**Matlab综合实验一 音乐合成实验报告**

学号：2017011090

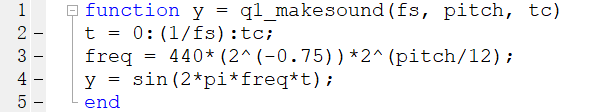
班级：无78

姓名：游子權

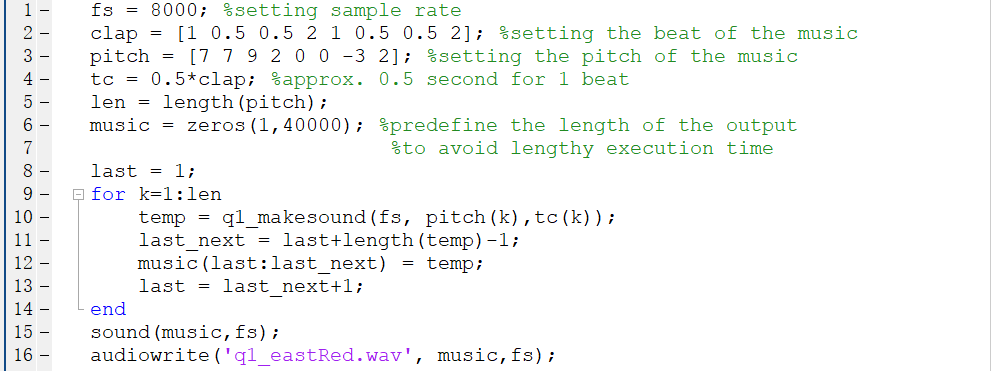
说明：在本作业的所有部分中，fs为取样率（sample rate）；pitch被定义为距离中央C的「**半音**」数，当pitch为虚数单位i时，表示休止符；tc被定义为该音的持续时间。

1. **简单的合成音乐**
   1. 关键代码及方法如下：

* q1\_makesound.m

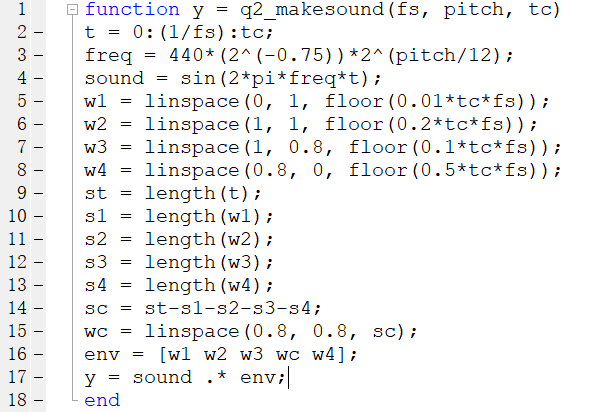


说明：第3行根据pitch计算相应的频率，第4行根据频率算出相应正弦信号。

* q1\_song.m

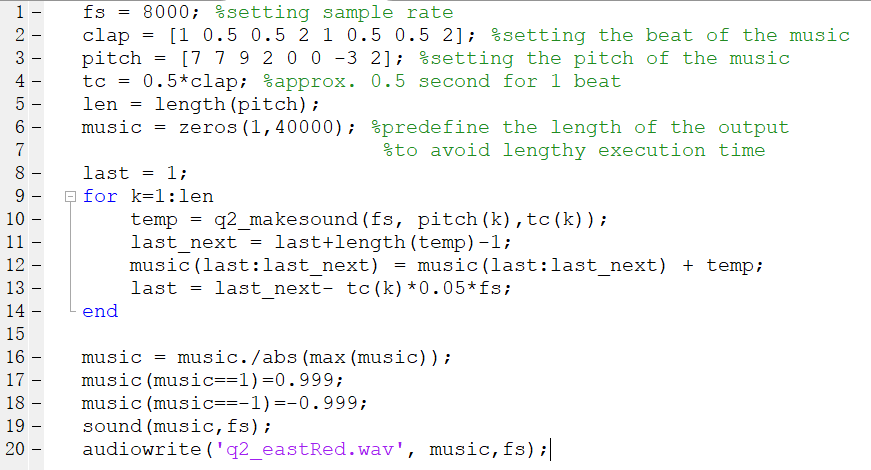
说明：第1~4行录入歌曲的节拍、音调、持续时间。第8~14行把各个音用q1\_makesound产生信号后依序填入music阵列内。第15行播放合成后的音乐。第16行写入到wav文件中。

* 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。
  1. 关键代码及方法如下：
* q2\_makesound.m



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



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。
  1. 方法：只要更换采样率为原本的P倍，音调便能提高个半音，**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。
  1. 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：



* 结果（q4\_eastRed.wav）：音调厚重了许多。
  1. 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。

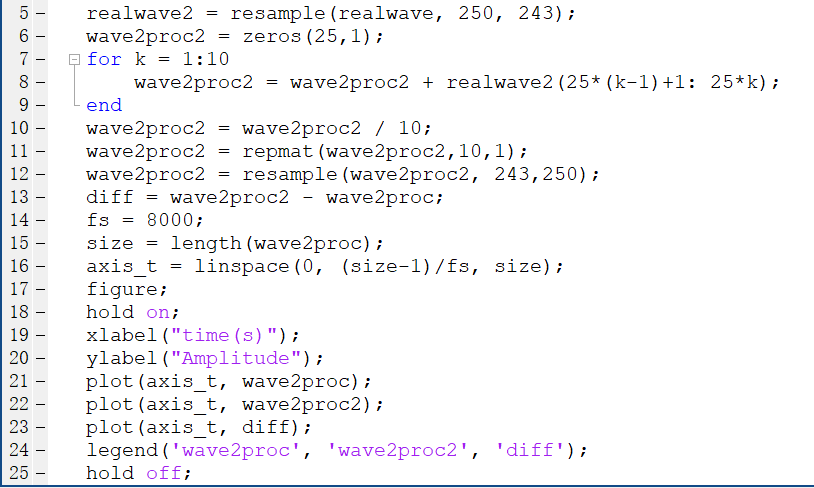
* 结果（q5\_tsinghua.wav）：**好听**。

1. **用Fourier级数分析音乐**
2. **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

1. **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个周期的信号疊加起来，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



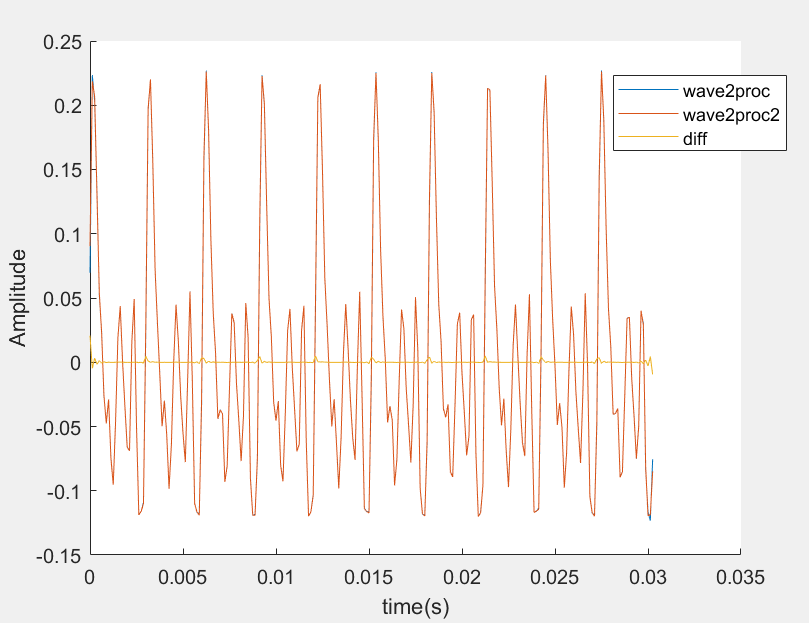
* 第5行：完成重采样
* 第7~9行：执行10个周期平均
* 第11行：完成平均后的单周期数据延拓。
* 第12行：重采样回原本的点数。
* 第13行：将我处理的结果（wave2proc2）和预处理的结果（wave2proc）相减以比较差异。
* 第17~25行：将三个信号画在同一张图中比较。

**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



**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。

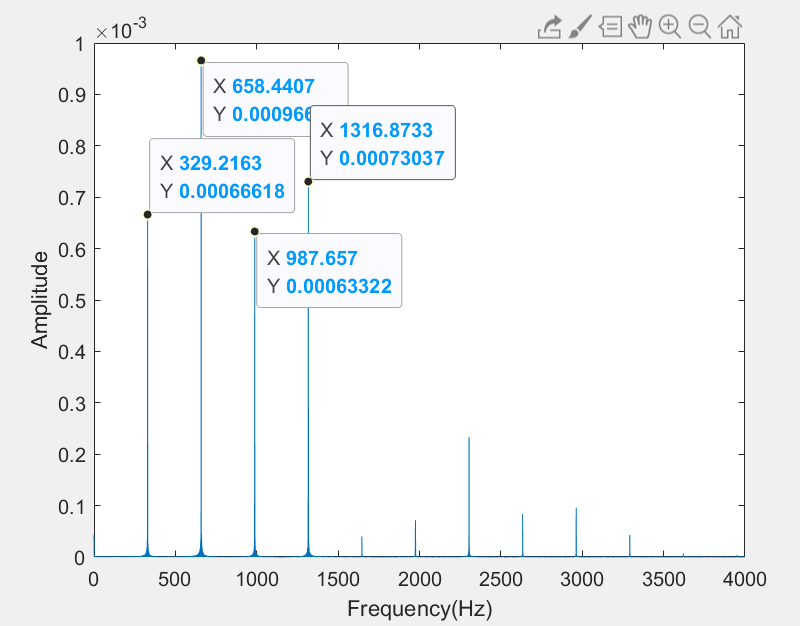
1. **客观结果：**这段乐音的基频是329.2163Hz。其音调与中央A（440Hz）相比，约相差个半音，即约为中央E。其谐波分量及幅值如表8所示：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 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**

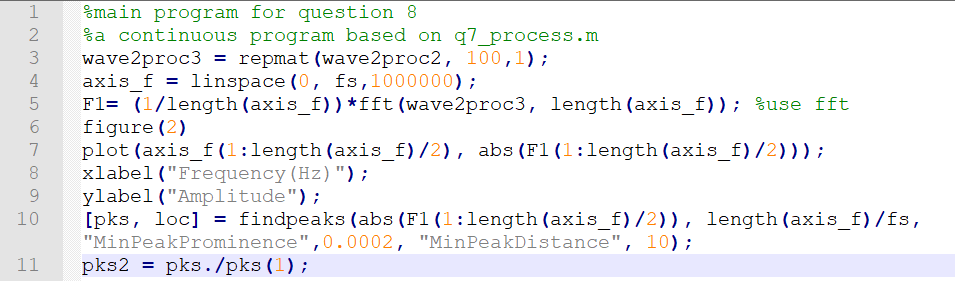
其频谱如图8所示。

**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



* 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。

1. **基于Fourier级数合成音乐**