EE 341 Summer 2018
Lab 3 The FFT and Digital Sound Transformations

Professor: Tai-Chang Chen

July 26th, 2018

He Feng 1427841
Hanling Lei 1673094
Zikai Yang 1435493

Introduction

In this lab we are going to process digital sound signal. The fft function and digital sound transformation are the core of this lab, and we are going to use them to determine the sound signal effect while they are shifting, scaling and series of processing in the following three exercises. The essence of this lab is to use fft function to figure out the frequency content, and we will analyze each sound signal based on the determined frequency content. In the first exercise, we will use plot function to plot the discrete function, and our goal is to plot out the magnitude of the FFT of the given signal before and after shifting. In the second and the third exercises, we use plot function as well to determine the frequency content in continuous function. For the second exercise, we focused on frequency shifting. For the last exercise, we initially focused on frequency shifting and scaling, finally we implement a high pass filter and let the sound signals passed through. The whole lab focused on digital sound transformation, and we used FFT as an important to in the process.

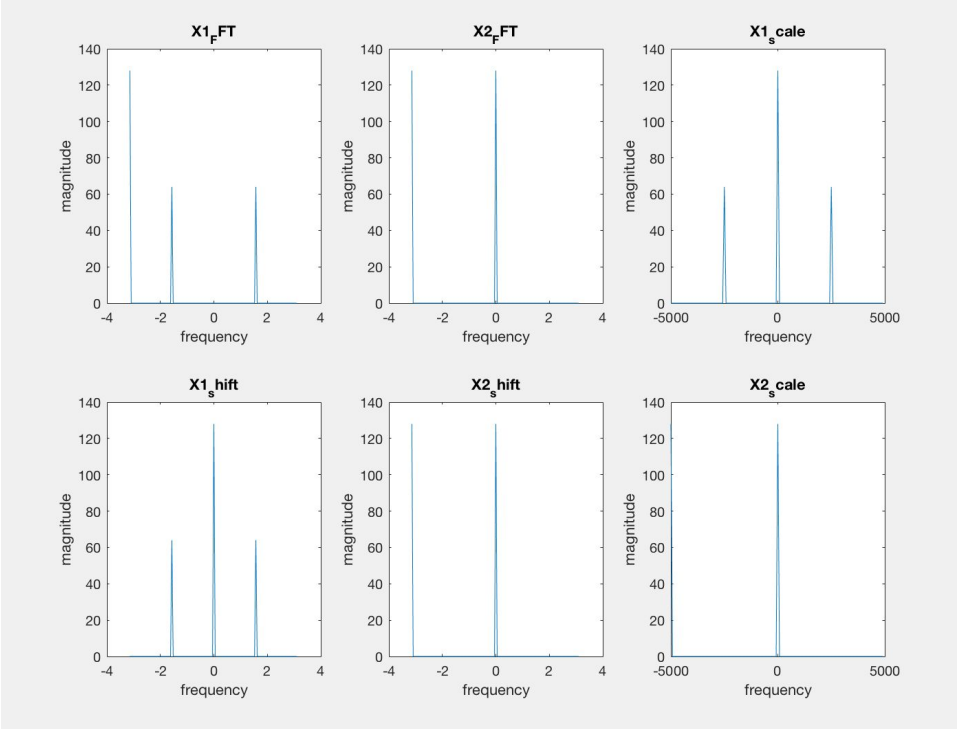
Results and Discussion

Exercise 1

The purpose of this exercise is letting us to use FFT function, and we need to plot out the magnitude of the FFT, which can be considered as the frequency content, before and after shifting the given signal.

We defined the domain of n at first, and then figure out there are 128 elements in this domain. Then we defined the signal frequency in one matrix, and defined a sampling rate which represented as Fs. Because we want to figure out the frequency content, therefore we also need to determine the frequency domain which will be shown on x-axis in the graph. We defined two frequency domain, one of them shows the graph initially format and the format after shifting, and the other one shows the graphs after scaling based on the given scaling rate.

Then we defined two signals with time domain based on the given equation, and we used fft function to make the given signals above within the frequency content. We use abs function to figure out the magnitude of two FFT function before shifting, these magnitudes will be used as we plot the graph laster. Then we used fftshift function to shift the signal, and used the same way to get the magnitude. Finally, we used the different frequency domain and the magnitudes above to plot six graphs, which represent the first signal before and after shifting, the second signal before and after shifting, and frequency content of the first and second signal after scaling with the supported sampling rate.



The frequency peak locations make sense which f is 0.25 and f is 0.5. When f is 0.25, the peak is at -0.5π, 0.5π and -π, which represents -0.25, 0.25 and -0.5 as the x-axis if f. We controlled our frequency domain and we let both of the before shifting graph and after shifting graph centered at 0, in this case the peak would be at 0, 0.25 and -0.25, which makes sense because the shifting process. After we used the scaling based on the sampling rate, the peak would stay at f is equal to -2500, 2500 and 0, which make sense because the value has been increased based on the shifted signal value. When f is 0.5, the whole process are similar to the previous method. The peak is at 0 and -0.5 as f is the unit on x-axis, and the peak will be at 0 and -5000 as we use the 10 kHz sampling rate. Therefore all of the graphs' peak make sense, which means we have the correct result. To conclude, both of the signals' peak make sense during the transformation, and we can make a say that we did this exercise correctly.

Exercise 2

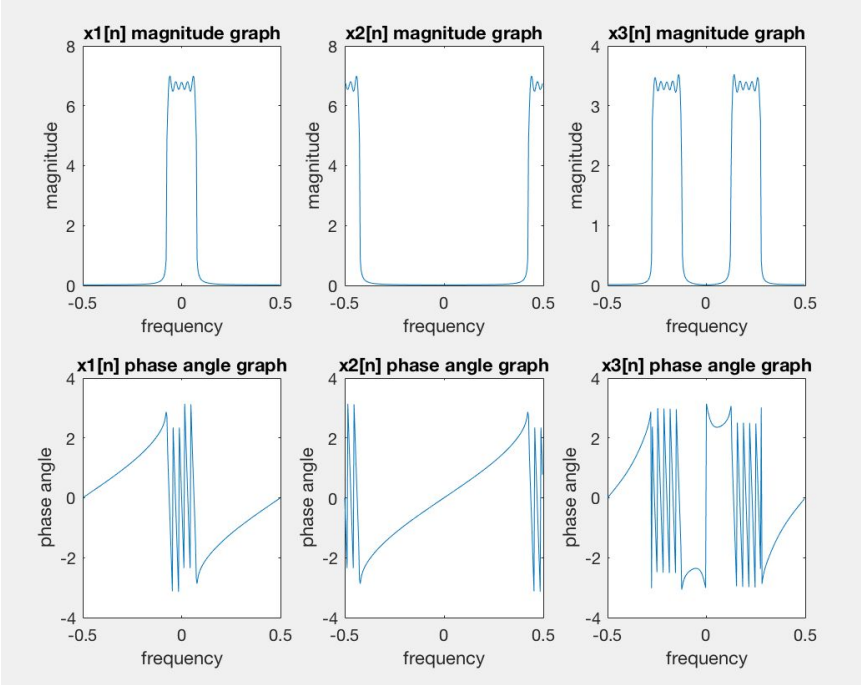
In this exercise we are going to focusing on frequency shifting. There are four different signals given in this section, and we need to get the magnitude and phase plots over the range of -0.5 to 0.5 for f. Similar to the exercise 1, we define the domain of n and the frequency domain which will be shown on x-axis at the initial step of the exercise. Then we created a matrix to represent different frequency. After setting the basic information, we create the plots of each signal one by one.

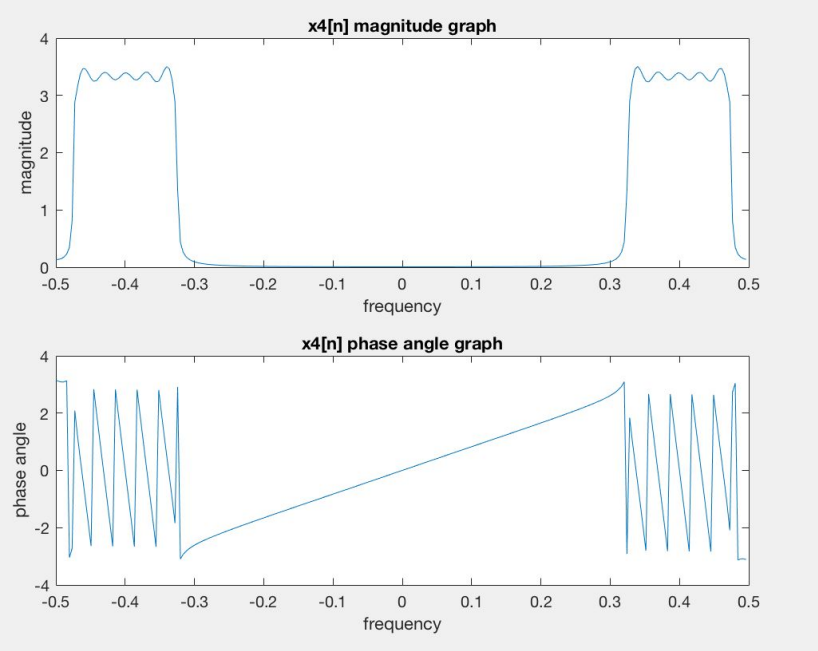
For the first equation, we wrote down the equation in time domain, and then we applied fft function to the previous signal and transfer it into frequency domain. Then we shifted the fft function, and used abs function and angle function to determine the magnitude and phase of this signal. We used the same method to justified the magnitude and phase of the total four functions, and we started to plot each graph based on the information above.

In this exercise, we plot the graphs within two figures. The first figure contains the six graphs from the first three functions, and the second figure contains the two graphs from the last function. Because the spec only requires us to plot the graphs of the first three functions, therefore we plot the fourth equation individually such that we can finish the discussion in this part easier. We used the plot function and here is one part of the code:

```
subplot(2,3,3);
plot(w,X3_shift_magnitude);
xlabel('frequency');
ylabel('magnitude');
title('x3[n] magnitude graph');
subplot(2,3,6);
plot(w,X3_shift_angle);
xlabel('frequency');
ylabel('phase angle');
title('x3[n] phase angle graph');
```

The code for the other functions are similar to the code above.





Based on the graphs above, we know the first signal corresponds to a low pass filter, the second signal corresponds to a high pass filter, and the third signal corresponds to a bandpass filter. For the last signal, it does not have a flat frequency response because it suffers from the aliasing, which will cause the magnitude of each parts not the same. The aliasing it suffered caused the not flat frequency response.

Exercise 3

The last exercise is an complicated one in this lab. We need to initially plot the sound signals, and then do different process on this signal to make different modified version of the sound. We initially defined the domain of n, then we defined the time sample and frequency sample which will be used later. At the beginning of this exercise, we used audioread function to load blm.wav and tiger.wav. Then we used fft function to justify the original signals, and we used abs function to determine the magnitude of each fft function. Then we plot the time sample and frequency sample based on the information above. We used audioinfo function to find the sampling rate, length of the data and samples, here are the output we got:

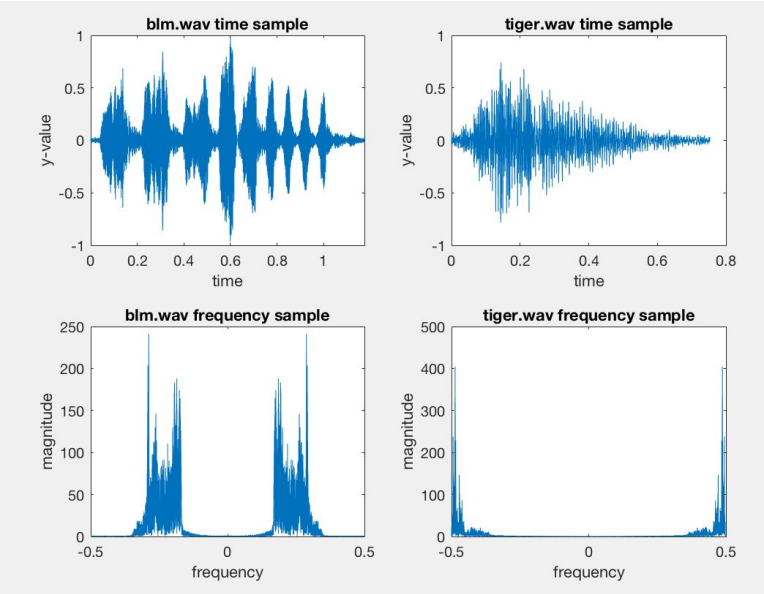
```
info1 =

  struct with fields:

           Filename: '/Users/hefeng/Desktop/EE 341 lab 3/blm.wav'
    CompressionMethod: 'Uncompressed'
          NumChannels: 1
           SampleRate: 22050
         TotalSamples: 25917
             Duration: 1.1754
                Title: []
              Comment: []
               Artist: []
        BitsPerSample: 8


info2 =

  struct with fields:

           Filename: '/Users/hefeng/Desktop/EE 341 lab 3/tiger.wav'
    CompressionMethod: 'Uncompressed'
          NumChannels: 1
           SampleRate: 22255
         TotalSamples: 16763
             Duration: 0.7532
                Title: []
              Comment: []
               Artist: []
        BitsPerSample: 8
```

We defined two new functions in the first part of this exercise, and we used fft function to process each signals. Then we use abs function to determine the magnitude of each new fft functions, and we play each sound with a two second delay between each sound. We used pause function to represent the delay. Then we used the frequency domain and the upper new fft function magnitude to plot the frequency content of two sound signals. This part can be considered as the frequency shifting.

In the second part of this exercise, we downsampled two signal functions to speed up. We used downsample function to downsample two signals, and we used fft function to determine the fft function of the signal, then we used abs function to determine the magnitude. The following steps are same as the previous part. We used soundsc function to display the signal, and we used plot function to plot each signal in the figure.

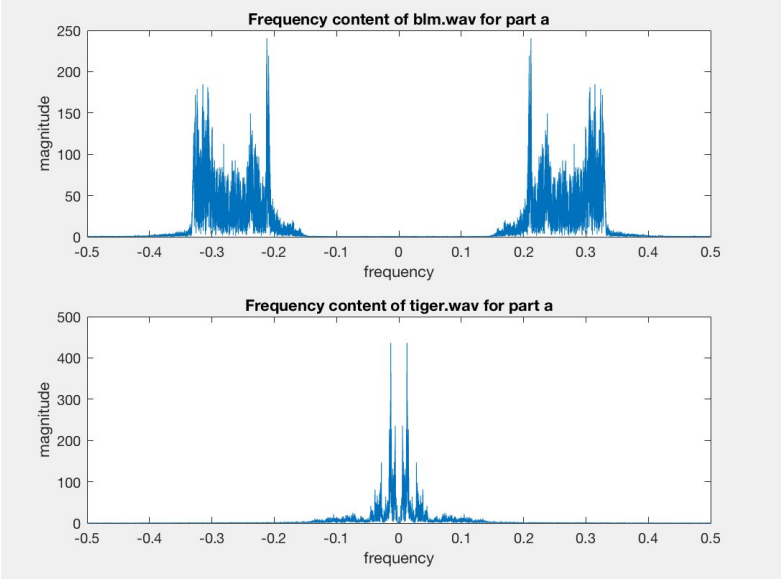
In the last section we implement a high pass filter, and we let two given signals passed through this implemented filter and get the new signals. We used the method from the previous sections to process these two signals. We use fft function to process the signals, which will be used to find the frequency content. Then we use ifft function to find the inverse of the signals above, we will based on this function, real function and double function to display the sound. In the figure 4 we plot the frequency content of the signals. All the output are shown below:

Figure 1:



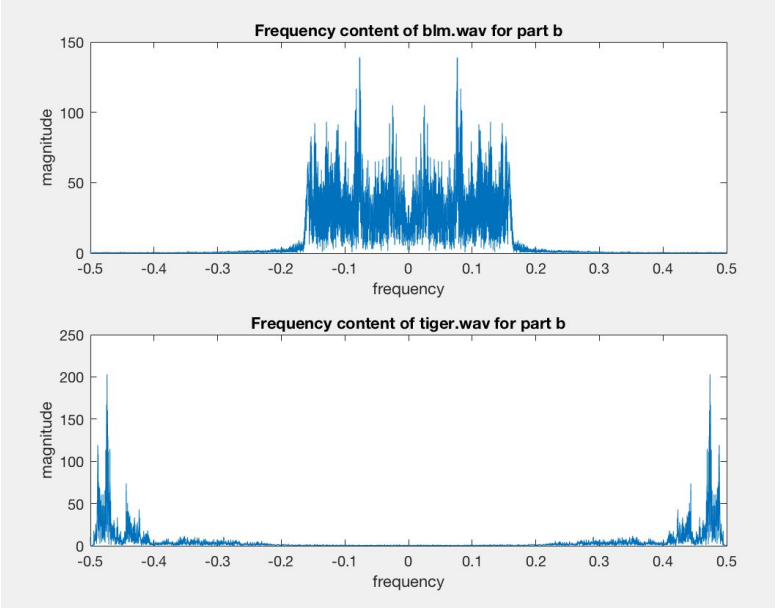
At the beginning of this exercise, we predicted the blm.wav has more low frequency content, and tiger.wav has less low frequency content. Based on the frequency content above, we found that the output result matches our expectation.

Figure 2:



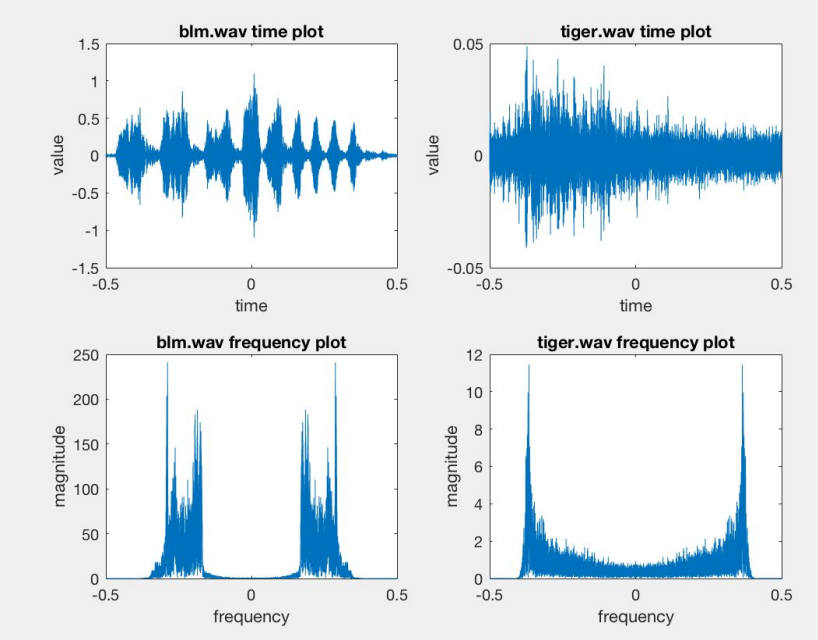
We can still hear the blm.wav sound pretty clear but a little bit different from the previous blm.wav sound, but the sound of tiger.wav is not clear at all, and it is hard to figure out the tiger's sound. Even though frequency shifting will only make the signal move left or right, but the result shifted signal we got in this case has so much difference than the original one, therefore the tiger.wav has a big difference in this case compared to the original one.

Figure 3:



The sound from the blm.wav are still clear to hear, and the low frequency content nearby zero are more than the previous one because of the scaling process. For the tiger.wav, the sound is much more clearer than the sound in section a, and it sound like the tiger.wav sound in figure 1. There are some little differences between these two sounds, but we can hear the voice clear enough compared to the last section. The main difference in scaling and shifting is while the signal is scaling, it can either be maximized or minimized, and shifting can only move the signal to the left or right instead of doing process described above.

Figure 4:



The blm.wav sounds pretty different than the previous sounds, but we can still say it sounds good without too much noise. We actually zero out some frequency terms, and the final sound sounds less energetic than the original sound because lots of energy has been eliminated during filtering.

Conclusion:

In this lab we used fft function to process a signal, and we used digital sound transformation to transfer the sound signal between within time domain and within frequency domain. At the beginning of the lab, we transfer a signal into fft function and figure out the magnitude. Then we generate three plots, which are unshifted DFT, shifted DFT and shifted DFT with a Hz sampling rate. We used these plots to observe the changing for a signal while they are shifting. In the second part of the lab, we mainly focused on frequency shifting, and we figured out that each signal can represent one type of filter for the top three signal functions, and we also discussed the reason why the fourth signal's frequency response is not flat. At the end of the lab, we did a combined exercise, which contain frequency shifting and frequency scaling. At the end of this section, we implement a high pass filter, and we let two signals passed through this filter to generate a less energetic sound. To conclude, this lab is shorter than the previous lab, but the thing conclude in this lab is trickier than the previous one. We learned a lot through this section and we are ready to face more struggle question about signal transformation.

Attached Code:

Ex1.m

```
% He Feng, Hanling Lei, Zikai Yang

% We used FFT to compute two signals, and draw the shifted,
% unshifted, and frequency scaling version of each signal
% in one figure.

clear all;
close all;

% Define the domain of n.
n = 0:1:127;
N = 128;
% We have two signal frequency rate here.
f = [0.25, 0.5];
Fs = 10000;
% This is the frequency which will shown as the x-axis.
w_shift = 2*pi*(-N/2:N/2-1)*(1/N);
% The frequency shown on x-axis after scaling.
w_scale = (-N/2:N/2-1)*(Fs/N);

% Two signals.
x1 = 1+cos(2*pi*f(1)*n);
x2 = 1+cos(2*pi*f(2)*n);
% Define the FFT functions which computes the signals above.
X1 = fft(x1);
X2 = fft(x2);
% The magnitude of two FFT functions.
X1_magnitude = abs(X1);
X2_magnitude = abs(X2);

 % Shift the FFT functions.
 X1_shift = fftshift(X1);
 X2_shift = fftshift(X2);
 % The magnitude of the shifted FFT functions
 X1_shift_magnitude = abs(X1_shift);
 X2_shift_magnitude = abs(X2_shift);

 % We show the result in one 2x3 figure
 % This subplot plots the unshifted DFT for f=0.25
 figure(1);
 subplot(2,3,1);
 plot(w_shift,X1_magnitude);
 xlabel('frequency');
 ylabel('magnitude');
 title('X1_FFT');

 % This subplot plots the shifted DFT for f=0.25
 subplot(2,3,4);
 plot(w_shift,X1_shift_magnitude);
 xlabel('frequency');
 ylabel('magnitude');
 title('X1_shift');

 % This subplot plots the unshifted DFT for f=0.5
 subplot(2,3,2);
 plot(w_shift,X2_magnitude);
 xlabel('frequency');
```

```
    ylabel('magnitude');
    title('X2_FFT');

    % This subplot plots the shifted DFT for f=0.5
    subplot(2,3,5);
    plot(w_shift,X2_shift_magnitude);
    xlabel('frequency');
    ylabel('magnitude');
    title('X2_shift');

    % This subplot plots the shifted DFT with a 10kHz frequency scale
    % for f=0.25
    subplot(2,3,3);
    plot(w_scale,X1_shift_magnitude);
    xlabel('frequency');
    ylabel('magnitude');
    title('X1_scale');

    % This subplot plots the shifted DFT with a 10kHz frequency scale
    % for f=0.5
    subplot(2,3,6);
    plot(w_scale,X2_shift_magnitude);
    xlabel('frequency');
    ylabel('magnitude');
    title('X2_scale');
```

Ex2.m

```
% He Feng, Hanling Lei, Zikai Yang

% We used the same method from the last exercise to compute some built-in
% sinc functions, and we use FFT function to determine the magnitude and
% phase plot.

clear all;
close all;

% Ddefine the domain of n.
n = 0:1:255;
N = 256;
f = [0.15 0.2 0.4];
% The frequency which will shown on x-axis.
w = (-N/2:N/2-1)*(1/N);

% Define the functions and use FFT to compute it.
% Then shift the function and determine the magnitude and angle.
x1 = sinc(f(1)*(n-32));
X1 = fft(x1);
X1_shift = fftshift(X1);
X1_shift_magnitude = abs(X1_shift);
X1_shift_angle = angle(X1_shift);

x2 = sinc(f(1).*(n-32)).*(-1).^n;
X2 = fft(x2);
X2_shift = fftshift(X2);
X2_shift_magnitude = abs(X2_shift);
X2_shift_angle = angle(X2_shift);

```

```
31 -    x3 = sinc(f(1).*(n-32)).*cos(2*pi*f(2).*n);
32 -    X3 = fft(x3);
33 -    X3_shift = fftshift(X3);
34 -    X3_shift_magnitude = abs(X3_shift);
35 -    X3_shift_angle = angle(X3_shift);
36
37 -    x4 = sinc(f(1).*(n-32)).*cos(2*pi*f(3).*n);
38 -    X4 = fft(x4);
39 -    X4_shift = fftshift(X4);
40 -    X4_shift_magnitude = abs(X4_shift);
41 -    X4_shift_angle = angle(X4_shift);
42
43      % Plot the graphs in two figures. The first figure contains the six graphs
44      % from the first three functions, and the second figure contains two graphs
45      % from the last function.
46 -    figure(1);
47 -    subplot(2,3,1);
48 -    plot(w,X1_shift_magnitude);
49 -    xlabel('frequency');
50 -    ylabel('magnitude');
51 -    title('x1[n] magnitude graph');
52 -    subplot(2,3,4);
53 -    plot(w,X1_shift_angle);
54 -    xlabel('frequency');
55 -    ylabel('phase angle');
56 -    title('x1[n] phase angle graph');
57
58 -    subplot(2,3,2);
59 -    plot(w,X2_shift_magnitude);
60 -    xlabel('frequency');
61 -    ylabel('magnitude');
62 -    title('x2[n] magnitude graph');
63 -    subplot(2,3,5);
64 -    plot(w,X2_shift_angle);
65 -    xlabel('frequency');
66 -    ylabel('phase angle');
67 -    title('x2[n] phase angle graph');
68
69 -    subplot(2,3,3);
70 -    plot(w,X3_shift_magnitude);
71 -    xlabel('frequency');
72 -    ylabel('magnitude');
73 -    title('x3[n] magnitude graph');
74 -    subplot(2,3,6);
75 -    plot(w,X3_shift_angle);
75 -    plot(w,X3_shift_angle);
76 -    xlabel('frequency');
77 -    ylabel('phase angle');
78 -    title('x3[n] phase angle graph');
79
80 -    figure(2);
81 -    subplot(2,1,1);
82 -    plot(w,X4_shift_magnitude);
83 -    xlabel('frequency');
84 -    ylabel('magnitude');
85 -    title('x4[n] magnitude graph');
86 -    subplot(2,1,2);
87 -    plot(w,X4_shift_angle);
88 -    xlabel('frequency');
89 -    ylabel('phase angle');
90 -    title('x4[n] phase angle graph');
91
```

Ex3.m

```
% He Feng, Hanling Lei, Zikai Yang

% We chose two sound signals from the class websites, and we load each
% sound and play the original signal initially. Then we applied the
% frequency shift and time scaling in the following parts. We implement
% a high pass filter and at the end of the exercise we determined the
% inverse FFT of the signal passed through the implemented high pass
% filter.

clear all;
close all;

% Load each sound.
[y1,Fs1] = audioread('blm.wav');
[y2,Fs2] = audioread('tiger.wav');
n1 = 0:1:25916;
% Length of y1F
N1 = 25917;
n2 = 0:1:16762;
% Length of y2
N2 = 16763;
new_n1 = transpose(n1);
new_n2 = transpose(n2);

% Play each sound with 2 seconds pause.
soundsc(y1,Fs1);
pause(2);
soundsc(y2,Fs2);
% Length of data in seconds.
info1 = audioinfo('blm.wav')
info2 = audioinfo('tiger.wav')

Y1 = fft(y1);
Y2 = fft(y2);
Y1_magnitude = abs(Y1);
Y2_magnitude = abs(Y2);

% Calculate the time sample
t1 = (0:N1-1)*(1/Fs1);
t2 = (0:N2-1)*(1/Fs2);
% Calculate the frequency sample
w1 = (-N1/2:N1/2-1)*(1/N1);
w2 = (-N2/2:N2/2-1)*(1/N2);

% Plot the time sample plot and frequency sample plot of blm.wav
figure(1);
subplot(2,2,1);
plot(t1, y1);
xlabel('time');
ylabel('y-value');
title('blm.wav time sample');
subplot(2,2,3);
plot(w1,Y1_magnitude);
xlabel('frequency');
ylabel('magnitude');
title('blm.wav frequency sample');
subplot(2,2,2);
plot(t2, y2);
xlabel('time');
ylabel('y-value');
title('tiger.wav time sample');
subplot(2,2,4);
```

```
63 -    plot(w2,Y2_magnitude);
64 -    xlabel('frequency');
65 -    ylabel('magnitude');
66 -    title('tiger.wav frequency sample');
67
68      % Part a
69 -    new_y1 = y1 .* (-1).^new_n1;
70 -    new_y2 = y2 .* (-1).^new_n2;
71 -    new_Y1 = fft(new_y1);
72 -    new_Y2 = fft(new_y2);
73 -    new_Y1_magnitude = abs(new_Y1);
74 -    new_Y2_magnitude = abs(new_Y2);
75 -    pause(2);
76 -    soundsc(new_y1,Fs1);
77 -    pause(2);
78 -    soundsc(new_y2,Fs2);
79
80 -    figure(2);
81 -    subplot(2,1,1);
82 -    plot(w1,new_Y1_magnitude);
83 -    xlabel('frequency');
84 -    ylabel('magnitude');
85 -    title('Frequency content of blm.wav for part a');
86 -    subplot(2,1,2);
87 -    plot(w2,new_Y2_magnitude);
88 -    xlabel('frequency');
89 -    ylabel('magnitude');
90 -    title('Frequency content of tiger.wav for part a');
91
92
93      % Part b
94      % We downsample the y1 and y2 functions to speed up.
95 -    downsample_y1 = downsample(y1,2);
96 -    downsample_y2 = downsample(y2,2);
97
98 -    downsample_Y1 = fft(downsample_y1);
99 -    downsample_Y2 = fft(downsample_y2);
100 -   down_Y1_magnitude = abs(downsample_Y1);
101 -   down_Y2_magnitude = abs(downsample_Y2);
102 -   pause(2);
103 -   soundsc(downsample_y1,Fs1);
104 -   pause(2);
105 -   soundsc(downsample_y2,Fs2);
106
107 -   N1_new = length(downsample_y1);
108 -   N2_new = length(downsample_y2);
109 -   w1_new = (-N1_new/2:N1_new/2-1)*(1/N1_new);
110 -   w2_new = (-N2_new/2:N2_new/2-1)*(1/N2_new);
111
```

```
112     % Plot the frequency content of two signals after scaling.
113 -   figure(3);
114 -   subplot(2,1,1);
115 -   plot(w1_new,down_Y1_magnitude);
116 -   xlabel('frequency');
117 -   ylabel('magnitude');
118 -   title('Frequency content of blm.wav for part b');
119
120 -   subplot(2,1,2);
121 -   plot(w2_new,down_Y2_magnitude);
122 -   xlabel('frequency');
123 -   ylabel('magnitude');
124 -   title('Frequency content of tiger.wav for part b');
125
126     % Part c
127     % Create a high pass filter.
128 -   [b,a] = butter(20,0.25,'high');
129 -   y1_filter = filter(b,a,y1);
130 -   y2_filter = filter(b,a,y2);
131 -   new_Y1 = fft(y1_filter);
132 -   new_Y2 = fft(y2_filter);
133 -   inverse_y1_filter = ifft(new_Y1);
134 -   inverse_y2_filter = ifft(new_Y2);
135 -   pause(2);
136 -   soundsc(real(double(inverse_y1_filter)));
137 -   pause(2);
```

```
137 -   pause(2);
138 -   soundsc(real(double(inverse_y2_filter)));
139 -   new_Y1_magnitude = abs(new_Y1);
140 -   new_Y2_magnitude = abs(new_Y2);
141
142
143 -   figure(4);
144 -   subplot(2,2,1);
145     % blm.wav go through high pass filter
146 -   plot(w1,inverse_y1_filter);
147 -   xlabel('time');
148 -   ylabel('value');
149 -   title('blm.wav time plot');
150 -   subplot(2,2,2);
151     % tiger.wav go through high pass filter
152 -   plot(w2,inverse_y2_filter);
153 -   xlabel('time');
154 -   ylabel('value');
155 -   title('tiger.wav time plot');
156
157 -   subplot(2,2,3);
158 -   plot(w1,new_Y1_magnitude);
159 -   xlabel('frequency');
160 -   ylabel('magnitude');
161 -   title('blm.wav frequency plot');
162
163 -   subplot(2,2,4);
164 -   plot(w2,new_Y2_magnitude);
165 -   xlabel('frequency');
166 -   ylabel('magnitude');
167 -   title('tiger.wav frequency plot');
168
```