MATLAB 综合实验之音乐合成

聂浩　无 31　2013011280

2015 年 8 月 12 日

[1]

1　简单的合成音乐

[2]

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？

   思路：利用 12 平均律的定义，并假设每拍为构造 m_ note 函数, 将音调和持续时间传过去[3]，调用一次函数发一次音，这样只需要按照次序产生各个音即可。[4]

   听起来调子是对的，但是各个音之间有爆破音。

   代码如下 (dfh_1.m 和 m_note_1.,m)：

```
clear;clc;
m_note_1(12,0.5,196,8000);
m_note_1(12,0.25,196,8000);
```

[1]所有的.m 文件均采用 utf8 编码，在 matlab 中打开可能会出现中文乱码的情况，请用其它编辑器打开

[2]本部分在目录 dfh 中执行

[3]与输入的 base 相差的音阶，从 -5 到 19 共 25 个音，传 -10 表示这是空节拍

[4]这里参考了 matlab help kron 中的内容

```
m_note_1(14,0.25,196,8000);
m_note_1(7,1,196,8000);
m_note_1(5,0.5,196,8000);
m_note_1(5,0.25,196,8000);
m_note_1(2,0.25,196,8000);
m_note_1(7,1,196,8000);
```

```
function m_note_1(note,time,base,f)
    %输入参数
    %音符note,
    %between -5~25,
    %-10 means an empty beat
    %持续时间time
    %基调base
    %采样频率f
    n=[-5:19];
    freq=base*2.^kron(1/12,n);
    t=[0:time*f]/f;
    if note==10
        s=0*t;
    else
        s=cos(2*pi*freq(note+6)*t);
        sound(s,f);
        pause(time+0.01);
    end
    return
```

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。

   思路：直接利用.* 给生成的 s 信号乘以一个指数变换的序列来改变包

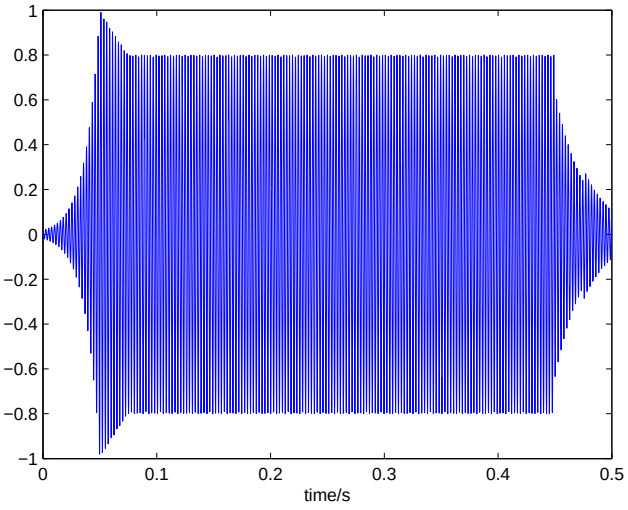

图 1: 增加了包络的信号

络，使得每一个音看起来像图 1，虽然开始结束的时间点不严格为 0，但是通过修改参数使其很小。

同时为了之后方便处理，将每个音的产生和播放分开，将每次产生的音乐序列连在数组 s 后面，最后统一播放。这里消除了爆音，但是听起来音色很奇怪。

代码如下 (dfh_2.m 和 m_note_2.,m):

```
clear;clc;format long;
base=196;f=8000;s=zeros(f*0.05,1);
s=[s;m_note(12,0.5,196,8000)];
s=[s;m_note(12,0.25,base,f)];
s=[s;m_note(14,0.25,base,f)];
s=[s;m_note(7,1,base,f)];
s=[s;m_note(5,0.5,base,f)];
s=[s;m_note(5,0.25,base,f)];
s=[s;m_note(2,0.25,base,f)];
s=[s;m_note(7,1,base,f)];
sound(s,f);
```

```
function [s]=m_note_2(note,time,base,f)
    %输入参数
    %音符note,
    %between -5~25,
    %-10 means an empty beat
    %持续时间time
    %基调base
    %采样频率f
    %声音信号s
    n=-5:19;
    freq=base*2.^kron(1/12,n);
    t=((1:(time)*f)/f)';
    if note==-10
        s=0*t;
    else
        e=(t<0.1*time).*(t-0.1*time)*80+(t
            >=0.1*time&t<0.15*time).*(0.1*
            time-t)*4.46/time-(t>=0.15*time&
            t<0.95*time)*0.223+(t>=0.9*time)
            .*((0.9*time-t)*40-0.223);
        %exp(e)为指数序 ?
        s=exp(e).*(cos(2*pi*freq(6+note)*t)
            );
    end
    return
```

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）

   答：最简单的方法是改变播放时 sound 函数的采样率来进行播放，本质上是改变播放的速率。利用十二品平均律的定义，利用上一问中得到的 s，可以有：

二倍速播放，高了一个八度

```
sound(s,16000)
```

即 0.5 倍速播放，低了一个八度。

```
sound(s,4000)
```

resample 函数为修改函数的采样率，然后以原来的采样率播放，就会改变音调。提高半个音阶使 s 的采样率变为 $\frac{8000}{2^{\frac{1}{12}}}$, 然后仍然以 8000 的采样率播放即可，这样做依然会有播放时间改变的问题。具体做法为：

但是, 无论是

```
resample(s,10000,10594);
sound(s,8000)
```

由于只改变了半个音阶，虽然没有对音乐的时间进行调整，但是听起来变化不大。

思考：本题的思路比较简单，主要在于对 resample 函数使用和原理的了解。这里利用 matlab 提供的文档以及搜索引擎进行学习也是掌握 matlab 函数的常规方法。

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？）

   思路：只需在产生每个音时把谐波加上就可以了，其他并不用改变，代码如下 (dfh_3.m,m_note_3.m)

```
clear;clc;format long;
base=196;f=8000;s=zeros(f*0.05,1);
s=[s;m_note_3(12,0.5,base,f)];
s=[s;m_note_3(12,0.25,base,f)];
s=[s;m_note_3(14,0.25,base,f)];
s=[s;m_note_3(7,1,base,f)];
```

```
s=[s;m_note_3(5,0.5,base,f)];
s=[s;m_note_3(5,0.25,base,f)];
s=[s;m_note_3(2,0.25,base,f)];
s=[s;m_note_3(7,1,base,f)];
sound(s,f);
```

```
function [s]=m_note_3(note,time,base,f)
    %输入参数
    %音符note
    %between-5~25
    %-10 means an empty beat
    %持续时间time
    %基调base
    %采样频率f
    %声音信号s
    n=-5:19;
    freq=base*2.^kron(1/12,n);
    t=((1:(time)*f)/f)';
    if note==-10
        s=0*t;
    else
        e=(t<0.1*time).*(t-0.1*time)*80+(t
            >=0.1*time&t<0.15*time).*(0.1*
            time-t)*4.46/time-(t>=0.15*time&
            t<0.95*time)*0.223+(t>=0.9*time)
            .*((0.9*time-t)*40-0.223);
        %exp(e)为指数序 ?
        s=exp(e).*(cos(2*pi*freq(6+note)*t)
            +0.2*cos(2*pi*2*freq(6+note)*t)
            +0.3*cos(2*pi*3*freq(6+note)*t))
            ;
    end
    return
```

思考：听起来还是有点风琴的感觉，但是似乎更像笛子，这和音的包络也有关系。

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。

思路：合成的是致爱丽丝的第二节。出于方便输入的考量，首先将标准的简谱和对应的音的时间长存在两个数组中，然后用 matlab 转换谱子为十二平均律的形式，再循环调用之前的 m_note_3 函数代码如下 (my_music.m):

```
clear;clc;clear all;
%tone 中
%1~7 与简谱一致
tone=[13 12.5 13 12.5 13 7 12 11 6 -10 1 3
    6 7 -10 3 11 7 6];
time=[0.25 0.25 0.25 0.25 0.25 0.25 0.25
    0.25 0.5 0.25 0.25 0.25 0.25 0.5 0.25
    0.25 0.25 0.25 0.5];
%谱子转十二平均律
msheet2tone=[-3 -4 -4.5 -5 -5.5 -6 -6.5 -7
    1 1.5 2 2.5 3 4 4.5 5 5.5 6 6.5 7 11
    11.5 12 12.5 13];
for i=1:length(tone)
    [~,num]=find(msheet2tone==tone(i));
    if isempty(num)
        num=-4;
    end
    tone(i)=num;
end
s=[];
%演奏
for i=1:length(tone)
    s=[s;m_note_3(tone(i)-6,time(i)
        ,220,8000)];
end
```

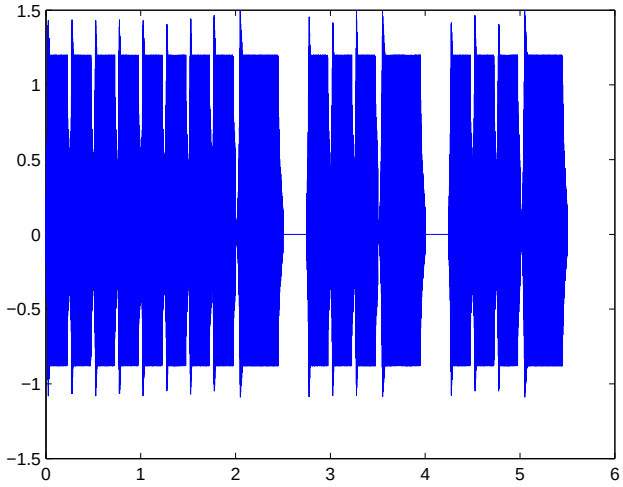

图 2: 合成音乐波形图

```
20 sound(s,8000);
```

波形图如图 2：

2 用傅里叶级数分析音乐

5

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？ 听起来是很明显的吉他声，确实真实很多。操作如下：

```
1 [R Fs]=audioread(fmt.wav);
2 sound(R,Fs);
```

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

[5] 本部分在目录 fmt 中执行

析音调是非常重要的。提示：从时域做，可以继续使用 resample 函数。

思路：先对 wave2proc 和 realwave 做傅立叶分析，画出其频谱图如图 3. 可以看出，两者的频谱除了低频部分基本是一致的，思路首先是滤去 realwave 的低频部分然后进行傅里叶逆变换。但题目要求以时域的方法处理，同时由于 realwave 中的低频分量不是很好处理，所以没有采用这种方案。

傅立叶分析代码[6](anay_wave.m)

```
load('guitar.mat');
R1=wave2proc;
R2=realwave;
Fs=8000;
L=length(R1);
NFFT=2^nextpow2(L);
Y1=fft(R1,NFFT)/L;
Y2=fft(R2,NFFT)/L;
nf=NFFT/2+1;
f= Fs/2*linspace(0,1,nf);
plot(f,2*abs(Y1(1:nf)),'b',f,2*abs(Y2(1:nf)),'r');
legend('wave2proc','realwave');
```

这里观察 wave2proc 和 realwave 的波形如图 4, 可以看出，大致有十个峰，将 wave2proc 的十个峰画在一起如图 5，可以看出，其周期性相当强。因此采用将 realwave 分为 10 个周期，然后对其求平均得到一个周期，再延拓的方法。需要注意的是，realwave 有 243 个采样点，应先用 resample 函数将此采样为 250 个点，处理完后再恢复为 243 个。生成的波形如图 6

画图代码为

```
        clc;clear;close all;
        Fs=8000;
```

[6]这里参照了 matlab fft demo 的代码

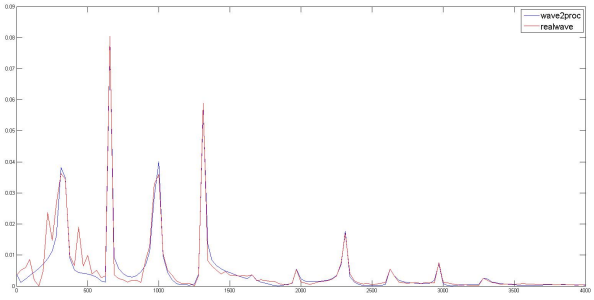

图 3: wave2proc 与 realwave 的频谱

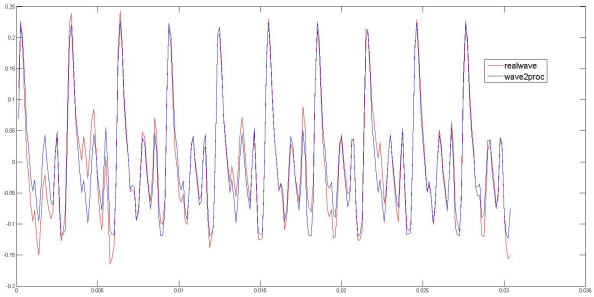

图 4: wave2proc 与 realwave 的波形

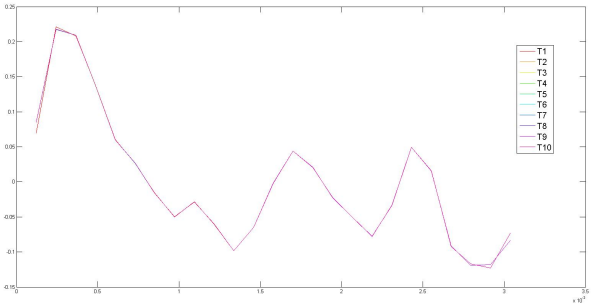

图 5: wave2proc 的十个周期

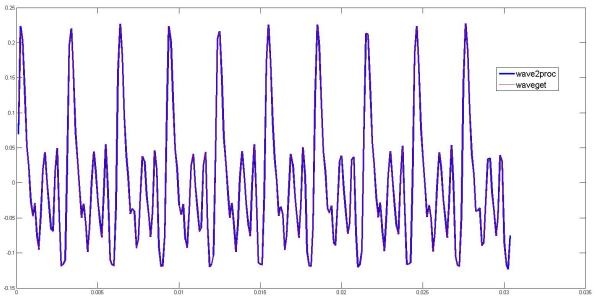

图 6: wave2proc 与得到的波形

```
load guitar.mat;
tmp=resample(wave2proc,250,243);
t=(1:25)/(8000/243*250);
colors= hsv(10);
labels=[];
for d=1:10
plot(t,tmp((d-1)*25+1:d*25),'color'
    ,colors(d,:));
labels{d}=['T' num2str(d)];
hold on
end
legend(labels);
```

波形生成代码 (wave4proc.m)

```
clc;clear;close all;
Fs=8000;
load guitar.mat;
tmp=resample(realwave,250,243);
a=zeros(250,25);
for d=1:25
    a((0:9)*25+d,d)=1/10;
end
```

```
wavetoproc=tmp'*a;
wavetoproc=kron(ones(10,1),wavetoproc');
wavetoproc=resample(wavetoproc,243,250);
t=(1:243)/8000;
plot(t,wave2proc,'LineWidth',3);
hold on;
plot(t,wavetoproc,'r');
legend('wave2proc','waveget')
```

可以看出，所得到的波形和 wave2proc 是非常相似的，这样做通过平均减弱了噪音和多次谐波的影响。

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。

   思路：将该段音乐重复 100 次后利用 fft 函数进行傅里叶变换，代码如下：

```
clear;clc;
Fs=8000;
load guitar.mat;
y=repmat(wave2proc,100,1);
L=length(y);
NFFT=2^nextpow2(L);
Y=fft(y,NFFT)/L;
nf=NFFT/2+1;
f= Fs/2*linspace(0,1,nf);
reso=Fs/2/nf;
Y=2*abs(Y(1:nf)');
```

```
plot(f,Y);
xlabel('f/Hz');
Y1=zeros(2,nf);
Y1(1,2:nf)=Y(1:nf-1);
Y1(2,1:nf-1)=Y(2:nf);
Y1=(Y1-[Y;Y]);
Y1=[1  1]*(Y1<0);
Y1=(Y1==2);
f=f.*Y1;

if (max(Y.*Y1))>0.03
    f=f.*((Y.*Y1)>0.03);
else
    f=f.*((Y.*Y1)>0.013);
end

[list,I]=sort(f+10000.*(f==0));
base=(1:6)*list(1);
a=zeros(1,6);
a(1)=Y(I(1));
for n=2:6
    [a1,f1]=max(Y(n*I(1)-fix(2/reso):n*I(1)
        +fix(2/reso)));
    base(n)=base(n)-2+f1*reso;a(n)=a1;
end
A=a';
base
A
```

得到如表 1

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

	基波	一次谐波	二次谐波	三次谐波	四次谐波	五次谐波
频率/Hz	328.10	658.40	987.26	1316.36	1645.22	1974.32
幅度	0.0439	0.0791	0.0438	0.0594	0.0025	0.0059
相对幅度[7]	1	1.80	1.00	1.35	0.06	0.13

表 1: realwave 的基波及谐波

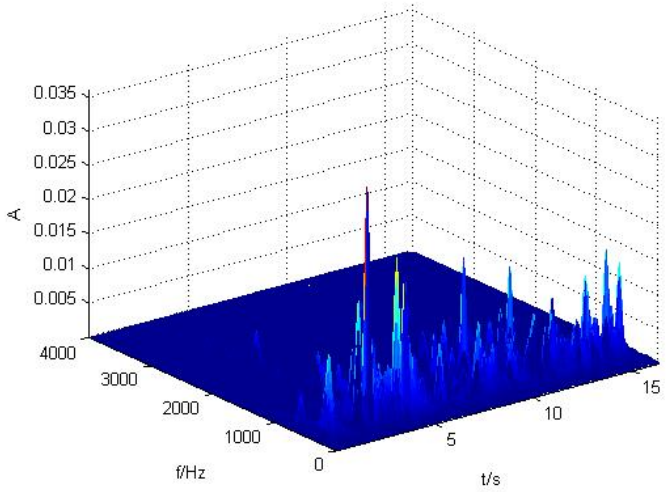

图 7: fmt 音乐的时频图

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。

思路: 因为之前做过小白鲸找妈妈的作业，所以第一想法是对信号进行短时傅里叶变换[8]，然后每隔 $\frac{1}{4}$ 拍读取一次频率，这样就可以得到各个时间点的音调, 因为音乐约为 16.3 秒，约为 32.5 拍，所以分为 140 段进行短时傅里叶变换。变换结果如图 7.

代码是这样的（short_fourier.m）

```
[s Fs]=audioread('fmt.wav');
L=length(s);
NFFT=2^nextpow2(L)/64;
```

[8]此处参照网络教程http://blog.sina.com.cn/s/blog_6163bdeb0102dwfw.html

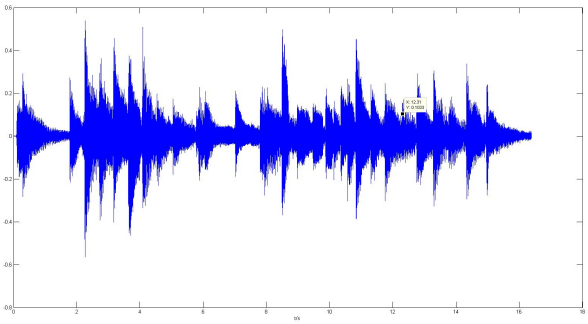

图 8: fmt 波形图

```
[S1,F1,T1,P1]=spectrogram(s,128,16,NFFT,Fs)
    ;
surf(T1,F1,sqrt(P1),'edgecolor','none');
    axis tight;
xlabel('t/s');ylabel('f/Hz');zlabel('A');
t=(L/Fs)
```

然而很难分清得到的 4 分音符究竟是独立的一个音还是和前后的音连在一起的长音。需要分析音乐的节拍。方案有：

- 提取音乐的包络，但是在用高斯滤波提取后，得到的波形图与原波形契合度不高，无法获得音节划分。
- 仿照 cooledit 节拍分化的功能，通过判断时域信号在短时间（指定时间，取 6ms）内的变化率（取 6.5dB 临界值）进行节奏分化。但是如图 8所示，12.37s 时幅度变化太小，小于一些拍子内部的变化, 无法划分。在调整参数后，也无法解决该问题

因此，采用最基本的手工标定法，利用 cooledit 将音乐分为 32 拍，分别存储在 fmt (2).wav fmt (3).wav 中，批处理获得各个音的音调即对应的的谐波强度。代码如下（anay.m,f_f.m）：

```
clc;clear;close all;
freq=[];time=[];A=[];
%freq 存储每段的基波与谐波频率
```

```
%time 存储每段音乐时长
%A 存储每段基波与谐波的振幅
for i=2:32
    [freq,time,A]=f_f(freq,time,['fmt(' num2str(i) ').wav'],A);
end

note=round(log2(freq(1,:)/220)*12);
```

```
function [freq,time,A]=f_f(freq,time,name,A)
% time 为本段的时长
% name 为文件名
% freq 返回基波和谐波频率
%A 返回振幅
%use demo fft of matlab
[R,Fs]=audioread(name);
y=repmat(R,100,1);
L=length(y);
time=[time length(R)/Fs];

NFFT=2^nextpow2(L);
Y=fft(y,NFFT)/L;
nf=NFFT/2+1;
f= Fs/2*linspace(0,1,nf);
reso=Fs/2/nf;
Y=2*abs(Y(1:nf)');

Y1=zeros(2,nf);
Y1(1,2:nf)=Y(1:nf-1);
Y1(2,1:nf-1)=Y(2:nf);
Y1=(Y1-[Y;Y]);
Y1=[1 1]*(Y1<0);
```

```
Y1=(Y1==2);
f=f.*Y1;
%取极大点对应的频率
if (max(Y.*Y1))>0.03
    f=f.*((Y.*Y1)>0.03);
else
    f=f.*((Y.*Y1)>0.013);
end
%加幅值的限制
[list,I]=sort(f+10000.*(f==0));
%得到基波
base=(1:6)*list(1);
a=zeros(1,6);
a(1)=Y(I(1));
%得到谐波
for n=2:6
    [a1,f1]=max(Y(n*I(1)-fix(2/reso):n*I(1)
        +fix(2/reso)));
    base(n)=base(n)-2+f1*reso;a(n)=a1;
end
freq=[freq base'];
A=[A a'];
return
```

最终的得到的结果为??

3 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？ 思路：将谐波的振幅关系传给 m_note_guitar 函数，使产生的音有谐波分量，代码如下:(dfh_guitar.m,m_note_guitar.m):

```
clear;clc;format long;
Fs=8000;
```

		基波	一次谐波	二次谐波	三次谐波	四次谐波	五次谐波
1	频率/Hz	219.67	439.31	658.95	878.58	1098.22	1317.85
	振幅	0.08166	0.00051	0.01535	0.00041	0.00175	0.00157
2	频率/Hz	219.84	440.35	661.52	881.36	1101.20	1320.36
	振幅	0.01373	0.00738	0.00448	0.00219	0.00089	0.00117
3	频率/Hz	247.94	495.86	743.79	993.77	1239.62	1487.55
	振幅	0.06010	0.00516	0.00205	0.00112	0.00017	0.00028
4	频率/Hz	217.04	434.08	651.09	868.12	1085.14	1302.17
	振幅	0.03314	0.00140	0.00086	0.00008	0.00014	0.00120
5	频率/Hz	247.18	494.34	741.50	988.66	1235.81	1482.97
	振幅	0.03062	0.00138	0.00173	0.00208	0.00015	0.00163
6	频率/Hz	329.99	659.97	989.94	1319.91	1649.89	1979.86
	振幅	0.04808	0.01838	0.02096	0.00402	0.00067	0.00069
7	频率/Hz	193.27	386.53	579.79	773.04	966.29	1159.54
	振幅	0.05014	0.00178	0.00287	0.00074	0.00053	0.00039
8	频率/Hz	173.71	347.41	521.12	694.81	868.50	1042.19
	振幅	0.07271	0.00062	0.00321	0.00141	0.00046	0.00084
9	频率/Hz	174.87	349.73	522.67	699.42	874.27	1049.12
	振幅	0.04760	0.01557	0.00190	0.00284	0.00437	0.00419
10	频率/Hz	174.25	349.88	524.12	698.36	872.60	1046.84
	振幅	0.02362	0.00930	0.00062	0.00048	0.00053	0.00040
11	频率/Hz	208.44	416.82	625.23	833.63	1042.01	1250.41
	振幅	0.03025	0.00538	0.00886	0.00038	0.00022	0.00024
12	频率/Hz	246.20	492.40	738.59	985.82	1232.02	1478.21
	振幅	0.01489	0.00822	0.00345	0.00443	0.00064	0.00077
13	频率/Hz	328.89	657.78	986.66	1315.54	1644.43	1973.31
	振幅	0.01709	0.02016	0.01450	0.00879	0.00060	0.00127
14	频率/Hz	221.07	442.15	663.21	884.26	1105.33	1326.38
	振幅	0.03992	0.01272	0.01351	0.00249	0.00018	0.00951
15	频率/Hz	222.23	444.44	666.61	888.81	1111.01	1333.21
	振幅	0.04759	0.00120	0.00139	0.00039	0.00006	0.00034
16	频率/Hz	220.78	441.56	662.31	883.07	1103.83	1324.60
	振幅	0.03763	0.00286	0.02018	0.00071	0.00039	0.00362
17	频率/Hz	440.29	880.58	1320.86	1761.14	2201.41	2641.69
	振幅	0.04344	0.01702	0.01354	0.00166	0.00205	0.00245
18	频率/Hz	221.10	442.21	663.30	884.40	1105.48	1326.57
	振幅	0.05490	0.01542	0.00635	0.00036	0.00012	0.00146
19	频率/Hz	393.10	786.21	1179.27	1572.37	1965.44	2358.53
	振幅	0.04801	0.02425	0.00497	0.00278	0.00178	0.00059
20	频率/Hz	349.61	699.20	1048.81	1398.39	1747.97	2097.55
	振幅	0.03477	0.00857	0.00117	0.00403	0.00194	0.00087
21	频率/Hz	330.11	660.17	990.25	1320.30	1650.37	1980.45
	振幅	0.03088	0.04671	0.03176	0.00407	0.00070	0.00038
22	频率/Hz	294.77	589.49	884.23	1178.94	1473.68	1768.42
	振幅	0.03460	0.03521	0.00643	0.00316	0.00200	0.00174
23	频率/Hz	262.57	525.15	787.70	1050.26	1312.82	1575.38
	振幅	0.06187	0.00556	0.01253	0.01083	0.00845	0.00422
24	频率/Hz	247.01	494.02	741.01	988.01	1235.00	1481.99
	振幅	0.04424	0.00933	0.00716	0.00281	0.00031	0.00081
25	频率/Hz	247.80	493.76	741.55	991.18	1237.12	1484.91
	振幅	0.03396	0.00139	0.00326	0.00284	0.00014	0.00043
26	频率/Hz	262.22	524.44	786.65	1048.87	1311.08	1573.29
	振幅	0.03060	0.00246	0.00339	0.00063	0.00031	0.00025
27	频率/Hz	175.31	348.68	523.97	699.26	874.56	1049.86
	振幅	0.04658	0.01264	0.00180	0.00741	0.00585	0.00315
28	频率/Hz	221.25	442.49	663.73	884.95	1108.24	1327.41
	振幅	0.06692	0.00938	0.01335	0.00087	0.00034	0.00257
29	频率/Hz	165.04	330.08	493.36	658.39	825.15	991.90
	振幅	0.01478	0.02411	0.01033	0.00143	0.00754	0.00351
30	频率/Hz	221.95	442.31	665.85	886.18	1109.72	1331.66
	振幅	0.08120	0.00481	0.00565	0.00012	0.00025	0.00089
31	频率/Hz	209.14	416.88	626.02	835.15	1044.28	1253.41
	振幅	0.01509	0.00495	0.00349	0.00007	0.00004	0.00003

图 9: fmt 分析结果**??**

```
load guitar.mat;
y=repmat(wave2proc,100,1);
L=length(y);

NFFT=2^nextpow2(L);
Y=fft(y,NFFT)/L;
nf=NFFT/2+1;
f= Fs/2*linspace(0,1,nf);
reso=Fs/2/nf;
Y=2*abs(Y(1:nf)');

Y1=zeros(2,nf);
Y1(1,2:nf)=Y(1:nf-1);
Y1(2,1:nf-1)=Y(2:nf);
Y1=(Y1-[Y;Y]);
Y1=[1  1]*(Y1<0);
Y1=(Y1==2);
f=f.*Y1;
%取极大  点对应的频率?
if (max(Y.*Y1))>0.03
    f=f.*((Y.*Y1)>0.03);
else
    f=f.*((Y.*Y1)>0.013);
end
%加幅值的限制
[list,I]=sort(f+10000.*(f==0));
%得到基频
base=(1:6)*list(1);
a=zeros(1,6);
a(1)=Y(I(1));
for n=2:6
    [a1,f1]=max(Y(n*I(1)-fix(2/reso):n*I(1)
        +fix(2/reso)));
```

```
        base(n)=base(n)-2+f1*reso;a(n)=a1;
end
A=a';


base=220;f=8000;s=zeros(f*0.05,1);
s=[s;m_note_guitar(10,0.5,base,f,A)];
s=[s;m_note_guitar(10,0.25,base,f,A)];
s=[s;m_note_guitar(12,0.25,base,f,A)];
s=[s;m_note_guitar(5,1,base,f,A)];
s=[s;m_note_guitar(3,0.5,base,f,A)];
s=[s;m_note_guitar(3,0.25,base,f,A)];
s=[s;m_note_guitar(0,0.25,base,f,A)];
s=[s;m_note_guitar(5,1,base,f,A)];
sound(s,f);
```

```
function [s]=m_note_guitar(note,time,base,f
    ,A)
    %输入参数
    %音符note
    %between -5~25,-10 means an empty beat
    %持续时间time
    %基调base
    %采样频率f
    %声音信号s
    %谐波强度A


    A=A/A(1);
    n=-5:19;
    freq=base(1)*2.^kron(1/12,n);
    t=((1:(time)*f)/f)';
    if note==-10
```

```
17          s=0*t;
18      else
19          s=zeros(time*f,1);
20          for i=1:6
21              s=s+A(i)*cos(2*pi*freq(note+6)*
                    i*t);
22          end
23          %包络
24          e=(t<0.05*time).*(t-0.05*time)*80/
                time+(t>=0.05*time&t<0.3*time)
                .*(0.05*time-t)*4.43/time+(t
                >=0.3*time).*(2.7*(0.3*time-t)/
                time-1.10);
25          s=s.*exp(e);
26      end
27      return
```

思考：第一次做完后听起来并不像，但是这里的谐波分布和之前应该是一致的，为了更符合要求，我仿照图 8修改了包络，使得新的波形从 2变为 10. 修改之后听起来确实像吉他了，但是还有一些奇怪的杂音。

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。

    代码如下:

```
1  run anay.m;
2
3  %A=A./repmat(A(1,:),6,1);
4  %幅度改为相对 220Hz 为基准音
```

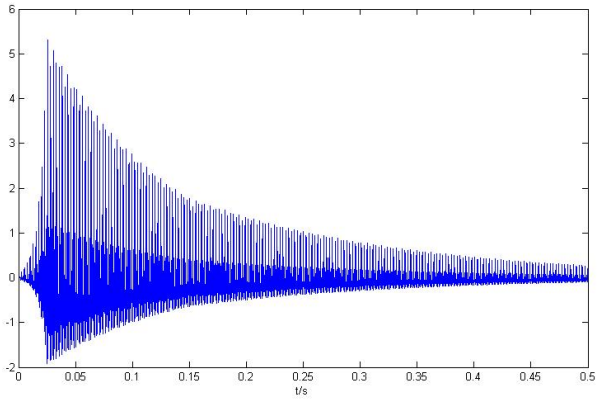

图 10: 新包络

```
f=round(log2(freq(1,:)/220)*12);
%以音名为索引
[C,I]=sort(f);
A_tmp=zeros(6,18);
tmp=C(1);count=0;
%将重复音的频谱特征取平均
for i=1:31
    if C(i)==tmp;
        A_tmp(:,C(i)+6)=A_tmp(:,C(i)+6)+A(:,I(i));
        count=count+1;
    else
        A_tmp(:,C(i-1)+6)=(A_tmp(:,C(i-1)+6)+A(:,I(i-1)))/count;
        tmp=C(i);
        count=1;
    end
end
A_tmp(:,C(31)+6)=A(:,I(31));
%不存在的音以两侧音的特征的平均值替代
```

```
for i=1:18
    if A_tmp(:,i)==0
        A_tmp(:,i)=(A_tmp(:,i+1)+A_tmp(:,i-1))/2;
    end
end

base=220;f=8000;s=zeros(f*0.05,1);
s=[s;m_note_guitar(10,0.5,base,f,A_tmp(:,10+6))];
s=[s;m_note_guitar(10,0.25,base,f,A_tmp(:,10+6))];
s=[s;m_note_guitar(12,0.25,base,f,A_tmp(:,12+6))];
s=[s;m_note_guitar(5,1,base,f,A_tmp(:,5+6))];
s=[s;m_note_guitar(3,0.5,base,f,A_tmp(:,3+6))];
s=[s;m_note_guitar(3,0.25,base,f,A_tmp(:,3+6))];
s=[s;m_note_guitar(0,0.25,base,f,A_tmp(:,0+6))];
s=[s;m_note_guitar(5,1,base,f,A_tmp(:,5+6))];
sound(s,f);
```

生成的波形如图 11

思考: 听起来很像吉他了，但是两个长音中还是有杂音. 但单独播放这个音的短音时并无杂音。所以这是拖得过长而导致的，需要给长音增加特殊的包络就能解决这一问题。本题的难点在于将第（9）问的数据应用在这里，这里进行了较为复杂的运算才得到，是自己有了不小的提升。

12. 现在只要你掌握了某乐器足够多的演奏资料，就可以合成出该乐器

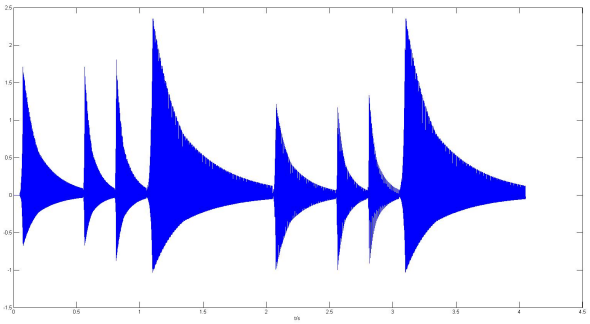

图 11: 生成的波形

演奏的任何音乐，在学完本书后面内容之后，试着做一个图形界面把上述功能封装起来。 GUI 界面如图 12其中内置了三首曲子：《东方红》《茉莉花》《致爱丽丝》; 提供 C 大调和 A 大调两种音调；提供吉他、风琴（如第四题，但是仍使用吉他的包络）、电子音（只有基波）三种音色。

在简谱栏中输入音符如 1 2 3 就是 do re mi，11 就是高八度的 do，-6 就是低八度的 la，1.5 就是降 re，-10 表示是空音。可以演奏从 -3 到 13 的所有音（C 大调，A 大调为 -5 到 15）在时间栏中输入各个音符对应的时间，然后样例乐谱选无，点击演奏就能听到音乐了。同时，用户可以点击分析音乐按钮，选择的一小段 wav 文件，程序会进行分析（无法进行分拍分析，只会当做一个音），然后音色选择 custom 就能使用该音色。

思考: 本题的核心是 GUI，主要的逻辑都需要做在按钮的 callback 函数中。因为给用户的反馈是按下按钮后才发声。这有几个难点，一个是空音，需要选择一个不会出现的数字。另一方面，允许用户输入，所以处理输入的音符时应该对其进行限制来增强鲁棒性。代码如下:(gui.m)

```
function varargout = gui(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',
    mfilename, ...
```

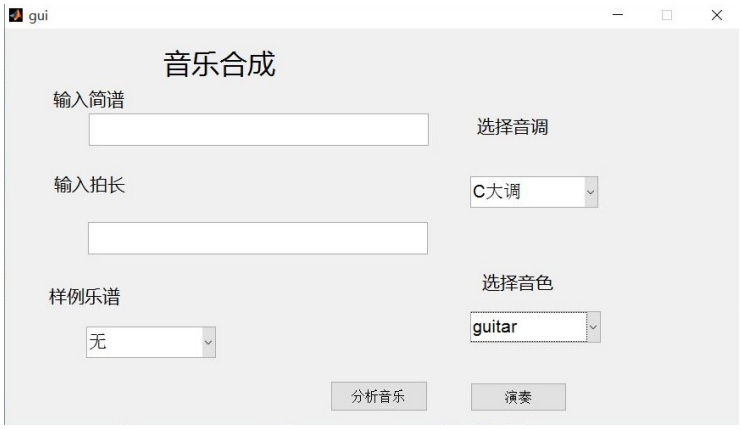

图 12: GUI 界面

```
4                       'gui_Singleton',
                            gui_Singleton, ...
5                       'gui_OpeningFcn',
                            @gui_OpeningFcn, ...
6                       'gui_OutputFcn',
                            @gui_OutputFcn, ...
7                       'gui_LayoutFcn',  [] ,
                            ...
8                       'gui_Callback',   []);
9   if nargin && ischar(varargin{1})
10      gui_State.gui_Callback = str2func(
            varargin{1});
11  end
12
13  if nargout
14      [varargout{1:nargout}] = gui_mainfcn(
            gui_State, varargin{:});
15  else
16      gui_mainfcn(gui_State, varargin{:});
17  end
18  % End initialization code - DO NOT EDIT
```

```
19
20
21 % --- Executes just before gui is made
       visible.
22 function gui_OpeningFcn(hObject, eventdata,
       handles, varargin)
23 % Choose default command line output for
       gui
24 handles.output = hObject;
25
26 % Update handles structure
27 %不同音调guitar
28 handles.guitar=[0.0295583566761820
       0.0410941345134757  0.0456155248168245
       0.0501369151201733  0.0150942327948900
       0.0506918143957047  0.0411115638971236
       0.0315313133985425  0.0306018188528348
       0.0326034060041743  0.0346049931555137
       0.0304438595623305  0.0262827259691473
       0.0347713477835479  0.0413905767692186
       0.0480098057548894  0.0457266497451795
       0.0434434937354695;
29 0.0482126003883897  0.0125374140745269
       0.00716059137425992 0.00178376867399290
       0.00495078360762451 0.00666430595013039
       0.00550363690446728 0.00434296785880417
       0.00246330674711421 0.0188350822355054
       0.0352068577238966  0.0365342250971077
       0.0378615924703188  0.00857053773470121
       0.0164106182936462  0.0242506988525912
       0.0206351382394561  0.0170195776263210;
30 0.0206558500193487  0.00152760756396231
       0.00219890018041650 0.00287019279687068
```

```
    0.00349294202926450  0.00793580678314517
    0.00585318846766221  0.00377057015217924
    0.00338667456050978  0.00490771858196665
    0.00642876260342353  0.0162190114081411
    0.0260092602128588   0.00117258606592246
    0.00307346918072569  0.00497435229552892
    0.00925582041626151  0.0135372885369941;
31  0.00285336304856058  0.00453640669792585
    0.00263949648145414  0.000742586264982436
        6.68401761333680e-05
    0.000814065440693232
    0.00190609132895165  0.00299811721721007
    0.000628637998314440
    0.00189444843869244  0.00316025887907043
    0.00440233177136729  0.00564440466366414
    0.00403417874801128  0.00340693225315865
    0.00277968575830602  0.00221847975503006
    0.00165727375175410;
32  0.0150876502796980   0.00414709484646134
    0.00233614670375124  0.000525198561041149
        4.20504657581285e-05
    0.000291684580679916
    0.000283492071550838
    0.000275299562421759
    0.000309384394071862
    0.00115327671849602  0.00199716904292017
    0.00132983323130851  0.000662497419696851
        0.00194061187195759
    0.00186164673013615  0.00178268158831472
    0.00191739094342452  0.00205210029853432;
33  0.00702799658954647  0.00272518393583796
    0.00155648685901399  0.000387789782190024
        3.17892339543552e-05
```

```
    0.00240451218998288 0.00160935705722773
    0.000814201924472573
    0.000248809480213054
    0.000996328456682373
    0.00174384743315169 0.00120906410397970
    0.000674280774807711
    0.000872029083003068
    0.000729265978417851
    0.000586502873832633
    0.00151829881512565
    0.00245009475641867];
handles.guitar=[handles.guitar repmat(
    handles.guitar(:,18),1,7)];
%organ
handles.organ=kron([1 0.2 0.3 0 0 0]',ones
    (1,25));
%基本
handles.basic=zeros(6,25);handles.basic
    (1,:)=1;
handles.custom=handles.basic;
handles.A=handles.guitar;
handles.tone=[];
handles.time=[];

guidata(hObject, handles);

% --- Outputs from this function are
    returned to the command line.
function varargout = gui_OutputFcn(hObject,
    eventdata, handles)
varargout{1} = handles.output;


```

```
51 % --- Executes on selection change in
       ch_tone.
52 function ch_tone_Callback(hObject,
       eventdata, handles)
53 switch get(hObject,'Value')
54     case 1
55         handles.A=handles.guitar;
56     case 2
57         handles.A=handles.organ;
58     case 3
59         handles.A=handles.basic;
60     case 4
61         handles.A=handles.custom;
62 end
63 guidata(hObject, handles);
64
65
66 % --- Executes during object creation,
       after setting all properties.
67 function ch_tone_CreateFcn(hObject,
       eventdata, handles)
68 if ispc && isequal(get(hObject,'
       BackgroundColor'), get(0,'
       defaultUicontrolBackgroundColor'))
69     set(hObject,'BackgroundColor','white');
70 end
71
72
73
74
75 % --- Executes on selection change in
       popupmenu2.
76 function popupmenu2_Callback(hObject,
```

```
    eventdata, handles)
function popupmenu2_CreateFcn(hObject,
    eventdata, handles)

if ispc && isequal(get(hObject,'
    BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function ch_tone_ButtonDownFcn(hObject,
    eventdata, handles)


function get_rhy_Callback(hObject,
    eventdata, handles)
 handles.tone=str2num(get(hObject,'String')
    );
guidata(hObject, handles);


% --- Executes during object creation,
    after setting all properties.
function get_rhy_CreateFcn(hObject,
    eventdata, handles)
if ispc && isequal(get(hObject,'
    BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


```

```
99  % --- Executes on button press in anay_file
        .
100 function anay_file_Callback(hObject,
        eventdata, handles)
101 [Filename,Pathname,~]=uigetfile('*.wav');
102 A=[];freq=[];time=[];
103 [~,~,A]=f_f(freq,time,[Pathname Filename],A
        );
104 handles.custom=repmat(A,1,25);
105 guidata(hObject, handles);
106
107
108 % --- Executes on button press in play.
109 function play_Callback(hObject, eventdata,
        handles)
110 switch get(handles.get_rhy,'Value')
111     case 1
112         handles.A=handles.guitar;
113     case 2
114         handles.A=handles.organ;
115     case 3
116         handles.A=handles.basic;
117     case 4
118         handles.A=handles.custom;
119 end
120
121 switch get(handles.get_demo,'Value')
122     case 1%用户输入
123         handles.tone=str2num(get(handles.
              get_rhy,'String'));
124         handles.time=str2num(get(handles.
              get_time,'String'));
125     case 2%东方红
```

```
126             handles.tone=[5 5 6 2 1 1 -6 2];
127             handles.time=[0.5 0.25 0.25 1 0.5
                    0.25 0.25 1 ];
128         case 3%茉莉花
129             handles.tone=[3 3 5 6 11 11 6 5 5 6
                     5 3 3 5 6 11 11 6 5 5 6 5 5 5 5
                     3 5 6 6 5 3 2 3 5 3 2 1 1 2 1];
130             handles.time=[0.5 0.25 0.25 0.25
                    0.25 0.25 0.25 0.5 0.25 0.25 1
                    0.5 0.25 0.25 0.25 0.25 0.25
                    0.25 0.5 0.25 0.25 1 0.5 0.5 0.5
                     0.25 0.25 0.5 0.5 1 0.5 0.25
                    0.25 0.5 0.25 0.25 0.5 0.25 0.25
                     1];
131         case 4%致爱丽丝
132             handles.tone=[13 12.5 13 12.5 13 7
                    12 11 6 -10 1 3 6 7 -10 3 11 7
                    6];
133             handles.time=[0.25 0.25 0.25 0.25
                    0.25 0.25 0.25 0.25 0.5 0.25
                    0.25 0.25 0.25 0.5 0.25 0.25
                    0.25 0.25 0.5];
134         end
135
136 msheet2tone=[-3 -4 -4.5 -5 -5.5 -6 -6.5 -7
        1 1.5 2 2.5 3 4 4.5 5 5.5 6 6.5 7 11
        11.5 12 12.5 13 14 14.5 15];
137 for i=1:length(handles.tone)
138     if handles.tone~=-10&&(handles.tone
            <(-1+get(handles.get_base,'Value')
            *2)||handles.tone>(11+get(handles.
            get_base,'Value')*2))
139             warning('wrong input');
```

```
140         end
141         [~,num]=find(msheet2tone==handles.tone(
                i));
142          if isempty(num)
143              num=-4;
144          end
145         handles.tone(i)=num;
146     end
147     switch get(handles.get_base,'Value')
148         case 2
149             if handles.tone~=-4
150                 handles.tone=handles.tone-3;
151             end
152             handles.A(:,(4:25))=handles.A
                    (:,(1:22));
153             handles.A(:,(1:3))=repmat(handles.A
                    (:,4),1,3);
154     end
155     s=[];
156     %220为基准音Hz
157     for i=1:length(handles.tone)
158         s=[s;m_note_guitar(handles.tone(i)-6,
                handles.time(i),220,8000,handles.A
                (:,abs(handles.tone(i))))];
159         %由于空音出现了索引为负数的情况，所以加上abs
160     end
161     sound(s,8000);
162
163     function get_time_Callback(hObject,
            eventdata, handles)
164     handles.time=str2num(get(hObject,'String'))
            ;
165     guidata(hObject, handles);
```

```
166
167 function get_time_CreateFcn(hObject,
        eventdata, handles)
168 if ispc && isequal(get(hObject,'
        BackgroundColor'), get(0,'
        defaultUicontrolBackgroundColor'))
169     set(hObject,'BackgroundColor','white');
170 end
171
172 function get_demo_Callback(hObject,
        eventdata, handles)
173 function get_demo_CreateFcn(hObject,
        eventdata, handles)
174 if ispc && isequal(get(hObject,'
        BackgroundColor'), get(0,'
        defaultUicontrolBackgroundColor'))
175     set(hObject,'BackgroundColor','white');
176 end
177
178
179 % --- Executes on selection change in
        get_base.
180 function get_base_Callback(hObject,
        eventdata, handles)
181 function get_base_CreateFcn(hObject,
        eventdata, handles)
182 if ispc && isequal(get(hObject,'
        BackgroundColor'), get(0,'
        defaultUicontrolBackgroundColor'))
183     set(hObject,'BackgroundColor','white');
184 end
```