Matlab 综合实验一 音乐合成实验报告

学号：2017011090
班级：无 78
姓名：游子權

说明：在本作业的所有部分中，fs 为取样率（sample rate）；pitch 被定义为距离中央 C 的「**半音**」数，当 pitch 为虚数单位 i 时，表示休止符；tc 被定义为该音的持续时间。

一、 简单的合成音乐

1. 关键代码及方法如下：

- q1_makesound.m

```
1   function y = q1_makesound(fs, pitch, tc)
2 -   t = 0:(1/fs):tc;
3 -   freq = 440*(2^(-0.75))*2^(pitch/12);
4 -   y = sin(2*pi*freq*t);
5 -   end
```

说明：第 3 行根据 pitch 计算相应的频率，第 4 行根据频率算出相应正弦信号。

- q1_song.m

```
1 -   fs = 8000; %setting sample rate
2 -   clap = [1 0.5 0.5 2 1 0.5 0.5 2]; %setting the beat of the music
3 -   pitch = [7 7 9 2 0 0 -3 2]; %setting the pitch of the music
4 -   tc = 0.5*clap; %approx. 0.5 second for 1 beat
5 -   len = length(pitch);
6 -   music = zeros(1,40000); %predefine the length of the output
7                              %to avoid lengthy execution time
8 -   last = 1;
9 -   for k=1:len
10 -      temp = q1_makesound(fs, pitch(k),tc(k));
11 -      last_next = last+length(temp)-1;
12 -      music(last:last_next) = temp;
13 -      last = last_next+1;
14 -  end
15 -  sound(music,fs);
16 -  audiowrite('q1_eastRed.wav', music,fs);
```

说明：第 1~4 行录入歌曲的节拍、音调、持续时间。第 8~14 行把各个音用 q1_makesound 产生信号后依序填入 music 阵列内。第 15 行播放合成后的音乐。第 16 行写入到 wav 文件中。

- **结果（q1_eastRed.wav）**：很不自然。相邻乐音若为同音，则听不出明显间隔，只有因相位不连续产生的「啪啪」高频分量。

2. **关键代码及方法如下：**

- q2_makesound.m

```
1   function y = q2_makesound(fs, pitch, tc)
2   t = 0:(1/fs):tc;
3   freq = 440*(2^(-0.75))*2^(pitch/12);
4   sound = sin(2*pi*freq*t);
5   w1 = linspace(0, 1, floor(0.01*tc*fs));
6   w2 = linspace(1, 1, floor(0.2*tc*fs));
7   w3 = linspace(1, 0.8, floor(0.1*tc*fs));
8   w4 = linspace(0.8, 0, floor(0.5*tc*fs));
9   st = length(t);
10  s1 = length(w1);
11  s2 = length(w2);
12  s3 = length(w3);
13  s4 = length(w4);
14  sc = st-s1-s2-s3-s4;
15  wc = linspace(0.8, 0.8, sc);
16  env = [w1 w2 w3 wc w4];
17  y = sound .* env;
18  end
```

说明：第 5~8、15 行利用 linspace 加入线性包络，为使不同部分线段拼接起来时不因舍入误差而造成包络矩阵大小与 sound 矩阵大小不一致，第 14 行计算「持续」阶段的包络 wc 的大小时，采用总大小减去其他包络大小而得，确保拼接起来的包络大小与 sound 一致。

- q2_song.m

```
1   fs = 8000; %setting sample rate
2   clap = [1 0.5 0.5 2 1 0.5 0.5 2]; %setting the beat of the music
3   pitch = [7 7 9 2 0 0 -3 2]; %setting the pitch of the music
4   tc = 0.5*clap; %approx. 0.5 second for 1 beat
5   len = length(pitch);
6   music = zeros(1,40000); %predefine the length of the output
7                           %to avoid lengthy execution time
8   last = 1;
9   for k=1:len
10      temp = q2_makesound(fs, pitch(k),tc(k));
11      last_next = last+length(temp)-1;
12      music(last:last_next) = music(last:last_next) + temp;
13      last = last_next- tc(k)*0.05*fs;
14  end
15
16  music = music./abs(max(music));
17  music(music==1)=0.999;
18  music(music==-1)=-0.999;
19  sound(music,fs);
20  audiowrite('q2_eastRed.wav', music,fs);
```

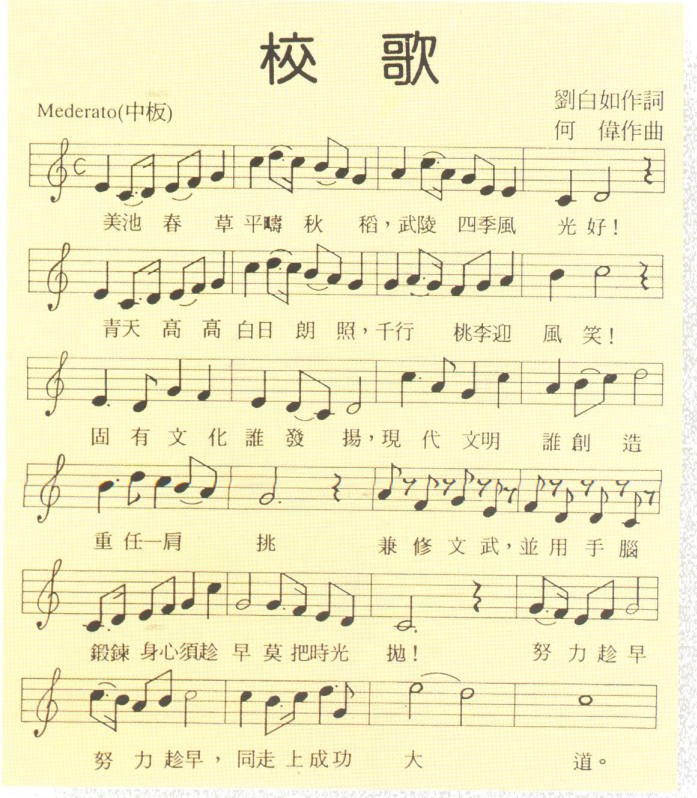
说明：第 13 行用 last 变量实现乐音间的疊接，其重疊的长度暂定为乐音持续时间的 5%。由于输入 wav 格式的音乐，其幅值仅能在-1 到 1 之间，故需将其正规化，如第 16~18 行所示，否则会出现”Data Clipped while writing into file”错误。

- **结果（q2_eastRed.wav）：**听起来比较舒服了，但是一听还是觉得是合成的音乐，因为音调太「单纯」了。

3. **方法**：只要更换采样率为原本的 P 倍，音调便能提高$12\log_2 P$个半音，**但同时速度亦变为原本的 P 倍。**
   - 欲听将其音调提高 8 度（12 个半音）后的结果，只需将采样率设为原本的 2 倍即可。可运行：`sound(resample(music,1,2),fs)`，或 `sound(music,2*fs)`其中 `music` 为上一题的音乐输出。
   - 欲将其音调升高 1 个半音，需将原采样率变为原本的$2^{\frac{1}{12}} \approx 1.05946 \approx \frac{5297}{5000}$倍，可运行：可运行：`sound(resample(music,5000,5297),fs)`或 `sound(music,1.05946*fs)`。其中 `music` 为上一题的音乐输出。
4. **关键代码及方法如下：**
   - q4_makesound.m：只要将 q2_makesound.m 的第 4 行加入高次谐波分量即可，如下所示：

```
4 -  sound = sin(2*pi*freq*t)+0.15*sin(4*pi*freq*t)+0.4*sin(6*pi*freq*t);
```

   - **结果**（q4_eastRed.wav）：音调厚重了许多。
5. 本人选用我的母校——武陵高中校歌做合成音乐
   （http://www.wlsh.tyc.edu.tw/files/11-1002-110.php?Lang=zh-tw）



关键代码如下：

- q5_wuling.m：只要将上述谱中的每一个音符的音高和时值记录于数组 pitch 和 clap 中，再反覆调用 q4_makesound 函数并将各音疊接，即可合成好听的武陵高中校歌了。
- **结果**（q5_wuling.wav）：**好听**。

意犹未尽，我再次用同样的方法合成清华大学校歌。

- **结果**（q5_tsinghua.wav）：**好听**。

二、 用 Fourier 级数分析音乐

6. **废话**。效果当然很好，当然真实啊。因为有多次的谐波分量，而且还有和弦，不是单音。

   关键代码：q6_play.m

7. **方法**：乍看之下，realwave 包含 10 个周期。但因为每个周期都受到杂讯影响，故并未呈现完美之周期性。但理论而言，杂讯长时间的平均值为 0。所以我们可以把 10 个周期的信号叠加起来，除以 10，再延拓为 10 个周期，应会比较接近真实信号。又原信号 realwave 有 243 个数据点，不是 10 的倍数，故可用 resample 重采样为 250 个数据点，再把每 25 个数据点当作一个周期处理之。处理后再重采样为 243 个数据点。

   关键代码：q7_process.m

```
 5 -    realwave2 = resample(realwave, 250, 243);
 6 -    wave2proc2 = zeros(25,1);
 7 -    for k = 1:10
 8 -        wave2proc2 = wave2proc2 + realwave2(25*(k-1)+1: 25*k);
 9 -    end
10 -    wave2proc2 = wave2proc2 / 10;
11 -    wave2proc2 = repmat(wave2proc2,10,1);
12 -    wave2proc2 = resample(wave2proc2, 243,250);
13 -    diff = wave2proc2 - wave2proc;
14 -    fs = 8000;
15 -    size = length(wave2proc);
16 -    axis_t = linspace(0, (size-1)/fs, size);
17 -    figure;
18 -    hold on;
19 -    xlabel("time(s)");
20 -    ylabel("Amplitude");
21 -    plot(axis_t, wave2proc);
22 -    plot(axis_t, wave2proc2);
23 -    plot(axis_t, diff);
24 -    legend('wave2proc', 'wave2proc2', 'diff');
25 -    hold off;
```

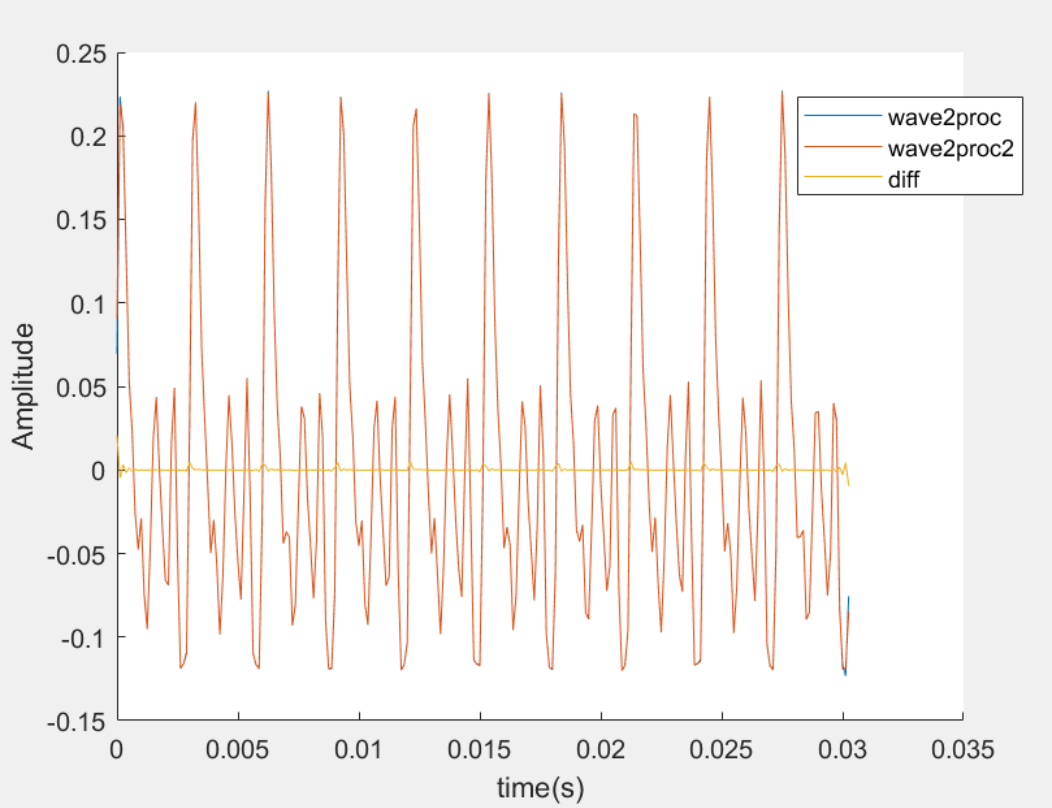
- 第 5 行：完成重采样
- 第 7~9 行：执行 10 个周期平均
- 第 11 行：完成平均后的单周期数据延拓。
- 第 12 行：重采样回原本的点数。
- 第 13 行：将我处理的结果（wave2proc2）和预处理的结果（wave2proc）相减以比较差异。
- 第 17~25 行：将三个信号画在同一张图中比较。

客观结果：

其中 wave2proc 为预先处理的结果

wave2proc2 为本人处理的结果

diff = wave2proc2-wave2proc 为两者之差



▲图 7：wave2proc 的时域波形

主观解释：可以见到两者几乎吻合，惟在每个周期的一开始（峰值部分）有差异较大之处。推测是因为我使用的 resample 的函数先插值再舍入的方式造成了与原信号的一些差异。

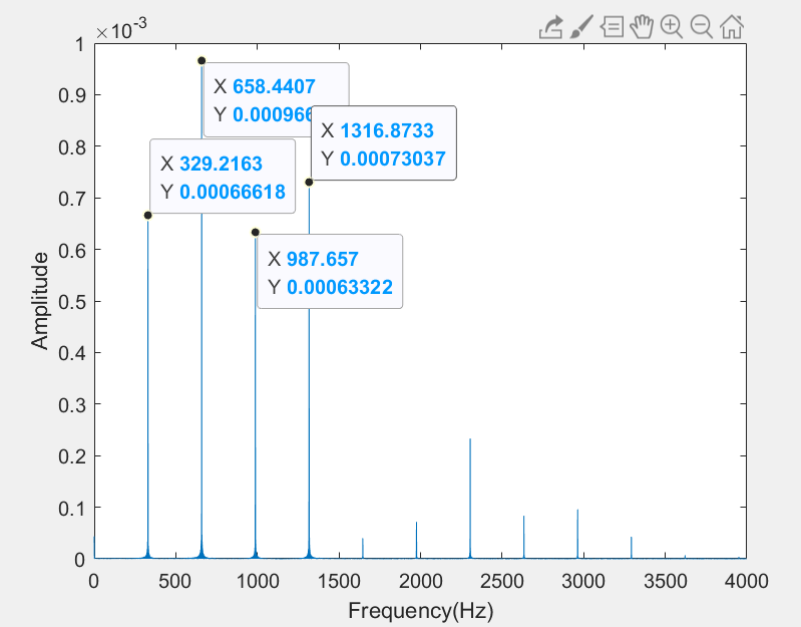
8. **客观结果：**这段乐音的基频是 329.2163Hz。其音调与中央 A（440Hz）相比，约相差$12\log_2\frac{329.2163Hz}{440Hz}\approx -5$个半音，即约为中央 E。其谐波分量及幅值如表 8 所示：

谐波	1(基波)	2	3	4	5	6	7
频率(Hz)	329.22	658.44	987.65	1316.9			2304.5
相对幅值	1	1.4501	0.9505	1.0964	0	0	0.3436

▲表 8：wave2proc 的各频率成份

其频谱如图 8 所示。

方法：利用 fft 函数对原信号作快速傅立叶变换，求出频谱，再利用 findpeaks 函数，输入适当之参数，找出频谱中的极大值，即为潜在的基频与谐波分量。在本例中，用 findpeaks 找到的分别就是基波与 2,3,4,7 次谐波的幅值。



▲图 8：wave2proc 的频谱

关键代码如下：q8_process.m

```
1   %main program for question 8
2   %a continuous program based on q7_process.m
3   wave2proc3 = repmat(wave2proc2, 100,1);
4   axis_f = linspace(0, fs,1000000);
5   F1= (1/length(axis_f))*fft(wave2proc3, length(axis_f)); %use fft
6   figure(2)
7   plot(axis_f(1:length(axis_f)/2), abs(F1(1:length(axis_f)/2)));
8   xlabel("Frequency(Hz)");
9   ylabel("Amplitude");
10  [pks, loc] = findpeaks(abs(F1(1:length(axis_f)/2)), length(axis_f)/fs,
    "MinPeakProminence",0.0002, "MinPeakDistance", 10);
11  pks2 = pks./pks(1);
```

- 第 5 行：对原信号做快速傅立叶变换得到频谱，频谱范围为 0~fs（其中 fs 为抽样频率）
- 第 7 行：实信号的 fft 频谱对称于中央，故只要画出左半边即可。
- 第 10 行：用 findpeaks 找出频谱上的极值。
- 第 11 行：将找到的极值以基频为标准作归一化。

9. **关键代码与方法：**
   - q9/analyze.m：用 envelope 函数分析样本音乐 fmt.wav 的包络。再用 findpeaks 函数找到 fmt.wav 包络的极值，约略是每个音开始的位置。
   - q9/findFreq.m：先对 analyze.m 所分析出各段的乐音做 fft 得到其频谱，再用 findpeaks 函数设定适当经验参数得到频谱的极值，从小到大遍历每个极值，维护一个基频数组，将目前经历的极值点除以目前基频数组中的各值，设为数组 t(n)，若存在 n 使得 abs(t(n)-round(t(n)))<0.01，则视其不为基频，将其加入对应基频的类；若不存在上述的 n，从该极值点为新的基频，建立新的一类。最后在各类中，参酌人耳的频率响应，选择能量最大的一类，输出其基频与各次谐波的相对幅值（以幅值最大者的幅度为 1）。

- 结果：运行完 q9/analyze.m 后，会出现 FREQ_INFO 矩阵，该矩阵每一（row）的第一个元素为基频，随后依次为基波、二次谐波、三次谐波等的分量。有些音在上述算法下分析得不错，但有些音只分析得到基波、有些音的基频仅几 10Hz，应是 findFreq.m 选错了「类」，有待日后精进改善。

三、 基于 Fourier 级数合成音乐

10. **关键代码与方法：**

- q10_makesound.m

```
function y = q10_makesound(fs, pitch, tc, amp)
t = linspace(0, tc, tc*fs);
freq = 440*(2^(-0.75))*2^(pitch/12);
K  = 2*pi*freq : 2*pi*freq : 2*7*pi*freq;
sound = amp*sin(kron(K', t));
```

将第 4 问的 q4_makesound 中，各次谐波的幅值不写死在函数中，改用一个 1*7 的 amp 矩阵传入基波至第 7 谐波的幅值。

- q10_song.m：将第 8 题计算出来的幅值引入，并调用 q10_makesound。

结果（q10_eastRed.wav）：嗯……应该有像吧。

- 意犹未尽，我再次用同样的方法合成武陵高中和清华大学校歌。

结果（q10_wuling.wav、q10_tsinghua.wav）：果然，歌曲一长就听出端倪了，低音的时候确实蛮像的，但当到了高音，其更高频的谐波分量听起来显得很不自然，依照我们的生活经验与对乐器的主观想像，高音的时候不致再有很高频的谐波了。

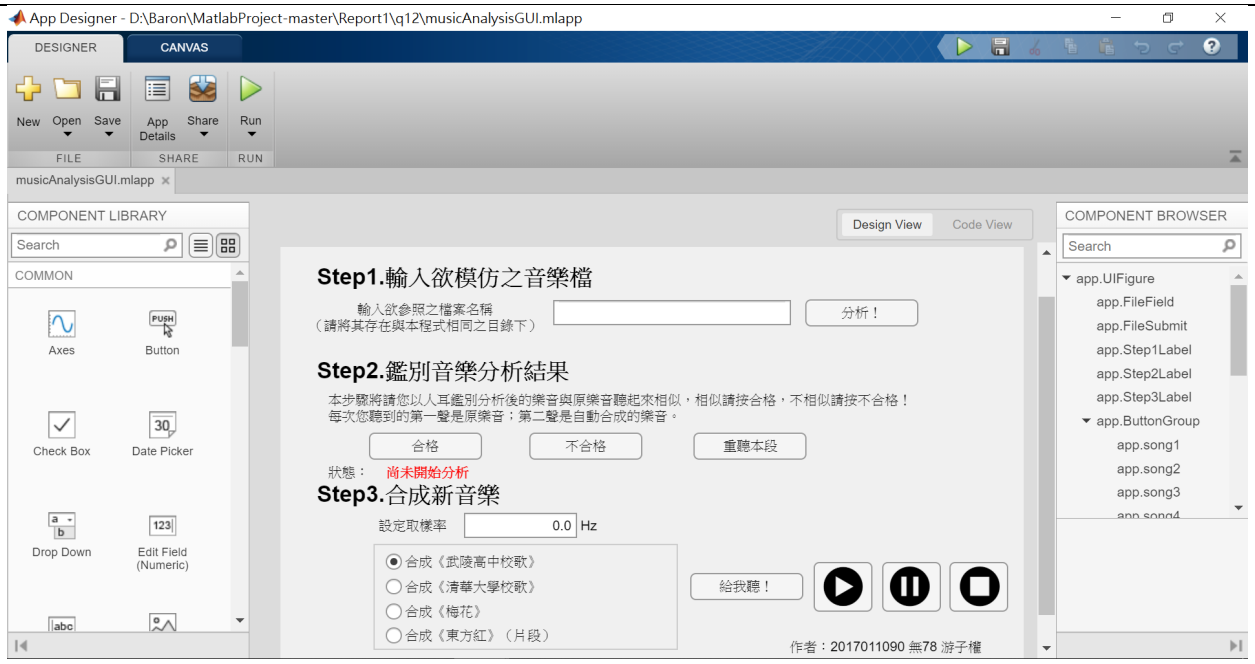
11. **关键代码与方法：**

- q9/ makeWulingSong.m：输入由第 9.题产生的 FREQ_INFO 矩阵，对于每个音，会自动挑选 FREQ_INFO 中最接近的频率的幅值矩阵，调用 makesound_harmonic.m 合成乐音，输出《武陵高中校歌》。
- q9/makesound_harmonic.m： 将第 4 问的 q4_makesound 中，各次谐波的幅值不写死在函数中，改用可变大小的 amp 矩阵传入基波至第 n 谐波的幅值，并据以合成音乐。
- **结果（output_wuling.wav）**：还行吧，我不希望它再更像了。

12. **关键代码与方法：**

- q12/ musicAnalysisGUI.mlapp：本 GUI 是用 Matlab R2019a 的 App Designer 所设计，与 GUIDE 有所不同。请以 Matlab R2019a 方可开启。

- **结果：如下页图所示。**



▲图 12-1：Matlab App Designer 介面

▲图 12-2：设计 GUI 可成功运行