Southern University of Science and Technology

Speech Signal Processing

# Lab 4 Report

11510478 郭锦岳

# Question 1

Code:

```
function [ ] = lab4q1( file_name, start_sample, frame_length )
%LAB3Q1 calculate the STFT, and generate figures for lab4q1
[signal, Fs] = audioread(file_name);
L = round((Fs/1000)*frame_length);
window = hamming(L);

% plot the original signal
subplot(2,2,1);plot(signal);title('input signal');

% plot the windowed signal
signal_w = signal(start_sample:start_sample+L-1).*window;
subplot(2,2,2);plot([start_sample:start_sample+L-
1],signal_w);title('windowed signal');

% plot the STFT
signal_w_s = fft(signal_w);
subplot(2,2,3);stem(abs(signal_w_s(1:round(L/2))));title('STFT
of windowed signal');

% plot the STFT in dB
signal_w_s_db = 20*log(abs(signal_w_s));
subplot(2,2,4);stem(signal_w_s_db(1:round(L/2)));title('STFT
of windowed signal in dB');

end
```

Running result:

```
>> lab4q1('s5.wav',1000,40);
```

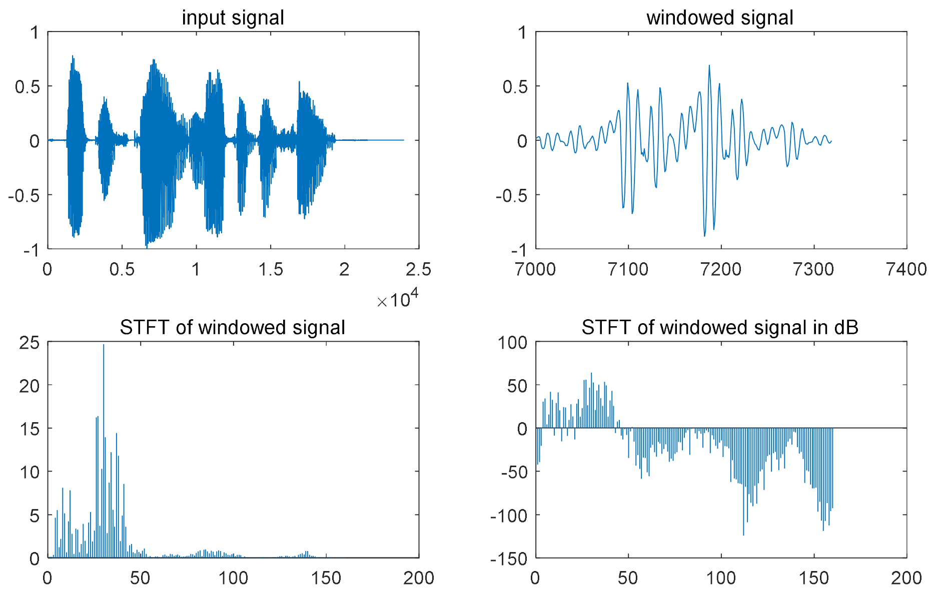


Figure 1

```
>> lab4q1('vowel_iy_100hz.wav',1000,40);
```

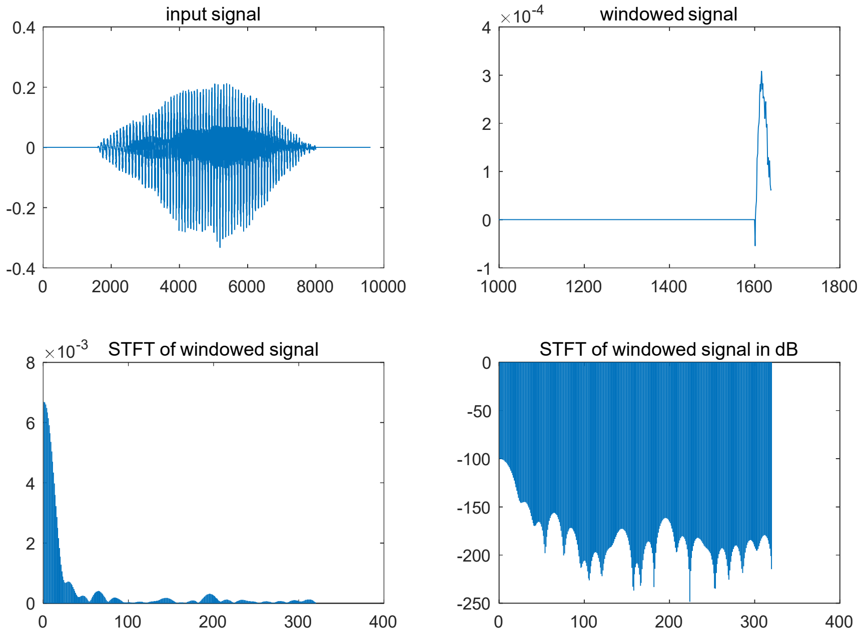


Figure 2

The result of STFT is different if we change the starting frame value. For s5.wav, the piece we chose has some regular frequency components, so we can see the STFT has many high-value points. But for the second signal, since the starting point is very early, we can only get little information from the signal. Then, we can see that there is only low frequency components in the result of STFT.

# Question 2

In question 2, the length of our window is changing, so we can only define the window length and plot the figure inside the for loops.

Code:

```
function [ ] = lab4q2( anal_num, file_name, start_sample, frame_length )
%LAB3Q1 calculate the STFT, and generate figures for lab4q1
[signal, Fs] = audioread(file_name);

% hamming window plot
figure;
for i = 1:anal_num
    L = round((Fs/1000)*frame_length(i));% calculate length
    window = hamming(L);% generate hamming window
    signal_w = signal(start_sample:start_sample+L-1).*window;% timesignal
    signal_w_s = fft(signal_w);% compute STFT

subplot(anal_num,1,i);stem(abs(signal_w_s(1:round(L/2))),'Marker','none');xlim([0,L]);
    title(sprintf('STFT, hamming window, length = %d msec', frame_length(i)));
end

% rectangular window plot
figure;
for i = 1:anal_num
    L = round((Fs/1000)*frame_length(i));% calculate length
    window = ones(L,1);% generate hamming window
    signal_w = signal(start_sample:start_sample+L-1).*window;% timesignal
    signal_w_s = fft(signal_w);% compute STFT

subplot(anal_num,1,i);stem(abs(signal_w_s(1:round(L/2))),'Marker','none');
    title(sprintf('STFT, rectangular window, length = %d msec', frame_length(i)));xlim([0,L]);
end

end
```

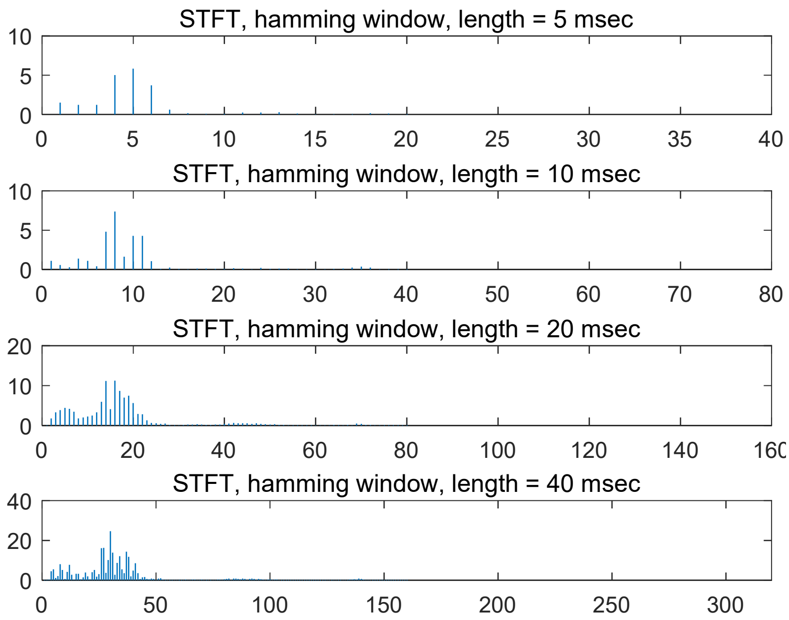


Figure 3, Hamming window

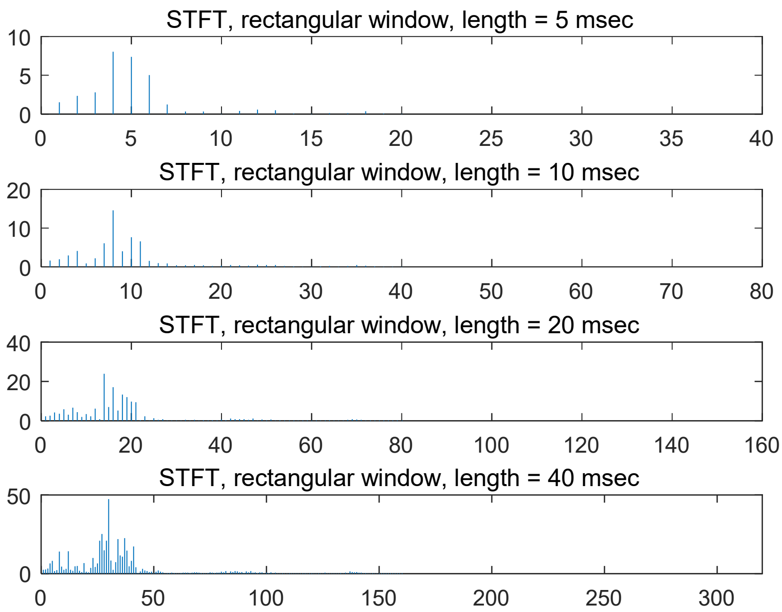


Figure 4, Rectangular window

As the window length becomes longer, we can have a better resolution of the STFT in frequency domain, but the peaks are still in the same place.

However, there seems to be a slight difference between hamming window and rectangular window. The result of hamming window has a smoother curve, while the one of rectangular window shows a more significant difference in value.

# Question 3

Since the question asks for two subplot in one figure, I write another function plotSpec, and calls it two times to plot the figure for narrow and wide band results, and run the program two times for color or gray scale figures.

Code:

```
function [ ] = lab4q3( file_name, s_rate, len_narrow, len_wide,
plot_scale, dyn_range, col_gry)
%LAB4Q3
%s_rate : integer, minimum is 0
%len_narrow and len_wide: window length in miliseconds
%plot_scale: 'db' for dB scale, 'linear' = linear scale;
%dyn_range: desired dynamic range in dB
%col_gry: 'C' for color, 'G' for gray

[signal,Fs] = audioread(file_name);
% convert the input from ms to frame
L_n = round((Fs/1000)*len_narrow);
L_w = round((Fs/1000)*len_wide);
% generate window
window_n = hamming(L_n);
window_w = hamming(L_w);

% down sampling, if needed
if s_rate > 0
    signal = resample(signal, 1, s_rate);
end
% call the function plotSpec to generate figures for narrow
and wide band
subplot(2,1,1);plotSpec(signal, Fs, window_n, plot_scale,
dyn_range, col_gry);
title(sprintf('narrow-band spectrum, window length = %d ms
(%s)', len_narrow, plot_scale));
subplot(2,1,2);plotSpec(signal, Fs, window_w, plot_scale,
dyn_range, col_gry);
title(sprintf('wide-band spectrum, window length = %d ms (%s)',
len_wide, plot_scale));


end
```

```
function [ ] = plotSpec(signal, Fs , window, plot_scale,
dyn_range, col_gry)

[sp, F, T] = spectrogram(signal,
window,round(length(window)/2),1024,Fs);% get the spectrogram
BA = 20*log10(abs(sp));% convert into dB
BA_max = max(max(BA));% find the maximum
BA(find(BA < BA_max - dyn_range)) = BA_max - dyn_range;%
convert the dynamic range

% determine dB scale or log scale
if strcmp(plot_scale,'db')
    imagesc(T,F,BA);
elseif strcmp(plot_scale,'linear')
    imagesc(T,F,10.^(BA/20));
end

axis xy; % flip the Y Axis so lower frequencies are at the
bottom
xlabel('time (s)');ylabel('frequency (Hz)');

% determine color or gray
if strcmp(col_gry,'C')
    t=colormap;
    colormap(1-t);
    colormap(jet);
elseif strcmp(col_gry,'G')
    t=colormap(gray);
    colormap(1-t);
end

end
```

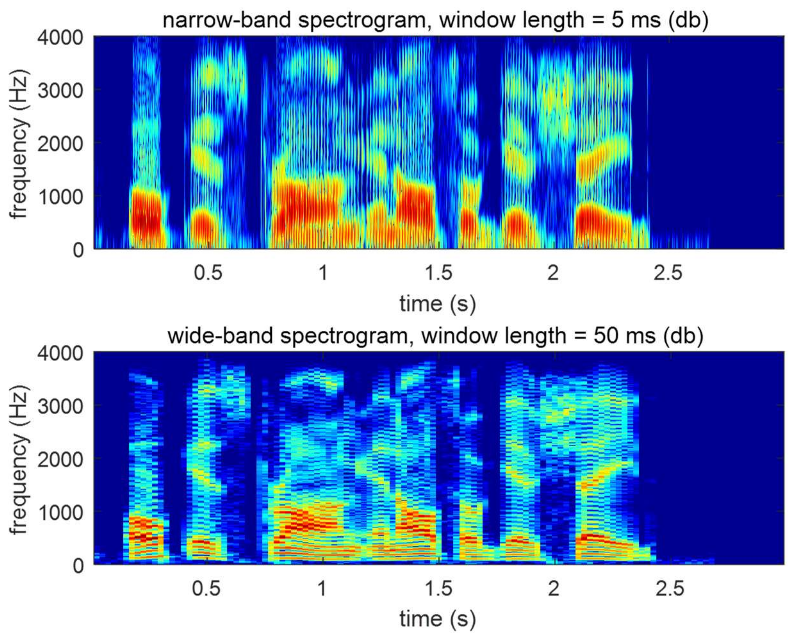


Figure 5, in color

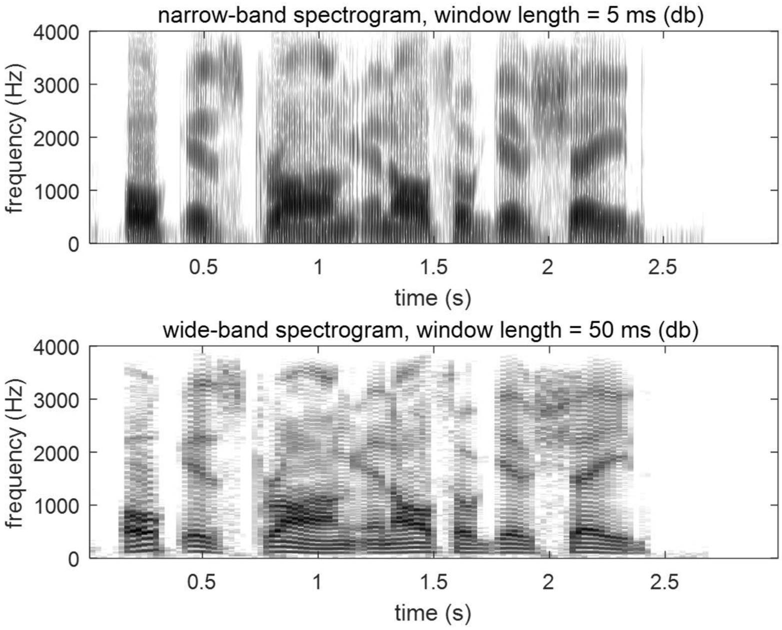


Figure 6, in gray scale

In both colored and gray figure can we recognize the formants and their changes of the input audio signal. The narrow band spectrogram has higher temporal resolution, so we can clearly see the change of the spectrum along with time, while the wide band spectrogram has higher spectrum resolution, so we can precisely locate the formants and the details in frequency domain.