




MATLAB 综合实验之音乐合成-实验报告

 王治 无 03 2020010699

一、实验名称

信号与系统 - MATLAB 综合实验之音乐合成

在 MATLAB R2020b(64-bit)平台上

二、实验目的

- 1.学习并了解乐理知识
- 2.学习并了解 MATLAB 这个强大的工程工具
- 3.学习并了解 MATLAB 的声音处理、频谱分析

三、实验原理

人类听觉可以感受到的声音大体上可划分为噪音、语音、乐音，等等。 其中， 乐音的基本特征可以用**基波频率**、 **谐波频谱**和**包络波形**三个方面来描述。

1.基波频率

乐音的声调由基频决定

我们用大写英文字母 CDEFGAB 表示每个音的“音名”（或称为“音调”）， 当指定某一音名时， 它对应固定的基波信号频率

2.谐波频谱

不同乐器发出的乐音的音色则由高次谐波分量决定

如果只考虑乐音的基波成分， 每个音名对应不同频率的正弦（余弦） 波； 当引入谐波分量之后， 波形不再是简单的正弦函数， 例如， 可能接近矩形波、 锯齿波

3.包络波形

包络也是描述乐音特性的重要因素


除了前述基频和谐波表征了乐音特性之外， 对于不同类型的乐器它们的包络形状也不相同， 在电子乐器制作中习惯上称此包络为“音型”或“音形”。 实际的波形好像通信系统中经过调制的信号

四、实验内容

1.简单的合成音乐

1) 合成《东方红》

(1) 请根据《东方红》片断的简谱和“十二平均律”计算出该片断中各个乐音的频率， 在 MATLAB 中生成幅度为 1、 抽样频率为 8kHz 的正弦信号表示这些乐音。 请用 sound 函数播放每个乐音， 听一听音调是否正确。 最后用这一系列乐音信号拼出《东方红》片断， 注意控制每个乐音持续的时间要符合节拍， 用 sound 播放你合成的音乐， 听起来感觉如何？

 对应
./src/music_01.m
./generated_music/music_01.wav

(1) 请根据《东方红》片断的简谱和“十二平均律”计算出该片断中各个乐音的频率， 在 MATLAB 中生成幅度为 1、 抽样频率为 8kHz 的正弦信号表示这些乐音。 请用 sound 函数播放每个乐音， 听一听音调是否正确。 最后用这一系列乐音信号拼出《东方红》片断， 注意控制每个乐音持续的时间要符合节拍， 用 sound 播放你合成的音乐， 听起来感觉如何？

1>首先将 F 大调的各个音的频率存储到 tunes 中

MATLAB

```
1  for i = 1 : 7
2      if (i ==1)
3          tunes = zeros([7 4]);
4          tunes(3, 1) = 220; %f major-> 1 corresponding to f
5          tunes(3, 2) = 440;
6          tunes(3, 3) = 880;
7          frequency_diff = 2^(1/12).^[-4, -2, 0, 2, 3, 5, 7]'; %The semitone frequency multiplier is 2^(1/12)
8      end
9      tunes(i, 1:end) = tunes(3, 1:end) .* frequency_diff(i);
10 end
```

2>再将《东方红》的这前四小节曲谱存到 song_pitch（音调）和 song_length（音长）中。

MATLAB

```
1  %The pitch of each tone
2  song_pitch = [...
3  treble(5); treble(5); treble(6);... %5- 56
4  treble(2);... %2- --
5  treble(1); treble(1); bass(6);... %1- 16
6  treble(2)];
7  %The length of each tone
8  song_length = [...
9  1;0.5;0.5;... %5- 56
10 2;... %2- --
11 1;0.5;0.5;... %1- 16
12 2];
```

$$1 = F \frac{2}{4} \quad | \quad 5 \quad \widehat{56} \quad | \quad 2 \quad - \quad | \quad 1 \quad \widehat{16} \quad | \quad 2 \quad - \quad |$$

图 1.3: 乐曲《东方红》前四小节曲谱

3>然后每个音产生出单频正弦波（8KHZ），之后拼接起来

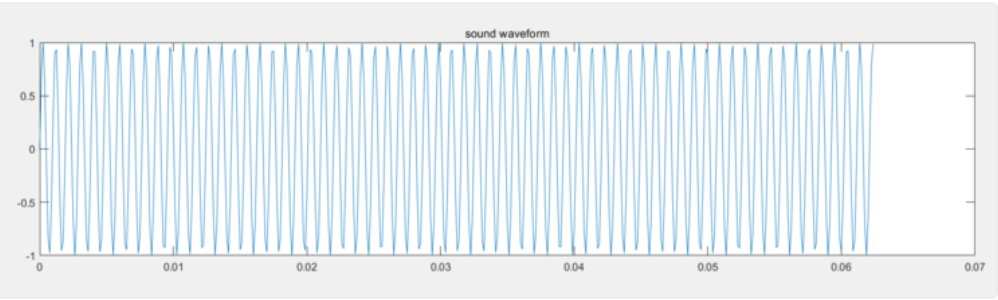
MATLAB

```
1 tone_sampling = [];  
2 for i = 1 : 1 : length(song_length)  
3     f = tunes(song_pitch(i));  
4     length_of_each_tone = song_length(i) * length_of_beat;  
5     t = linspace(0, length_of_each_tone - 1 / frequency_sampling, frequency_sampling * length_of_each_tone)';  
6     tone_sampling_temp = sin(2 * pi * f * t);  
7     %with an amplitude of 1 and sampling frequency of 8kHz  
8     tone_sampling = [tone_sampling; tone_sampling_temp];  
end
```

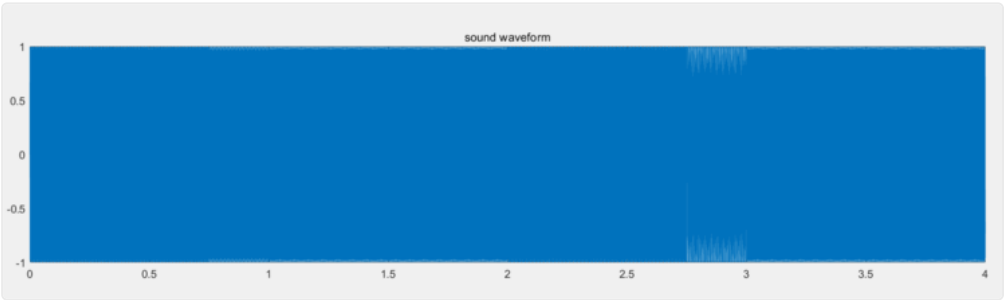
4>绘制音乐的波形并开始播放音乐

MATLAB

```
1 subplot(2,1,1);  
2 plot([0 : 500 - 1] / frequency_sampling, tone_sampling(1 : 500));  
3 title("sound waveform");  
4 subplot(2,1,2);  
5 plot([0 : length(tone_sampling) - 1] / frequency_sampling, tone_sampling);%draw sound waveform  
6 title("sound waveform");  
7 sound(tone_sampling, frequency_sampling);  
8 audiowrite('music_01.wav', tone_sampling, frequency_sampling);
```




取前 500 个采样点绘制波形



取所有采样点绘制波形

2) 包络修正每个乐音

(2) 你一定注意到 (1) 的乐曲中相邻乐音之间有“啪”的杂声，这是由于相位不连续产生了高频分量。这种噪声严重影响合成音乐的质量，丧失真实感。为了消除它，我们可以用图 1.5 所示包络修正每个乐音，以保证在乐音的邻接处信号幅度为零。此外建议用指数衰减的包络来表示

 对应

```
./src/music_02.m  
./generated_music/music_02.wav
```

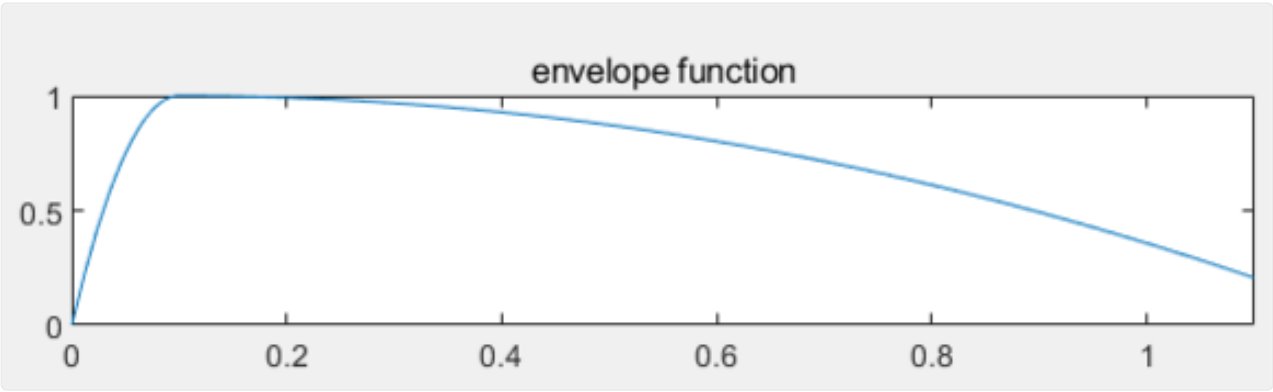
1>引入包络函数

为了解决相位不连续的问题，并模拟真实乐器的强度变化，需要给正弦波加上包络。

MATLAB

```
1 %envelope fuction  
2 function [y, z] = my_envelope(x)  
3     z = 1;  
4     y = (x >= 0 & x < 0.1) .* ( (-1./(0.1.^2)).* (x-0.1).^2+ 1) + (x >= 0.1 & x < 1.5) .* (((1 + (1.5-0.1)./(0.1-1))  
5     -1)./(1.5-0.1).^2) .* (x -0.1).^2 + 1) + (x >= 1.5 & x < 1) .* ((1.5-0.1)./(1.5-1) .*(((1 + (1.5-0.1)./(0.1-1))-1)./(
```

```
1.5-0.1).^2)) .* (x -1).^2;
end
```

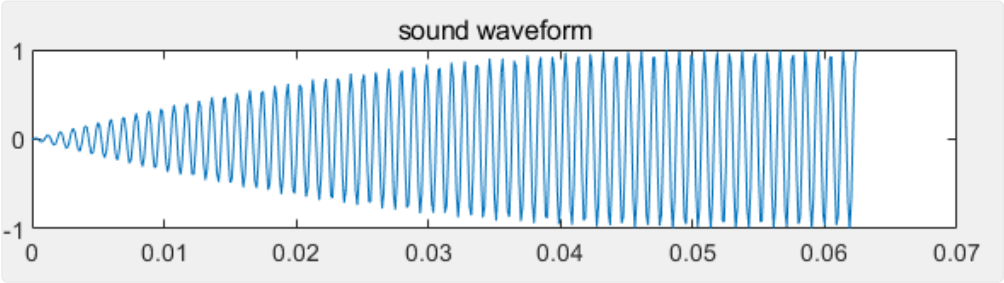


包络函数波形

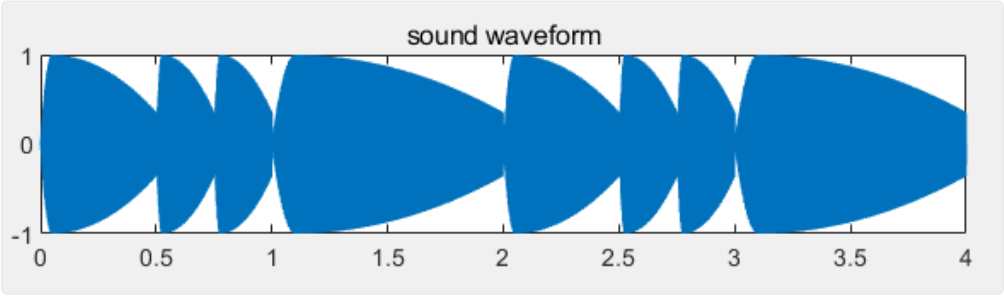
2>然后每个音产生出单频正弦波（8KHZ），再乘以包络函数，之后拼接起来

MATLAB

```
1 tone_sampling = []; %store sampled tones
2 padding = 1;
3 last_padding = 0; %Record the length of the overlap of a note(Sampl
4 ing points)
5 for i = 1 : 1 : length(song_length)
6     f = tunes(song_pitch(i)); %The pitch of each tone
7     length_of_each_tone = song_length(i) * length_of_beat; %The length of each tone
8     length_of_each_tone_padding = frequency_sampling * length_of_each_tone * padding;
9     t = linspace(0, length_of_each_tone * padding - 1 / frequency_sampling, length_of_each_tone_padding)';
10    tone_sampling_temp = my_envelope(t/length_of_each_tone) .* sin(2 * pi * f * t);%These sounds are represented by a
11    sinusoidal signal with an amplitude of 1 and sampling frequency of 8kHz(after padding)
12    if (last_padding == 0)
13        tone_sampling = [tone_sampling; tone_sampling_temp];
14    else
15        tone_sampling = [tone_sampling(1:end-last_padding); tone_sampling(end-last_padding)+tone_sampling_temp(1:last
16        _padding); tone_sampling_temp(last_padding+1:end)];
17    end
18    last_padding = round(length_of_each_tone_padding - frequency_sampling * length_of_each_tone);
19 end
```




取前 500 个采样点绘制波形



取所有采样点绘制波形

3) 音调变化

(3) 请用最简单的方法将 (2) 中的音乐分别升高和降低一个八度。（提示： 音乐播放的时间可以变化） 再难一些， 请用 resample 函数（也可以用 interp 和 decimate 函数） 将上述音乐升高半个音阶。（提示： 视计算复杂度， 不必特别精确）

 对应

```
./src/music_03.m
./generated_music/music_03_bass.wav
./generated_music/music_03_treble.wav
./generated_music/music_03_treble_half.wav
```

将音乐升高八度的方法非常简单。由于两个相邻的八度之间频率相差一倍，因此只需要在播放时将采样率扩大一倍即可

1>升高八度

MATLAB

```
1 % Raise an octave
```



```
2 subplot(3,2,1);
3 plot([0 : 500 - 1] / frequency_sampling*2, tone_sampling(1 : 500));
4 title("treble sound waveform");
5 subplot(3,2,2);
6 plot([0 : length(tone_sampling) - 1] / frequency_sampling * 2, tone_sampling); %draw sound waveform
7 title("treble sound waveform");
8 % sound(tone_sampling, frequency_sampling * 2); %play sampled tones
9 audiowrite('music_03_treble.wav', tone_sampling, frequency_sampling * 2); %store .wav
```

2>降低八度

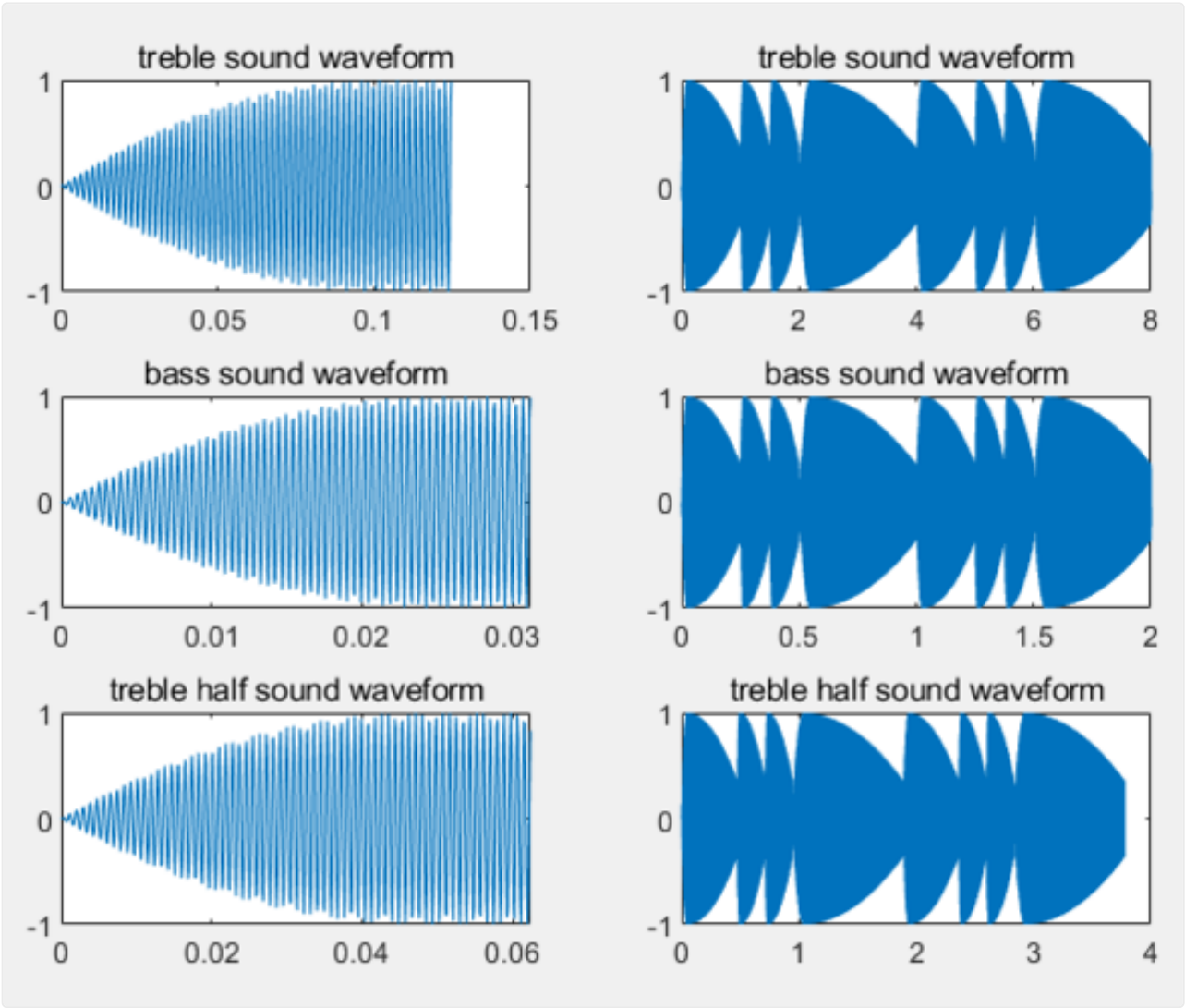
MATLAB

```
1 % Lower an octave
2 subplot(3,2,3);
3 plot([0 : 500 - 1] / frequency_sampling/2, tone_sampling(1 : 500));
4 title("bass sound waveform");
5 subplot(3,2,4);
6 plot([0 : length(tone_sampling) - 1] / frequency_sampling / 2, tone_sampling); %draw sound waveform
7 title("bass sound waveform");
8 % sound(tone_sampling, frequency_sampling / 2); %play sampled tones
9 audiowrite('music_03_bass.wav', tone_sampling, frequency_sampling / 2); %store .wav
```

3>升高半个音阶

MATLAB

```
1 %Raise half an octave
2 subplot(3,2,5);
3 tone_sampling_resample = resample(tone_sampling, round(frequency_sampling ./ 2.^(1/12)), frequency_sampling);
4 plot([0 : 500 - 1] / frequency_sampling, tone_sampling_resample(1 : 500));
5 title("treble half sound waveform");
6 subplot(3,2,6);
7 plot([0 : length(tone_sampling_resample) - 1] / frequency_sampling, tone_sampling_resample);%draw sound waveform
8 title("treble half sound waveform");
9 sound(tone_sampling_resample, frequency_sampling); %play sampled tones
10 audiowrite('music_03_treble_half.wav', tone_sampling_resample, frequency_sampling); %store .wav
```



音调变化

4) 增加谐波分量

(4) 试着在 (2) 的音乐中增加一些谐波分量， 听一听音乐是否更有“厚度” 了？ 注意谐 波分量的能量要小， 否则掩盖住基音反而听不清音调了。
(如果选择基波幅度为 1， 二次谐波幅度 0:2， 三次谐波幅度 0:3， 听起来像不像象风琴？)

对应
./src/music_04.m
./generated_music/music_04.wav

1>高次谐波分量的添加

在产生单频正弦信号的时候，加入高频正弦分量， 设置为基波分量振幅为 1， 二次谐波分量振幅为 0.2， 三次谐波分量振幅为 0.3

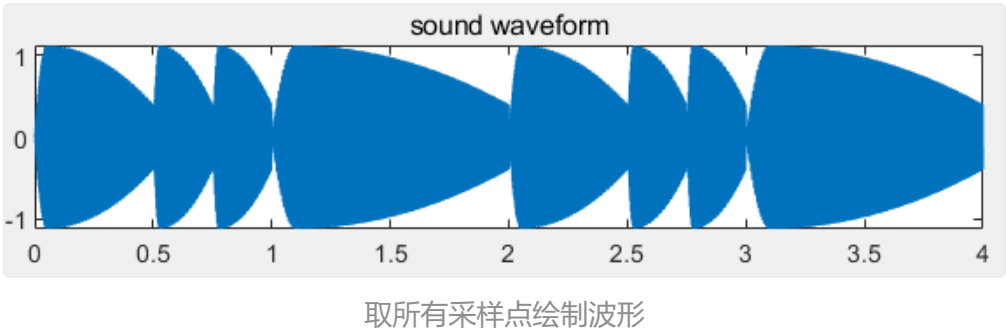
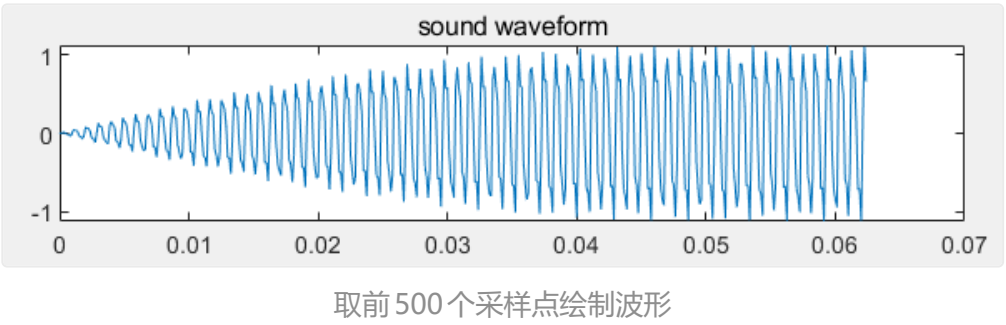
MATLAB

```
1 tone_sampling_temp = my_envelope(t/length_of_each_tone) .* (sin(2 * pi * f * t) + 0.2 * sin(2 * pi * 2 * f * t) + 0.3 * sin(2 * pi * 3 * f * t)); %These sounds are represented by a sinusoidal signal with an amplitude of 1 and sampling frequency of 8kHz(after padding)[The fundamental wave amplitude is 1, the second harmonic amplitude is 0:2, the third harmonic amplitude is 0:3]
```

2>绘制波形

MATLAB

```
1 subplot(3,1,1); %draw envelope functiofn
2 fplot(@(t) my_envelope(t), [-padding+1, padding * 1.1]);
3 title("envelope function");
4 subplot(3,1,2);
5 plot([0 : 500 - 1] / frequency_sampling, tone_sampling(1 : 500));
6 title("sound waveform");
7 subplot(3,1,3);
8 plot([0 : length(tone_sampling) - 1] / frequency_sampling, tone_sampling);
9 title("sound waveform");
```



5) 合成周杰伦的《最伟大的作品》

(5) 自选其它音乐合成， 例如贝多芬第五交响乐的开头两小节。

对应
./src/music_05.m
./generated_music/music_05.wav

1>首先将 B 大调的各个音的频率存储到 tunes 中

MATLAB

```
1 for i = 1 : 7
2     if (i ==1)
3         tunes = zeros([7 4]);
4         tunes(7, 1) = 220; %f major-> 1 corresponding to f
5         tunes(7, 2) = 440;
6         tunes(7, 3) = 880;
7         frequency_diff = 2^(1/12).^[-10, -9, -7, -5, -4, -2, 0]'; %The semitone frequency multiplier is 2^
8     (1/12)
9
```

```
10     end
    tunes(i, 1:end) = tunes(7, 1:end) .* frequency_diff(i);
end
```

2>再将《最伟大的作品》曲谱存到 song_pitch（音调）和 song_length（音长）中

```

1 %The pitch of each tone
2 song_pitch = [...
3     pause(0);...
4     % xiao chuan jing jing wang fan
5     alto(3);...
6     alto(6);...
7     treble(3);...
8     treble(3);...
9     treble(2);...
10    treble(4);...
11    pause(0);...
12    % ma di si de hai an
13    alto(3);...
14    alto(5);...
15    treble(2);...
16    treble(2);...
17    treble(1);...
18    treble(3);...
19    pause(0);...
20    %xing kong xia de ye wan
21    alto(3);...
22    alto(6);...
23    treble(1);...
24    treble(1);...
25    alto(7);...
26    treble(2);...
27    pause(0);...
28    %jiao gei fan gu dian ran
29    alto(3);...
30    alto(4);...
31    treble(1);...
32    treble(1);...
33    treble(2);...
34    alto(7);...
35    pause(0);...
36    %meng mei de tai duan zan
37    alto(3);...
38    alto(6);...
39    treble(3);...
40    treble(3);...
41    treble(2);...
42    treble(7);...
43    %meng ke qiao shang na han
44    treble(8);...
45    treble(7);...
46    treble(3);...
47    treble(2);...
48    treble(3);...
49    treble(1);...
50    pause(0);...
51    %zhe shi shang de re nao
52    treble(1);...
53    alto(7);...
54    treble(1);...
55    treble(3);...
56    treble(2);...
57    alto(6);...
58    pause(0);...
59    %chu zi gu dan
60    alto(3);...
61    treble(1);...
62    alto(7);...
63    alto(6)];

```

```

1 %The length of each tone
2 song_length = [...
3     0.5;...
4     % xiao chuan jing jing wang fan
5     0.5;...
6     0.5;...
7     0.5;...
8     0.75;...
9     0.25;...
10    1;...
11    0.5;...
12    % ma di si de hai an
13    0.5;...
14    0.5;...
15    0.5;...
16    0.75;...
17    0.25;...
18    1;...
19    0.5;...
20    %xing kong xia de ye wan
21    0.5;...
22    0.5;...
23    0.5;...
24    0.75;...
25    0.25;...
26    1;...
27    0.5;...
28    %jiao gei fan gu dian ran
29    0.5;...
30    0.5;...
31    0.5;...
32    0.75;...
33    0.25;...
34    1;...
35    0.5;...
36    %meng mei de tai duan zan
37    0.5;...
38    0.5;...
39    0.5;...
40    0.75;...
41    0.25;...
42    1.5;...
43    %meng ke qiao shang na han
44    0.5;...
45    0.5;...
46    0.5;...
47    0.75;...
48    0.25;...
49    1;...
50    0.5;...
51    %zhe shi shang de re nao
52    0.5;...
53    0.5;...
54    0.5;...
55    0.75;...
56    0.25;...
57    1;...
58    0.5;...
59    %chu zi gu dan
60    0.5;...
61    0.5;...
62    0.75;...
63    2];

```


3>然后每个音产生出单频正弦波（8KHZ），然后添加高次谐波分量，再乘以包络函数，之后拼接起来

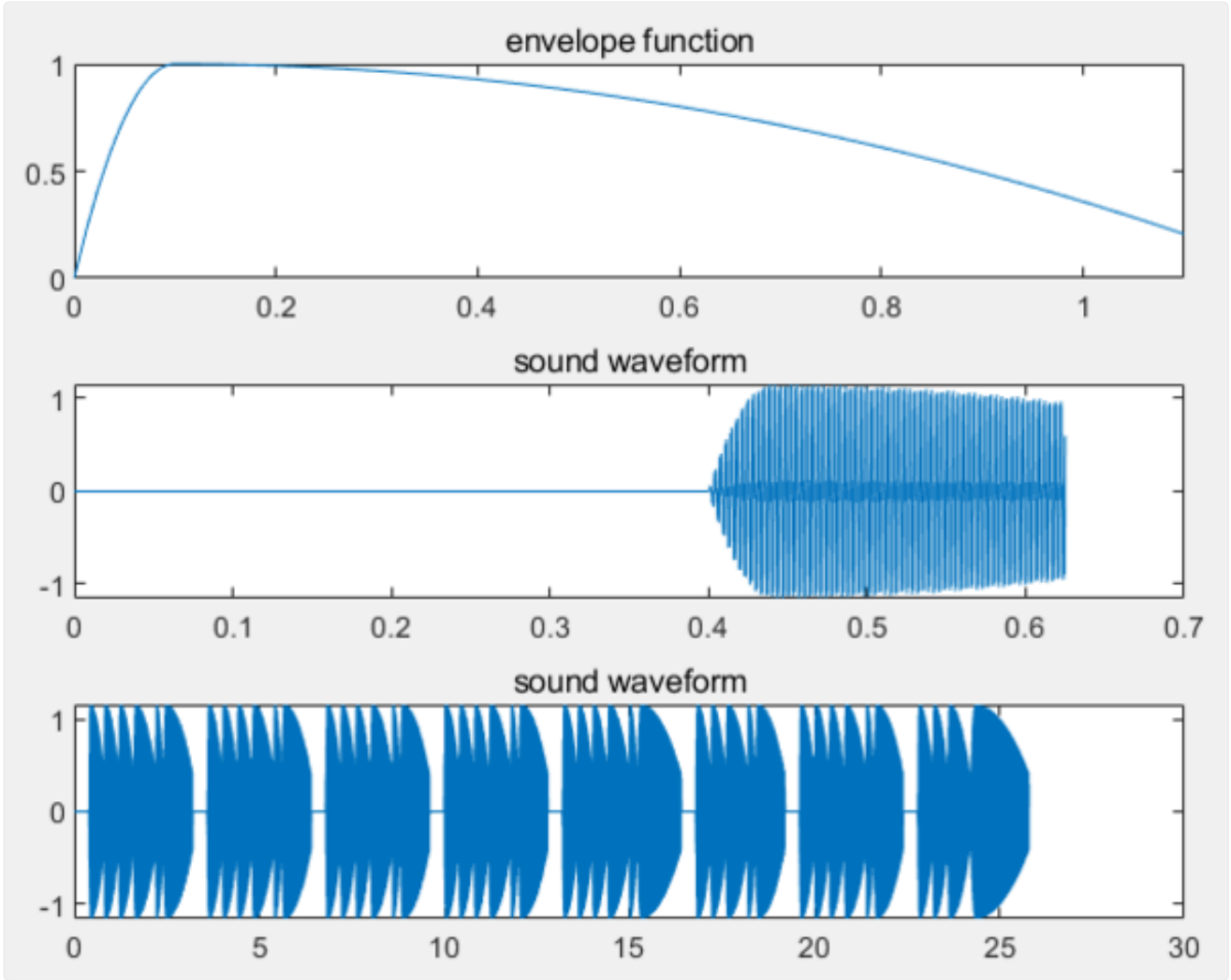
MATLAB

```
1 tone_sampling = [];  
2 padding = 1;  
3 last_padding = 0;  
4 for i = 1 : 1 : length(song_length)  
5     f = tunes(song_pitch(i)); %The pitch of each tone  
6     length_of_each_tone = song_length(i) * length_of_beat; %The length of each tone  
7     length_of_each_tone_padding = frequency_sampling * length_of_each_tone * padding;  
8     t = linspace(0, length_of_each_tone * padding - 1 / frequency_sampling, length_of_each_tone_padding)';  
9     tone_sampling_temp = my_envelope(t/length_of_each_tone) .* (sin(2 * pi * f * t) + 0.2 * sin(2 * pi * 2 * f * t) +  
0.1 * sin(2 * pi * 3 * f * t)+0.4 * sin(2 * pi * 4 * f * t)); %These sounds are represented by a sinusoidal s  
10 econd harmonic amplitude is 0:2, the third harmonic amplitude is 0:3]  
11     if (last_padding == 0)  
12         tone_sampling = [tone_sampling; tone_sampling_temp];  
13     else  
14         tone_sampling = [tone_sampling(1:end-last_padding); tone_sampling(end-last_padding)+tone_sampling_temp(1:last  
15 _padding); tone_sampling_temp(last_padding+1:end)];  
16     end  
    last_padding = round(length_of_each_tone_padding - frequency_sampling * length_of_each_tone);  
end
```

4>绘制波形开始播放

MATLAB


```
1 subplot(3,1,1); %draw envelope functiofn  
2 fplot(@t my_envelope(t), [-padding+1, padding * 1.1]);  
3 title("envelope function");  
4 subplot(3,1,2);  
5 plot([0 : 5000 - 1] / frequency_sampling, tone_sampling(1 : 5000));  
6 title("sound waveform");  
7 subplot(3,1,3);  
8 plot([0 : length(tone_sampling) - 1] / frequency_sampling, tone_sampling);  
9 title("sound waveform");  
10  
11 sound(tone_sampling, frequency_sampling); %play sampled tones  
12 audiowrite('music_05.wav', tone_sampling, frequency_sampling); %store .wav
```



2. 用傅里叶级数分析音乐

6) 导入 Guitar.mat，播放 fmt.wav

(6) 先用 wavread 函数载入光盘中的 fmt.wav 文件， 播放出来听听效果如何？ 是否比刚才的合成音乐真实多了？

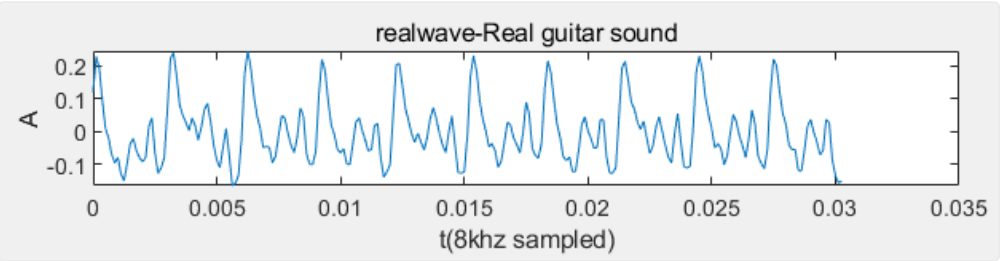
 对应

```
./src/music_06.m
./generated_music/music_06_realwave.wav
./generated_music/music_06_wave2proc.wav
./generated_music/music_06_fmt.wav
```

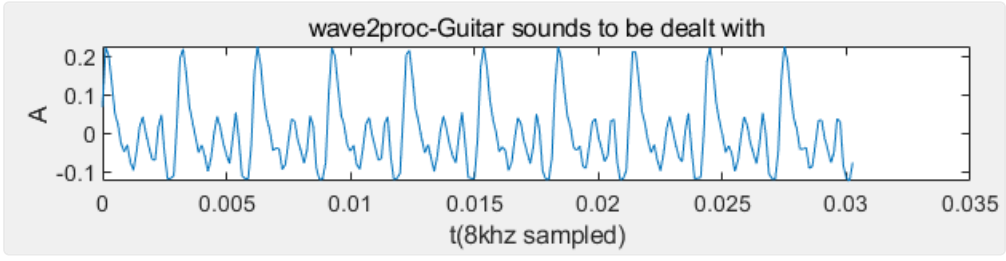
1>载入 realwave 和 wave2proc，并绘制波形

MATLAB


```
1 subplot(3, 1, 1);
2 t1 = linspace(0, length(realwave) / frequency_sampling_guitar - 1/frequency_sampling_guitar, length(realwave));
3 plot(t1, realwave);
4 title('realwave-Real guitar sound');
5 xlabel('t(8khz sampled)');
6 ylabel('A');
7 % sound(realwave, frequency_sampling_guitar);
8 audiowrite('music_06_realwave.wav', realwave, frequency_sampling_guitar); %store .wav
9
10 subplot(3, 1, 2);
11 t2 = linspace(0, length(wave2proc) / frequency_sampling_guitar - 1/frequency_sampling_guitar, length(wave2proc));
12 plot(t2, wave2proc);
13 title('wave2proc-Guitar sounds to be dealt with');
14 xlabel('t(8khz sampled)');
15 ylabel('A');
16 % sound(wave2proc, frequency_sampling_guitar);
17 audiowrite('music_06_wave2proce.wav', wave2proc, frequency_sampling_guitar);
```



realwave 波形



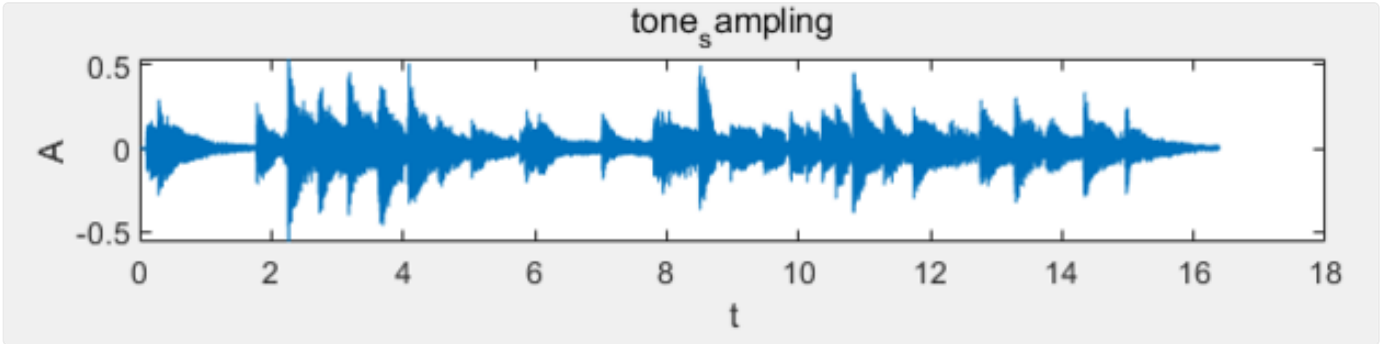
wave2proc 波形

 wave2proc 的波形非常均匀规律，而 realwave 的波形则不是非常规律。

2>播放 fmt.wav

MATLAB

```
1 [tone_sampling, frequency_sampling] = audioread('resource/fmt.wav');
2 subplot(3, 1, 3);
3 t3 = linspace(0, length(tone_sampling) / frequency_sampling - 1/frequency_sampling, length(tone_sampling));
4 plot(t3, tone_sampling);
5 title('tone_sampling');
6 xlabel('t');
7 ylabel('A');
8 sound(tone_sampling, frequency_sampling);
9 audiowrite('music_06_fmt.wav', tone_sampling, frequency_sampling);
```



fmt.wav 波形

7) 去除真实乐曲中的非线性谐波和噪声

(7) 你知道待处理的 wave2proc 是如何从真实值 realwave 中得到的么？ 这个预处理过程可以去除真实乐曲中的非线性谐波和噪声， 对于正确分析音调是非常重要的。 提示： 从时域做， 可以继续使用 resample 函数


 对应

```
./src/music_07.m
./generated_music/music_07_modified_realwave.wav
```

1> 将去除真实乐曲中的非线性谐波和噪声的功能包装成一个函数

MATLAB

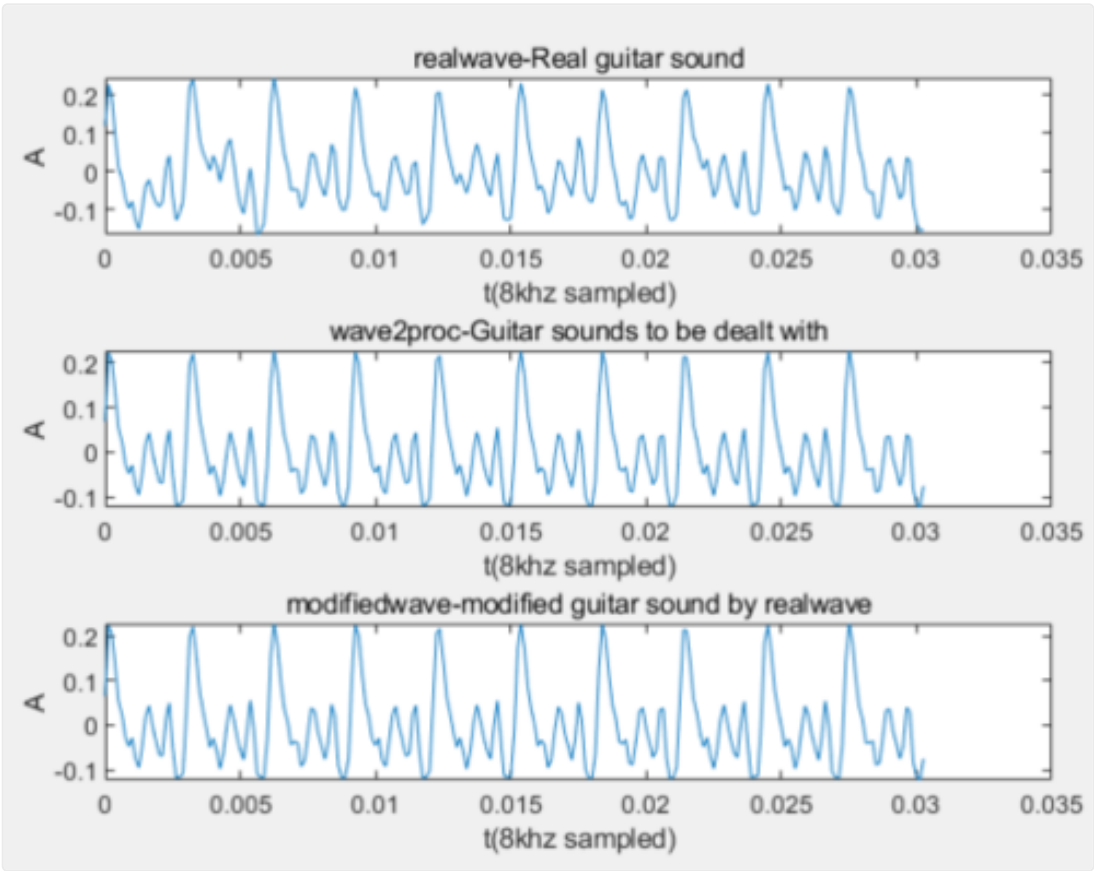
```
1 function sampling_resample = my_resample(x)
2     resample_temp = resample(x,100,1);
3     len = round(length(resample_temp)/10);
4     sampling_resample = zeros([len,1]);
5     for i = 1:1:10
6         sampling_resample = sampling_resample+resample_temp((i-1)*len+1:i*len);
7     end
8     sampling_resample = [sampling_resample/10; sampling_resample/10];
9     sampling_resample = [sampling_resample; sampling_resample; sampling_resample; sampling_resample; sampling_resample];
10 e];
11     sampling_resample = resample(sampling_resample, 1, 100);
    end
```

 输入原始信号， 输出真实乐曲中的非线性谐波和噪声

2> 绘制波形

MATLAB


```
1 subplot(3, 1, 1);
2 plot([0 : length(realwave)-1] / frequency_sampling, realwave);
3 title('realwave-Real guitar sound');
4 xlabel('t(8khz sampled)');
5 ylabel('A');
6
7 subplot(3, 1, 2);
8 plot([0 : length(wave2proc)-1] / frequency_sampling, wave2proc);
9 title('wave2proc-Guitar sounds to be dealt with');
10 xlabel('t(8khz sampled)');
11 ylabel('A');
12
13 subplot(3, 1, 3);
14 realwave_sampling_resample = my_resample(realwave);
15 plot([0 : length(realwave_sampling_resample)-1] / frequency_sampling, realwave_sampling_resample );
16 title('modifiedwave-modified guitar sound by realwave');
17 xlabel('t(8khz sampled)');
18 ylabel('A');
19 sound(realwave_sampling_resample, frequency_sampling);
20 audiowrite('music_07_modified_realwave.wav', realwave_sampling_resample, frequency_sampling);
%store .wav
```



realwave-Real guitar sound
wave2proc-Guitar sounds to be dealt with
modifiedwave-modified guitar sound by realwave

8) 分析谐波分量

(8) 这段音乐的基频是多少？是哪个音调？请用傅里叶级数或者变换的方法分析它的谐波分量分别是什么。提示：简单的方法是近似取出一个周期求傅里叶级数但这样明显不准确，因为你应该已经发现基音周期不是整数（这里不允许使用 `resample` 函数）。复杂些的方法是对整个信号求傅里叶变换（回忆周期性信号的傅里叶变换），但你可能发现无论你怎么提高频域的分辨率，也得不到精确的包络（应该近似于冲激函数而不是 `sinc` 函数），可选的方法是增加时域的数据量，即再把时域信号重复若干次，看看这样是否效果好多了？请解释之。

 对应

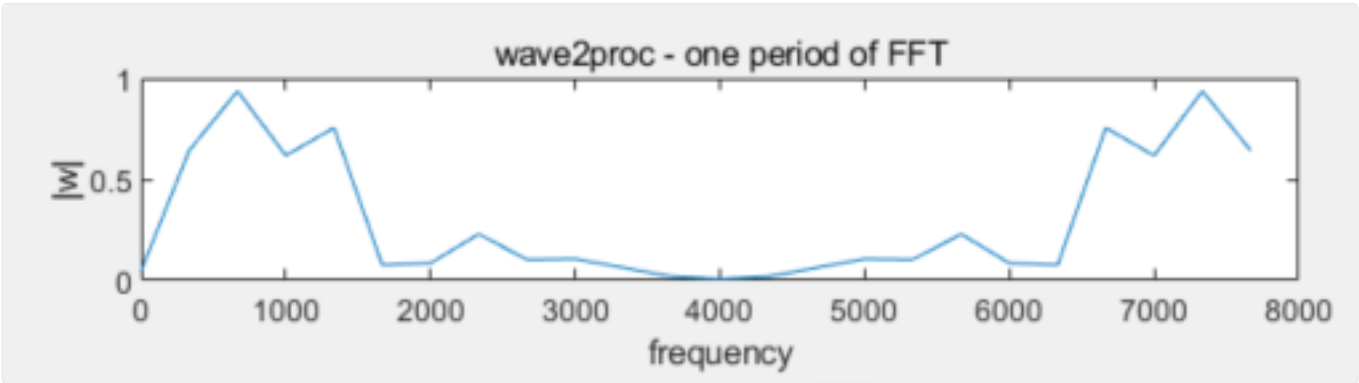
`./src/music_08.m`

wave2proc 有 10 个周期

1>对 wave2proc 的一个周期进行 fft

```
1 % wave2proc - one period of FFT
2 subplot(3, 1, 1);
3 fft1 = fft(wave2proc(1:round(end/10)));
4 plot([0 : length(fft1) - 1] / (round(length(wave2proc)/10)/frequency_sampling), abs(fft1));
5 title('wave2proc - one period of FFT');
6 xlabel('frequency');
7 ylabel('|w|');
```

MATLAB



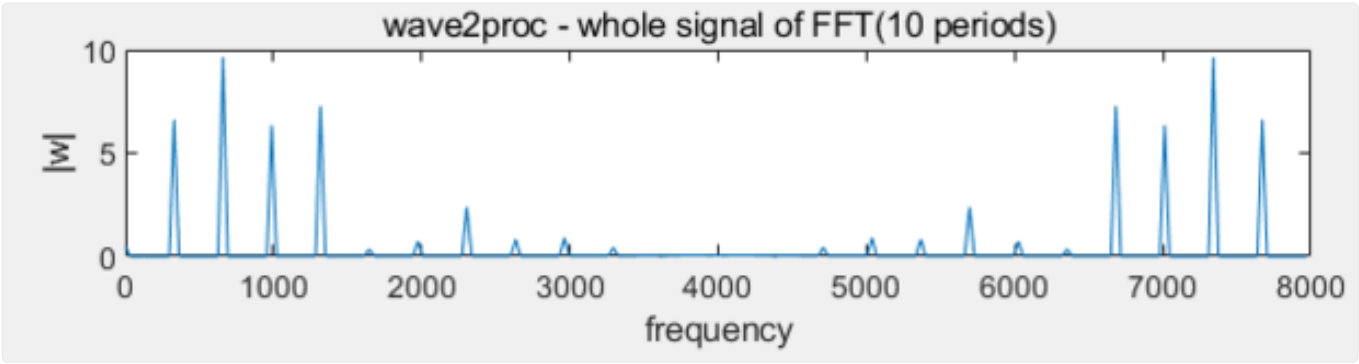
wave2proc-one period of FFT

2>对整个 wave2proc (10 个周期) 进行 fft

```
1 % wave2proc - whole signal of FFT(10 periods)
2 subplot(3, 1, 2);
3 fft2 = fft(wave2proc);
```

MATLAB

```
4 plot([0 : length(fft2) - 1] / (length(wave2proc)/frequency_sampling), abs(fft2));
5 title('wave2proc - whole signal of FFT(10 periods)');
6 xlabel('frequency');
7 ylabel('|w|');
```

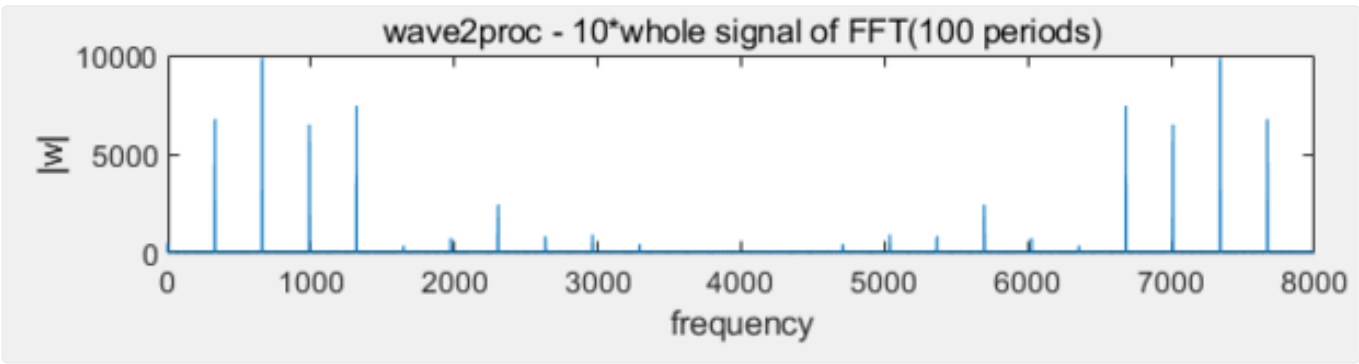


wave2proc-whole signal of FFT(10 peiods)

3>对 10 个 wave2proc（10 个周期） 进行 fft

MATLAB

```
1 %wave2proc - 10*whole signal of FFT(100 periods
2 subplot(3, 1, 3);
3 for i = 1 : 1 : 10
4     wave2proc = [wave2proc; wave2proc];
5 end
6 fft3 = fft(wave2proc);
7 plot([0 : length(fft3) - 1] / (length(wave2proc)/frequency_sampling), abs(fft3));
8 title('wave2proc - 10*whole signal of FFT(100 periods)');
9 xlabel('frequency');
10 ylabel('|w|');
```



wave2proc-10 * whole signal of FFT(100 periods)

4>获得谐波分量

将获得谐波分量的功能包装成一个函数，输入为 fft（）分析得到的频谱，输出为谐波，并打印出来

MATLAB

```
1 %The intensity of each harmonic component
2 harmonic= my_get_harmonic(fft3);
3 disp([0 : length(harmonic) - 1]', harmonic]);
4
5 function harmonic = my_get_harmonic(fft)
6     delta = 10 * 2^10;
7     n = [0 : length(fft) - 1];
8     harmonic = abs(fft(mod(n, delta) == 0));
9     harmonic = harmonic / harmonic(2);
10 end
```

输出谐波分量：

MATLAB


```
1      0      0.0739
2     1.0000     1.0000
3     2.0000     1.4572
4     3.0000     0.9587
5     4.0000     1.0999
6     5.0000     0.0523
7     6.0000     0.1099
8     7.0000     0.3589
9     8.0000     0.1240
```



```
10      9.0000      0.1351
11     10.0000      0.0643
12     11.0000      0.0019
13     12.0000      0.0058
14     13.0000      0.0068
15     14.0000      0.0047
16     15.0000      0.0033
17     16.0000      0.0026
18     17.0000      0.0021
19     18.0000      0.0018
20     19.0000      0.0014
21     20.0000      0.0012
22     21.0000      0.0011
23     22.0000      0.0009
24     23.0000      0.0008
25     24.0000      0.0008
```

9) 自动分析乐曲的音调和节拍

(9) 再次载入 `fmt.wav`，现在要求你写一段程序，自动分析出这段乐曲的音调和节拍！如果你觉得太难就允许手工标定出每个音调的起止时间，再不行你就把每个音调的数据都单独保存成一个文件，然后让 MATLAB 对这些文件进行批处理。注意：不允许逐一地手工分析音调。编辑音乐文件，推荐使用“CoolEdit”编辑软件。

 对应
./src/music_09.m

进行模块化设计：

1>Plot

```
1 function my_plot(x,y1,y2,y3,y4,y5,frequency_sampling,find_idx,component_record)
```

MATLAB

2>load_file

```
1 function [x,frequency_sampling] = my_load_file(file_path)
```

MATLAB

3>Split music

```
1 function [y1,y2,y3,y4,y5] = my_split_music(x,frequency_sampling)
```

MATLAB

4>Find idx

```
1 function [find_idx] = my_find_idx(y5,frequency_sampling)
```

MATLAB

5>Find basic frequency in each piece of music

```
1 function [base_freqs,base_freq_idx,Xs,T1s,modified_base_freqs,std_freqs] = my_find_base_freq(find_idx,frequency_sampling,x)
```

MATLAB

6>Sort by freqs

```
1 function [base_freqs,base_freq_idx,Xs,T1s,modified_base_freqs] = my_sort_by_freqs(base_freqs,base_freq_idx,Xs,T1s,modified_base_freqs)
```

MATLAB

7>Get parameters

MATLAB

```
1 function [component_res,component_record] = my_get_parameters(std_freqs,modified_base_freqs,base_freq_idx,Xs)
```

8>record

MATLAB

```
1 function [component_record,base_freq_record]= my_record(std_freqs,component_record)
```

9>generate_std_freqs

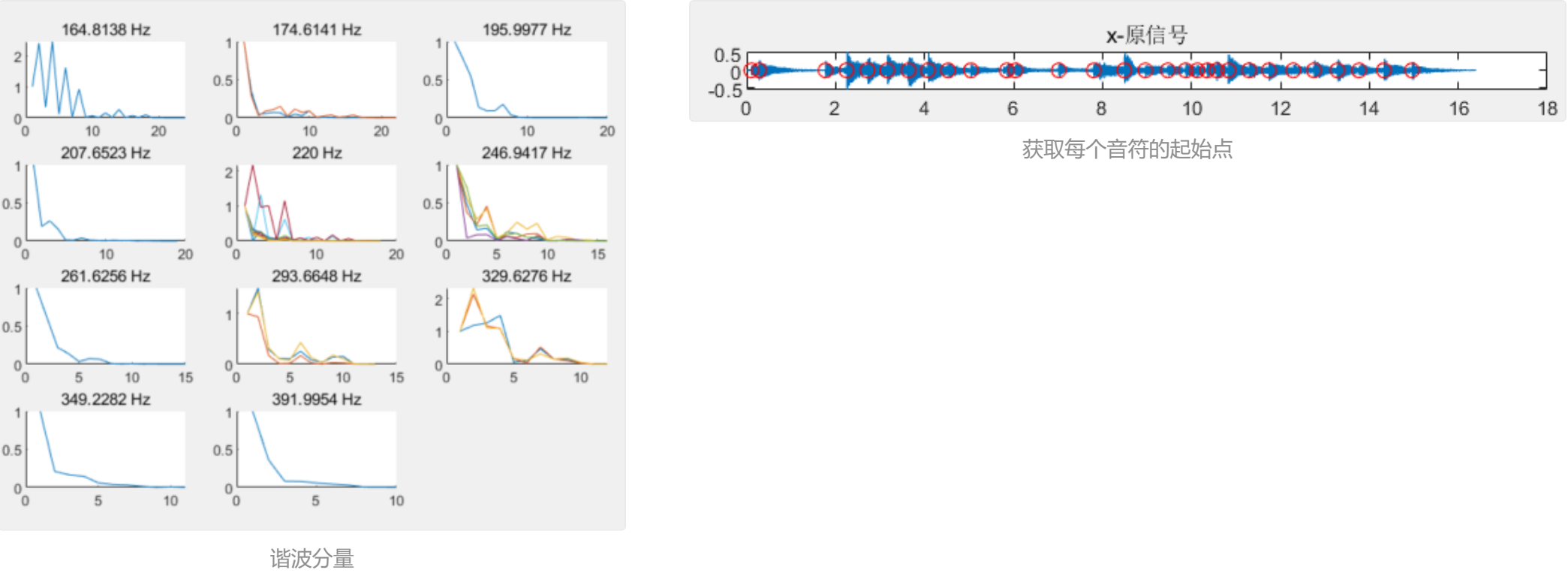
MATLAB

```
1 function std_freqs = generate_std_freqs(base_freq, max_freq)
```

10>nearest_search: find the nearest element


MATLAB

```
1 function [val, idx] = nearest_search(list, target, is_sorted_list)
```



10) 用 wave2proc 的傅里叶级数演奏《东方红》

(10) 用 (7) 计算出来的傅里叶级数再次完成第 (4) 题， 听一听是否像演奏 fmt.wav 的吉他演奏出来的？

 对应
./src/music_10.m
./generated_music/music_10.wav

1>用 wave2proc 的 fft 获得谐波分量

MATLAB

```
1 fft3 = fft(wave2proc);
2 % plot([0 : length(fft3) - 1] / (length(wave2proc)/frequency_sampling), abs(fft3));
3 % title('wave2proc - 10*whole signal of FFT(100 periods)');
4 % xlabel('frequency');
5 % ylabel('|w|');
6
7 %The intensity of each harmonic component
8 harmonic= my_get_harmonic(fft3);
9 disp([[0 : length(harmonic) - 1]', harmonic]);
```

MATLAB

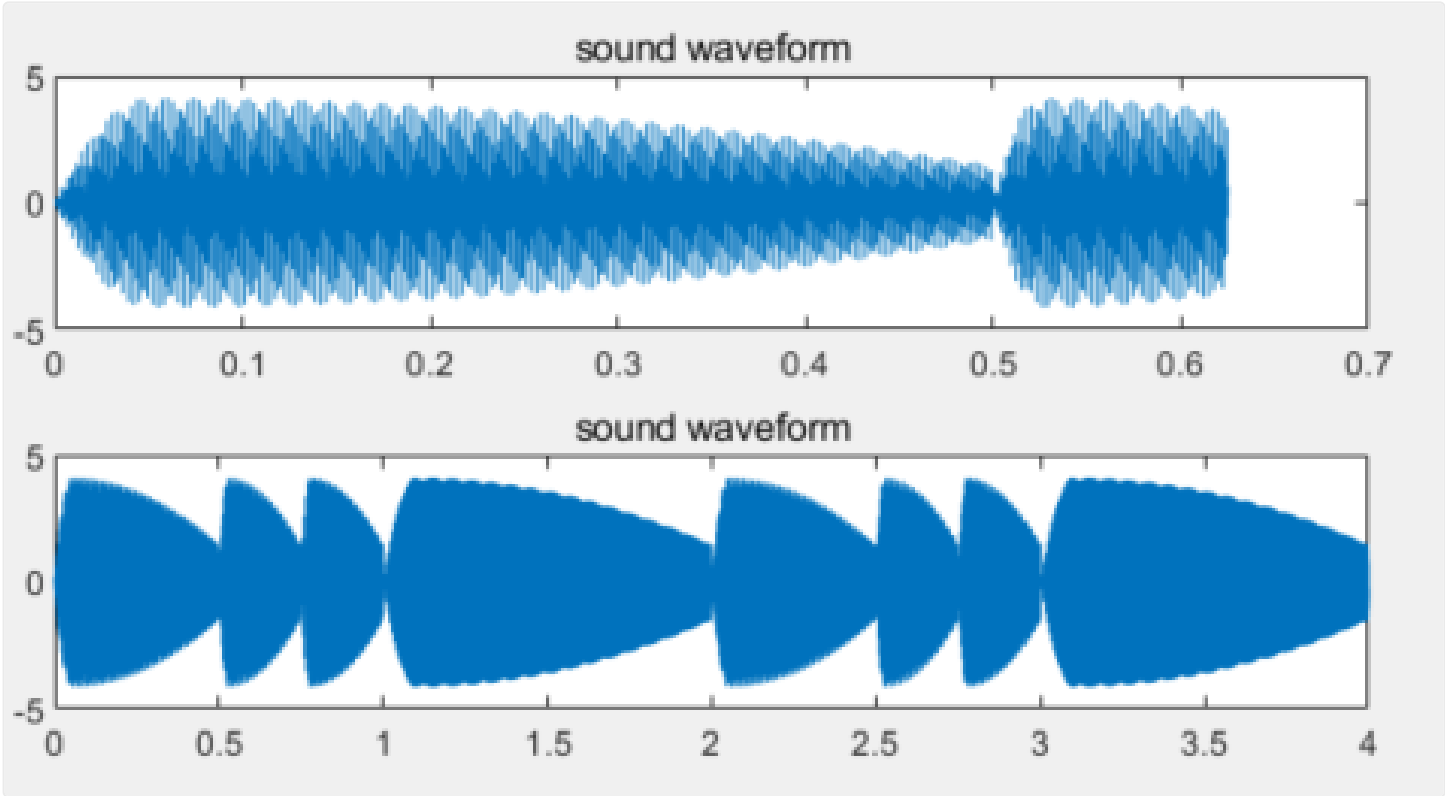
```
1 function harmonic = my_get_harmonic(fft)
```

```
2    delta = 10 * 2^10;
3    n = [0 : length(fft) - 1];
4    harmonic = abs(fft(mod(n, delta) == 0));
5    harmonic = harmonic / harmonic(2);
6    end
```

2>然后每个音产生出单频正弦波（8KHZ），然后添加高次谐波分量，再乘以包络函数，之后拼接起来

MATLAB

```
1    tone_sampling = [];
2    padding = 1;
3    last_padding = 0;
4    for i = 1 : 1 : length(song_length)
5        f = tunes(song_pitch(i)); %The pitch of each tone
6        length_of_each_tone = song_length(i) * length_of_beat; %The length of each tone
7        length_of_each_tone_padding = frequency_sampling * length_of_each_tone * padding;
8        t = linspace(0, length_of_each_tone * padding - 1 / frequency_sampling, length_of_each_tone_padding)';
9        tone_sampling_temp = zeros(size(t));
10       for j = 1 : 1 : length(harmonic)
11           tone_sampling_temp = tone_sampling_temp + harmonic(j) * sin(2 * pi * j * f * t);
12       end
13       tone_sampling_temp = my_envelope(t/length_of_each_tone) .* tone_sampling_temp;
14       % tone_sampling_temp = my_envelope(t/length_of_each_tone) .* (sin(2 * pi * f * t) + 0.2 * sin(2 * pi * 2 * f * t)
+ 0.1 * sin(2 * pi * 3 * f * t)+0.4 * sin(2 * pi * 4 * f * t)); %These sounds are represented by a sinusoidal
signal with an amplitude of 1 and sampling frequency of 8kHz(after padding)[The fundamental wave amplitude is 1, the
second harmonic amplitude is 0:2, the third harmonic amplitude is 0:3]
15       if (last_padding == 0)
16           tone_sampling = [tone_sampling; tone_sampling_temp];
17       else
18           tone_sampling = [tone_sampling(1:end-last_padding); tone_sampling(end-last_padding)+tone_sampling_temp(1:last
20 _padding); tone_sampling_temp(last_padding+1:end)];
21       end
        last_padding = round(length_of_each_tone_padding - frequency_sampling * length_of_each_tone);
    end
```



加入 wave2proc 的高次谐波之后演奏的声音波形

11) 用 fmt.wav 中提取的谐波分量演奏奏《东方红》

(11) 也许(9) 还不是很像， 因为对于一把泛音丰富的吉他而言， 不可能每个音调对应的 泛音数量和幅度都相同。 但是通过完成第 (8) 题， 你已经提取出 fmt.wav 中的很多音调， 或者说， 掌握了每个音调对应的傅里叶级数， 大致了解了这把吉他的特征。 现在就来演奏一曲《东方红》吧。 提示： 如果还是音调信息不够， 那就利用相邻音调的信息近似好了， 毕竟可以假设吉他的频响是连续变化的。



对应

./src/music_11.m

./generated_music/music_11.wav

1>从 fmt.wave 中提取谐波分量

MATLAB

```
1    [~, idx] = nearest_search(base_freq_record, f);
2    harmonic = component_record{idx};
```

MATLAB

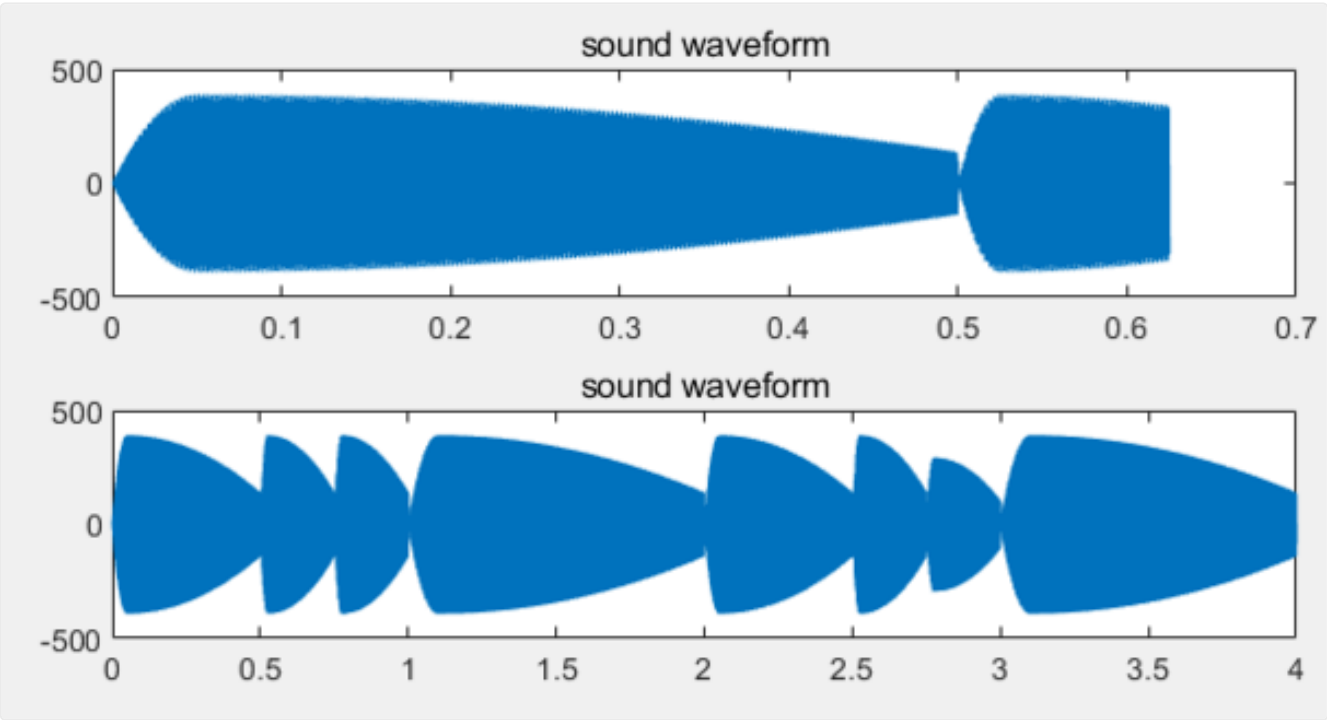
```
1 function [val, idx] = nearest_search(list, target, is_sorted_list)
2     % nearest_search: find the nearest element
3
4     if nargin < 3
5         is_sorted_list = true;
6     elseif is_sorted_list == false
7         if issorted(list) == false
8             list = sort(list);
9         end
10    end
11
12    if length(list) == 0
13        error('The list is empty!');
14    end
15
16    if length(target) == 0
17        val = [];
18        idx = [];
19    else
20        idx = zeros(size(target));
21        for i = 1 : 1 : length(target)
22            [~, idx(i)] = min(abs(list - target(i)));
23        end
24        val = list(idx);
25    end
26 end
```

2>然后每个音产生出单频正弦波（8KHZ），然后添加高次谐波分量，再乘以包络函数，之后拼接起来

MATLAB

```
1 tone_sampling = [];
2 padding = 1;
3 last_padding = 0;
4 for i = 1 : 1 : length(song_length)
5     f = tunes(song_pitch(i)); %The pitch of each tone
6     length_of_each_tone = song_length(i) * length_of_beat; %The length of each tone
7     length_of_each_tone_padding = frequency_sampling * length_of_each_tone * padding;
8     t = linspace(0, length_of_each_tone * padding - 1 / frequency_sampling, length_of_each_tone_padding)';
9     tone_sampling_temp = zeros(size(t));
10
11    [~, idx] = nearest_search(base_freq_record, f);
12    harmonic = component_record{idx};
13
14    for j = 1 : 1 : length(harmonic)
15        tone_sampling_temp = tone_sampling_temp + harmonic(j) * sin(2 * pi * j * f * t);
16    end
17    tone_sampling_temp = my_envelope(t/length_of_each_tone) .* tone_sampling_temp;
18    if (last_padding == 0)
19        tone_sampling = [tone_sampling; tone_sampling_temp];
20    else
21        tone_sampling = [tone_sampling(1:end-last_padding); tone_sampling(end-last_padding)+tone_sampling_temp(1:last
22 _padding); tone_sampling_temp(last_padding+1:end)];
23    end
24 end
```


```
last_padding = round(length_of_each_tone_padding - frequency_sampling * length_of_each_tone);
end
```



加入 fmt.wav 的谐波之后演奏的声音波形

12) GUI 设计

(12) 现在只要你掌握了某乐器足够多的演奏资料， 就可以合成出该乐器演奏的任何音乐， 在学完本书后面内容之后， 试着做一个图形界面把上述功能封装起来。

 对应

./src/music_12.m

./generated_music/music_12_the_greatest_work.wav

具体代码设计较为复杂，请看代码

1>模块化设计

2>README.md

::

0

静态文本框（直接在.fig 里修改就好）

1

普通按钮

2

列表框

3

坐标区

4

表格

5

可编辑文本

GUI-模块化设计

README.md

目录

• CH

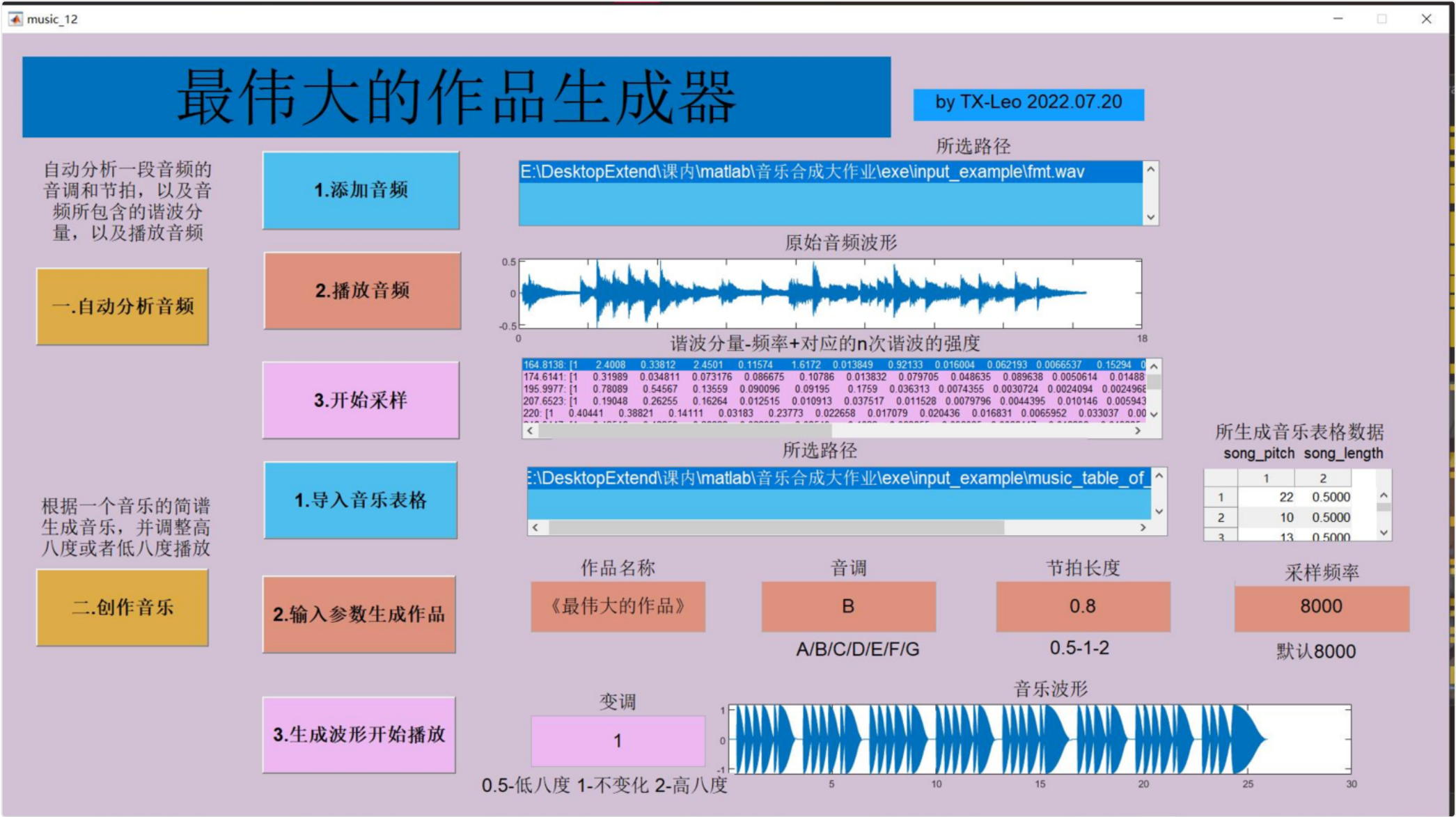
- 功能介绍
 - 1.自动分析音频
 - 2.创作音乐
- 使用方法
 - 1.自动分析音频
 - 1) 添加音频
 - 2) 播放音频
 - 3) 开始采样
 - 2.创作音乐
 - 1) 添加音乐表格
 - 2) 输入参数生成作品
 - 3) 生成波形开始播放

• ENG

- 1.Prerequisites for Deployment
- 2.Files to Deploy and Package
- 3.Definitions

README.md 框架

3>最终效果



《最伟大的作品生成器》最终效果

四、附：《最伟大的作品生成器》设计方案

CH

《最伟大的作品生成器》.exe

Author: TX-Leo

Data:2022.07.20

功能介绍

1.自动分析音频

自动分析一段音频的音调和节拍，以及音频所包含的谐波分量，以及播放音频

2.创作音乐

根据一个音乐的简谱生成音乐，并调整高八度或者低八度播放

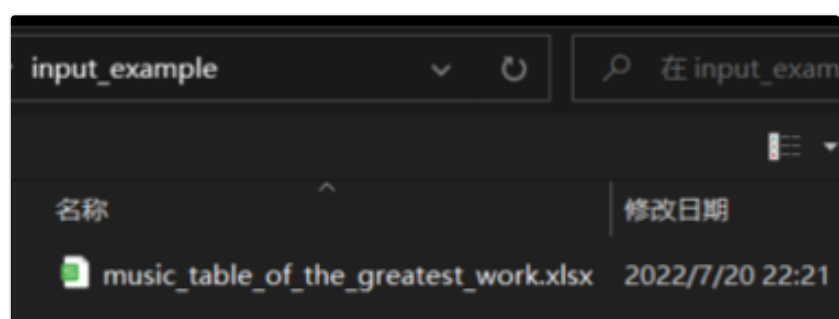
使用方法

1.自动分析音频

1) 添加音频

添加音频的例子：

input_example/fmt.wav



格式说明：

第一列对应为是低音 (1) 还是中音 (2) 还是高音 (3) 还是 pause (0)

第二列对应为音调

第三列对应为音长

如下图:

	A	B	C
1	0	0	0.5
2	2	3	0.5
3	2	6	0.5
4	3	3	0.5
5	3	3	0.75
6	3	2	0.25
7	3	4	1
8	0	0	0.5
9	2	3	0.5
10	2	5	0.5
11	3	2	0.5
12	3	2	0.75
13	3	1	0.25
14	3	3	1
15	0	0	0.5
16	2	3	0.5
17	2	6	0.5
18	0	1	0.5

(20)

<u>X X X X</u>	<u>X X X X</u>	<u>X XXX X</u>	<u>0 X X X</u>		<u>X X X X</u>	<u>X X X X</u>	<u>X X X X</u>	<u>0 3</u>	<u>6 3</u>
海	的	乡	愁	种	在	一	无	所	有
的	温	柔		寂	寞	的		枝	头
				才	能	长	出	常	玉
								小	船
									静

(22)

<u>3̣.</u>	<u>2̣</u>	<u>4̣</u>	<u>0</u>	<u>3</u>	<u>5̣</u>	<u>2̣</u>		<u>2̣.</u>	<u>1̣</u>	<u>3̣</u>	<u>0</u>	<u>3</u>	<u>6</u>	<u>1̣</u>
静	往	返		马	谛	斯		的	海	岸		星	空	下

(24)

<u>1̣.</u>	<u>7</u>	<u>2̣</u>	<u>0</u>	<u>3</u>	<u>4̣</u>	<u>1̣</u>		<u>1̣.</u>	<u>2̣</u>	<u>7</u>	<u>0</u>	<u>3</u>	<u>6</u>	<u>3̣</u>
的	夜	晚		交	给	梵		谷	点	燃		梦	美	的

(26)

<u>3̣.</u>	<u>2̣</u>	<u>2̣</u>	<u>7̣.</u>	<u>7̣</u>	<u>1̣</u>	<u>7̣</u>	<u>3̣</u>		<u>2̣.</u>	<u>3̣</u>	<u>1̣</u>	<u>0</u>	<u>1̣</u>	<u>7̣</u>	<u>1̣</u>
太	短	暂		孟	克	桥		上	呐	喊		这	世	上	

(28)

<u>3̣.</u>	<u>2̣</u>	<u>6</u>	<u>0</u>	<u>3</u>	<u>1̣.</u>	<u>7̣</u>		<u>6</u>	<u>-</u>	<u>0</u>	<u>3</u>	<u>6</u>	<u>3̣</u>
的	热	闹		出	自	孤	单						

(30)

2) 输入参数生成作品

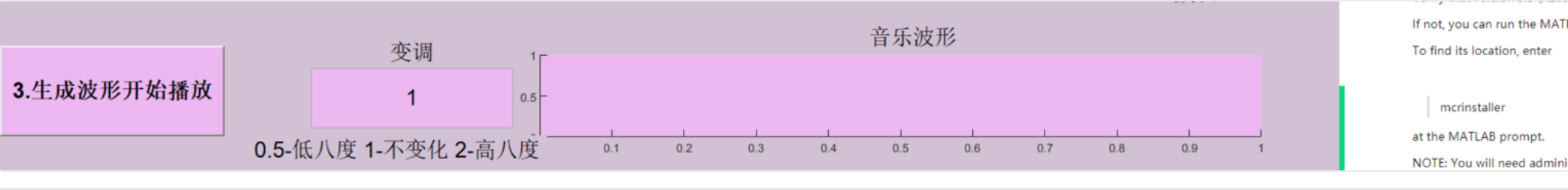
图中所示为默认参数



等一会再开始下一步

3) 生成波形开始播放

这里可以选择变调播放



ENG

1.Prerequisites for Deployment

Verify that version 9.9 (R2020b) of the MATLAB Runtime is installed.

If not, you can run the MATLAB Runtime installer.

To find its location, enter

```
mcrinstaller
```

at the MATLAB prompt.

NOTE: You will need administrator rights to run the MATLAB Runtime installer.

Alternatively, download and install the Windows version of the MATLAB Runtime for R2020b from the following link on the MathWorks website:

<https://www.mathworks.com/products/compiler/mcr/index.html>

For more information about the MATLAB Runtime and the MATLAB Runtime installer, see "Distribute Applications" in the MATLAB Compiler documentation in the MathWorks Documentation Center.

2.Files to Deploy and Package

Files to Package for Standalone

- music_12_export.exe
- MCRInstaller.exe

Note: if end users are unable to download the MATLAB Runtime using the instructions in the previous section, include it when building your component by clicking the "Runtime included in package" link in the Deployment Tool.

- This readme file

3.Definitions

For information on deployment terminology, go to

<https://www.mathworks.com/help> and select MATLAB Compiler >

Getting Started > About Application Deployment >

Deployment Product Terms in the MathWorks Documentation Center.

六、实验总结

总结来看，本次实验对我来说难度还是不小的，之前没有用 matlab 做这么大的工程，我从中学到了很多如何处理音频，如何用傅里叶变换，以及如何在 matlab 上实现以上功能

最让我收获多的是 MATLAB-GUI 设计，我想做一个集成所有功能的 GUI，所以导致我在这个设计的逻辑布置上需要思考很多，学到了 GUI 的方便与直观性，也意识到了一个 GUI 背后的代码框架应该是怎么样的。

最后感谢老师助教们对于此次实验的付出让我收获颇多，谢谢 ~

七、文件清单

MATLAB

```
1
2 |   tree.txt
3 |
4 |├docs
5 |   音乐合成大作业.pdf
6 |
7 |├exe
8 |   |   mccExcludedFiles.log
9 |   |   music_12_the_greatest_work.wav
10 |  |   README.md
11 |  |   requiredMCRProducts.txt
12 |  |   splash.png
13 |  |   《最伟大的作品生成器》.exe
14 |  |
15 |  └image
16 |      image_30UMS5vTQk.png
17 |      image_5uRwNZ0DPV.png
18 |      image_b-LjWkbXLf.png
19 |      image_kRh_0CJEDv.png
20 |      image_PfLdW_CR61.png
21 |      image_rTp08B34ZM.png
22 |      image_su6mpMsoCx.png
23 |      image_wCB5my7p_S.png
24 |      image_zeRtYZh4IM.png
25 |
26 |└input_example
27 |   fmt.wav
28 |   music_table_of_the_greatest_work.xlsx
29 |
30 |└generated_music
31 |   music_01.wav
32 |   music_02.wav
33 |   music_03_bass.wav
34 |   music_03_treble.wav
35 |   music_03_treble_half.wav
36 |   music_04.wav
37 |   music_05.wav
38 |   music_06_fmt.wav
39 |   music_06_realwave.wav
40 |   music_06_wave2proce.wav
41 |   music_07_modified_realwave.wav
42 |   music_10.wav
43 |   music_11.wav
44 |   music_12_the_greatest_work.wav
45 |
46 |
```



```
47 |report
48 |    report.pdf
49 |
50 |resource
51 |    fmt.wav
52 |    Guitar.MAT
53 |    music_12.xlsx
54 |
55 |src
56 |    music_01.m
57 |    music_02.m
58 |    music_03.m
59 |    music_04.m
60 |    music_05.m
61 |    music_06.m
62 |    music_07.m
63 |    music_08.m
64 |    music_09.m
65 |    music_10.m
66 |    music_11.m
67 |    music_12.fig
68 |    music_12.m
69 |    music_12_export.m
70 |    music_12_export.mat
```