MATLAB 高级编程与工程应用

音乐合成

学号：2020010768

班级：无 05

姓名：付宇辉

时间：2022 年 7 月 29 日

目录

一、实验名称与目的

名称：音乐合成

目的：可以增进对傅里叶级数的理解；

能够熟练运用 MATLAB 基本指令。

二、实验内容

（一）简单的音乐合成

（1）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。

解答：将《东方红》片段的音调和每个乐音的持续时间存在相应的矩阵中，简单用相应频率的正弦波拼接即可。

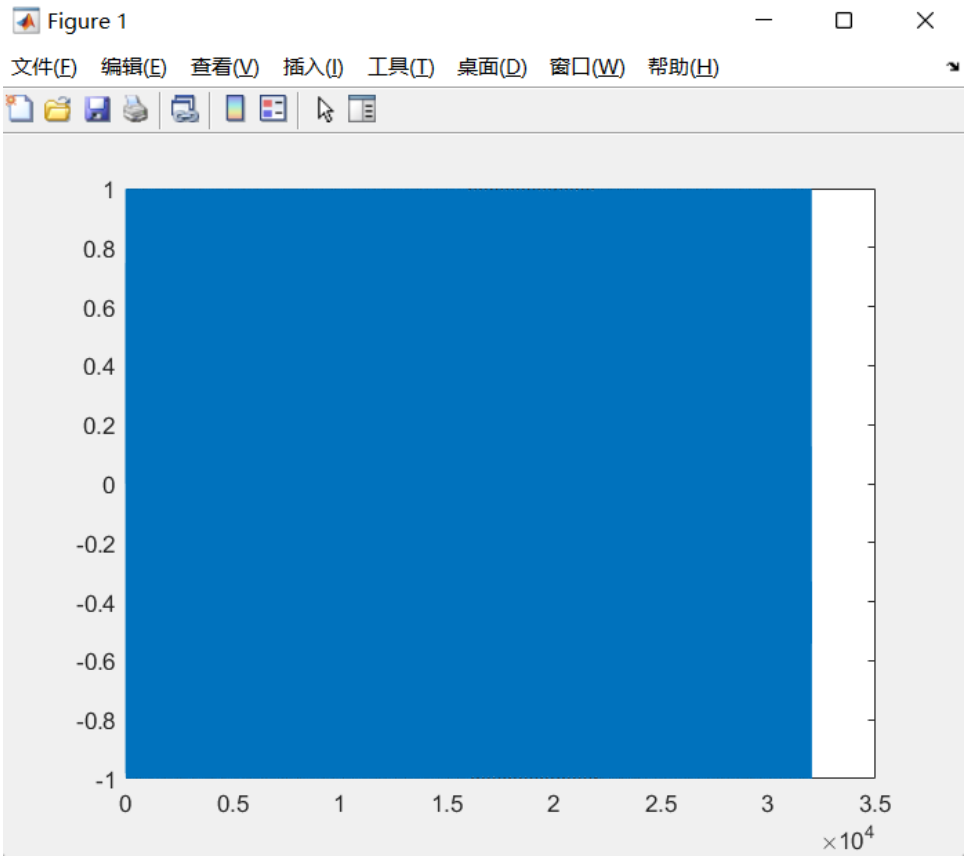
关键代码如下：

```
%合成音乐
time = fs * content(:, 2) * meter;
song = zeros(1, sum(time));
n = 1;
for tune_num = 1:size(content,1)
    t = linspace(0, content(tune_num, 2) * meter - 1/fs,
time(tune_num));
    song(n:n+time(tune_num)-1) = sin(2 * pi * tunes(content(tune_num,1))
* t);
    n = n+time(tune_num);
end

sound(song, fs);
plot(song);
```

由此得到的 song 矩阵即为音乐。可以通过 sound 函数播放，可以听到大致的东方红旋律，十分有成就感。

波形图如下：



（2）用指数衰减的包络来对乐音进行优化。

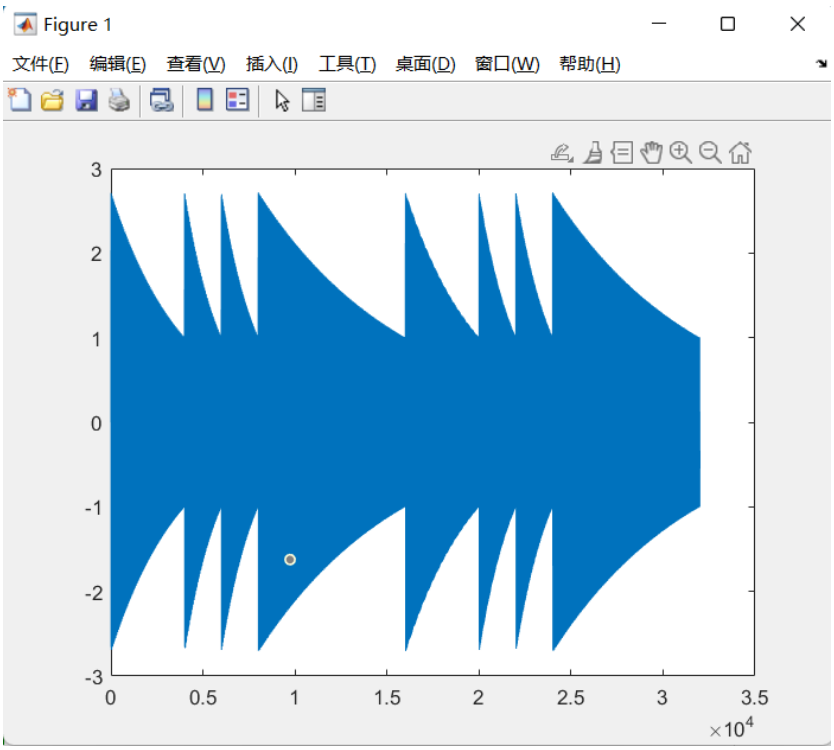
解答：我们可以在正弦波的基础上，直接增加指数包络，修正乐音，使得音乐听起来更加连续。

关键代码如下：

```
%合成音乐
time = fs * content(:, 2) * meter;
song = zeros(1, sum(time));
n = 1;
for tune_num = 1:size(content,1)
    t = linspace(0, content(tune_num, 2) * meter - 1/fs,
time(tune_num));
    env = zeros(1, time(tune_num));
    env(:) = exp(1:(-1/time(tune_num)):1/fs);
    song(n:n+time(tune_num)-1) = sin(2 * pi * tunes(content(tune_num,1))
* t) .* env;
    n = n+time(tune_num);
end
```

结果听起来确实比第一次合成的音乐连贯许多。

波形图如下：



（3）

最简单的方法将 (2) 中的音乐分别升高和降低一个八度。

解答：最简单的方法就是将（2）中的各个乐音的频率乘 2（或除以 2）即为升高（或降低）一个八度。

关键代码如下：

```
%最简单的方法将音乐升高一个八度
song(n:n+time(tune_num)-1) = sin(2*2*pi *tunes(content(tune_num,1)) *
t) .* env;
```

用 resample 函数将上述音乐升高半个音阶。

解答：将音乐升高半个音阶，只需要将频率变为$2^{\frac{1}{12}}$倍，用 resample 函数重新采样即可。

关键代码如下：

```
%利用 resample 函数将音乐升高半个音阶, 2^(1/12)=1.06
song = resample(song, 100, 106);
```

（4）在 (2) 的音乐中增加一些谐波分量

增加谐波分量的方法很简单，只需要在生成单频正弦波的基础上加上一定强度的高频正弦波分量即可。 设基波分量的振幅为 1，取二次谐波为 0.2、三次谐波

为 0.3，关键代码如下：

```
for i = 1:length(harmonic_amplitude)
    harmonic_wave = harmonic_wave + ...
       harmonic_amplitude(i)*sin(2*i*pi*tunes(content(tune_num,1)) * t);
end
```

由于有高次谐波分量，因此波形相对于正弦波有一些变形，声音的波形不是单频正弦波了。音色确实与风琴声类似。

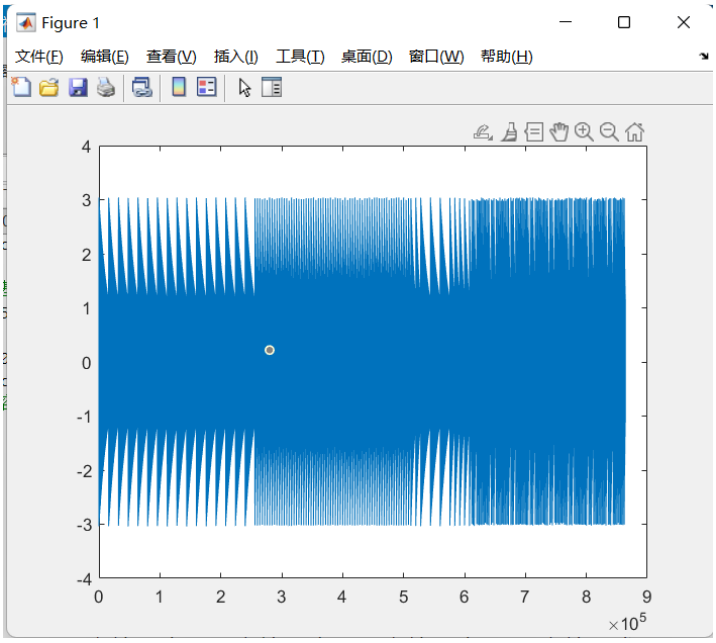
（5）自选音乐合成

我选择了 D 大调卡农中的一小部分音乐进行合成。

总体的步骤与原本的《东方红》大差不差，只是音乐的长度加长，需要大量的时间进行简谱的转化。

最后的听起来的效果是非常不错的。

合成音乐的波形图如下：



（二）用傅里叶级数分析音乐

（6）用 wavread 函数载入光盘中的 fmt.wav 文件， 播放出来听听效果。

关键代码如下：

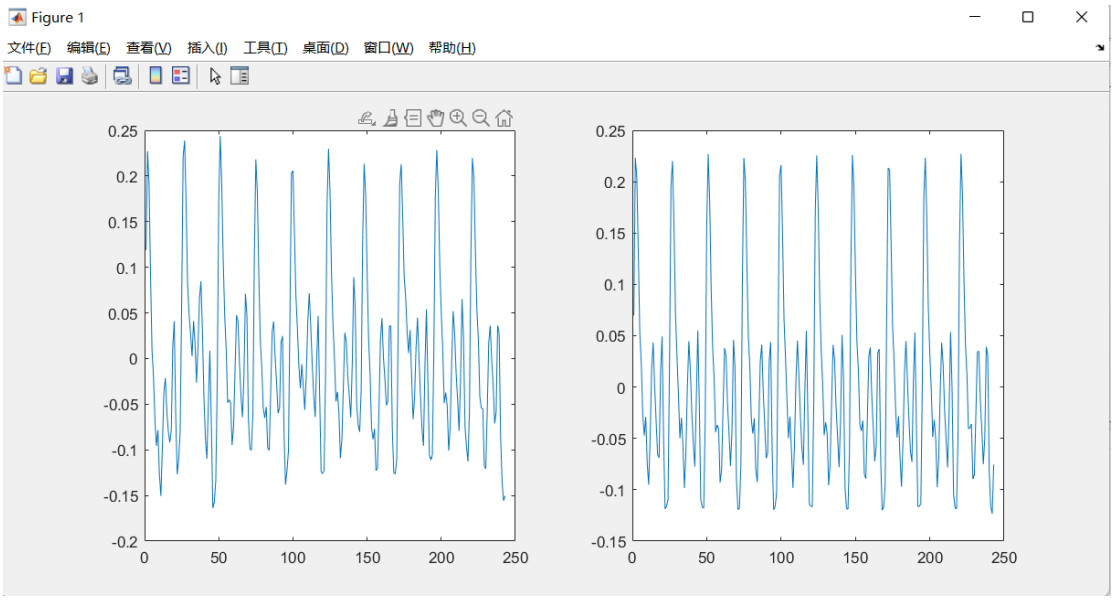
```
wave = audioread("../res/fmt.wav");
```

```
sound(wave);
```

音乐确实十分真实，与吉他的音色类似。

（7）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。

我们将 realwave 和 wave2proc 的波形图画出进行分析：左侧是 realwave，右侧是 wave2proc。

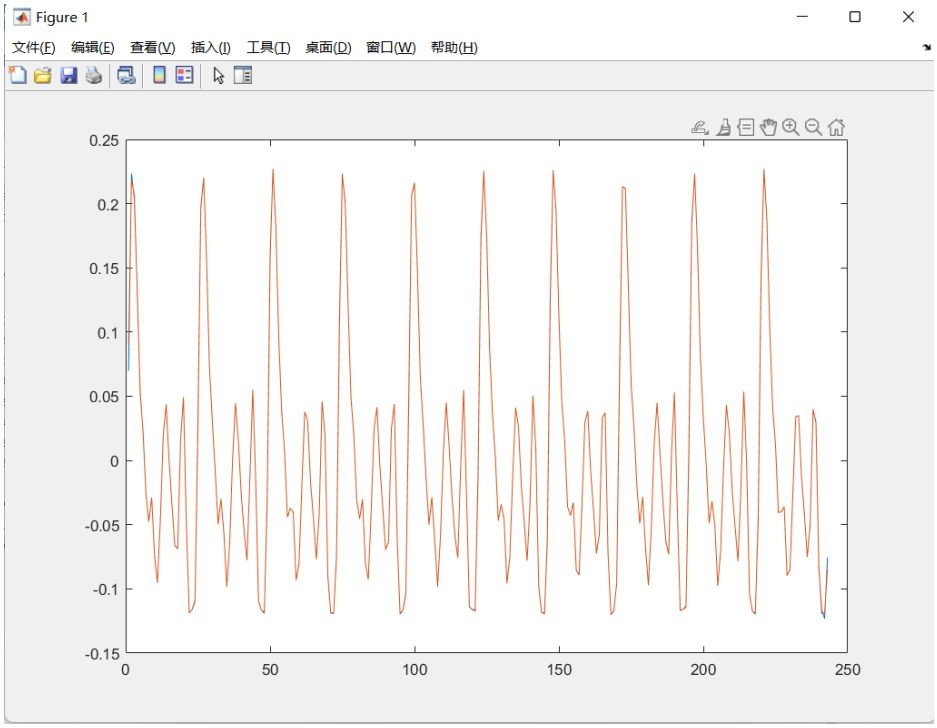


我们可以看出 wave2proc 比 realwave 的周期性好，去除了真实音乐当中的非线性谐波和噪声。realwave 和 wave2proc 的数据都是 243 个点，观察 wave2proc 它的周期大约是 25，所以我们想到可以先重新对 realwave 进行采样成 250 个点，然后可以通过对 10 个周期内进行算术平均，重新扩充为 250 个点，再重新采样为 243 个点，就可以实现较好的周期性，去除非线性谐波和噪声。

关键代码如下：

```
wav = reshape(resample(realwave, 250, 243), [25, 10]);
tmp = sum(wav, 2) / 10;
wav = repmat(tmp, [10, 1]);
wav = resample(wav, 243, 250);
```

处理后的信号与 wave2proc 相比的效果如下：



可以看到，两个信号基本完全重合。

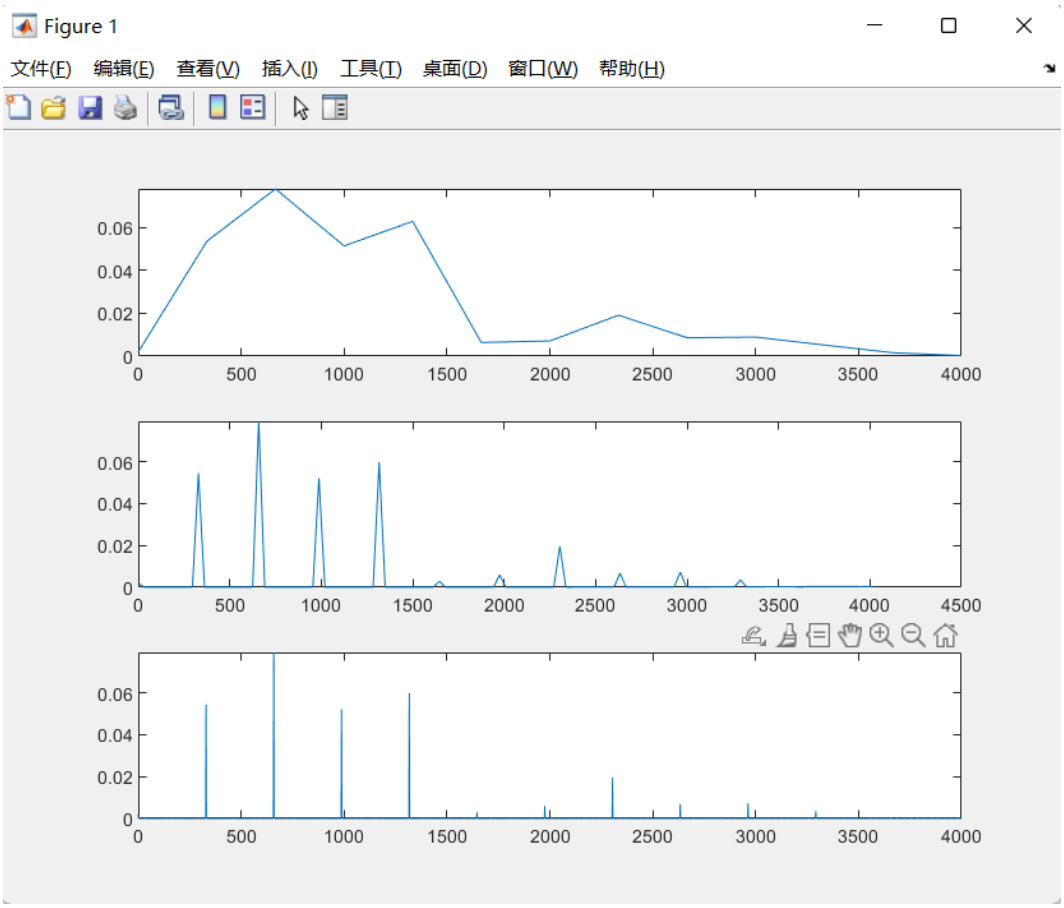
（8）分析 wave2proc 的频谱我们用到了 matlab 当中的 fft 快速傅里叶变换函数。基于 help 文档当中的用法（如下图），我们照搬了样例进行傅里叶分析。

现在，采用原始的、未破坏信号的傅里叶变换并检索精确幅值 0.7 和 1.0。

```
Y = fft(S);
P2 = abs(Y/L);
P1 = P2(1:L/2+1);
P1(2:end-1) = 2*P1(2:end-1);

plot(f,P1)
title('Single-Sided Amplitude Spectrum of S(t)')
xlabel('f (Hz)')
ylabel('|P1(f)|')
```

我们分别从一个周期，从 10 个周期，以及重复了 20 次之后分别进行了傅里叶分析，得到的频谱如下图：



将 wave2proc 信号重复较多次之后，它本身的周期性就会变得非常明显，使得傅里叶变换后的效果比较好。

（9）再次载入 fmt.wav，写一段程序，自动分析出这段乐曲的音调和节拍。

解答：我们首先在 CoolEdit 中手动标定了每个音调的起止时间，将他们存储下来，进行分析。

可以想到，我们可以从小到大遍历每个频点，当我们遇到一个幅度较大的频点极大值时，可以将它定位这个音调的基频。然后通过这个基频，我们就可以在它的倍数上进行进一步地搜索，找到谐波频点的幅度。通过（8）的经验，我们可以看出将每一个音调进行多次重复，可以提高音调的周期性，从而可以得到更好的频谱效果。综上我们代码在 analysis_component.m 当中，关键代码如下：

```
function [freq, amplitude] = analysis_component(wave, fs)
    freq = zeros(1, 7);
    amplitude = zeros(1, 7);
    n = 2^nextpow2(length(wave));
    X = fft(wave, n) / length(wave);
    f = fs/2 * linspace(0, 1, n/2+1);
```

```
    X = 2 * abs(X(1:n/2+1));
    %plot(f, X);
    err = floor(n/fs);
    for i = 2:length(X)-1
        if (X(i) > 0.02) && (X(i) > X(i-1)) && (X(i) > X(i+1))
            freq(1) = f(i); amplitude(1) = 1;
            break;
        end
    end
    ……
end
```

获得的 33 个音调的频率成分和相应的幅度大小如下表：

freq

33x7 double

	1	2	3	4	5	6	7	8	9
4	246.1872	492.3744	736.4197	984.7469	0	1.4771e+...	1.7212e+...		
5	199.5754	0	0	796.1788	993.6314	0	0		
6	222.2214	444.4447	664.3982	886.6215	0	0	0		
7	328.2604	656.5075	984.7832	1.3109e+...	1.6435e+...	1.9739e+...	2.2913e+...		
8	173.3332	345.5553	522.2225	0	0	0	0		
9	133.3618	266.6626	400.0244	533.3252	666.6870	799.9878	933.3496		
10	174.8524	349.7047	0	699.4114	874.2638	1.0491e+...	0		
11	294.8542	588.3150	881.7806	0	0	1.7636e+...	0		
12	165.9393	331.8787	497.8180	0	825.3288	991.2682	0		
13	247.8600	493.7916	740.6874	987.5832	0	0	0		
14	657.0606	1.3127e+...	1.9741e+...	0	0	0	0		
15	213.8290	0	641.4871	0	0	0	0		
16	163.4598	326.8890	490.3870	653.8467	0	980.7663	0		
17	220.7794	441.5588	662.3383	883.1177	1.1017e+...	1.3247e+...	0		
18	219.6159	439.2319	656.7001	0	0	1.3134e+...	0		
19	439.5161	879.0321	1.3165e+...	1.7601e+...	2.2016e+...	2.6351e+...	0		

编辑器 - ex_9.m 变量 - amp

freq amp

33x7 double

	1	2	3	4	5	6	7	8	9
4	1	0.6709	0.0510	0.2390	0	0.0739	0.0523		
5	1	0	0	0.1270	0.1786	0	0		
6	1	0.1789	0.0523	0.0611	0	0	0		
7	1	0.0860	0.2315	0.1739	0.0655	0.1614	0.0695		
8	1	0.0925	0.1806	0	0	0	0		
9	1	0.4319	0.3153	0.2156	0.2231	0.0668	0.0607		
10	1	0.3222	0	0.0537	0.0851	0.0880	0		
11	1	0.0788	0.1520	0	0	0.1422	0		
12	1	0.6368	0.0829	0	0.2587	0.1071	0		
13	1	0.1295	0.0644	0.3290	0	0	0		
14	1	0.1566	0.0724	0	0	0	0		
15	1	0	0.1884	0	0	0	0		
16	1	0.0609	0.1652	0.4270	0	0.1133	0		
17	1	0.4076	0.1346	0.0841	0.0531	0.1383	0		
18	1	0.0640	0.0701	0	0	0.5909	0		
19	1	0.4857	0.3489	0.0614	0.0854	0.1054	0		

（三）基于傅里叶级数的合成音乐

（10）用(8)计算出来的傅里叶级数再次完成第(4)题

解答：只需要将第（4）题当中的 harmonic_amplitude 矩阵换为（8）傅里叶分析得到的结果即可。

```
harmonic_amplitude = [0.05442, 0.07929, 0.5217, 0.05985, 0.0028,
0.0060, 0.0195, 0.0067, 0.0073]; %谐波强度
```

我们可以听到音乐的效果确实比较像吉他的音色了，但仍然不是太好。

（11）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。

解答：我们可以通过（9）当中分析的结果，将每一个音色的谐波分量具体的找到，并且存储下来，可以实现更好的效果，我们对东方红当中的 8 个音调找到了近似的频率成分，然后对应的谐波频率也找到了，再次进行合成。效果更好了一些。

```
harmonic_amplitude = [1, 0.3071, 0.07074, 0.05729, 0.07538,  0, 0;
                      1, 0.3071, 0.07074, 0.05729, 0.07538,0, 0;
                      1, 0.0788, 0.15203, 0, 0, 0.1422,0;
                      1, 0, 0, 0.1270, 0.1786, 0, 0;
                      1, 0.3222, 0, 0.05368, 0.08507, 0.08801, 0;
                      1, 0.3222, 0, 0.05368, 0.08507, 0.08801, 0;
                      1, 0.0569, 0.05953, 0.05032, 0.1000, 0.06168, 0.05279;
                      1, 0, 0, 0.1270, 0.1786, 0, 0]; %谐波强度
```

（12）做一个图形界面把上述功能封装起来。

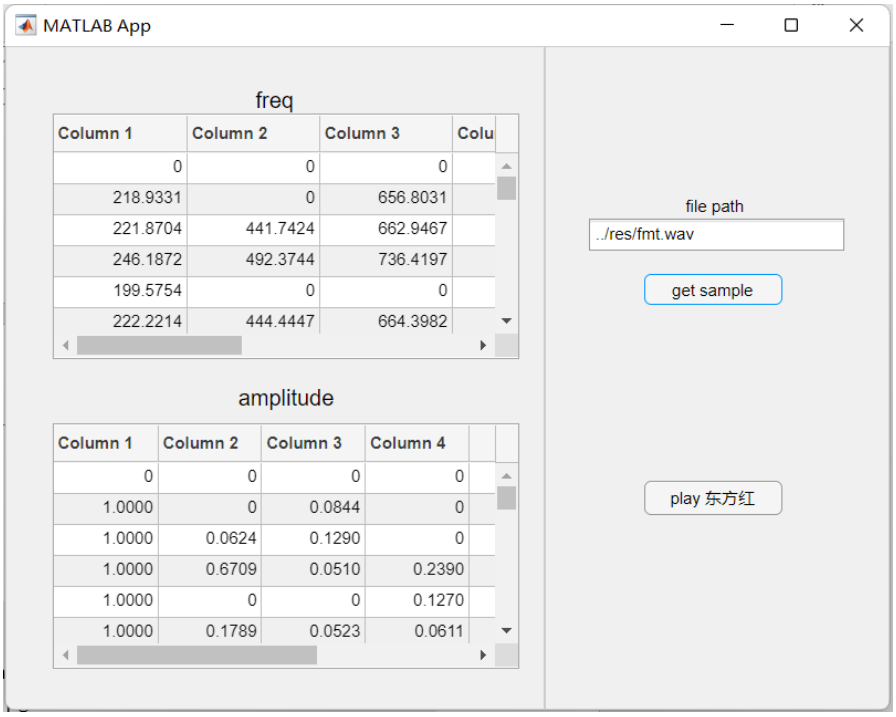
解答：我简单地将这些功能封装在了一个 app 里面，代码见 gui.mlapp。

界面十分简单，左侧显示将音频文件进行傅里叶分析的结果，上方是频率分析，下方是对应的谐波幅度。

我们只需要将文件的路径写在 file path 当中，然后点击 get sample，即可在左侧看到分析的结果。

右下方是播放我们已经合成好的东方红音乐。

总体的画面十分简单直接，易于交互。



三、文件清单

本次报告的目录结构如下：

```
├─── doc
|    └─── report.docx
├─── res
|    ├─── Guitar.MAT
|    └─── fmt.wav
├─── src
|    ├─── analysis_component.m
|    ├─── ex_1.m
|    ├─── ex_10.m
|    ├─── ex_11.m
|    ├─── ex_2.m
|    ├─── ex_3.m
|    ├─── ex_4.m
|    ├─── ex_5.m
|    ├─── ex_6.m
|    ├─── ex_7.m
|    ├─── ex_8.m
|    ├─── ex_9.m
|    └─── gui.mlapp
└─── tree.txt
```

四、心得体会

这次实验让我体会到了信号与系统的有趣之处以及有用的地方。我是姚铮老师班上的学生，所以 matlab 课堂是我第一次接触谷老师，谷老师十分有趣，并且将内容活灵活现地展现在我面前。

这次音乐合成作业，让我巩固了信号与系统的知识以及让我对音乐有了初步的了解，总体来说还是非常有收获的。

但是实验过程中也有很多不足的地方，例如图形界面做的不完善；有些地方能用矩阵运算解决的地方并没有使用，而是用了较为缓慢的循环运算，等等。希望之后利用 matlab 可以避免这些问题。

在作业当中收获到很多，谢谢老师和助教的指导和帮助！

五、参考内容

- MATLAB 官方 help 文档
- 《信号与系统——MATLAB 综合实验》谷源涛 应启珩 郑君里