MATLAB 大作业一：音乐合成

姓名：陈彦旭

班级：无24 学号：2022010597

目录：

MATLAB 大作业一：音乐合成
- 实验目的
- 实验步骤
  - Part1：简单的合成音乐
  - Part2：用傅里叶级数分析音乐
  - Part3：基于傅里叶级数的合成音乐
- 感想与收获
- 文件结构

实验目的

本章将基于傅里叶级数和傅里叶变换等基础知识，应用第一篇讲授的 MATLAB 编程技术，在电子音乐合成方面做一些练习，增进对傅里叶级数的理解，并能够熟练运用 MATLAB 基本指令。

实验步骤

Part1：简单的合成音乐

（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?

表格如下：

唱名	1	2	3	4	5	6	7	i
音名	F	G	A	(b)B	C	D	E	F
频率	349.23	392	440	466.16	523.25	587.33	659.25	698.45

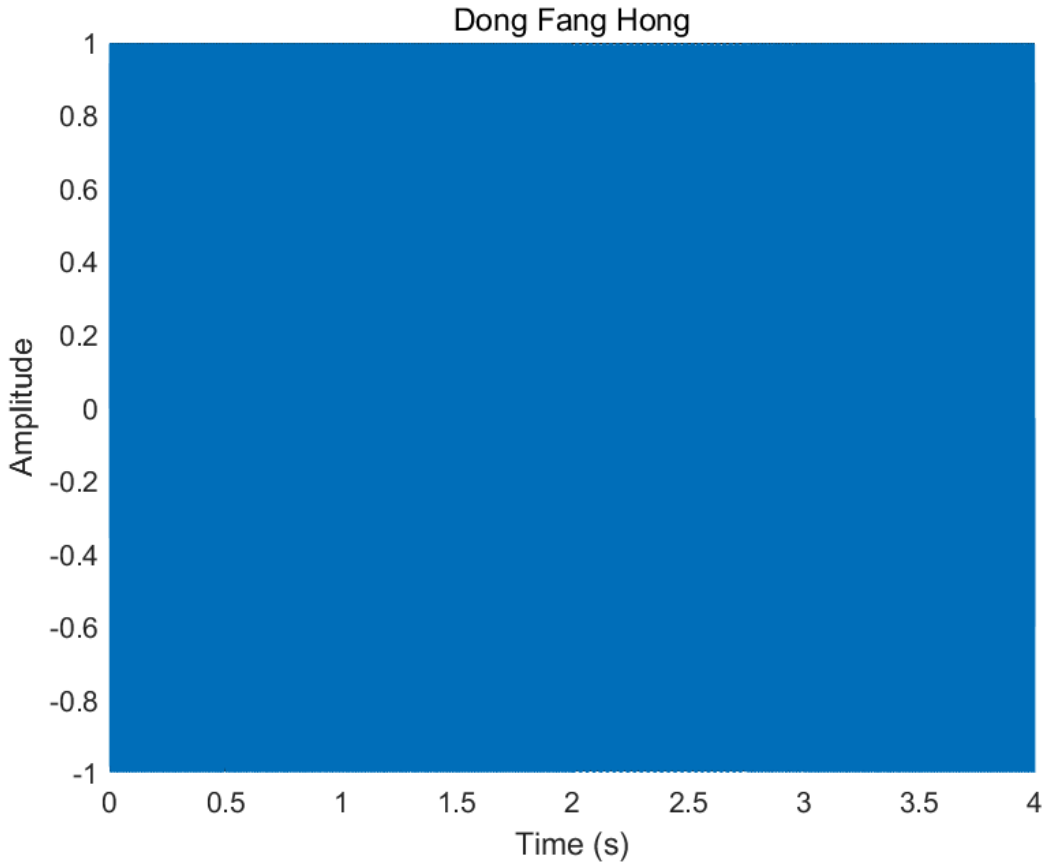
我根据两个 A 调频率（220Hz、440Hz）分别向上生成 12 个音调频率，得到一个二维数组 `freq` ，其中每一行是从 A 到 G 的完整 12 个频率。

然后根据《东方红》歌曲中每个乐音的音调以及节拍持续时间，创建二维数组 `DongFangHong` ，其中每一行的第一个元素是频率，第二个元素是持续时间。在向乐音数组 `melody` 中添加各个乐音的音调时，各次循环会改变数组大小，因此我使用元胞数组 `cell` ，最后使用 `cat` 将其展平，提升了循环的效率。

最后使用 `sound` 函数播放乐音，可以听出不同音调之间有较为明显的“不连续感”。

生成的音频文件保存在 `exp1.wav` 中。

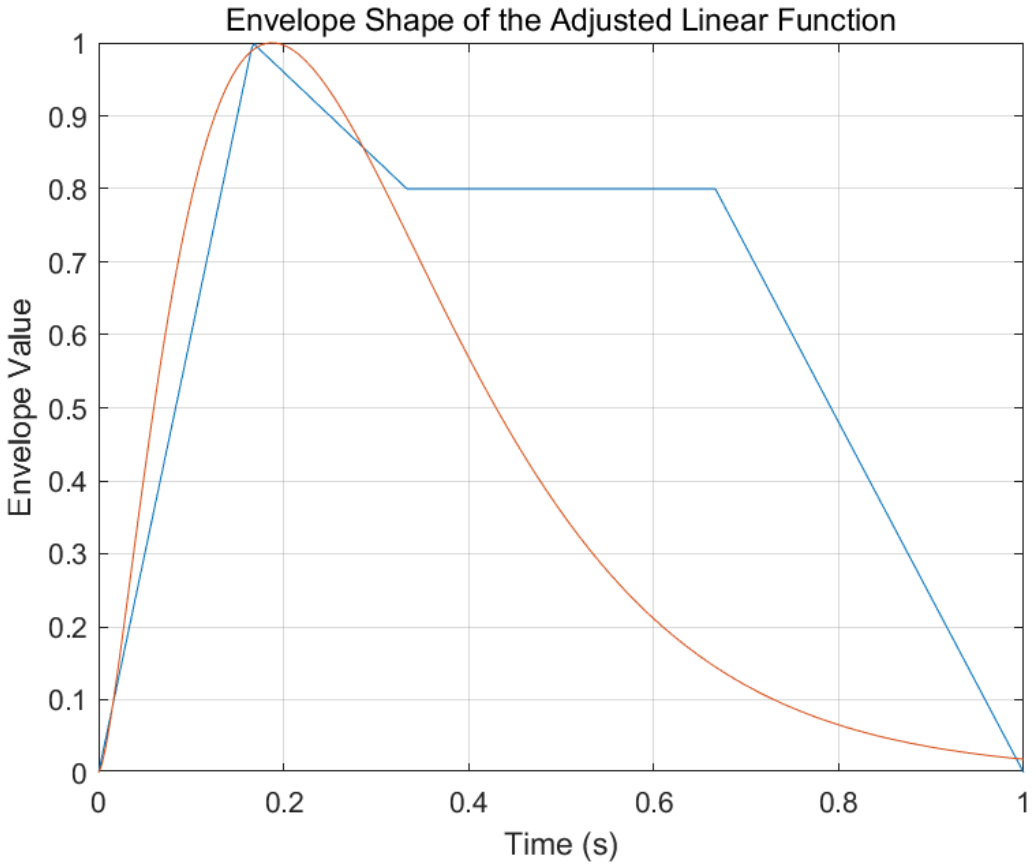
乐音波形图如下，可见各个音调幅度相同，均为1。



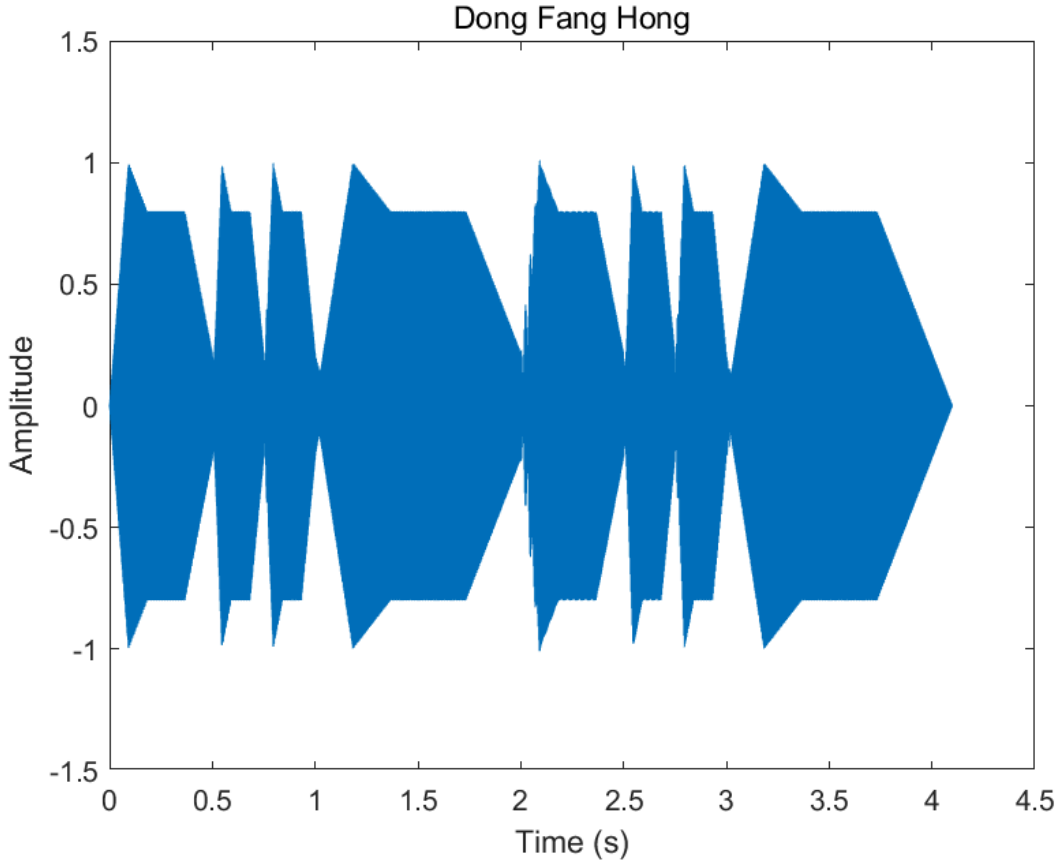
（2）你一定注意到 (1) 的乐曲中相邻乐音之间有“啪”的杂声，这是由于相位不连续产生了高频分量。这种噪声严重影响合成音乐的质量，丧失真实感。为了消除它，我们可以用图 1.5 所示包络修正每个乐音，以保证在乐音的邻接处信号幅度为零。此外建议用指数衰减的包络来表示。

为了使不同音调之间有“迭接”，增强乐音的连续性，需要使用包络修正乐音的波形。

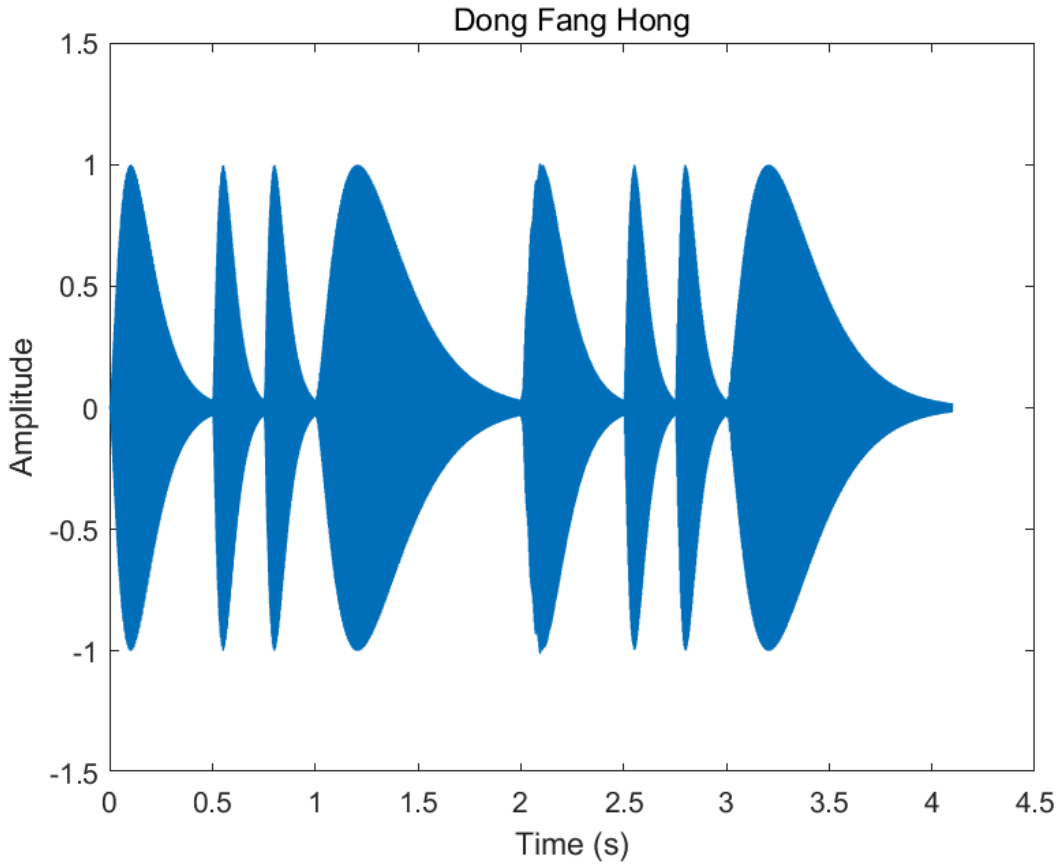
分别使用实验指导书中图 1.5 的分段折线包络和形式为 $t^A \exp(-Bt + C)$ 的指数衰减包络：



对于分段折线包络：



现在听起来比刚才好多了。然后再尝试用指数衰减包络：



可以发现效果更好了，不同音调之间的衔接明显更加顺滑。

生成的音频文件分别为 `exp2_1.wav, exp2_2.wav` 。

（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）

升高一个八度：原乐音不变，采样频率变为原来的两倍。播放时间变为原来的一半。

降低一个八度：原乐音不变，采样频率变为原来的一半。播放时间变为原来的两倍。

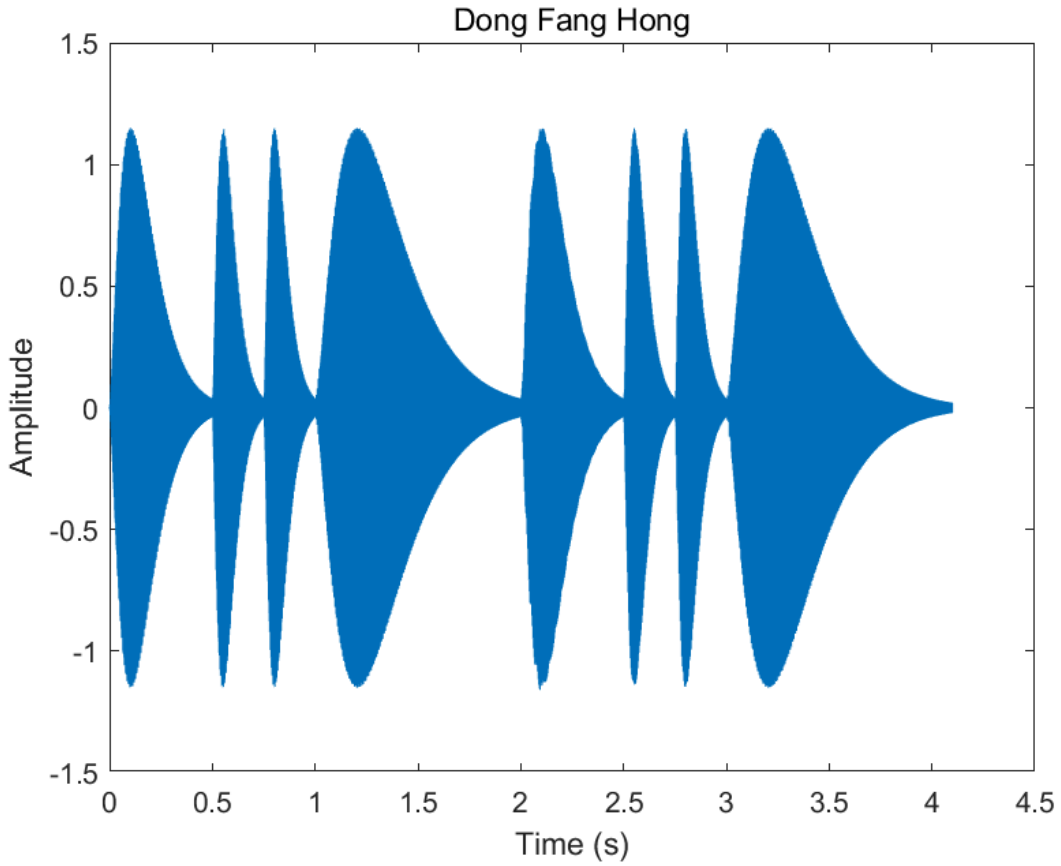
升高半个音阶：使用 `resample` 函数重新采样，采样率为原来的 $2^{1/12}$。

生成的音频文件分别为：`exp3_1.wav, exp3_2.wav, exp3_3.wav` 。

（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？）

使用谐波分量矩阵 `harmonics = [1; 0.2; 0.3]` 。高次谐波分量使得音色更加多样，加入谐波后确实更有“厚重感”，有些像风琴。

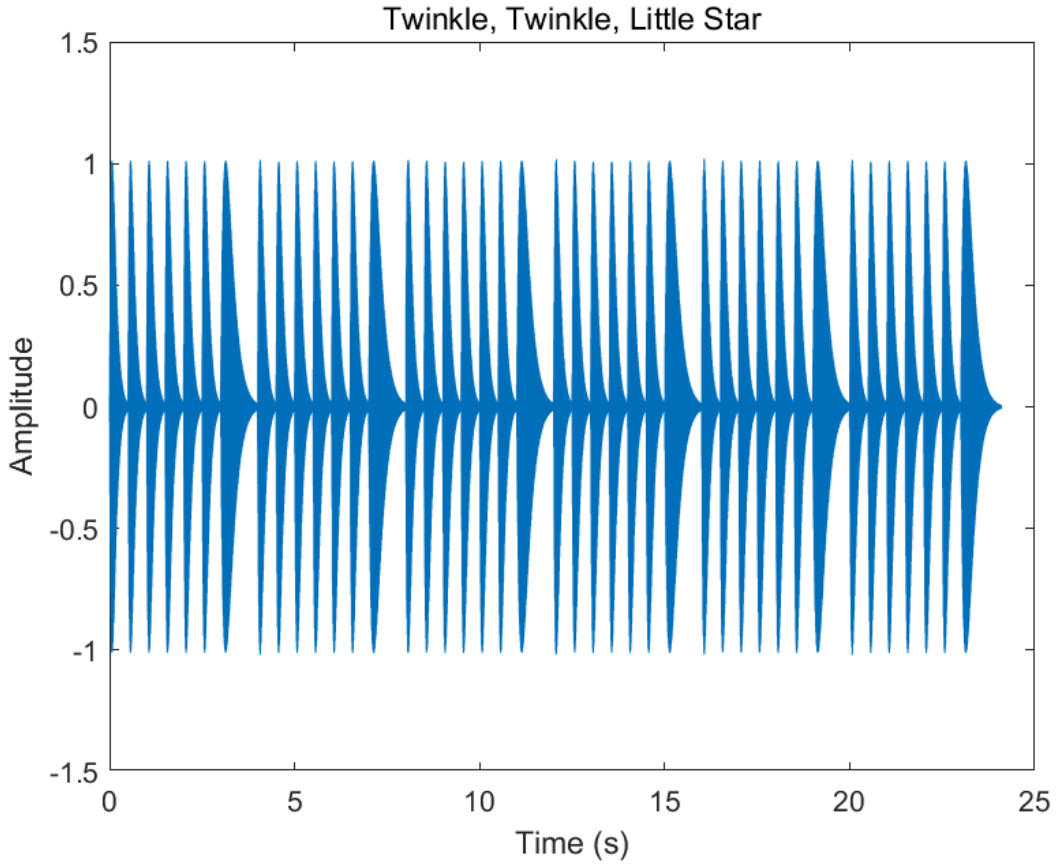
乐音波形如下：



生成的音频文件保存在 `exp4.wav` 中。

（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。

我选取的音乐是《小星星》，乐音波形如下：

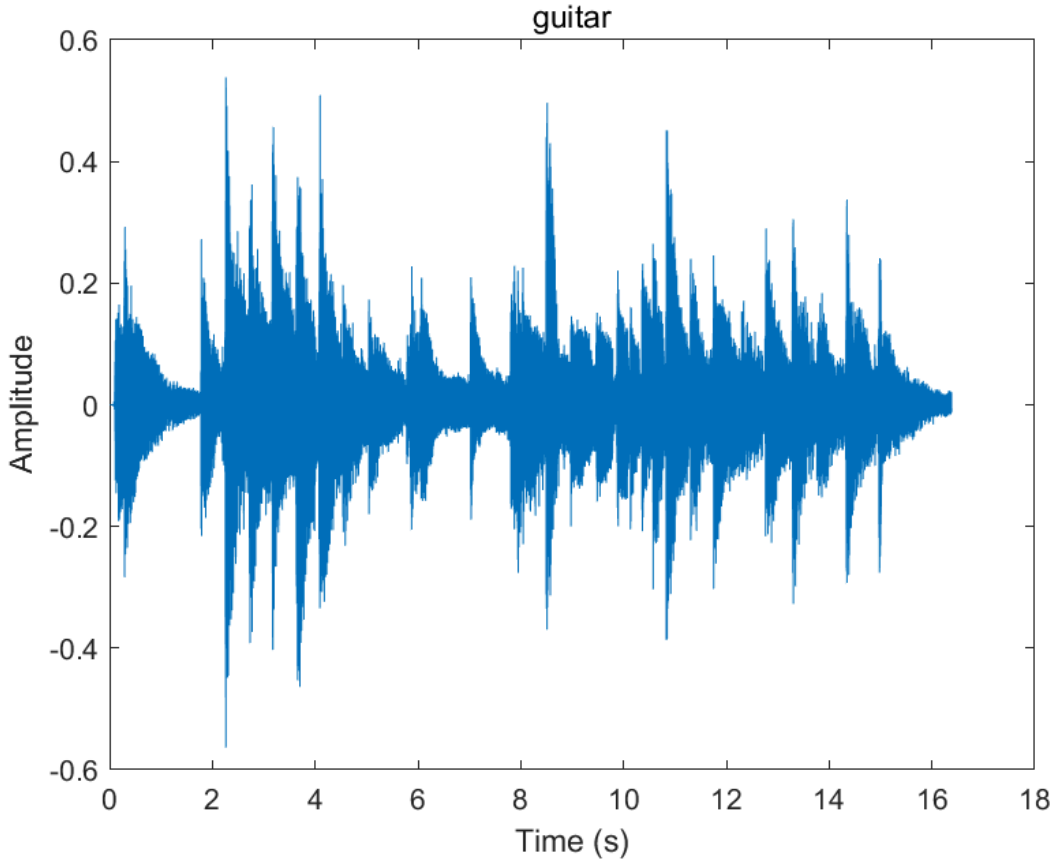


生成的音频文件保存在 `exp5.wav` 中。

Part2：用傅里叶级数分析音乐

（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？

播放出来的音乐像极了吉他音，非常真实。



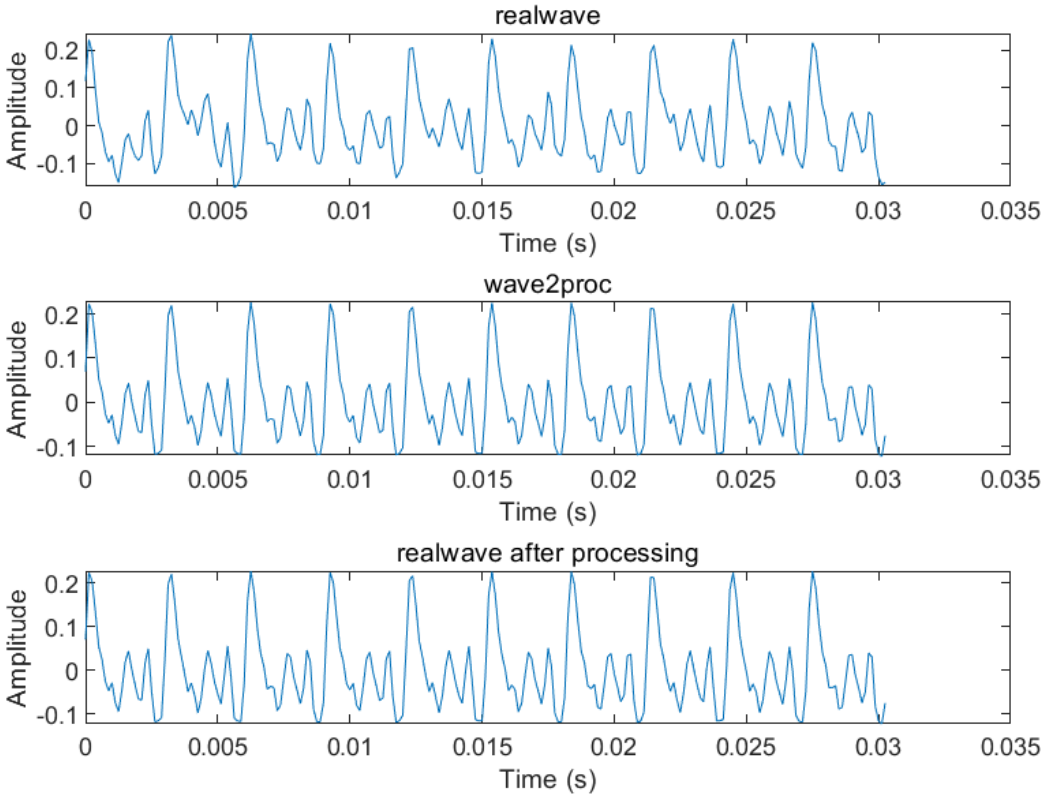
从波形图也可以看出，真实的乐音幅度不一、各个音调之间看起来“杂乱无章”、谐波分量千变万化，较为符合真实世界的情况。

（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。

首先绘制出 `realwave` 和 `wave2proc` ，可以发现该段乐音大致有 10 个周期。为了去除非线性谐波与噪声，可以采取分段累加求平均的方法：先将 `realwave` 的采样率变为原来的 10 倍，这样每个周期内的采样点数与原来的相同。然后将这 10 段求平均（累加除10），通过 `repmat` 扩展为 10 倍，再将采样率复原为 8KHz，得到处理过后的 `wave_proc` ，可以发现与 `wave2proc` 十分接近。

具体代码如下：

```
re_wave = resample(realwave, 10, 1);
wave_proc = zeros(rw_len, 1);

for i = 1:10
    wave_proc = wave_proc + re_wave((i - 1) * rw_len + 1:i * rw_len);
end

wave_proc = wave_proc / 10;
wave_proc = repmat(wave_proc, [10, 1]);
wave_proc = resample(wave_proc, 1, 10);
```



（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如何提高频域的分辨率，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。

方法一：只对第一个周期内的波形做傅里叶变换。此时效果极差，很难基波和谐波分量。

```
% method 1
waveform_1 = wave2proc;
len_1 = round(length(waveform_1) / 10);
spectrum_1 = abs(fft(waveform_1(1:len_1)));
subplot(3, 1, 1);
plot((0:len_1 - 1) * Fs / len_1, spectrum_1);
```

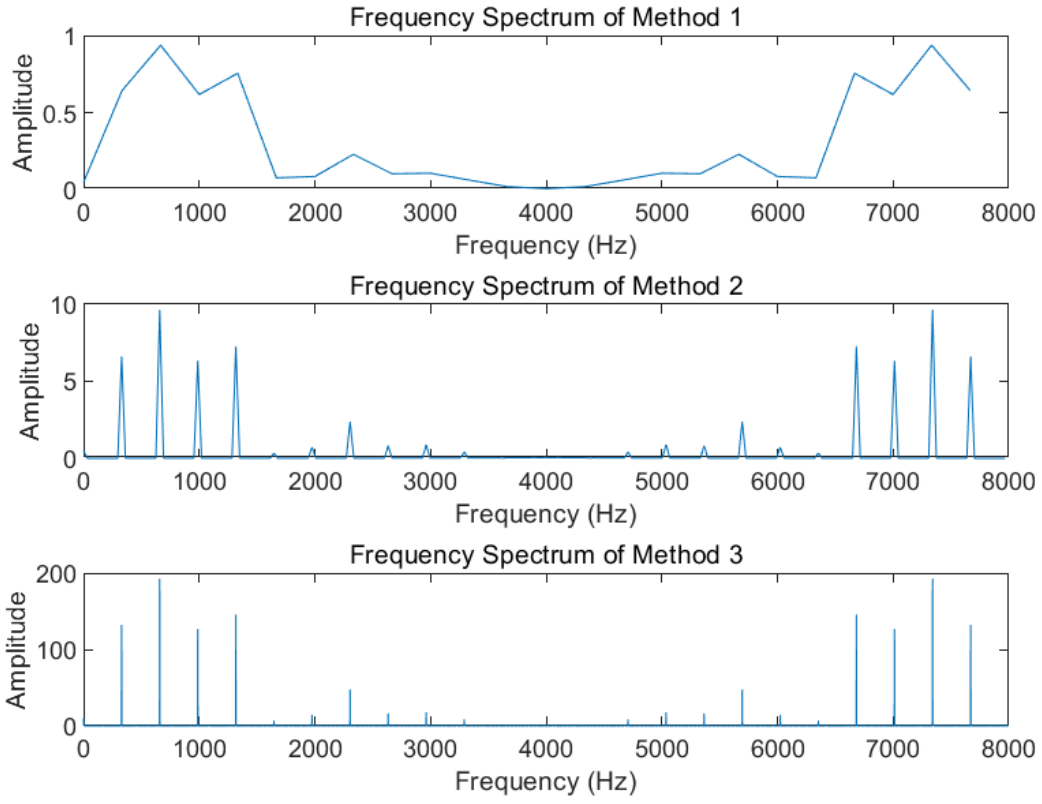
方法二：对完整的波形（十个周期）做傅里叶变换。此时效果稍好，频谱呈现三角脉冲的形状。

```
% method 2
waveform_2 = wave2proc;
len_2 = round(length(waveform_2));
spectrum_2 = abs(fft(waveform_2));
subplot(3, 1, 2);
plot((0:len_2 - 1) * Fs / len_2, spectrum_2);
```

方法三：对原波形使用 `repmat` 函数延拓 20 倍，然后做傅里叶变换。根据图像可知，频谱更加接近冲激函数，这是因为是时域波形周期性很强。

```
% method 3
waveform_3 = repmat(wave2proc, [20, 1]);
len_3 = length(waveform_3);
spectrum_3 = abs(fft(waveform_3));
subplot(3, 1, 3);
plot((0:len_3 - 1) * Fs / len_3, spectrum_3);
```

上述三种方法得到的频谱见下图：



对于上面的第三个频谱，我们通过 `findpeaks` 函数找到频谱中出现的脉冲，也就是波形中包含的频率分量。取出最小的那个即为基波频率，剩下的是高次谐波。由于采样率固定为 8KHz，因此最高频率分量不超过 4KHz。

```
% find base waves and harmonic waves
[peaks, locs] = findpeaks(abs(spectrum_3), 'MinPeakHeight', 0.01 *
max(abs(spectrum_3)));
peaks = [abs(spectrum_3(1)); peaks(1:end / 2)];
peaks = peaks / peaks(2);
locs = [0; locs(1:end / 2)];
locs = locs * Fs / len_3;

f_A = 220;
freq = f_A * 2 .^ (-1:1/12:2 -1/12);

[~, base] = min(abs(freq - locs(2)));
fprintf('Fundamental Frequency: %.3f Hz\n', freq(base));
```

最终得到基波频率约为 330.86 Hz，对应的标准乐音频率为 329.63 Hz，为 E 调。直流分量以及各次谐波分量的相对幅值（相对于基波分量）为：

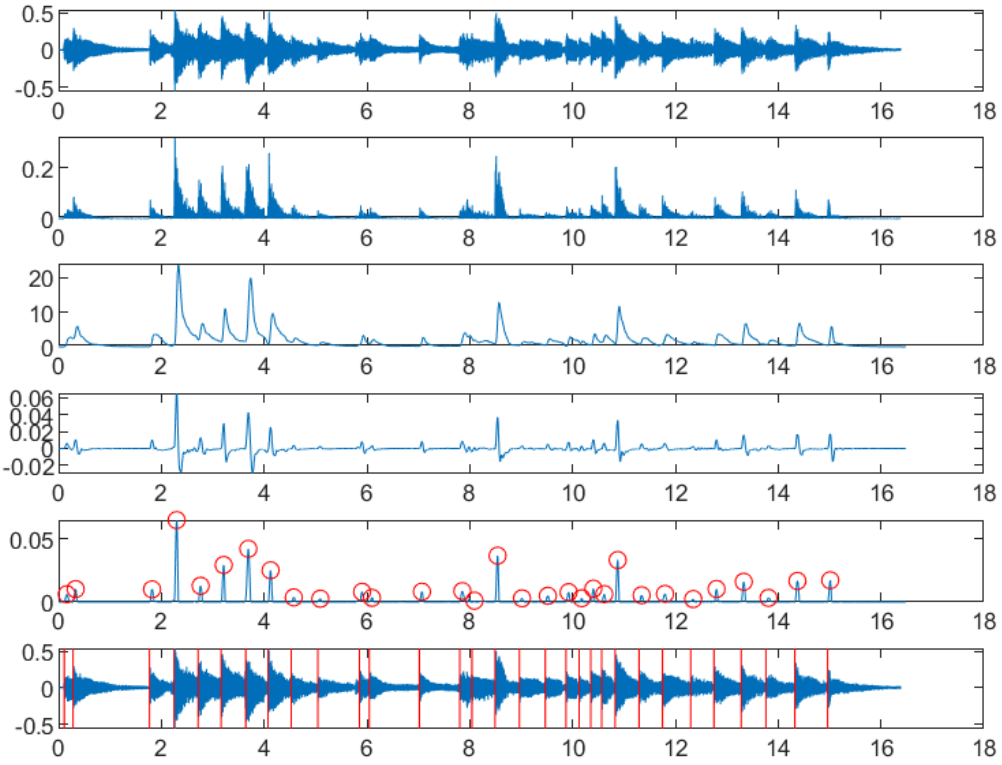
```
Fundamental Frequency: 329.628 Hz
DC component, Amplitude: 0.074
Harmoic 1, Amplitude: 1.000
Harmoic 2, Amplitude: 1.457
Harmoic 3, Amplitude: 0.959
Harmoic 4, Amplitude: 1.100
Harmoic 5, Amplitude: 0.052
Harmoic 6, Amplitude: 0.110
Harmoic 7, Amplitude: 0.359
Harmoic 8, Amplitude: 0.124
Harmoic 9, Amplitude: 0.135
Harmoic 10, Amplitude: 0.064
```

（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。

首先划分节拍：先对信号取平方，得到能量，然后与窗函数 `barthannwin` 做卷积，得到上述波形的包络。包络中冲激发生处是每个乐音的起始位置，由于发生冲激后衰减部分较“平缓”，不利于直接检测峰值，因此对该波形做差分，然乎取正值，此时在图像中清晰可见各处峰值。再使用 `findpeaks` 函数，设定峰值的最小幅度、峰值之间的最小间隔。

```
sq = guitar .^ 2;

con = conv(sq, barthannwin(round(Fs / 10)));

dif = con(2:end) - con(1:end - 1);

dif = dif .* (dif > 0);

[peaks, locs] = findpeaks(dif, ...
    'MinPeakHeight', 0.015 * max(abs(dif)), ...
    'MinPeakDistance', Fs * 0.15);

locs = locs - Fs / 20; % adjust the location of the beat
```

最后各个节拍开始的位置保存在数组 `locs` 中。由于 `findpeaks` 函数是以峰值为寻找标准的，因此在该位置处该音调实际已经开始了一段时间（冲激到最大值），并非刚开始的时间，因此我们对数组 `locs` 进行修正：`locs = locs - Fs / 20;`。最终经过划分的波形如下：



划分好节拍后，对每一个节拍分析音调：采用与（8）同样的策略，先将片段周期性重复足够多次，做傅里叶变换得到频谱。由于该片段中频谱连续性太强，使用 `findpeaks` 函数寻找频率点误差太大，我们需要采取其他方法。为了找到基波频率，我们先在频谱中找到幅度最大的频点，以此最大值的 1/4 为阈值，向下筛选基波频率：当最大幅度频段接近该频率的整数倍时（相对误差在 0.5% 之间），我们将此频率定为基波频率。

```
    wave_gt = repmat(wave_gt, [100, 1]);
    spect_gt = abs(fft(wave_gt));

    [sp_peak, sp_loc] = max(spect_gt);
    candidate = (1:sp_loc);
    candidate = candidate(spect_gt(1:sp_loc) > sp_peak / 4);
```

```
    for j = 1:length(candidate)
        k = sp_loc / candidate(j);

        if k < 5 && 0.995 < k / round(k) && k / round(k) < 1.005
            result = candidate(j);
            break;
        end

    end

    fund_freq = (result - 1) * Fs / length(spect_gt);
    [~, idx] = min(abs(std_freq - fund_freq));
    fund_freq = std_freq(idx);
```

寻找高次谐波分量时，在基波频率的整数倍的约 2% 左右范围内寻找幅度最大处，定位高次谐波的幅度。由此可以找到基波频率的各次谐波信息。如果出现重复的音调，那么就将新得到的谐波幅度向量与原有的做平均。最后我们找到该乐曲中出现的所有音调的谐波信息，存储在一个谐波矩阵中（为了使不同频率有不同的高次谐波数量，也就是矩阵的不同列的行数不同，我们使用元胞数组 `cell` ）。

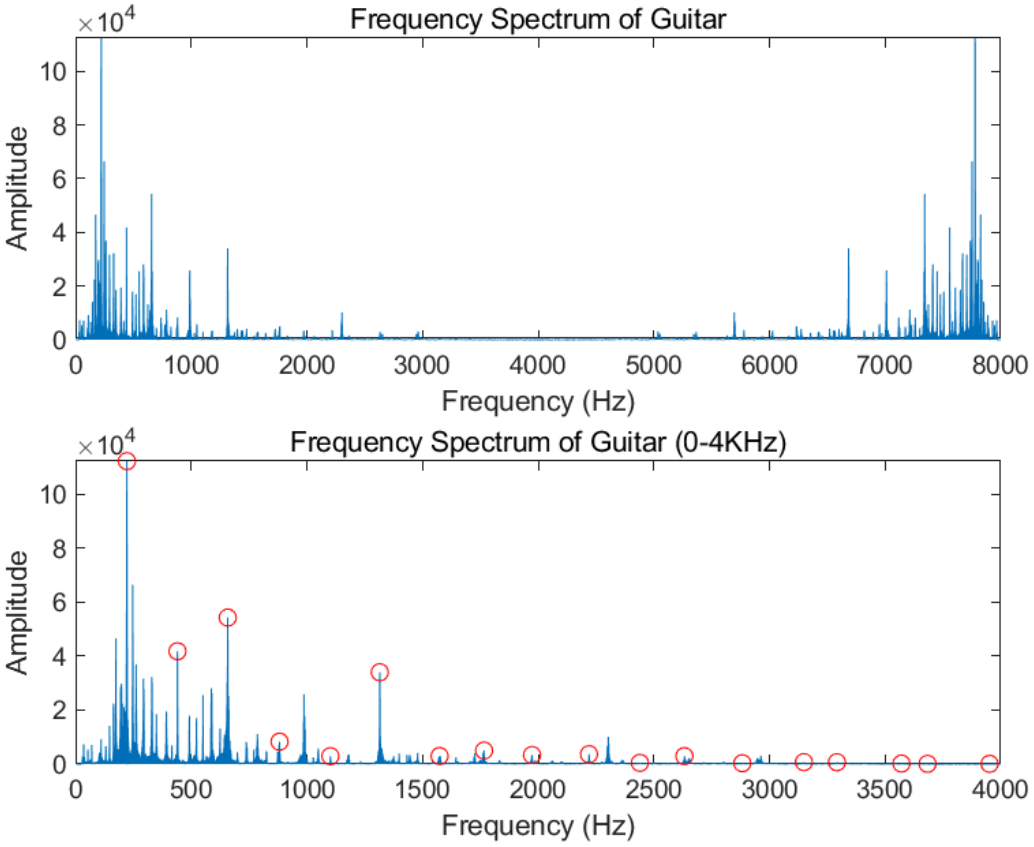
```
    temp_ampls = zeros(1, floor(Fs / 2 / fund_freq));

    for j = 1:floor(Fs / 2 / fund_freq)
        range = round(result * j * (1 - 0.02)):round(result * j * (1 + 0.02));
        temp_ampls(j) = max(spect_gt(range));
    end

    temp_ampls = temp_ampls / temp_ampls(1);

    if isempty(harmonics{idx})
        harmonics{idx} = temp_ampls;
    else
        harmonics{idx} = (harmonics{idx} + temp_ampls) / 2;
    end
```

现在我们不一定得到了全部音调的谐波信息，需要使用邻近的音调的谐波信息进行“插值”，最后我们才能得到完整的谐波矩阵。

```
for i = 1:length(std_freq)

    if isempty(harmonics{i})

        for j = 1:max(i - 1, length(std_freq) - i)

            if (i > j && ~isempty(harmonics{i - j}))
                harmonics(i) = harmonics(i - j);
                break;
            elseif (i + j <= length(std_freq) && ~isempty(harmonics{i + j}))
                harmonics(i) = harmonics(i + j);
                break;
            end

        end
```

```
16
17      end
18
19  end
```

下图是对第一个节拍的音调做频率分析的结果（基波频率约为 220Hz，A调）：



Part3：基于傅里叶级数的合成音乐

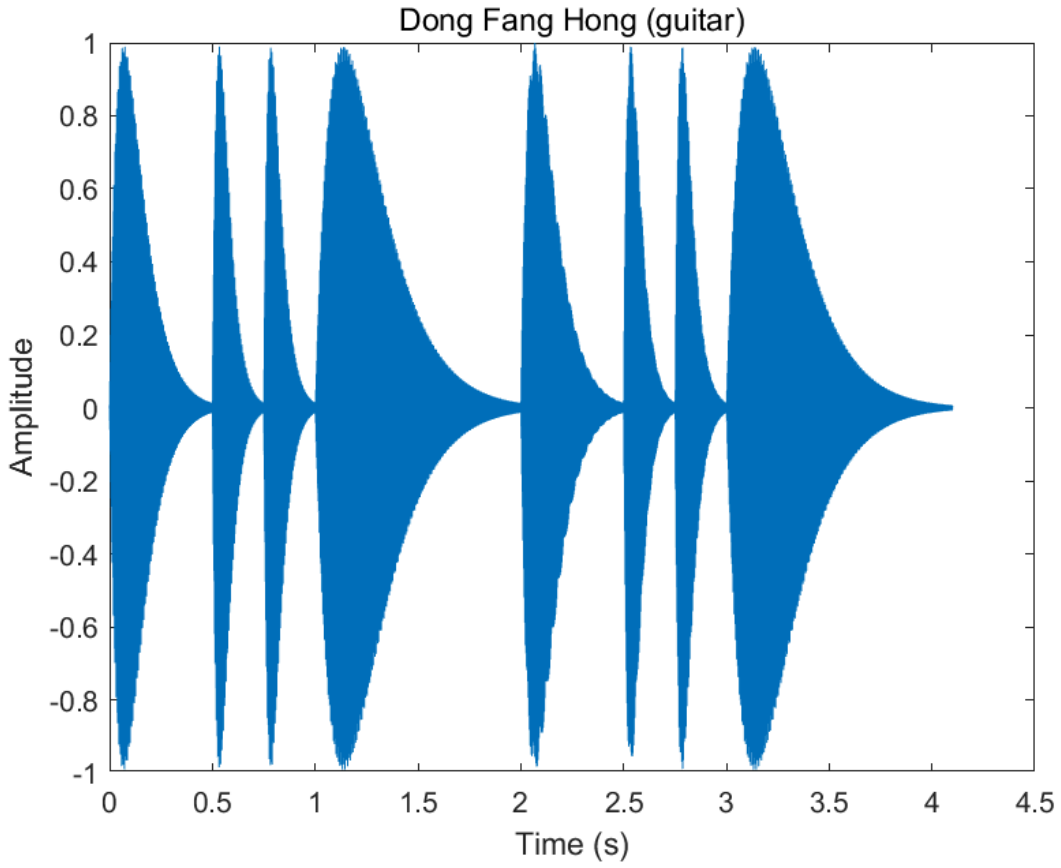
（10）用 (7) 计算出来的傅里叶级数再次完成第 (4) 题，听一听是否像演奏 fmt.wav 的吉他演奏出来的?

使用（7）中傅里叶变换得到的各次谐波的幅度数组，以基波频率幅度为基准归一化之后，加上各次谐波。此时播放出的声音并没有很像吉他音。

```
1  rw_len = length(realwave);
2  re_wave = resample(realwave, 10, 1);
3  wave_proc = zeros(rw_len, 1);
4
5  for i = 1:10
6      wave_proc = wave_proc + re_wave((i - 1) * rw_len + 1:i * rw_len);
7  end
8
9  wave_proc = wave_proc / 10;
10 wave_proc = repmat(wave_proc, [10, 1]);
11 wave_proc = resample(wave_proc, 1, 10);
12
13 wave_proc = repmat(wave_proc, [100, 1]);
14 spect = fft(wave_proc);
15 plot(abs(spect));
```

```
16
17 [peaks, ~] = findpeaks(abs(spect), 'MinPeakHeight', 0.01 * max(abs(spect)));
18
19 harmonics = peaks(1:end / 2);
20 harmonics = harmonics / harmonics(1);
21
22 for i = 1:size(DongFangHong, 1)
23     % ...
24     % add harmonics
25     sub_melody = sin(2 * pi * DongFangHong(i, 1) .* t *
   (1:length(harmonics))) * harmonics;
26     sub_melody = sub_melody .* Adjust_Exp(t / duration);
27     % ...
28 end
```

波形如下图：

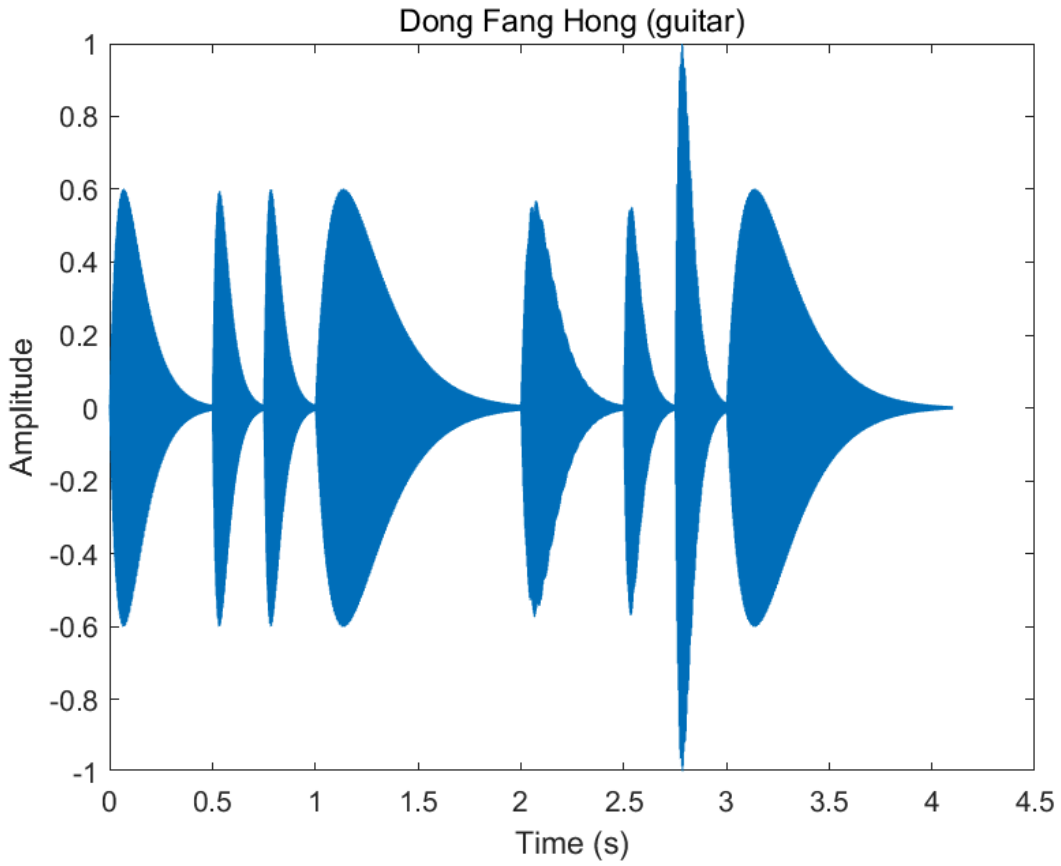


（11）也许(9) 还不是很像，因为对于一把泛音丰富的吉他而言，不可能每个音调对应的泛音数量和幅度都相同。但是通过完成第 (8) 题，你已经提取出 fmt.wav 中的很多音调，或者说，掌握了每个音调对应的傅里叶级数，大致了解了这把吉他的特征。现在就来演奏一曲《东方红》吧。提示：如果还是音调信息不够，那就利用相邻音调的信息近似好了，毕竟可以假设吉他的频响是连续变化的。

遍历节拍对应的音调时，先找到该音调在标准频率向量 `std_freq` 中的索引，然后从谐波矩阵中取出该音调对应的谐波向量。

```
for i = 1:size(DongFangHong, 1)
    % ...
    [~, idx] = min(abs(std_freq - DongFangHong(i, 1)));
    harmo = harmonics{idx};
    sub_melody = sin(2 * pi * DongFangHong(i, 1) * t * (1:length(harmo))) *
harmo';
    sub_melody = sub_melody .* Adjust_Exp(t / duration);
    % ...
end
```

波形如下图:



可以发现第 7 个节拍的谐波分量显著高于其他音调。

感想与收获

文件结构