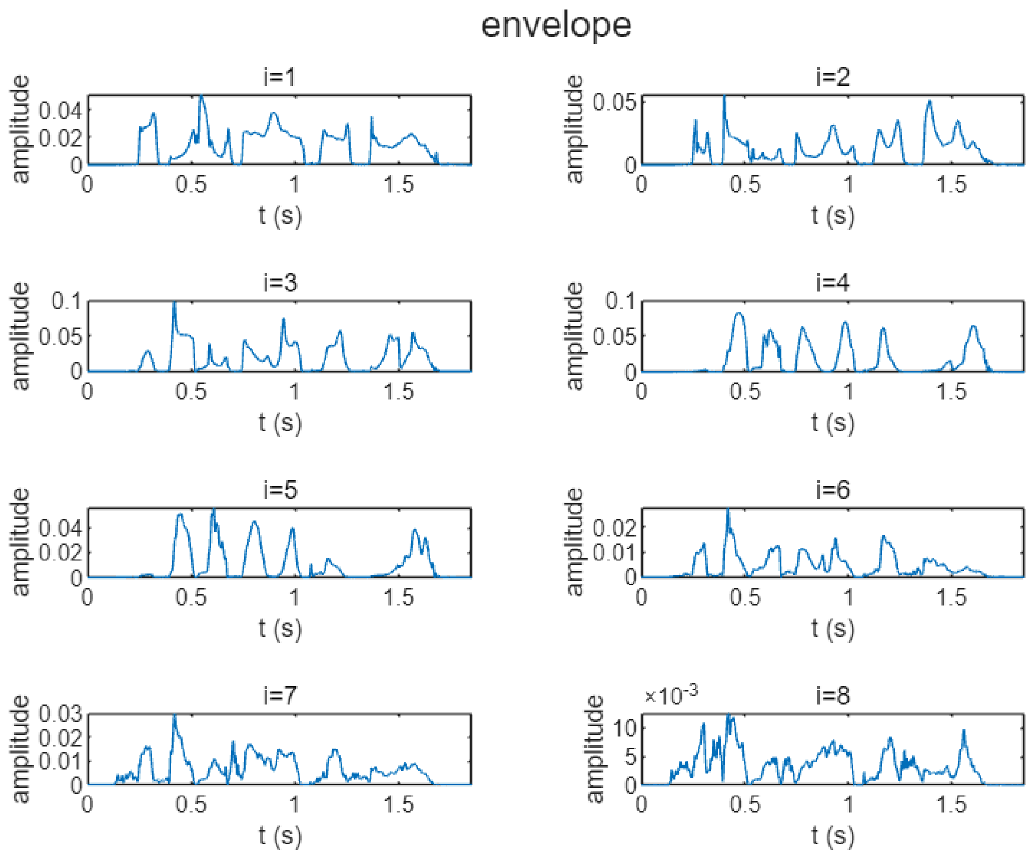


cut_off = 100 (Hz)

```
cut_off_100 = speech_synthesis(sig, fs, N, 100);
```

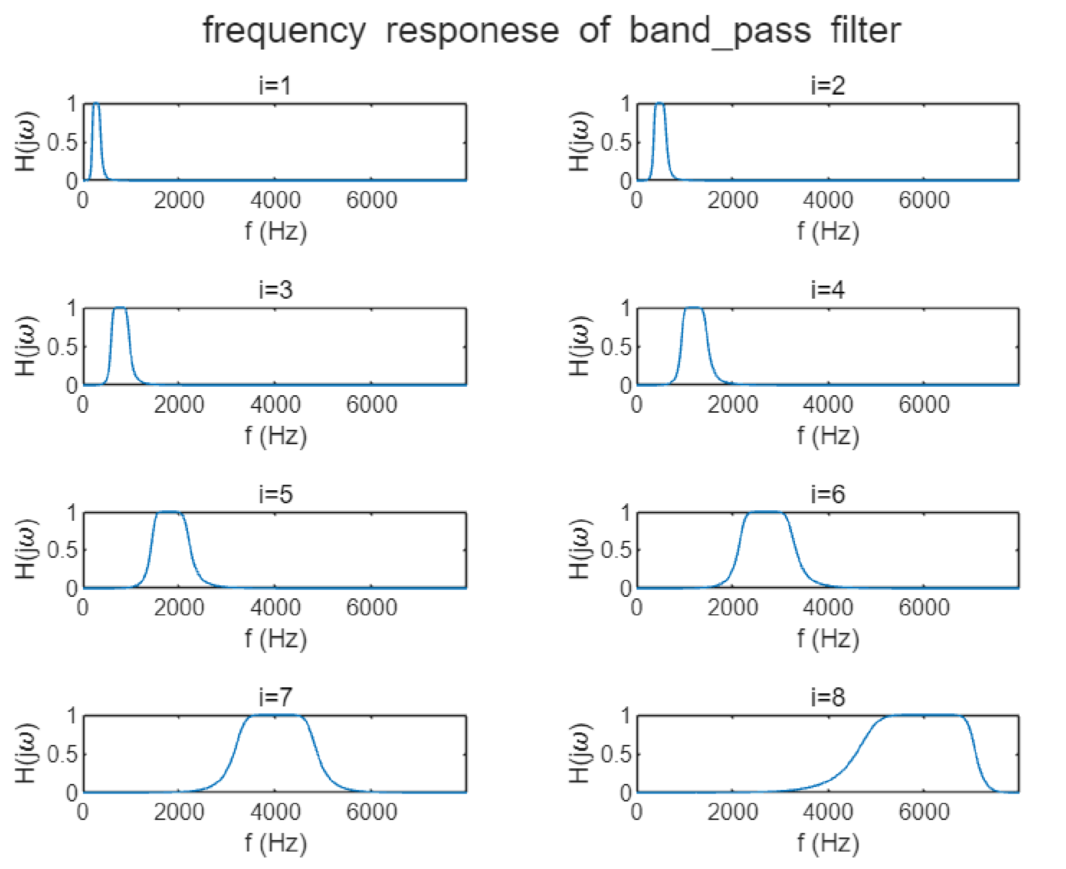

frequency responese of band_pass filter
i=1
i=2
i=3
i=4
i=5
i=6
i=7
i=8
H(jω)
f (Hz)
0
2000
4000
6000
0.5
1

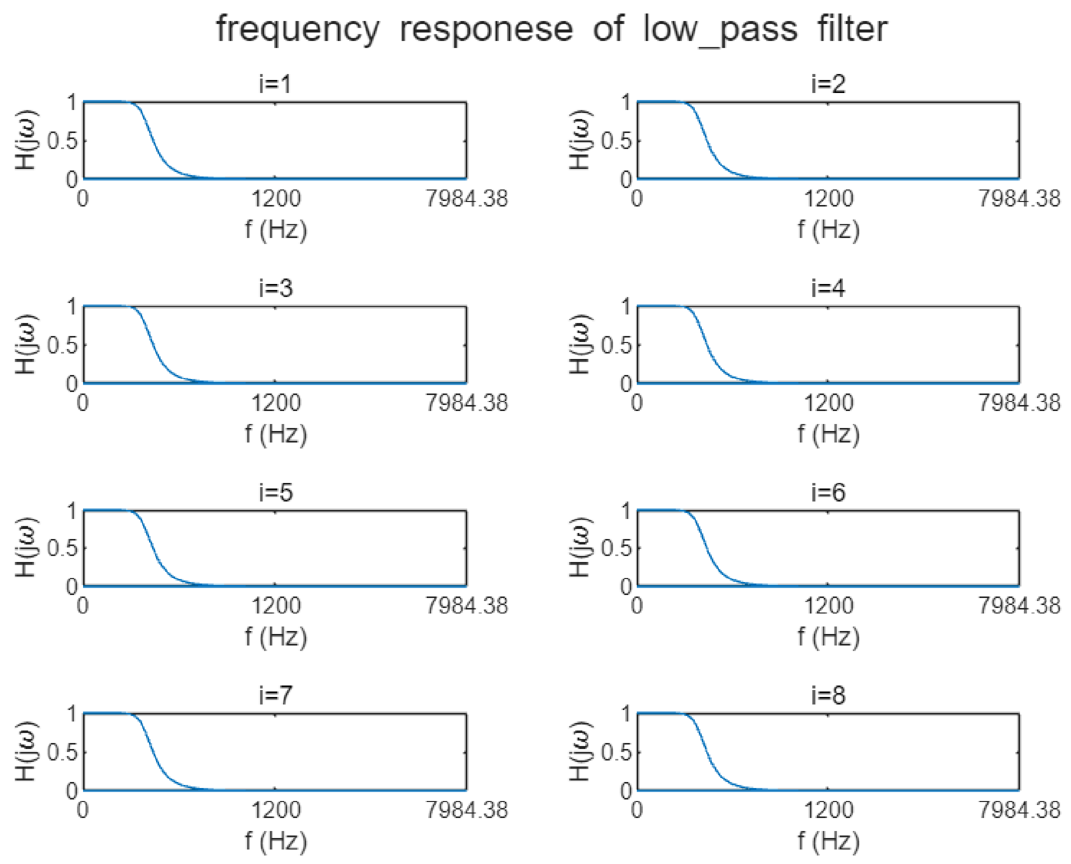

frequency responese of low_pass filter
i=1
i=2
i=3
i=4
i=5
i=6
i=7
i=8
H(jω)
f (Hz)
0
300
7984.38
0.5
1

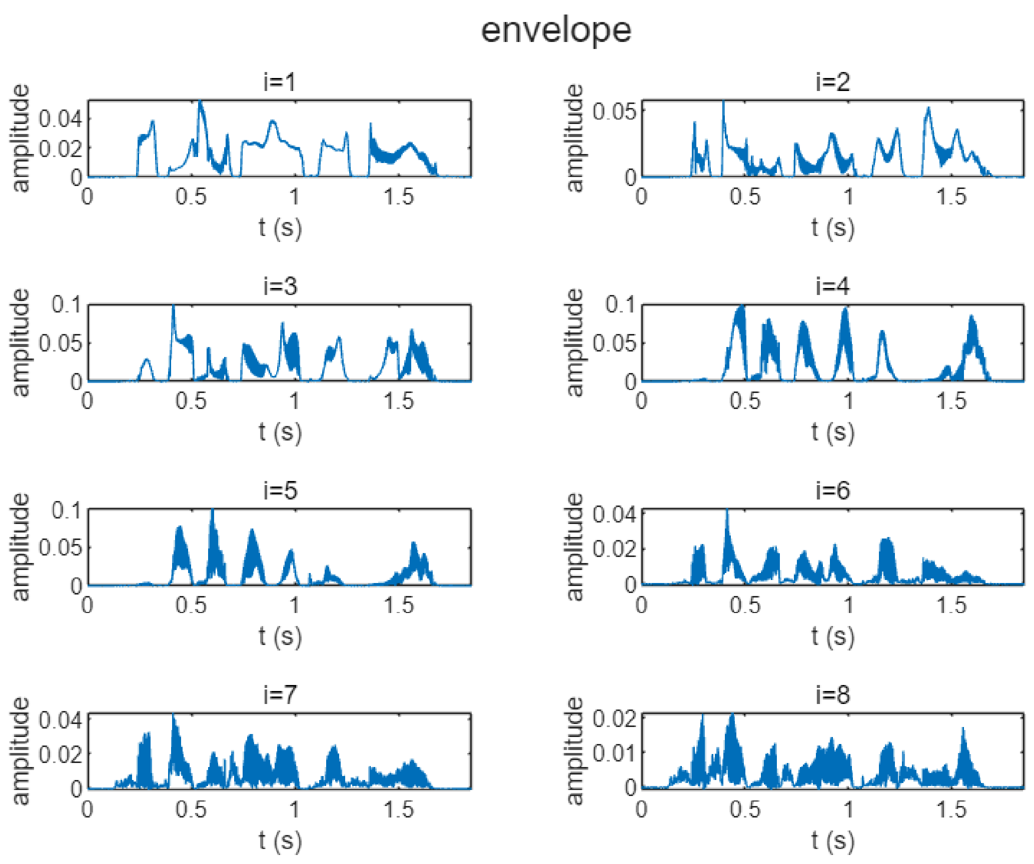


cut_off = 400 (Hz)

```
cut_off_400 = speech_synthesis(sig, fs, N, 400);
```


frequency responese of band_pass filter
i=1
i=2
i=3
i=4
i=5
i=6
i=7
i=8
H(jω)
1
0.5
0
0
2000
4000
6000
f (Hz)


frequency responese of low_pass filter
i=1
i=2
i=3
i=4
i=5
i=6
i=7
i=8
H(jω)
1
0.5
0
0
1200
7984.38
f (Hz)



Check energy normalization.

```
sig_norm = norm(sig);
feq(norm(cut_off_20), sig_norm)
```

```
ans = logical
   1
```

```
feq(norm(cut_off_50), sig_norm)
```

```
ans = logical
   1
```

```
feq(norm(cut_off_100), sig_norm)
```

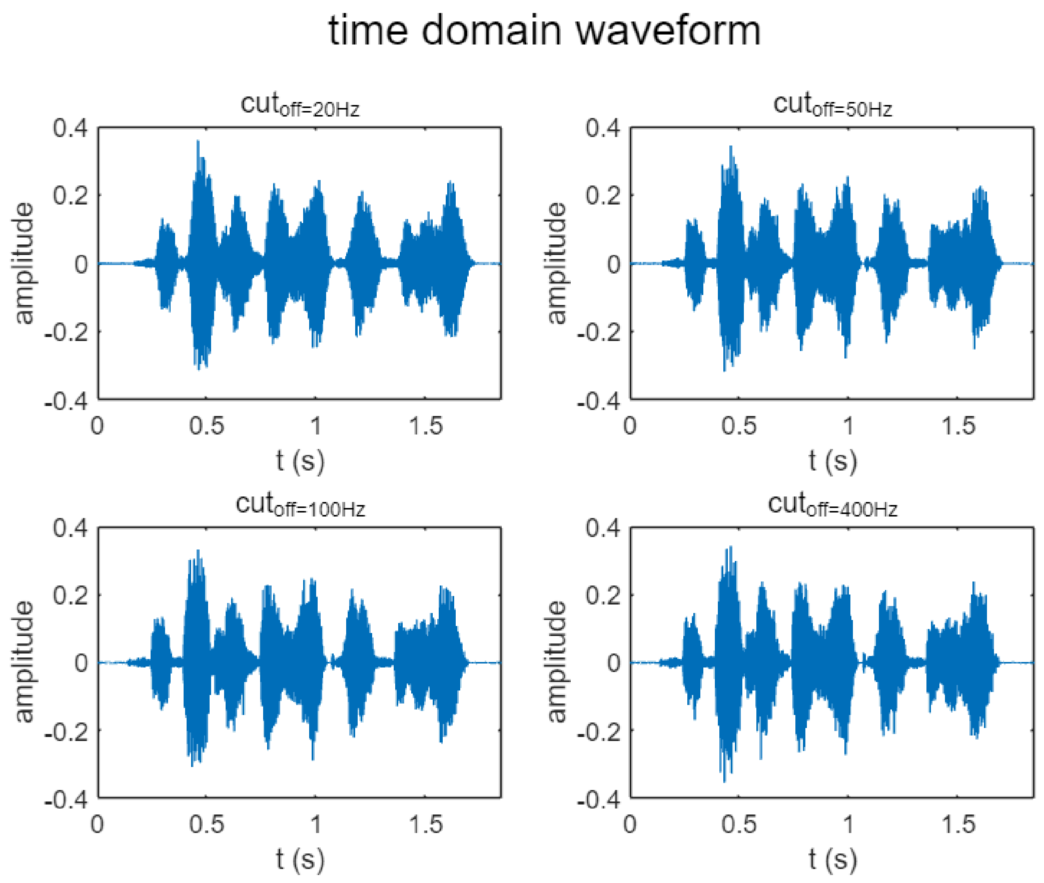
```
ans = logical
   1
```

```
feq(norm(cut_off_400), sig_norm)
```

```
ans = logical
   1
```

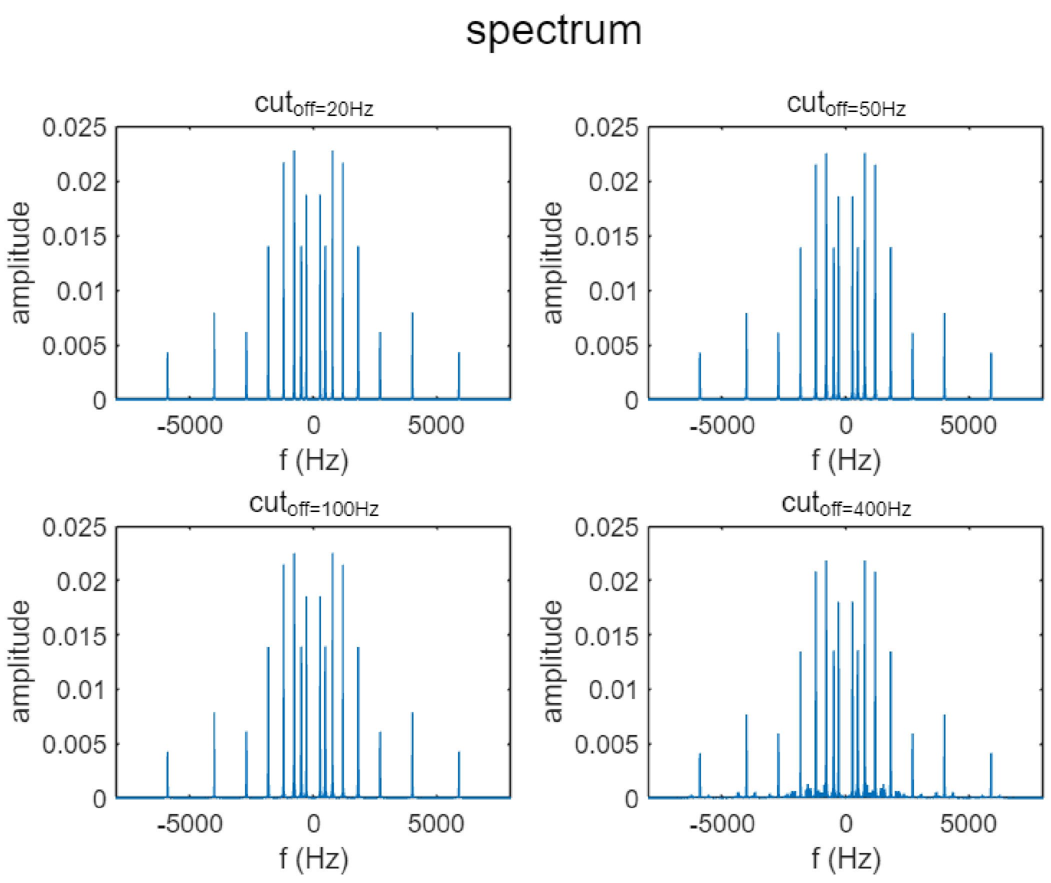
Plot each result.

```
t = (0:length(sig)-1)'./fs;
figure;
sgtitle('time domain waveform');
subplot(2, 2, 1);
plot(t, cut_off_20);
axis([t(1) t(end) ylim]);
title('cut_off=20Hz');
xlabel('t (s)');
ylabel('amplitude');
subplot(2, 2, 2);
plot(t, cut_off_50);
axis([t(1) t(end) ylim]);
title('cut_off=50Hz');
xlabel('t (s)');
ylabel('amplitude');
subplot(2, 2, 3);
plot(t, cut_off_100);
axis([t(1) t(end) ylim]);
title('cut_off=100Hz');
xlabel('t (s)');
ylabel('amplitude');
subplot(2, 2, 4);
plot(t, cut_off_400);
axis([t(1) t(end) ylim]);
title('cut_off=400Hz');
xlabel('t (s)');
ylabel('amplitude');
```



```
if mod(length(sig), 2) == 1
    freqency = linspace(-fs/2, fs/2, length(sig));
else
    freqency = linspace(-fs/2, fs/2, length(sig) + 1);
    freqency = freqency(1:end-1);
end
figure;
sgtitle('spectrum');
subplot(2, 2, 1);
plot(freqency, abs(fftshift(fft(cut_off_20))/fs));
axis([freqency(1) freqency(end) ylim]);
title('cut_off=20Hz');
xlabel('f (Hz)');
ylabel('amplitude');
subplot(2, 2, 2);
plot(freqency, abs(fftshift(fft(cut_off_50))/fs));
axis([freqency(1) freqency(end) ylim]);
title('cut_off=50Hz');
xlabel('f (Hz)');
ylabel('amplitude');
subplot(2, 2, 3);
plot(freqency, abs(fftshift(fft(cut_off_100))/fs));
axis([freqency(1) freqency(end) ylim]);
title('cut_off=100Hz');
```

```
xlabel('f (Hz)');
ylabel('amplitude');
subplot(2, 2, 4);
plot(freqency, abs(fftshift(fft(cut_off_400))/fs));
axis([freqency(1) freqency(end) ylim]);
title('cut_off=400Hz');
xlabel('f (Hz)');
ylabel('amplitude');
```



Save the processed audio.

```
audiowrite('proj1/T2_8_20.wav', cut_off_20, fs);
audiowrite('proj1/T2_8_50.wav', cut_off_50, fs);
audiowrite('proj1/T2_8_100.wav', cut_off_100, fs);
audiowrite('proj1/T2_8_400.wav', cut_off_400, fs);
```

Destcription

1、题目要求

Task2 探究的是提高截止频率对还原效果的影响。LPF 截止频率继续增加 能否提高还原效果，有没有上限?

①设置带数 N=8。

②调用 tone-vocoder 改变 LPF 截止频率依次为 20 赫兹，50 赫兹，100 赫兹 和 400 赫兹。

③说明低通滤波器的截止频率如何影响还原效果。

2、绘图及验证

（1）结果图：时域波形【横坐标以时间秒为单位，纵坐标为幅度】，频谱【横坐标以频率Hz为单位，纵坐标为幅度谱】，包络图均在对应的代码段落后

（2）使用norm函数与进行浮点数数值比较的feq函数进行能量归一化验证

3、对结果的分析和解释

分析：LPF截止频率高时，包络图蕴含更多高频信号，声音的可辨认程度增强，在LPF=400hz时声音可模糊辨认，但因为分段数N=8不够，所以无法清晰辨认

解释：LPF截止频率变高时，包络图蕴含更多高频信号，表现为出现更多上下波动。更多的语音信号被保留在了频谱上，所以合成语音的还原程度会提高

结论：语音合成的质量随着对包络滤波的截止频率的变高而提升

Task 3

- Generate a noisy signal by summing the clean sentence and speech-shaped noise (SSN) at SNR -5 dB.
- Set LPF cut-off frequency to 50 Hz.
- Implement tone-vocoder by changing the number of bands to N=4, N=8, N=16, and N=32.
- Describe how the number of bands affects the intelligibility of synthesized sentence and compare findings with those obtained in task 1.

Initialize and load audio.

```
clear;clc;
[sig, fs] = audioread('proj1/C_01_02.wav');
```

Generate -5 dB SNR.

```
snr = sig / sqrt(10);
```

Generate speech-shaped noise (SSN).

```
ssn = get_ssn(snr, fs) .* 10^(1/4);
```

Generate noisy signal.

```
noisy_sig = sig + ssn;
```

Check SNR -5 dB.

```
feq(20 * log10(norm(sig)/norm(ssn)), -5)
```

```
ans = logical
   1
```

Set LPF cut-off frequency.

```
cut_off = 50;
```

Implement tone-vocoder.

N = 4

```
N_4 = speech_synthesis(noisy_sig, fs, 4, cut_off, false);
```

N = 8

```
N_8 = speech_synthesis(noisy_sig, fs, 8, cut_off, false);
```

N = 16

```
N_16 = speech_synthesis(noisy_sig, fs, 16, cut_off, false);
```

N = 32

```
N_32 = speech_synthesis(noisy_sig, fs, 32, cut_off, false);
```

Check energy normalization.

```
noisy_sig_norm = norm(noisy_sig);
feq(norm(N_4), noisy_sig_norm)
```

```
ans = logical
   1
```

```
feq(norm(N_8), noisy_sig_norm)
```

```
ans = logical
   1
```

```
feq(norm(N_16), noisy_sig_norm)
```

```
ans = logical
   1
```

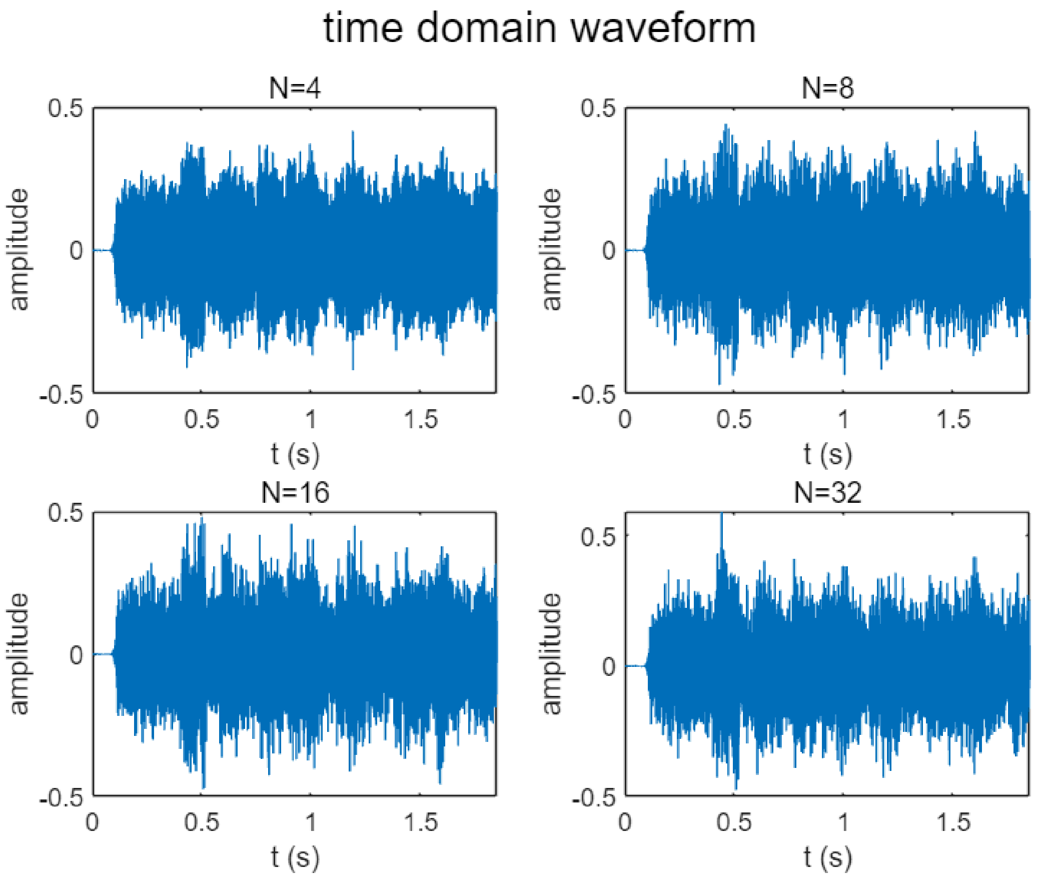
```
feq(norm(N_32), noisy_sig_norm)
```

```
ans = logical
   1
```

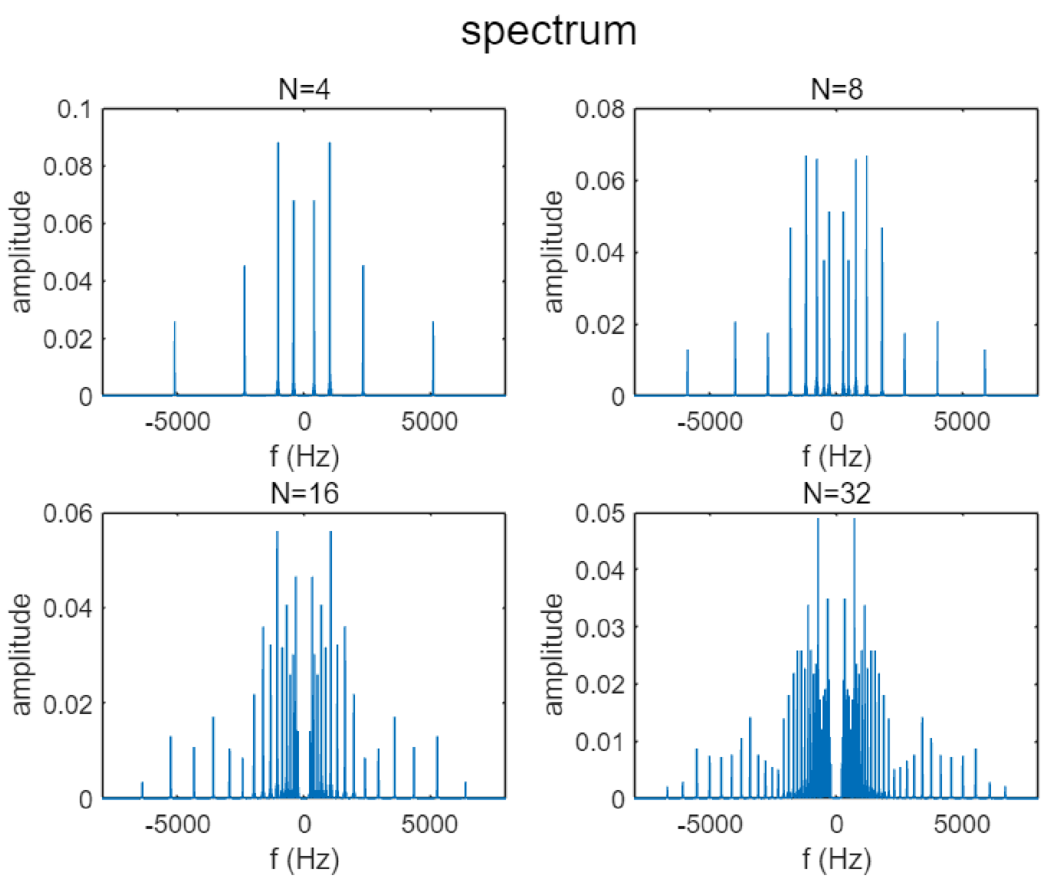
Plot each result.

```
t = (0:length(sig)-1)'./fs;
figure;
sgtitle('time domain waveform');
```

```
subplot(2, 2, 1);
plot(t, N_4);
axis([t(1) t(end) ylim]);
title('N=4');
xlabel('t (s)');
ylabel('amplitude');
subplot(2, 2, 2);
plot(t, N_8);
axis([t(1) t(end) ylim]);
title('N=8');
xlabel('t (s)');
ylabel('amplitude');
subplot(2, 2, 3);
plot(t, N_16);
axis([t(1) t(end) ylim]);
title('N=16');
xlabel('t (s)');
ylabel('amplitude');
subplot(2, 2, 4);
plot(t, N_32);
axis([t(1) t(end) ylim]);
title('N=32');
xlabel('t (s)');
ylabel('amplitude');
```



```
if mod(length(sig), 2) == 1
    freqency = linspace(-fs/2, fs/2, length(sig));
else
    freqency = linspace(-fs/2, fs/2, length(sig) + 1);
    freqency = freqency(1:end-1);
end
figure;
sgtitle('spectrum');
subplot(2, 2, 1);
plot(freqency, abs(fftshift(fft(N_4))/fs));
axis([freqency(1) freqency(end) ylim]);
title('N=4');
xlabel('f (Hz)');
ylabel('amplitude');
subplot(2, 2, 2);
plot(freqency, abs(fftshift(fft(N_8))/fs));
axis([freqency(1) freqency(end) ylim]);
title('N=8');
xlabel('f (Hz)');
ylabel('amplitude');
subplot(2, 2, 3);
plot(freqency, abs(fftshift(fft(N_16))/fs));
axis([freqency(1) freqency(end) ylim]);
title('N=16');
xlabel('f (Hz)');
ylabel('amplitude');
subplot(2, 2, 4);
plot(freqency, abs(fftshift(fft(N_32))/fs));
axis([freqency(1) freqency(end) ylim]);
title('N=32');
xlabel('f (Hz)');
ylabel('amplitude');
```



Save the processed audio.

```
audiowrite('proj1/T3_4_50.wav', N_4, fs);
audiowrite('proj1/T3_8_50.wav', N_8, fs);
audiowrite('proj1/T3_16_50.wav', N_16, fs);
audiowrite('proj1/T3_32_50.wav', N_32, fs);
```

Destcription

1、题目要求

在 task1 的基础上，task3 探究了在加入信噪比为-5dB 的噪声后，通频带数对 信号的还原程度的影响。

①生成一个信噪比-5dB的语谱噪声。

②设置低通滤波器的截止频率为 50Hz。

③利用 tone_vocoder 在 LPF 为 通带数目分别为 4，8，16，32 的条件 下还原信号。

④描述通频带数目如何影响声音还原效果。与task1进行对比。

2、绘图及验证

（1）通过判断相等的feq函数验证信噪比计算后为-5dB

（2）结果图：时域波形【横坐标以时间秒为单位，纵坐标为幅度】，频谱【横坐标以频率Hz为单位，纵坐标为幅度谱 】，均在对应的代码段落后

（3）使用norm函数与进行浮点数数值比较的feq函数进行能量归一化验证

3、对结果的分析和解释

分析：在时域图上，没有随着 N 的增大体现出明显差别； 但是在频谱图上可以观察到随着N的增大，复原信号保留的频率分量增多，频谱变得更加密集。

解释：阶数越高时，频域上的分段越多，合成的信号与原始信号越接近，但是由于混合了噪声，导致复原的效果变差 。

结论： 对比Task1，带有SSN的混合语音信号的语音合成的质量明显变差。 即使增大带数N，复原信号的质量也提升不多，无法通过人耳听出复原效果，在N = 32时仍然无法听清语音信号的内容。

Task 4

- Generate a noisy signal (summing clean sentence and SSN) at SNR -5 dB.
- Set the number of bands to N=8.
- Implement tone-vocoder by changing the LPF cut-off frequency to 20 Hz, 50 Hz, 100 Hz, and 400 Hz.
- Describe how the LPF cut-off frequency affects the intelligibility of synthesized sentence.

Initialize and load audio.

```
clear;clc;
[sig, fs] = audioread('proj1/C_01_02.wav');
```

Generate -5 dB SNR.

```
snr = sig / sqrt(10);
```

Generate speech-shaped noise (SSN).

```
ssn = get_ssn(snr, fs) .* 10^(1/4);
```

Generate noisy signal.

```
noisy_sig = sig + ssn;
```

Check SNR -5 dB.

```
feq(20 * log10(norm(sig)/norm(ssn)), -5)
```

```
ans = logical
   1
```

Set the number of bands.

```
N = 8;
```

Implement tone-vocoder.

cut_off = 20 (Hz)

```
cut_off_20 = speech_synthesis(noisy_sig, fs, N, 20, false);
```

cut_off = 50 (Hz)

```
cut_off_50 = speech_synthesis(noisy_sig, fs, N, 50, false);
```

cut_off = 100 (Hz)

```
cut_off_100 = speech_synthesis(noisy_sig, fs, N, 100, false);
```

cut_off = 400 (Hz)

```
cut_off_400 = speech_synthesis(noisy_sig, fs, N, 400, false);
```

Check energy normalization.

```
noisy_sig_norm = norm(noisy_sig);
feq(norm(cut_off_20), noisy_sig_norm)
```

```
ans = logical
   1
```

```
feq(norm(cut_off_50), noisy_sig_norm)
```

```
ans = logical
   1
```

```
feq(norm(cut_off_100), noisy_sig_norm)
```

```
ans = logical
   1
```

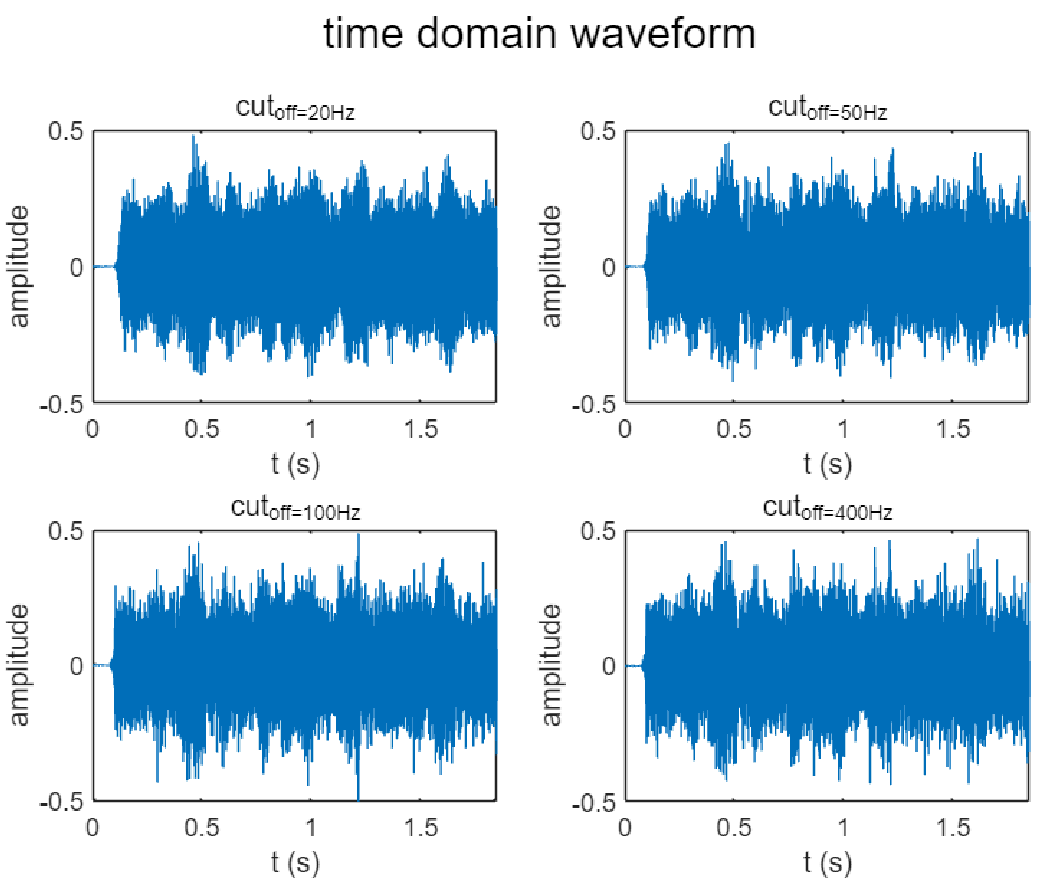
```
feq(norm(cut_off_400), noisy_sig_norm)
```

```
ans = logical
   1
```

Plot each result.

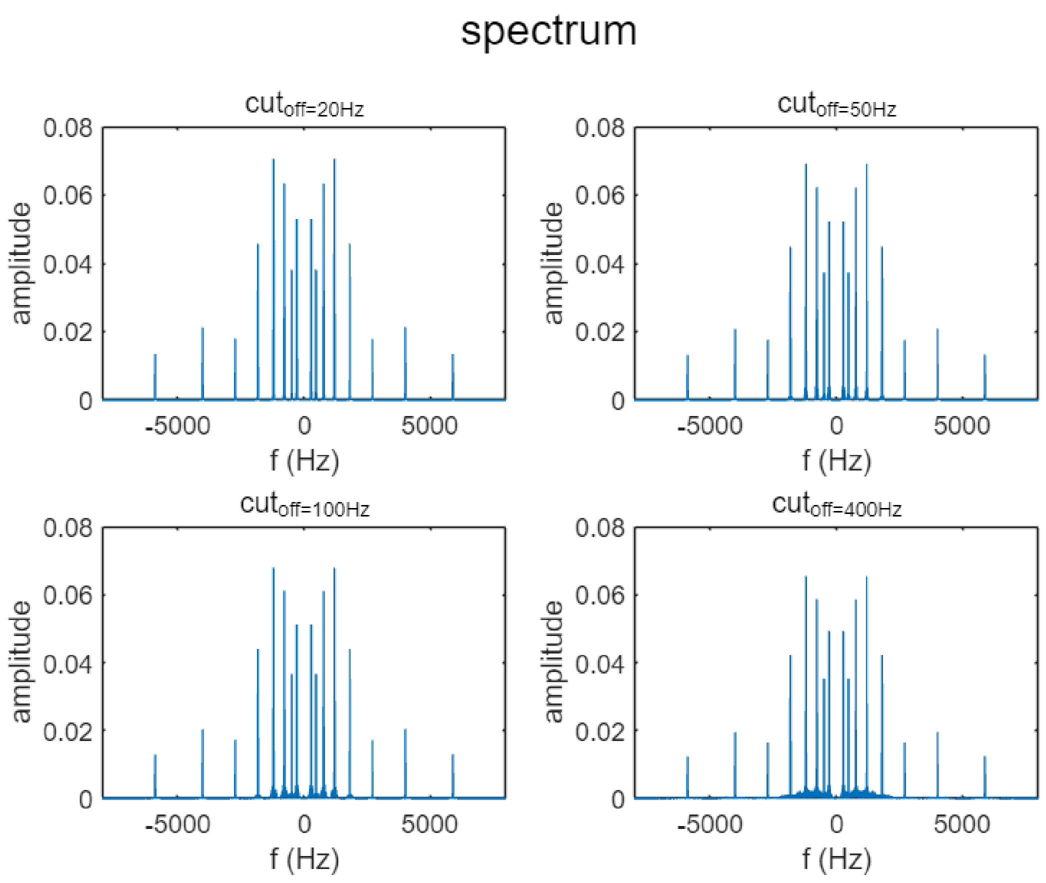
```
t = (0:length(sig)-1)'./fs;
figure;
sgtitle('time domain waveform');
subplot(2, 2, 1);
plot(t, cut_off_20);
axis([t(1) t(end) ylim]);
title('cut_off=20Hz');
xlabel('t (s)');
ylabel('amplitude');
```

```
subplot(2, 2, 2);
plot(t, cut_off_50);
axis([t(1) t(end) ylim]);
title('cut_off=50Hz');
xlabel('t (s)');
ylabel('amplitude');
subplot(2, 2, 3);
plot(t, cut_off_100);
axis([t(1) t(end) ylim]);
title('cut_off=100Hz');
xlabel('t (s)');
ylabel('amplitude');
subplot(2, 2, 4);
plot(t, cut_off_400);
axis([t(1) t(end) ylim]);
title('cut_off=400Hz');
xlabel('t (s)');
ylabel('amplitude');
```



```
if mod(length(sig), 2) == 1
    freqency = linspace(-fs/2, fs/2, length(sig));
else
    freqency = linspace(-fs/2, fs/2, length(sig) + 1);
    freqency = freqency(1:end-1);
end
```

```
figure;
sgtitle('spectrum');
subplot(2, 2, 1);
plot(freqency, abs(fftshift(fft(cut_off_20))/fs));
axis([freqency(1) freqency(end) ylim]);
title('cut_off=20Hz');
xlabel('f (Hz)');
ylabel('amplitude');
subplot(2, 2, 2);
plot(freqency, abs(fftshift(fft(cut_off_50))/fs));
axis([freqency(1) freqency(end) ylim]);
title('cut_off=50Hz');
xlabel('f (Hz)');
ylabel('amplitude');
subplot(2, 2, 3);
plot(freqency, abs(fftshift(fft(cut_off_100))/fs));
axis([freqency(1) freqency(end) ylim]);
title('cut_off=100Hz');
xlabel('f (Hz)');
ylabel('amplitude');
subplot(2, 2, 4);
plot(freqency, abs(fftshift(fft(cut_off_400))/fs));
axis([freqency(1) freqency(end) ylim]);
title('cut_off=400Hz');
xlabel('f (Hz)');
ylabel('amplitude');
```



Save the processed audio.

```
audiowrite('proj1/T4_8_20.wav', cut_off_20, fs);
audiowrite('proj1/T4_8_50.wav', cut_off_50, fs);
audiowrite('proj1/T4_8_100.wav', cut_off_100, fs);
audiowrite('proj1/T4_8_400.wav', cut_off_400, fs);
```

Destcription

1、题目要求

Task3 探究的是在原始信号中加入语谱噪声（SSN）后得到的噪声信号, 带通的数目对于最终合成语音信号的影响。Task4 则探究对噪声信号提取包络时的低通滤波器截止频率对于最终合成的语音信号的影响

1. 生成一个信噪比为-5dB的噪音信号（结合清晰语句与语谱噪声信号）
2. 设置带通滤波器的通带数目N = 8
3. 通过改变低通滤波器的截止频率为20Hz、50Hz、100Hz和400Hz，调整声音编码器
4. 描述低通滤波器的截止频率的变化如何影响语音合成的质量
5. 随着对包络滤波的截止频率的变高而提升，

2、绘图及验证

（1） 通过判断相等的feq函数验证信噪比计算后为-5dB

（2）结果图：时域波形【横坐标以时间秒为单位，纵坐标为幅度】，频谱【横坐标以频率Hz为单位，纵坐标为幅度谱 】，包络图均在对应的代码段落后

（3）使用norm函数与进行浮点数数值比较的feq函数进行能量归一化验证

3、对结果的分析和解释

分析：在时域图上，没有随着截止频率的增大体现出明显差别；但是在频谱图上可以观察到细微差别，即随着截止频率的增大，出现更多密集的微小幅度信号。

解释： 随着截止频率变高，不同频率包络分量增加，提高了语音合成的效果，复原信号保留的频率分量增多。

结论：与 Task2 对比,带有SSN的语音信号所合成的信号明显更加难以识别，无法听清。在信噪比-5dB的条件下,语谱噪声覆盖影响原信号。同时，截止频率会影响语音信号的合成效果，随着截止频率的升高，语音还原效果渐渐变好，但即使在截止频率400Hz 的情况下也仍然无法听清。

Task Enhancement

Since the description of the above, we wander to find a N which can as closer as possible to recovery the sound.

Initialize and load audio.

```
clear;clc;
[sig, fs] = audioread('proj1/C_01_02.wav');
```

Set LPF cut-off frequency.

```
cut_off = 50;
```

Implement tone-vocoder.

$$N = 4$$

```
N_64 = speech_synthesis(sig, fs, 64, cut_off, false);
```

$$N = 8$$

```
N_128 = speech_synthesis(sig, fs, 128, cut_off, false);
```

Check energy normalization.

```
sig_norm = norm(sig);
feq(norm(N_64), sig_norm)
```

```
ans = logical
   1
```

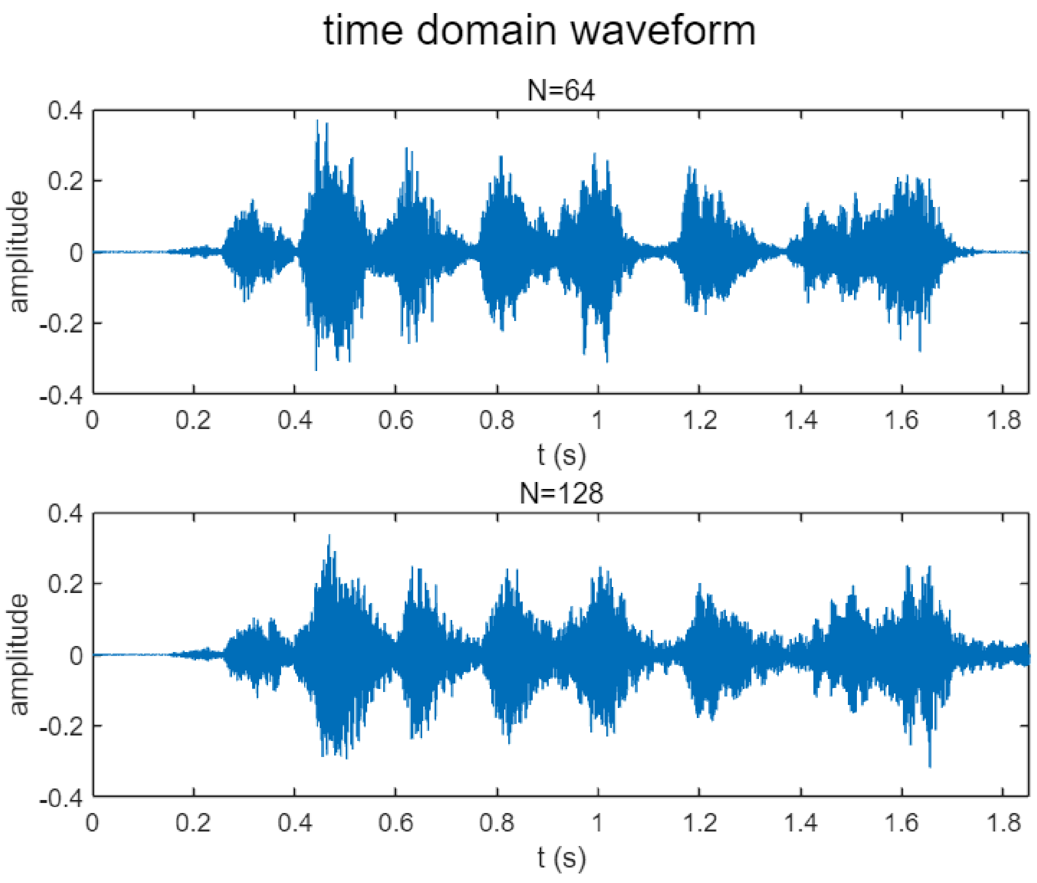
```
feq(norm(N_128), sig_norm)
```

```
ans = logical
```

```
1
```

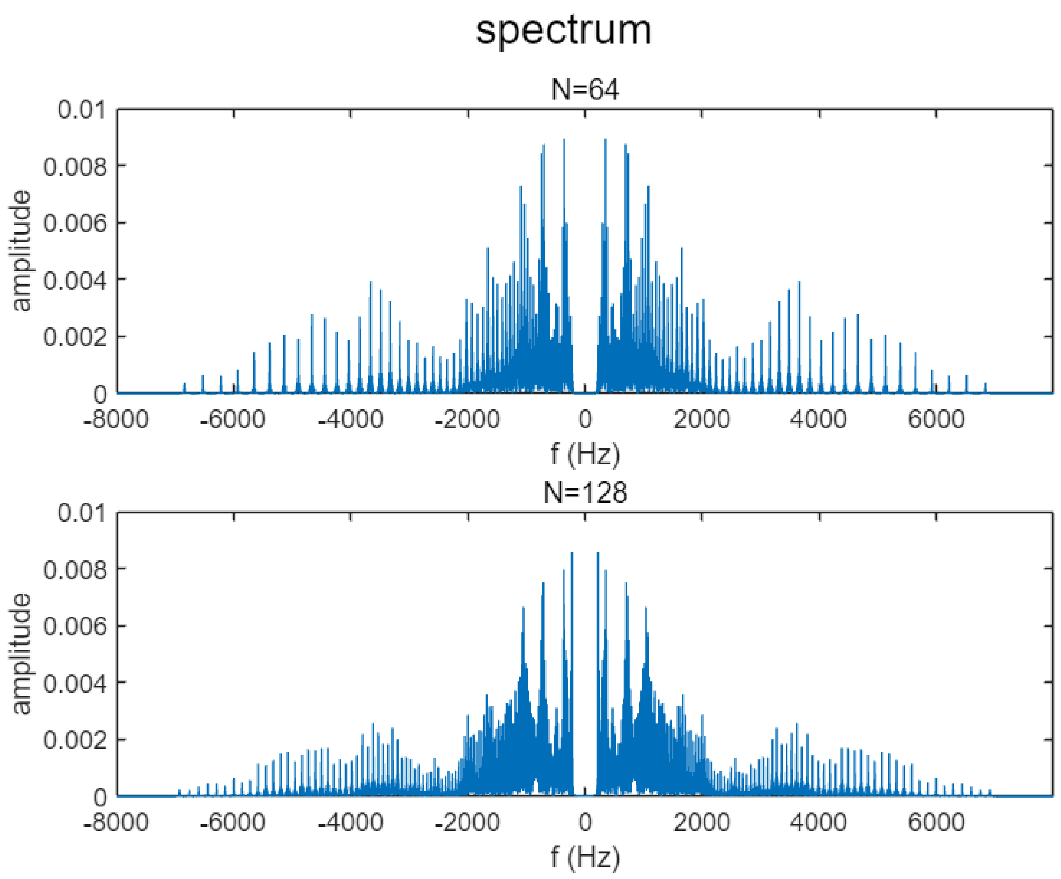
Plot each result.

```
t = (0:length(sig)-1)'./fs;
figure;
sgtitle('time domain waveform');
subplot(2, 1, 1);
plot(t, N_64);
axis([t(1) t(end) ylim]);
title('N=64');
xlabel('t (s)');
ylabel('amplitude');
subplot(2, 1, 2);
plot(t, N_128);
axis([t(1) t(end) ylim]);
title('N=128');
xlabel('t (s)');
ylabel('amplitude');
```



```
if mod(length(sig), 2) == 1
    freqency = linspace(-fs/2, fs/2, length(sig));
else
    freqency = linspace(-fs/2, fs/2, length(sig) + 1);
    freqency = freqency(1:end-1);
```

```
end
figure;
sgtitle('spectrum');
subplot(2, 1, 1);
plot(freqency, abs(fftshift(fft(N_64))/fs));
axis([freqency(1) freqency(end) ylim]);
title('N=64');
xlabel('f (Hz)');
ylabel('amplitude');
subplot(2, 1, 2);
plot(freqency, abs(fftshift(fft(N_128))/fs));
axis([freqency(1) freqency(end) ylim]);
title('N=128');
xlabel('f (Hz)');
ylabel('amplitude');
```



Save the processed audio.

```
audiowrite('proj1/Enh_64_50.wav', N_64, fs);
audiowrite('proj1/Enh_128_50.wav', N_128, fs);
```

By comparing the $Enh\_64\_50.wav$, $Enh\_128\_50.wav$ and $T1\_32\_50.wav$, there's little enhancement by increasing N to a larger scale, which means that $N = 32$ is a rather perfect parameter to recovery the signal in this method.

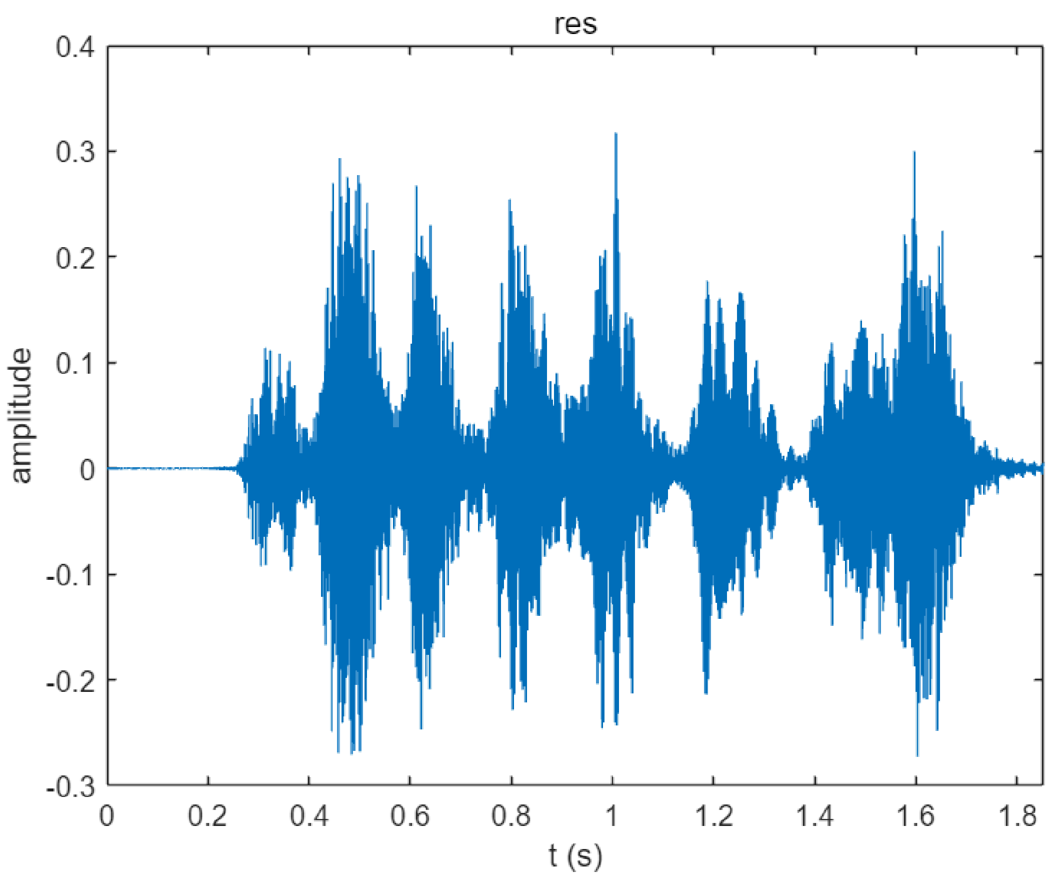
Additionally, we wander to find a way to recovery the speaking sounds less lossy meanwhile using less source.

As the speaking sound of human have different frequences in different situations, we set the common frequence of speaking as from 200Hz to 2000Hz.

```
clear;clc;
[sig, fs] = audioread('proj1/C_01_02.wav');
res = zeros(length(sig), 1);
t = (0:length(sig)-1)'./fs;
band_order = 4;
low_order = 6;
cut_off = 1000;
% get the band edge's frequency
N_speak = 64;
N_high = 8;
f_speak = divide_freq(N_speak,200,2000);
f_high = divide_freq(N_high,2000,7000);
% handle speak band
for i = 2:length(f_speak)
    lower = f_speak(i-1);
    higher = f_speak(i);
    % design band-pass filter
    [b,a] = butter(band_order, [lower higher]/(fs/2));
    % do band-pass filtering
    y = filter(b, a, sig);
    % do full-wave rectification
    y = abs(y);
    % design low-pass filtering
    [b,a] = butter(low_order, cut_off/(fs/2));
    % do low-pass filtering
    y = filter(b, a, y);
    % generate sinewave
    sinwave = sin((2*pi*(lower+higher)/2) .* t);
    % multiple the sinwave
    y = sinwave .* y;
    % add to the result
    res = res + 3*y;
end
% handle higher band
for i = 2:length(f_high)
    lower = f_speak(i-1);
    higher = f_speak(i);
    % design band-pass filter
    [b,a] = butter(band_order, [lower higher]/(fs/2));
    % do band-pass filtering
    y = filter(b, a, sig);
    % do full-wave rectification
    y = abs(y);
    % design low-pass filtering
    [b,a] = butter(low_order, cut_off/(fs/2));
    % do low-pass filtering
    y = filter(b, a, y);
    % generate sinewave
    sinwave = sin((2*pi*(lower+higher)/2) .* t);
    % multiple the sinwave
    y = sinwave .* y;
    % add to the result
    res = res + y;
end
% do energy normalization
res = res .* (norm(sig)/norm(res));
audiowrite('proj1/Enh_speaking.wav', res, fs);
```

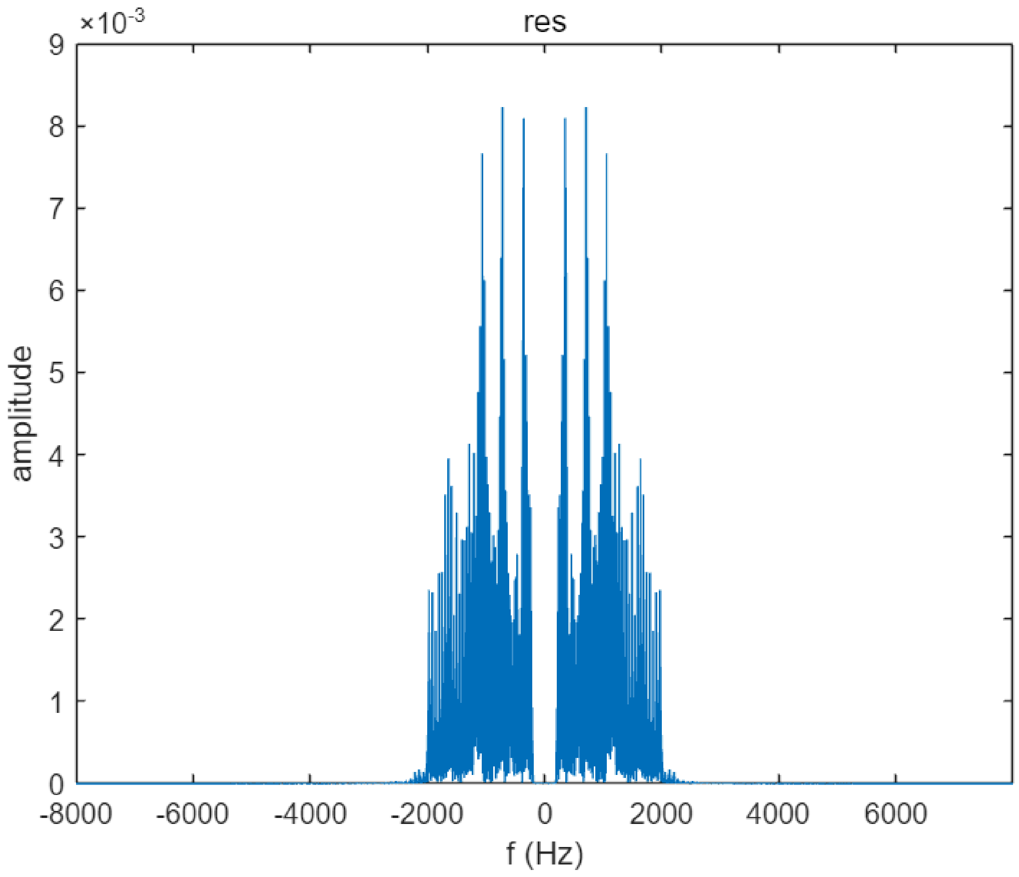
Plot the result.

```
t = (0:length(sig)-1)'./fs;
figure;
title('time domain waveform');
plot(t, res);
axis([t(1) t(end) ylim]);
title('res');
xlabel('t (s)');
ylabel('amplitude');
```



```
if mod(length(sig), 2) == 1
    freqency = linspace(-fs/2, fs/2, length(sig));
```

```
else
    freqency = linspace(-fs/2, fs/2, length(sig) + 1);
    freqency = freqency(1:end-1);
end
figure;
title('spectrum');
plot(freqency, abs(fftshift(fft(res))/fs));
axis([freqency(1) freqency(end) ylim]);
title('res');
xlabel('f (Hz)');
ylabel('amplitude');
```



But since the presentation audio may be generated by computer, has high frequence and has no other noise. We don't get a better version than $T1\_32\_50.wav$.

Functions

The function to judge two floating point number equal.

(Since floating point number comparison exists precision problem, when the difference between two floating point number is smaller than a very small value ϵ, we think they're the same)

(The precision problem will be magnified after complex calculation, so instead of using the eps in matlab, we loose the judging criteria and use a bigger ϵ)

```
function res = feq(a, b)
    res = abs(a-b) < 1e-12;
end
```

The function to get the corresponding cochlea length (in mm) from a certain frequency (in Hz).

```
function d = freq2cochlea (f)
    d = log10(f./165.4 + 1) ./ 0.06;
end
```

The function to get the corresponding frequency (in Hz) from a certain cochlea length (in mm).

```
function f = cochlea2freq (d)
    f = 165.4 .* (10.^(0.06.*d) - 1);
end
```

The function to get the frequency (in Hz) equally divided by the cochlea length.

```
function f = divide_freq (n, f0, f1)
if nargin == 1
    d0 = freq2cochlea(200);
    d1 = freq2cochlea(7000);
else
    d0 = freq2cochlea(f0);
    d1 = freq2cochlea(f1);
end

f = cochlea2freq(linspace(d0, d1, n + 1));
end
```

The function to get figure objects can contain given number of figure.

```
function figure_obj = get_figures(N, title, each_figure_num)
    if nargin < 3
        each_figure_num = 16;
        if nargin < 2
            title = '';
        end
    end

    figure_num = ceil(N/each_figure_num);
    figure_obj = gobjects(figure_num, 1);
    for i = 1:figure_num-1
        figure_obj(i) = figure();
        tiledlayout(4, 4);
        sgtitle(title);
    end
    last_figure_num = N - (figure_num-1) * each_figure_num;
    if last_figure_num == 0
        last_figure_num = each_figure_num;
    end
    last_figure_col = floor(sqrt(last_figure_num));
    last_figure_row = ceil(last_figure_num/last_figure_col);
    figure_obj(figure_num) = figure();
    tiledlayout(last_figure_row, last_figure_col);
    sgtitle(title);
end
```

The function to plot with nonuniform x-axis.

```
function nonuniform_plot(x, y, old_range, new_range)
    new_x = zeros(length(x), 1);
    x_idx = 1;
    for i = 2:length(old_range)
        rate = (new_range(i)-new_range(i-1)) / (old_range(i)-old_range(i-1));
        while(x(x_idx) < old_range(i))
            new_x(x_idx) = rate * (x(x_idx)-old_range(i-1)) + new_range(i-1);
            x_idx = x_idx + 1;
        end
    end
    plot(new_x, y);
    xticks(new_range);
    xticklabels(num2cell(old_range));
end
```

The speech synthesis function.

```
function res = speech_synthesis(s, fs, N, cut_off, draw_process, band_order, low_order)
if nargin < 6
    band_order = 4;
    low_order = 6;
    if nargin < 5
        draw_process = true;
    end
end
if draw_process
    % prepare the figure
    band_figure = get_figures(N, 'frequency responese of band\_pass filter');
    low_figure = get_figures(N, 'frequency responese of low\_pass filter');
    envelope_figure = get_figures(N, 'envelope');
end
% initialize res and t
res = zeros(length(s), 1);
t = (0:length(s)-1)'./fs;
% get the band edge's frequency
f = divide_freq(N);
% for each band
for i = 2:length(f)
    lower = f(i-1);
    higher = f(i);
    % design band-pass filter
    [b,a] = butter(band_order, [lower higher]/(fs/2));
    if draw_process
```

```
        % calculate and plot the frequency response of the band-pass filter
        [H, omega] = freqz(b,a);
        frequency = omega * (fs/(2*pi));
        figure(band_figure(ceil((i-1)/16)));
        nexttile;
        plot(frequency, abs(H));
        axis([frequency(1) frequency(end) ylim]);
        xlabel('f (Hz)');
        ylabel('H(j\omega)');
        title(['i=' num2str(i-1)])
    end
    % do band-pass filtering
    y = filter(b, a, s);
    % do full-wave rectification
    y = abs(y);
    % design low-pass filtering
    [b,a] = butter(low_order, cut_off/(fs/2));
    if draw_process
        % calculate and plot the frequency response of the low-pass filter
        [H, omega] = freqz(b,a);
        frequency = omega * (fs/(2*pi));
        figure(low_figure(ceil((i-1)/16)));
        nexttile;
        nonuniform_plot(frequency, abs(H), [frequency(1) min(cut_off * 3, frequency(end)) frequ
        axis([frequency(1) frequency(end) ylim]);
        xlabel('f (Hz)');
        ylabel('H(j\omega)');
        title(['i=' num2str(i-1)])
    end
    % do low-pass filtering
    y = filter(b, a, y);
    if draw_process
        % calculate and plot the envelope
        figure(envelope_figure(ceil((i-1)/16)));
        nexttile;
        plot(t, y);
        axis([t(1) t(end) ylim]);
        xlabel('t (s)');
        ylabel('amplitude');
        title(['i=' num2str(i-1)])
    end
    % generate sinewave
    sinwave = sin((2*pi*(lower+higher)/2) .* t);
    % multiple the sinwave
    y = sinwave .* y;
    % add to the result
    res = res + y;
end
% do energy normalization
res = res .* (norm(s)/norm(res));
end
```

The function to generate speed-shaped noise (SSN) with the same energy.

```
function ssn = get_ssn(sig, fs)
if iscolumn(sig)
    % generate white noise
    whitenoise = 1 - 2*rand(length(sig),1);
    % generate long-term spectrum of speech signal to estimate the power spectral density of th
    sig = repmat(sig, 10, 1);
else
    % generate white noise
    whitenoise = 1 - 2*rand(1,length(sig));
    % generate long-term spectrum of speech signal to estimate the power spectral density of th
    sig = repmat(sig, 1, 10);
end
[Pxx,f] =pwelch(sig, [], [], 512, fs);
% generate filter coefficients
b = fir2(3000, f / (fs/2), sqrt(Pxx/max(Pxx)));
% perform filtering for white noise signal to get the SSN
ssn = filter(b, 1, whitenoise);
% do energy normalization
ssn = ssn .* (norm(sig)/norm(ssn));
end
```

DTboard voice playback & the application of speech synthesis technology in real life

Burn the following code into the development board to enable voice playback

```
//};//wav的数据 32
uint16_t length = 0;
uint16_t rd = 0;
uint8_t play_times = 0;
uint32_t audio_buff;
uint8_t i=0;

void Sound_Int_Usr_Handler(void)
{
    if(Sound_GetInt()){
      Sound_ClearInt();
   audio_buff=abs(code_bjkjg_pcm[rd]);
   audio_buff=audio_buff<<16;
    rd=rd+1;
 }
  if (rd >= N){
   rd = 0;
   i= 0;
   play_times++;
  }
    if(Sound_GetFifoPointer()) Sound_Fill_FIFO1(audio_buff);//按照次序将wav的数据以此输入
    else Sound_Fill_FIFO0(audio_buff);

}
```

```
void Audio_GPIO_Init(void)
{
 GPIO_InitTypeDef GPIO_Init = { 0 };
 GPIO_Init.GPIO_Pin = GPIO_PIN_15;
 GPIO_Init.GPIO_Dir = GPIO_DIR_OUT;
 GPIO_Init.GPIO_Mode = GPIO_MODE_SF;
 GPIO_StructInit(GPIOA, &GPIO_Init);
 GPIO_Pins_Set(GPIOA,GPIO_PIN_15);

 GPIO_Init.GPIO_Pin = GPIO_PIN_14;
 GPIO_Init.GPIO_Dir = GPIO_DIR_OUT;
 GPIO_Init.GPIO_Mode = GPIO_MODE_GP;
 GPIO_StructInit(GPIOA, &GPIO_Init);
 GPIO_Pins_Set(GPIOA,GPIO_PIN_14);

 GPIO_Init.GPIO_Pin = GPIO_PIN_1;
 GPIO_Init.GPIO_Dir = GPIO_DIR_IN;
 GPIO_Init.GPIO_Mode = GPIO_MODE_GP;
 GPIO_StructInit(GPIOA, &GPIO_Init);
 GPIO_Pins_Set(GPIOA,GPIO_PIN_1);
}

void PlayAudio_Init(void)
{

  Sound_InitStruct Sound_Init = {0};

    RCC_SoundClk_Int(1);
    Sound_Init.unsigne_en   = 0;
    Sound_Init.sound_format = SOUND_FORMAT_PCM;//wav音频使用PCM采样
    Sound_Init.sample_rate  = SOUND_SAMPLE_RATE_16K;//wav音频的采样率，对应fs
  //对应SOUND_SAMPLE_RATE_8K。SOUND_SAMPLE_RATE_16K，SOUND_SAMPLE_RATE_44.1K
    Sound_Init.pcm_frist_value = 0;
    Sound_StructInit(&Sound_Init);

    Sound_Fill_FIFO0(0);
    Sound_Fill_FIFO1(0);
    Sound_EnableInt();
}

void playAudio_Stop(void)
{
 Sound_Fill_FIFO0(0);
 Sound_Fill_FIFO1(0);
 Sound_DisableInt();
}
int main(void)
{
 while(1){
  if(GPIO_Pin_ReadBit(GPIOA, 1<<1)==FALSE)//当按钮按下
   {
 Audio_GPIO_Init();
```

```
  PlayAudio_Init();


   if (play_times >= 4)//设置音频循环的播放
    {
       play_times = 0;
     playAudio_Stop();
   }

    }

  }
```

The application of speech synthesis technology in real life

As a efficient access to listen and communicate, the speech synthesis technology is widely applied in many fields,

1,Intelligence assistant such as Apple’s Siri , these devices can answer users’ questions and interact through speech synthesis technology.

2,Reading assistance: For people with visual impairments, speech synthesis technology can convert written information into speech, such as the text-to-speech function.

3,Vehicle navigation: Modern car navigation systems commonly use speech synthesis technology to provide real-time route guidance, improving driving safety.

4,Customer service robots: In fields such as banking, telecommunications, and e-commerce, speech synthesis technology is used for automatic customer service systems, providing 24-hour service and improving customer service efficiency.

Etc.

Expeience

1.掌握使用 MATLAB进行语音合成技术的实现，了解其中带数与截止频率作为两个基础变量对语音合成效果的影响以及原理。

2.了解了人工耳蜗根据耳蜗处理信号的分频分长度的特点来进行模拟，进行声音信号的分解与复原。

3.体验在加入一定信噪比的噪声后，语音质量下降，影响音质与语音合成的效果。

4.通过循环逻辑找出可以生成的最清晰的合成语音。

5.学习嵌入式32位的DT板语音播放，入门开发设计。

分工和贡献

12213010 涂峻绫 Task 1 Coding; Task 2 Coding;   Task Enhancement;

12212904 肖星辰 Task 1 Describe; Task 2 Describe;   Voice Playback;

12212961 欧阳莹轩 Task 3 Describe; Task 4 Describe;   Writing;

12211831 欧阳安男 Task 3 Coding; Task 4 Coding;   Task Enhancement;

贡献比：各 25%

....