音乐合成大作业

2021年9月1日　20:38

参考内容说明

c读取midi文件部分使用了开源的kts/matlab-midi: Matlab scripts to read and write MIDI files (github.com)项目，大作业通过该项目实现了midi文件的读取和转换，得到每个音符的开始、结束时间，强度和音名。src文件夹中的getTempoCHanges.m,midiinfo,m,readmidi.m为该项目所得，该项目所有代码在matlab-midi-master 文件夹中。
多周期平均除噪参考了学长的方法，代码部分自己实现。
其余内容有参考查阅资料，但代码均为原创。

基础知识：音乐调性与频率

下表是midi标准格式中每个十六进制数对应的音符，可以看到，按照十二平均律的划分方式，midi谱用0-127的数字表示了11个半的八度，频率也从C-1的8.176Hz覆盖到了G9的12.5kHz，这一频率范围几乎接近人儿听力极限。

实际的音乐创作中，往往并不需要用如此宽的音域。普通人通常能够唱出一个半八度已经是比较不错的水平，专业的歌手往往也只能覆盖两个八度24个音。并且，每个八度中的12个音，也并不是全部需要参与到谱曲。现实的作曲往往需要确定一个基音，每一个小节的结束，都需要作曲家回到基音。人们便选择不同的音符作为基音，也就产生了调性。如果将C大调中的F作为基音do,就是F大调。
但是，调性的划分不止包括基音的选择，还包括了基音与每个常用音符的跨度：常见的大调音乐按照2，2，1，2，2，2，1的步长夸过一个八度，日本常见的小调音乐使用2，1，2，2，1，2，2的步长，更特殊一些的，中国古代的宫商角徵羽五音调性，也是一种划分方式。

八度	音符编号											
	C	C#	D	D#	E	F	F#	G	G#	A	A#	B
–1	0	1	2	3	4	5	6	7	8	9	10	11
0	12	13	14	15	16	17	18	19	20	21	22	23
1	24	25	26	27	28	29	30	31	32	33	34	35
2	36	37	38	39	40	41	42	43	44	45	46	47
3	48	49	50	51	52	53	54	55	56	57	58	59
4	60	61	62	63	64	65	66	67	68	69	70	71
5	72	73	74	75	76	77	78	79	80	81	82	83
6	84	85	86	87	88	89	90	91	92	93	94	95
7	96	97	98	99	100	101	102	103	104	105	106	107
8	108	109	110	111	112	113	114	115	116	117	118	119
9	120	121	122	123	124	125	126	127				

为了确定不同调性不同音名频率，我们使用tone2midi函数：

```
function midi=tone2midi(tone,rof,oct,tonality,majo,raise)
    %举例，C大调的C4，输入参数为（1c,0(C音不升不降)，4，1（C调）
    %1（大调）,0（C大调不升不降））
    %举例，升F小调的D3，输入参数为（2（D）,0，3，4（F调）
    %0（minor），1（升调））
    %类似的，降调则输入-1.
    major=[0 2 4 5 7 9 11];
    minor=[0 2 3 5 7 8 10];
```

```
    %大调音阶全全半全全全半（2212221）
    %小调音阶全半全全半全全（2122122）
    if(majo>0)
        midi=12*oct+major(tone)+raise+tonality+11+rof;
    else
        midi=12*oct+minor(tone)+raise+tonality+11+rof;
    end
end
```

之后，再用freqc函数得到midi谱中音符与频率的关系：

```
function freqs=freqc(midi)
    %根据标准midi音符表示格式计算出频率
    freqs=220*2^((midi-57)/12);
end
```

这样就可以准确的获得每个音符的频率，例如，我们需要获得F大调中mi音的频率:

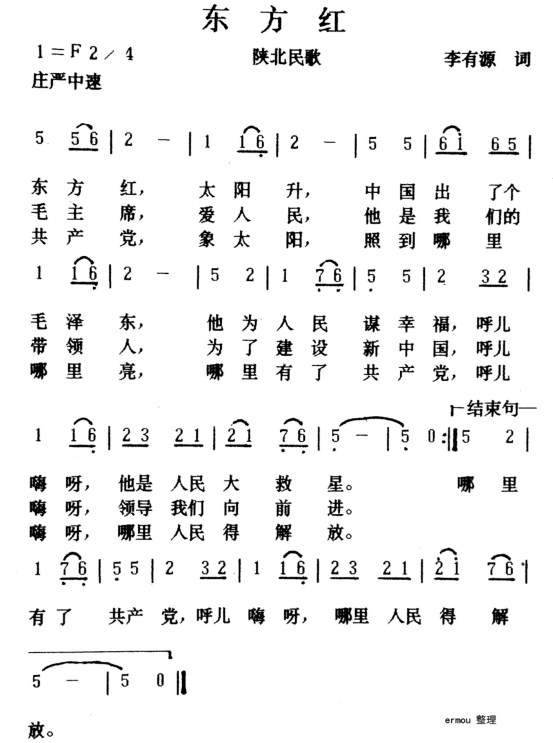
```
>> freqc(tone2midi(3,0,4,4,1,0))
%3表示mi音，0为不升不降，4为第四个八度，4为F调，1表示大调，0表示该调性即为F大调而不是升F
ans =

 391.9954
```

简单的音乐合成

1. 粗鄙的音乐合成

根据《东方红》简谱计算频率。



东 方 红

1＝F 2／4　　陕北民歌　　李有源 词

庄严中速

5 5 6 | 2 – | 1 1 6 | 2 – | 5 5 | 6 1 6 5 |

东 方 红， 太 阳 升， 中 国 出 了个
毛 主 席， 爱 人 民， 他 是 我 们的
共 产 党， 象 太 阳， 照 到 哪 里

1 1 6 | 2 – | 5 2 | 1 7 6 | 5 5 | 2 3 2 |

毛 泽 东， 他 为 人 民 谋 幸 福， 呼儿
带 领 人， 为 了 建 设 新 中 国， 呼儿
哪 里 亮， 哪 里 有 了 共 产 党， 呼儿

结束句

1 1 6 | 2 3 2 1 | 2 1 7 6 | 5 – | 5 0 :‖ 5 2 |

嗨 呀， 他是 人民 大 救 星。 哪 里
嗨 呀， 领导 我们 向 前 进。
嗨 呀， 哪里 人民 得 解 放。

1 7 6 | 5 5 | 2 3 2 | 1 1 6 | 2 3 2 1 | 2 1 7 6 |

有 了 共产 党，呼儿 嗨 呀， 哪里 人民 得 解

5 – | 5 0 ‖

放。

ermou 整理

我们尝试计算前四节的频率，根据F大调，我们按照如下模板计算：

freqc(tone2midi(x,0,y,4,1,0))，其中，x表示音名，y表示第y个八度，根据以上，可得如下换算关系。

x=(1:7)

x =

   1   2   3   4   5   6   7

>> freqc(tone2midi(x,0,4,4,1,0))

ans =

311.1270 349.2282 391.9954 415.3047 466.1638 523.2511 587.3295

音符	频率
C4	311.127
D4	349.2282
E4	391.9954
F4	415.3047
G4	466.1638
A4	523.2511
B4	587.3295

根据以上，我们尝试合成出前两句乐曲：

```
Fs=8000;
data=[];
%tones为乐谱，每行表示一个音符，第一列表示时间，单位秒，第二行为音名，第三行为八度
tones=[0.5,5,3;0.25,5,3;0.25,6,3;1,2,3;0.5,1,3;0.25,1,3;0.25,6,2;1,2,3];
lengthMusic=length(tones(:,1));
for i = 1:1:lengthMusic
    freq=freqc(tone2midi(tones(i,2),0,tones(i,3),4,1,0));
    tempWave=waveGen(tones(i,1),freq,1,Fs);
    data=[data,tempWave];
End
%归一化，实际上后来发现并不需要，因为这里不涉及叠加的过程
data=data/max(abs(data));
plot(data);
sound(data);
```

waveGen函数用来生成单一音符的波形，同时对后续的音色决定参数（如谐波、adsr参数）提供api。

```
function data=waveGen(time,freq,laud,fs)
    harmonicArray=[1];
    %谐波数组，此处暂不涉及谐波故为1
    lengthHarmonic=length(harmonicArray);
    data=0;
    for i=1:lengthHarmonic
        data=sin(2*pi*i*freq*(0:round(time*fs))/fs)*harmonicArray(i)+data;
    end
    %data=data.*adsr;
    %波形包络adse设定api
    data=data/max(abs(data)).*laud;
    %归一化，由于后续谐波叠加可能出现峰值超过1，削顶失真
end
```

至此，我们已经可以生成简单的音乐，虽然听感就很音叉

2. ADSR包络

由于前面已经为waveGen函数提供了方便的接口，我们可以搞一个简单的函数来做出一个adsr包络的样子

```
function asr=adsr(time,fs)
    asr=0;
    div=[0.02 0.38 0.3 0.3];
    %adsr四个阶段的占比
    st=0.6;
    x=linspace(0,div(1)*time,div(1)*time*fs);
    asr=2^20.^x-1;
    x=linspace(0,div(2)*time,div(2)*time*fs);
    asr=[asr,(st^(1/0.35)).^x];
    x=linspace(st,st,div(3)*time*fs);
    asr=[asr,x];
    x=linspace(0,div(4)*time,div(4)*time*fs);
    asr=[asr,(0.3.^x*st)];
end
```

Array div表示四个部分分别占比多少，st表示s阶段时音量的大小。

调试发现，s阶段不必太长，a阶段要远短于其他阶段，否则听感比较怪。

另，此函数实际上没有保证数组的长度，我们也懒得算具体应该多长，因此我选择了一个粗鄙的做法，在于data相乘时直接补0，虽然很粗鄙但是管用（

3. 变调

有多种办法，但普遍比较粗鄙

- 改谱子，

```
tones=[0.5,5,3;0.25,5,3;0.25,6,3;1,2,3;0.5,1,3;0.25,1,3;0.25,6,2;1,2,3];
Tones(:,3)=tones(:,3)+1;
```

- 用2倍的采样率播放，原采样率合成的音乐，虽然这会导致音乐时间变短。

```
plot(data);
sound(data,16384);
```

如此一来，音乐的时间就缩短了一半，这是我们不期望的。

进过查询，我们发现存在变速不变调、变调不变速的算法，例如OLA、WSOLA、LSEE-MSTFTM，但是我发现根本看不懂啊，这里就真的做不下去了。

4. 谐波

由于在waveGen函数中留下了方便的接口，这里只需手动输入不同的谐波数组即可。

```
function data=waveGen(time,freq,laud,fs)
    harmonicArray=[1 0.35 0.23 0.12 0.04 0.08 0.08 0.08 0.12];% 0.35 0.23 0.12 0.04 0.08 0.08 0.08 0.12
    %谐波数组
    lengthHarmonic=length(harmonicArray);
    data=0;
    for i=1:lengthHarmonic
        data=sin(2*pi*i*freq*(0:round(time*fs))/fs)*harmonicArray(i)+data;
    end
    asr=adsr(time,fs);
    lengthReq=length(data)-length(asr);
    asr=[asr,zeros(1,lengthReq)];
    data=data.*asr;
    data=data/max(abs(data)).*laud;
end
```

目前给出的这一个数组为我从现有吉他音频信号做FFT分析得出，但是后面听起来效果不是很好，更像是钢琴的声音。

5. 其他音乐合成

这里尝试了使用MIDI作为乐谱输入，合成音乐。使用了开源的kts/matlab-midi后，合成变得简单了不少。由于MIDI格式输入要求必须要有叠音，这里重写了合成器函数，以便实现叠音。

```
midiOrgin=readmidi("./jesu.mid");
midiProc=midiInfo(midiOrgin,0);
fs=8192;
musicLength=ceil((midiProc(end,6)+15)*fs);
%预先生成音乐数组
z=zeros(musicLength,1);
for i=1:size(midiProc)
    timeBegin=midiProc(i,5);
    timeEnd=midiProc(i,6);
    laud=midiProc(i,4)/100;
    tone=midiProc(i,3);
    freq=freqc(tone);
    x=waveGen(timeEnd-timeBegin,freq,laud,fs);
    nBegin=max(1,round(timeBegin*fs));
    z(nBegin:nBegin+length(x)-1)=z(nBegin:nBegin+length(x)-1)+x';
end
z=z/max(abs(z));
%归一化
```

合成之后，效果居然还不错！虽然仍然不像吉他的声音，但是不妨碍我们终于能够被称为是音乐的东西。

音乐分析

6. 试听fmt.wav

直接双击导入wav数据，之后用

```
plot(data);
sound(data,8000);
```

播放数据，注意，这里需要使用8000的采样率播放。实际上由于常见晶体振荡器为12MHz，8192及其倍数是更常见的采样率。

这里我们先准备好一个用来打印频谱的函数plotFFT：

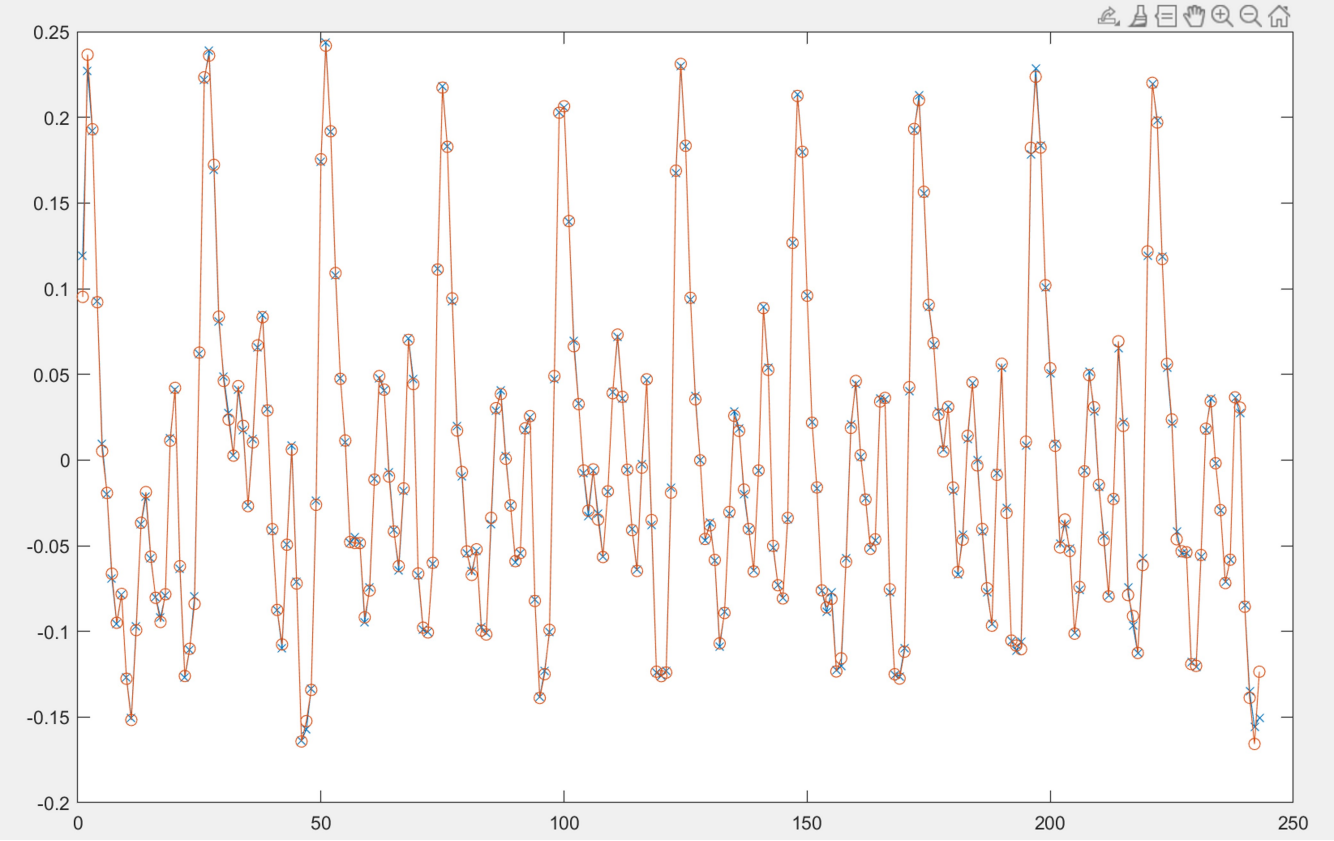
```
function [fre,ff]=plotFFT(sig,fs)
    N = length(sig);
    fre = (0:N-1)/N*fs;
    ff=abs(fft(sig));
    plot(fre,ff);
end
```

这里的返回值是频率和强度。

7. Realwave-Wave2Proc

这里我很长时间没有获得思路，一开始的想法是先做到频域上，然后消除掉所有能量非常小的分量，但是这样

```
ff=fft(realwave)
ff(find(abs(ff)<0.1))=0
y=ifft(ff)
```



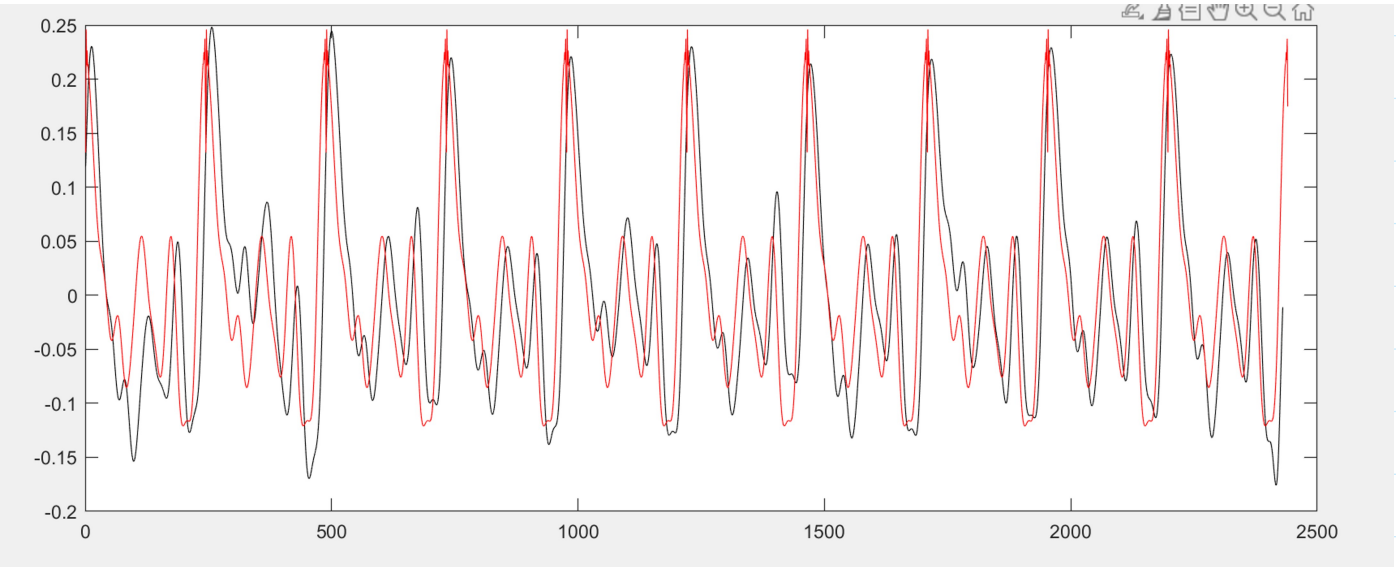
最终得到的如图所示，可以看到，去掉弱的频率分量并不是很理想。并且这会导致数据损失太大

不得已我参考了学长的做法：使用分周期平均，但我对这一方法进行了改进。学长的函数需要手动确定波的数量并限制输入的波形为完整的n个周期，这里我使用了一个粗鄙的做法来确定输入波形是完整的n个周期

首先在输入信号的前1/3和后1/3中分别找出最大值，截取两个最大值中间的部分作为输入，统计意义上来说这样操作可以避免输入非完整周期的情况。

```
function output = waveproc(input)
    input=resample(input,10,1);
    lengthInput=length(input);
    [temp,max1]=max(input(1:round(lengthInput/3)));
    [temp,max2]=max(input(round(lengthInput*2/3):end));
    max2=max2+round(lengthInput*2/3)-1;
    temp=input(max1:max2);
    lengthT=length(find(findpeaks(temp)>0.8*max(temp)))+1;
    lengthD=length(temp);
    sampleScale=lcm(lengthD,lengthT);
    x = resample(temp,sampleScale,lengthD);
    A = reshape(x,sampleScale/lengthT,lengthT).';
    p = mean(A);
    preal=resample(p,lengthD,sampleScale*10);
    output = repmat(preal.',10,1);
end
```

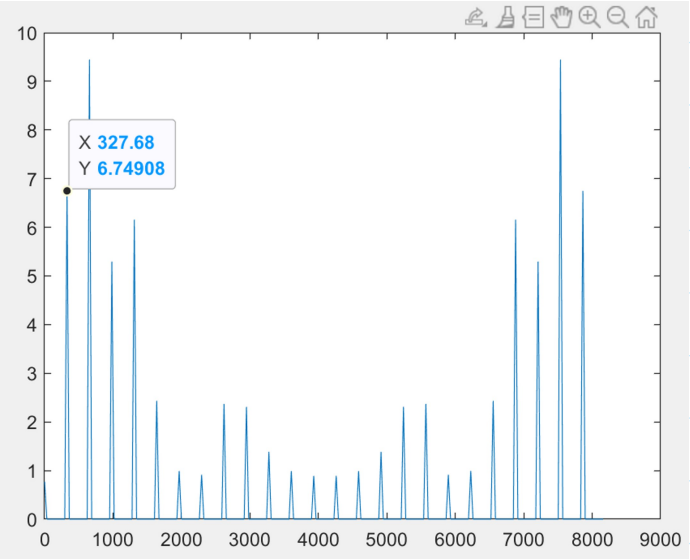
最终对realwave处理的结果如图所示



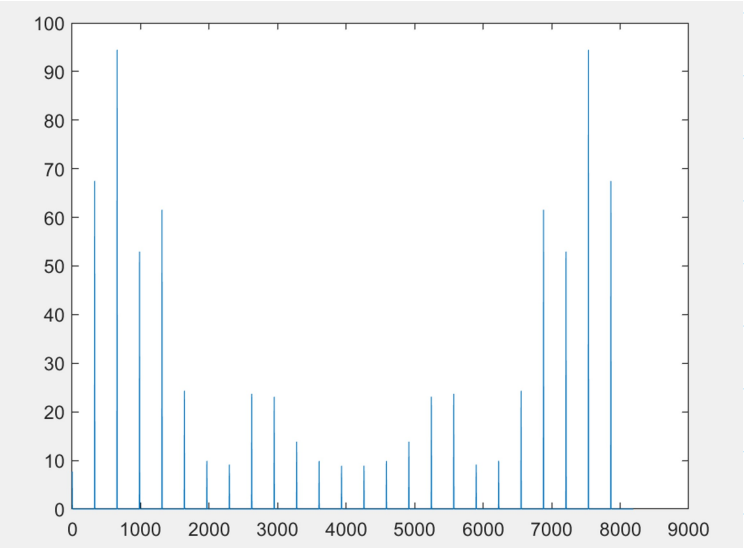
用傅里叶变换展示其频谱，可以看到比realwave更加接近离散谱

8. 音调分析

经过上面的处理，这一段音乐的基频可以清楚地观察出该音符的基频：327.68Hz。接近C大调的E4频率329Hz。



如果在处理前将output再重复10次，得到的频谱更接近冲击函数：



这是由于周期数约接近无穷，我们的信号就越接近于连续周期函数，此时的频谱就会退化为分立谱

为了方便下一步的自动音乐分析，我搞出了findFreq函数：

```
function freqc = findFreq(input,fs)
    wave2proc=waveProc(input);
    [ff,fa]=plotFFT(wave2proc,fs);
    fa(find(abs(fa)<10))=0;
    [pks,logs]=findpeaks(fa);
    freqc=min(ff(logs));
End
```

这玩意工作起来确实没问题，例如对于realfreq：

```
findFreq(realwave,8192)

ans =

  327.6800
```

9. 自动音乐分析

大体思路是每0.05秒做一次分割，做一次findFreq，获得freq数列后就差不多搞定了

为了实现0.05秒的分割，我们需要首先对data进行补齐，然后用reshape重新分割。

```
function output = waveProc(input)
    input=resample(input,10,1);
    lengthInput=length(input);
    [temp,max1]=max(input(1:round(lengthInput/3)));
    [temp,max2]=max(input(round(lengthInput*2/3):end));
    max2=max2+round(lengthInput*2/3)-1;
    temp=input(max1:max2);
```

```
    lengthT=length(find(findpeaks(temp)>0.8*max(temp)))+1;
    lengthD=length(temp);
    sampleScale=lcm(lengthD,lengthT);
    x = resample(temp,sampleScale,lengthD);
    A = reshape(x,sampleScale/lengthT,lengthT).';
    p = mean(A);
    preal=resample(p,lengthD,sampleScale*10);
    output = repmat(preal.',100,1);
end
```

利用上述函数即可较为容易的获得乐曲的曲调。

10. 高阶音乐合成

根据上面的plotFFT函数，我们可以很容易的提取出不同音符的谐波数组。将这一数组加入到waveGen函数中。

根据频率的范围选择不同的谐波函数来演奏。

```
function data=waveGen(time,freq,laud,fs)
    harmonicArray=[1 0.35 0.23 0.12 0.04 0.08 0.08 0.08 0.12 ; 1 0.47 0.35 0.12 0.14 0.08 0.11 0.7 0.16;1 0.24 0.37 0.17 0.24
0.13 0.19 0.05 0.05];% 0.35 0.23 0.12 0.04 0.08 0.08 0.08 0.12
    if(freq<263)
        flag=1;
    else
        if(freq<445)
            flag=2;
        else
            flag=3;
        end
    end
    %谐波数组
    lengthHarmonic=length(harmonicArray);
    data=0;
    for i=1:lengthHarmonic
        data=sin(2*pi*i*freq*(0:round(time*fs))/fs)*harmonicArray(flag,i)+data;
    end
    asr=adsr(time,fs);
    lengthReq=length(data)-length(asr);
    asr=[asr,zeros(1,lengthReq)];
    data=data.*asr;
    data=data/max(abs(data)).*laud;
end
```

实际上没有本质的困难。11题也是类似的