信号与系统 音乐合成 实验报告

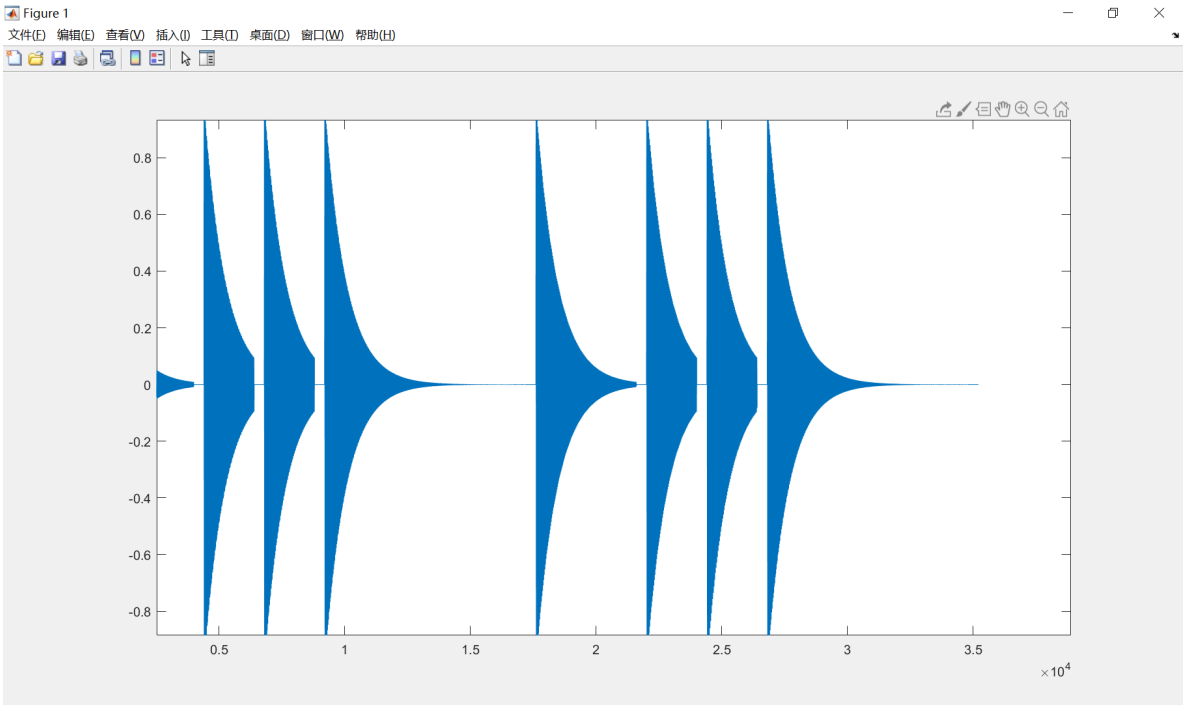
17341125 蒙亚愿 信息安全

Part1. 简单音乐合成

> (2) 你一定注意到 (1) 中的乐曲中相邻乐音之间由“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 1-1 表示的包络修正每个乐音，以保证在乐音的邻接处信号幅度为零（最简单的，也可以用指数衰减的包络来表示）。请画出两段音乐的部分片段波形，并简要比较。

最开始尝试指数衰减包络，但是感觉衰减效果不是很明显，仍然会有有些"pa"的声音，甚至一直都会有"dong"的声音，个人猜测是因为在每个音开始的时候的冲激导致的，于是改用了实验要求里的折线包络。

>指数衰减 $e^{-3\pi t_k}$



改用折现包络：

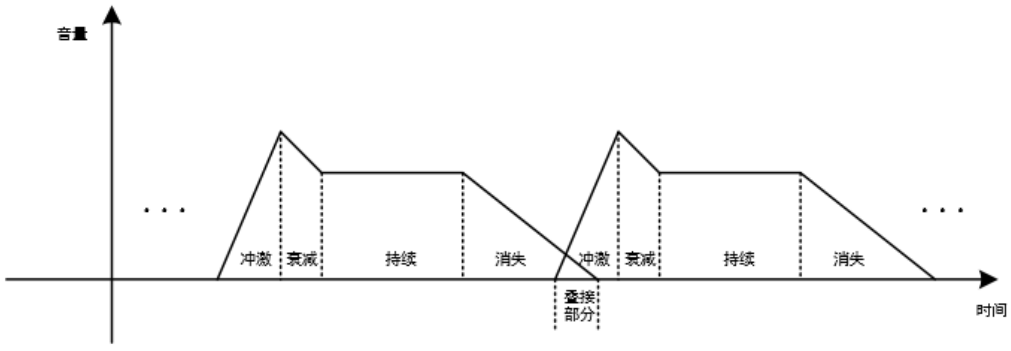
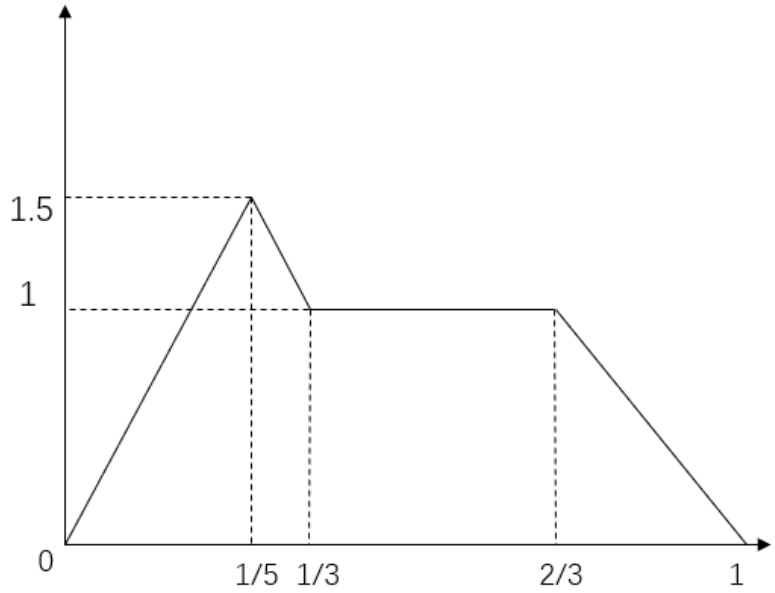


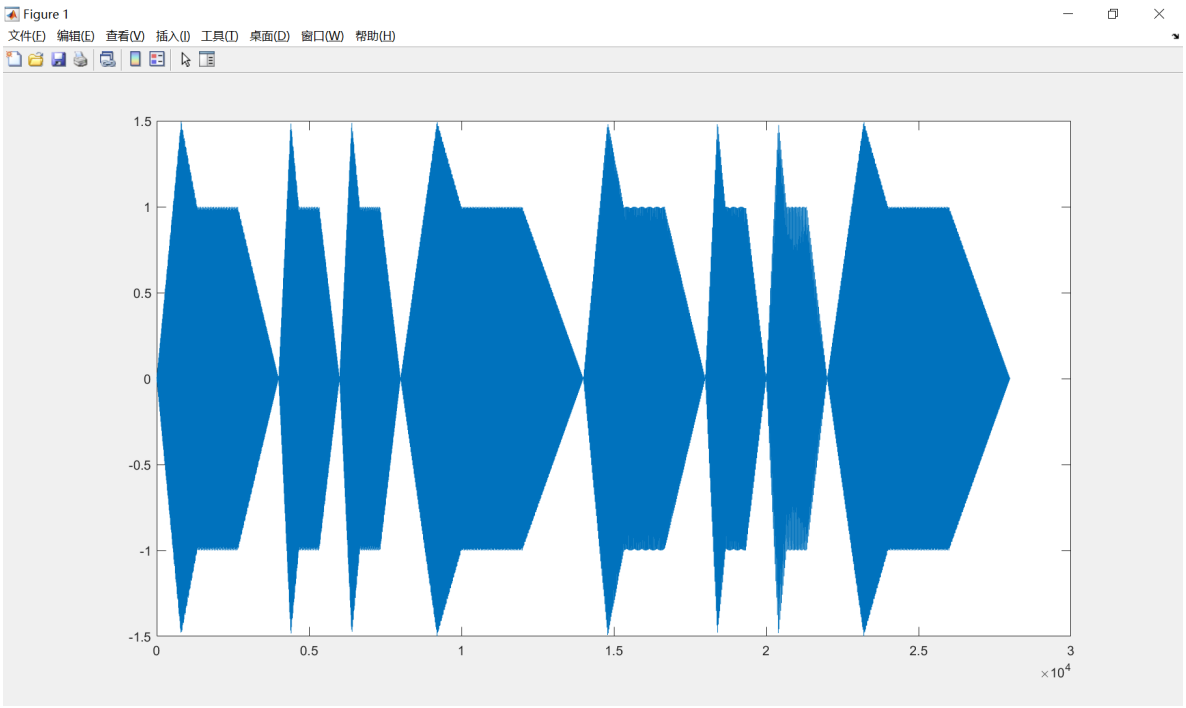
图 1-1 音量变化包络

从图中可以看出这个包络是由4个线段组成，因此只要确定了每段线段的断点，就可以根据端点写出直线方程的斜截式：

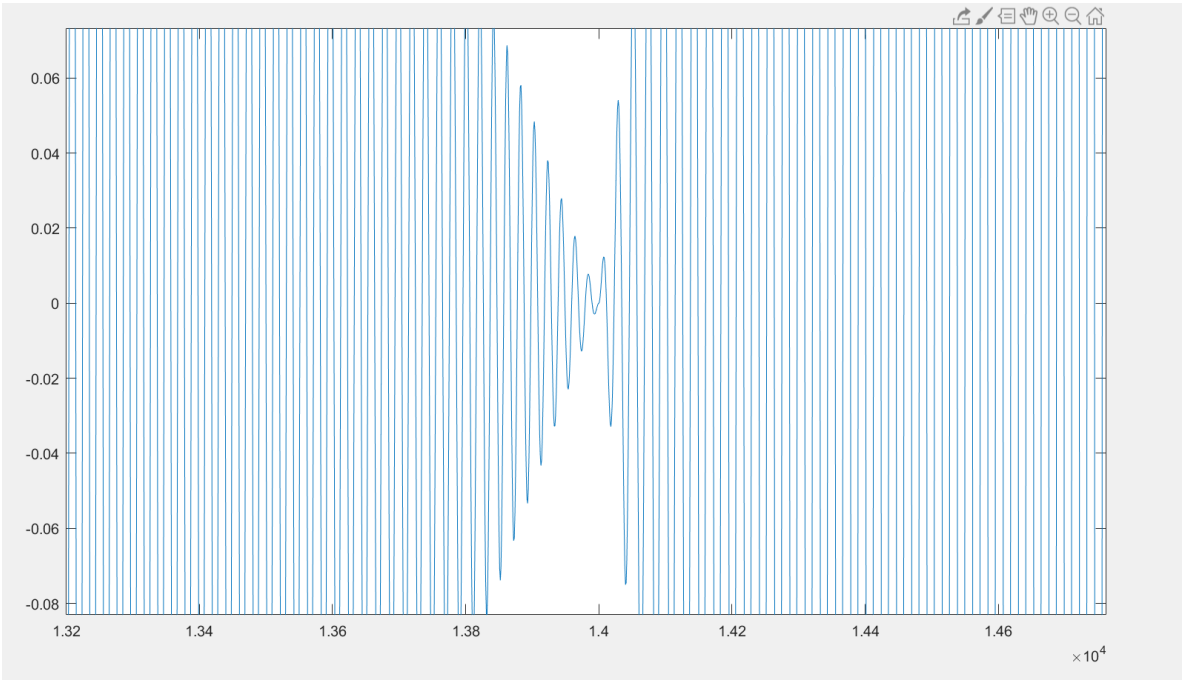



```
time = fs*beat ;     % 每个乐音对应的频率
N =length(time) ;    % 每段音乐的总的抽样点数
east = zeros(1,N) ;     %利用east向量来存储抽样点数
n=1 ;
for num = 1:N       %利用循环产生抽样，num表示乐音编号
    t = 1/fs:1/fs:(time(num))/fs ;      % 产生的第num个乐音的编号
    P = zeros(1,time(num)) ;    % P：存储包络数据的向量
    L = (time(num))*[0 1/5 333/1000 333/500 1] ;    % 包络线端点对应的横坐标

    T = [0 1.5 1 1 0] ;     %纵坐标
    s = 1 ;
    b = 1:1:time(num) ;     %产生包络线抽样点
    for k=1:4

        P(s:L(k+1)-1) = (T(k+1)-T(k))/(L(k+1)-L(k))*(b(s:L(k+1)-1)-
L(k+1)*ones(1,L(k+1)-s))+T(k+1)*ones(1,L(k+1)-s) ;       % 包络线直线方程式
        s = L(k+1) ;
    end
    east(n:n+time(num)-1) = sin(2*pi*note(num)*t).*P(1:time(num)) ;
    n = n+time(num) ;  %给第num个乐音加上包络
end
sound(east, 8000) ;
plot(east) ;
```

图像：



在两个调的交接处放大：可以看见前一个音调减小大0，接着下一个音调就从0开始递增，于是就没了"ding"和"pa"的声音了。



> (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：允许音乐播放的时间发生拉长/缩短）。再难些，考虑使用 resample 函数（也可以用 interp 和 decimate函数）将上述音乐升高半个音阶。改变音调后听听效果，并简述你实现以上两种变调的方法的基本原理。

升高一个八度就是每个乐音的频率都乘2，降低八度就是都除以2：

```
note = note*2 ;  %升高八度
note = note/2 ;  %降低八度
```

然后再包络。

> (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？注意谐波分量要比较小，否则掩盖住基音反而会使音调不准。（例如选择基波幅度 1，二次谐波幅度 0.2，三次谐波幅度 0.3)

在包络之后，加入谐波：

```
    m = [1 0.3 0.2] ;       %波形幅度矩阵
    ss = zeros(1, length(t)) ;
    for i=1:length(m)
        ss = ss+m(i)*sin(2*i*pi*note(num)*t) ;      %加谐波
    end
    east(n:n+time(num)-1) = sin(2*pi*note(num)*t).*P(1:time(num)) ;    % 给第num个乐音加上包络
    n = n+time(num) ;
```

但是感觉加入谐波之后，声音似乎什么变化，可能我耳朵不太好吧。

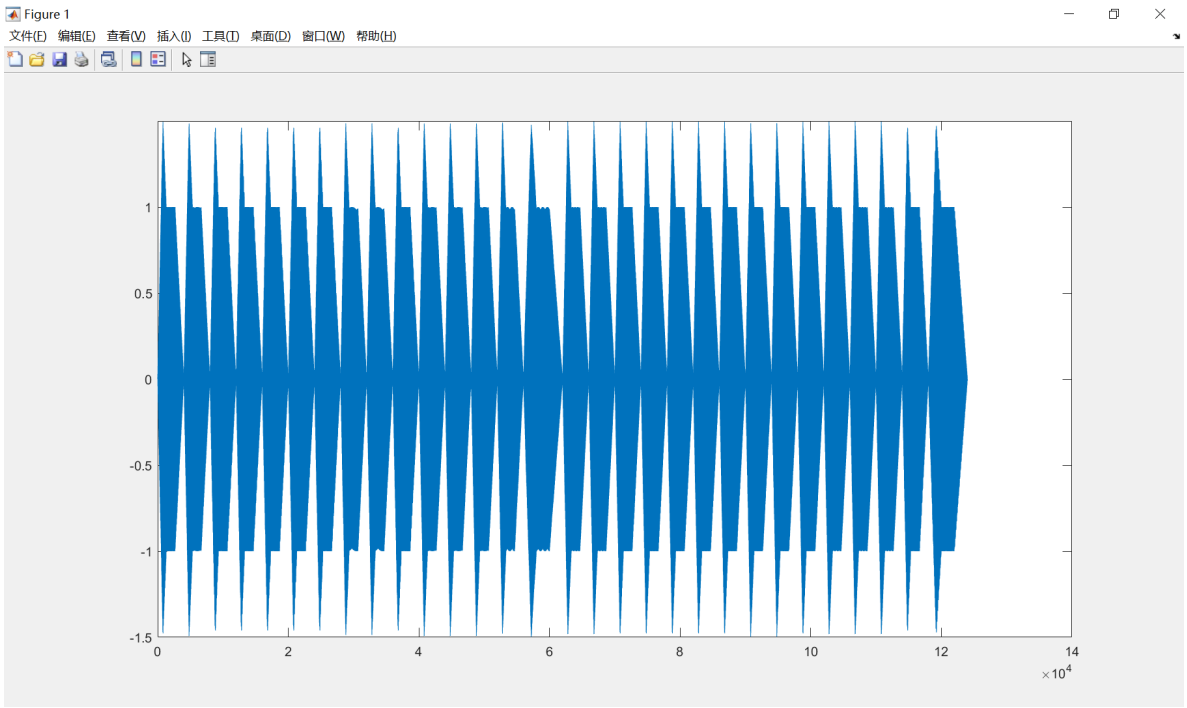
> (5) 自选其他你喜欢的音乐进行合成（选择 1-2 小节，时长控制在 10s 左右）。保存成 wav格式的音频文件（命名采用“歌曲名_1.wav”）。

上网查了有点甜的乐谱，就可以仿照东方红的写：

```
len = 37 ;
note = [f(1)*2,
f(2)*2,f(3)*2,f(3)*2,f(3)*2,f(3)*2,f(3)*2,f(5),f(5),f(3)*2,f(2)*2,f(2)*2,f(2)*2,
f(5)*2,f(5)*2,f(7),f(7),f(1)*2,f(1)*2,f(1)*2,f(1)*2,f(1)*2,f(3),f(3),f(1)*2,f(7)
,f(7),f(7),f(3)*2,f(3)*2] ;%,f(6),f(5),f(6),f(6),f(6),f(1)*2,f(7)] ;
beat = [2,2,2,2,2,2,2,2,2,2,2,2,2,2,3,2,2,2,2,2,2,2,2,2,2,2,2,2,3]
;%.2,2,2,2,2,2,3] ;
beat = beat/4 ;
note = note*0.9 ;

 time = fs*beat ;
 N =length(time) ;
 east = zeros(1,N) ;
 n=1 ;
 for num = 1:N
     t = 1/fs:1/fs:(time(num))/fs ;
     P = zeros(1,time(num)) ;
     L = (time(num))*[0 1/5 333/1000 333/500 1] ;

     T = [0 1.5 1 1 0] ;
     s = 1 ;
     b = 1:1:time(num) ;
     for k=1:4
         P(s:L(k+1)-1) = (T(k+1)-T(k))/(L(k+1)-L(k))*(b(s:L(k+1)-1)-
L(k+1)*ones(1,L(k+1)-s))+T(k+1)*ones(1,L(k+1)-s) ;
         s = L(k+1) ;
     end
     m = [1 0.3 0.2] ;
     ss = zeros(1, length(t)) ;
```

```
    for i=1:length(m)
        ss = ss+m(i)*sin(2*i*pi*note(num)*t) ;
    end
    east(n:n+time(num)-1) = sin(2*pi*note(num)*t).*P(1:time(num)) ;
    n = n+time(num) ;
end
sound(east, 8000) ;
plot(east) ;

fname = '有点甜.wav';          % 设定文件名称 注意格式
audiowrite(fname, east, fs);      % 输出文件
```

波形：



Part2.用傅里叶级数分析音乐

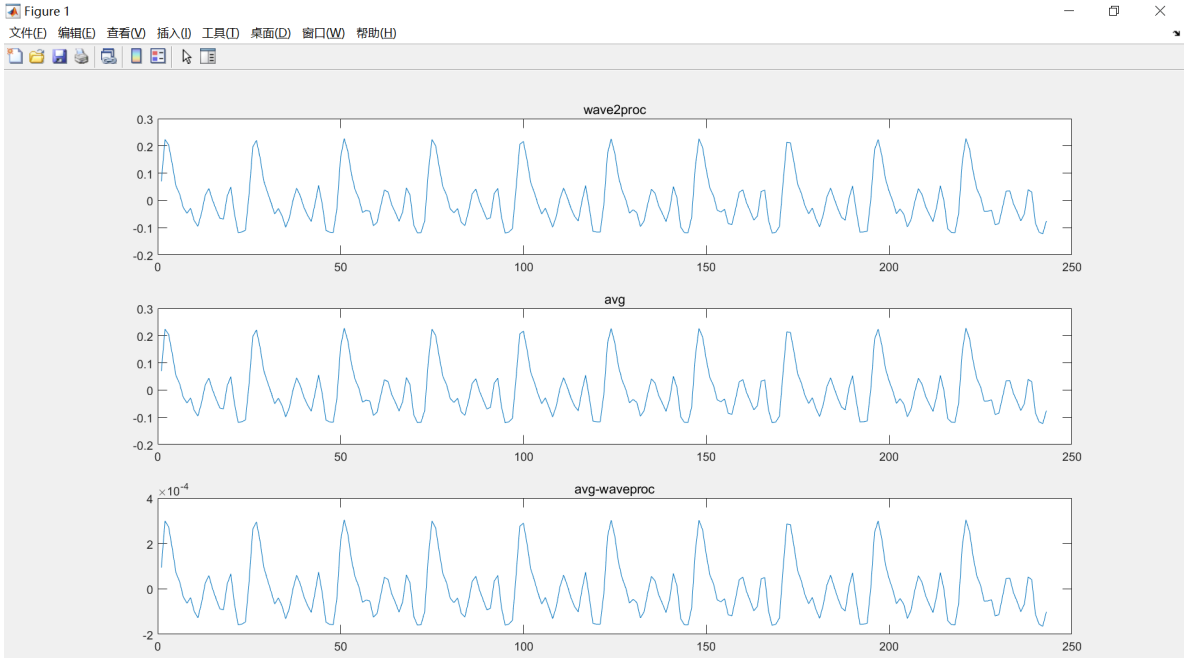
> (2) 画出 realwave 和 wave2proc 的波形并作对比，分析理论值和真实值都有哪些差异？请设计一个预处理过程，用于去除真实乐曲中的非线性谐波和噪声，并 简述你采用方法的基本原理和处理后的效果。
> □ 提示：可以考虑直接从时域做，采用一定方法让 realwave 的周期性更强并一定程度上消除噪声干扰，使其更接近 wave2proc。

非线性杂波和噪声都是随机产生的，不具有周期性，所以要在周期性的乐音中将其去除，可以把真实乐曲多次叠加再取平均值。理论上，叠加次数越多，最后的得到的平均后的乐音越具有周期性。

读出$realwave$波形，发现采样点为243，重复10次，可得周期为24.3，所以可以将其延长至10倍，于是周期就是整数。

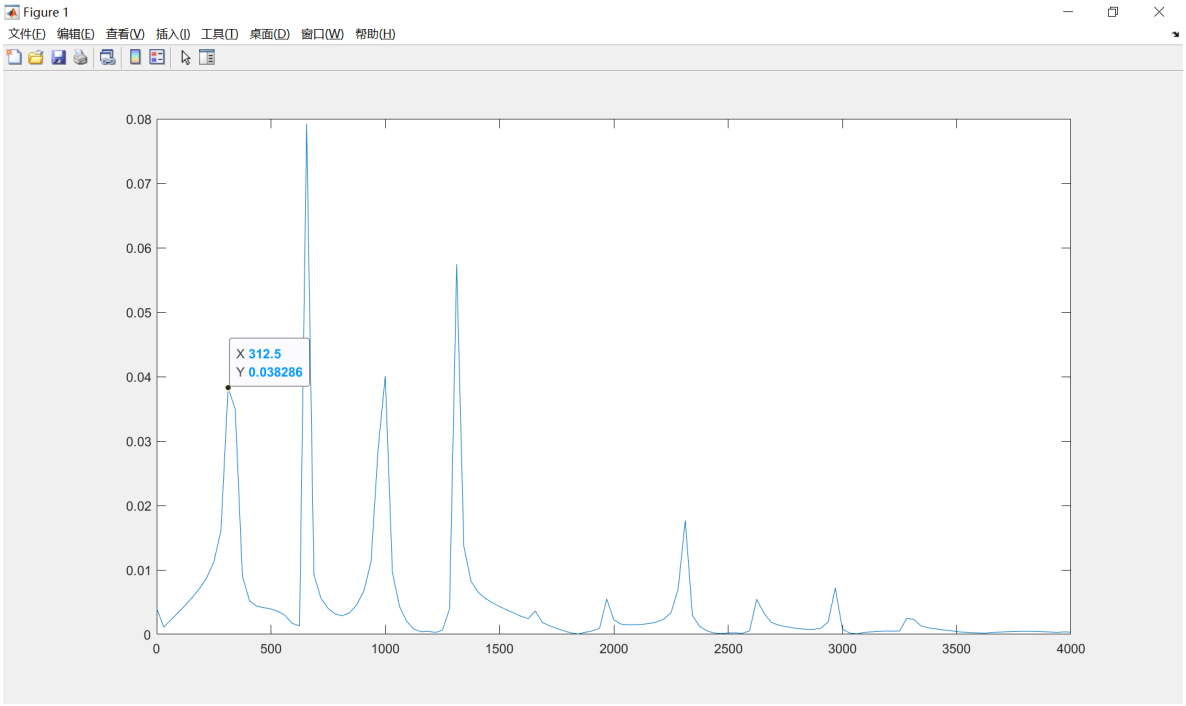
```
load('Guitar.mat') ;
len= length(realwave) ;
add = resample(realwave, 10,1) ;
avg = zeros(1,len) ;
for i= 1:10
    avg = avg + (add((i-1)*len+1:i*len))' ;
end
add = [avg, avg, avg, avg, avg, avg, avg, avg, avg, avg] ;
avg = resample(add/10, 1, 10) ;
figure ;
subplot(3,1,1) ;  plot(wave2proc) ; title('wave2proc') ;
subplot(3,1,2) ;  plot(avg) ;  title('avg') ;
subplot(3,1,3) ;  plot(avg-wave2proc') ;  title('avg-waveproc') ;
```

波形：虽然乍一看$wave2proc$和avg波形没有去区别，但是二者相减还是有细微的差异的。



> (3) 这段音乐（指 wave2proc）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？）但你可能发现无论如何提高频域的分辨率，也得不到精确的包络（我们希望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序的数据量，即再把时序信号重复若干次，看看这样效果是否好多了。对比前后两种方法的频谱图差异，并尝试解释之。

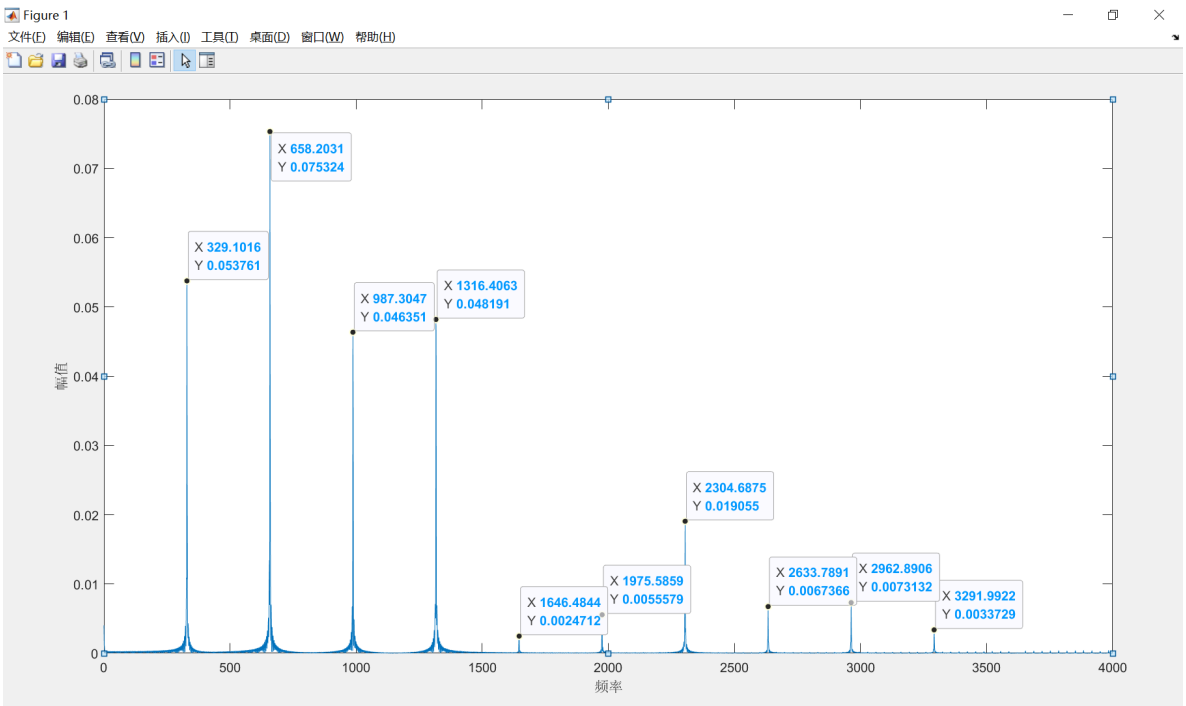
为了分析$wave2proc$的基波和谐波，可以对$wave2proc$进行傅里叶变换，得到的$wave2proc$的幅值谱，在频谱图上的第一个突出的波峰对应的频率即为$wave2proc$的基频。



但是从图上可以看出包络并不明显。因为原来的$wave2proc$只有243个数据点，并不能明显的体现出其周期性，所以它的幅值谱的离散化程度并不高，要体现出$wave2proc$的周期性，即让$wave2proc$在时域重复多次再进行傅里叶变换。

用FFT对该信号做DFT：

```
load 'Guitar.mat'
wave2proc = repmat(wave2proc, 20, 1) ;
NFFT = 2^nextpow2(length(wave2proc)) ;
Y = fft(wave2proc, NFFT)/ length(wave2proc) ;
g = fs/2*linspace(0, 1, NFFT/2+1) ;
plot(g, 2*abs(Y(1:NFFT/2+1)))
```



此时，幅值谱的离散化程度很高了，由图可知，$wav2proc$的基频为329.1016，幅值为0.052761。

谐波	2	3	4	5	6	7	8	9
频率	658.2031	987.3047	1316.4063	1646.4844	2304.6875	2633.7891	2962.8906	3291.9922
幅值	0.075324	0.046351	0.048191	0.0024712	0.019055	0.0067366	0.0073132	0.0033729

> (4) 再次载入 fmt.wav，现在要求编写一段程序，自动分析出这段乐曲的音调（和节拍）。如果觉得太难，允许手工标定出每个音调的起始时间，还可以把每个音调的数据都单独保存成一个文件，然后让 MATLAB 对这些文件进行批量处理，分析出每个音调的傅里叶级数（或者说谐波分量）。请简述你采用的分析方法的原理，出各个音调的分析结果（可以以表格呈现）。

因为幅值谱上极大值点的幅值足够大才能将其定位基频，因此在分析了几个音调之后发现基频处的幅值大多在0.025以上，有几个比0.025小于是可以规定基频处的限定条件是大于0.025的，如果所有的都小于0.02，那再以0.015作为幅值条件继续寻找，于是就能确定剩下的音调频率。

在程序找到了基频之后，再有基频去获取高次谐波的幅值时需要一定的容错能力，可以确定其误差为$\mp 1Hz$，在这个区间内寻找极大值点就是k次谐波的对应点。

```
function [y1, y2] = analysis(w,a)
    fs = a ;          % 傅里叶变换的抽样频率
    y1 = zeros(1,7) ;    %求7最大7次谐波
    y2 = zeros(1,7) ;
    NFFT = 2^nextpow2(length(w)) ;
    Y = fft(w,NFFT)/length(w) ;
    g = fs/2*linspace(0,1,NFFT/2+1) ;
    p = 2*abs(Y(1:NFFT/2+1)) ;
    plot(g,p) ;

    d = floor(NFFT/fs) ;    %把误差1Hz化成对应的点数
    for k=2:length(p) - 1
        if( p(k)>0.02 && p(k)>p(k-1) && p(k)>p(k+1) )
            y1(1) = g(k) ;  %存储基频
            y2(1) = p(k) ;  %存储几波幅值
            break ;
        elseif( p(k)>0.015 && p(k)>p(k-1) && p(k)>p(k+1) )  %如果没有找到，就把幅值限制改为0.015 继续寻找
            y1(1) = g(k) ;
            y2(1) = p(k) ;
        end
    end

    for t=2:7
        for i = t*(k-d):t*(k+d)      %误差允许的范围内寻找t次谐波点
            if( p(k)>0.02 && p(i)>0.05*p(k) && p(i)>p(i-1) && p(i)>p(i+1) )
                y2(t) = p(i)/y2(1) ;     %谐波幅值归一化
                y1(t) = g(i) ;
                break
            elseif( p(k)>0.015 && p(k)>0.05*p(k) && p(i)>p(i-1) && p(i)>p(i+1) )
                y2(t) = p(i)/y2(1) ;
                y1(t) = g(i) ;
            end
        end
    end
```

```
end
```

在cool Ediit中手动标记音调交界点，得到了$time$向量：

```
time = floor([0.096 0.267 1.767 2.234 2.706 3.146 3.606 4.056 4.520 5.030 5.749
5.978 7.015 7.709 7.923 8.028 8.490 8.959 9.454 9.852 10.125 10.356 10.565
10.822 11.292 11.741 12.284 12.741 13.269 13.758 14.315 14.939 15.432]/16.384*N)
;
```

其中，16.384是fmt.wav的总长度，N是数据点的总数。把时间转换成对应的数据点数。

```
wave = audioread('fmt.wav') ;
N = length(wave) ;
% 节点向量
time = floor([0.096 0.267 1.767 2.234 2.706 3.146 3.606 4.056 4.520 5.030 5.749
5.978 7.015 7.709 7.923 8.028 8.490 8.959 9.454 9.852 10.125 10.356 10.565
10.822 11.292 11.741 12.284 12.741 13.269 13.758 14.315 14.939 15.432]/16.384*N)
;
fs = N/16.384 ;      %确定数据的抽样抽样频率
n = length(time) ;
for k=1:n
    if k==1
        temp = wave([1:time(k)-1] );
    else
        temp = wave(time(k-1):time(k)-1) ;  %将第k个音调数据存入temp矩阵
    end

    temp = repmat(temp, 1000, 1) ;       % 将每个音调的处理结果分别保存  F：频率  U：幅
值
    [F(k,1:7) U(k,1:7)] = analysis(temp, fs) ;
end
```

可在工作区中查看变量U和F的取值。

实验总结

1. 这次实验的第一部分很简单，看了一会B站的教程就知道怎么搞，但是不知道为什么B站的方法和实验中给的例子方法一样，但是人家就没有"pa"的声音，但是实验例子中就有，做完之后声音超不自然，一直dongdongdongdong的响，于是参考了一下别人的代码，才想起来可以用老师给的包络的线段，于是声音果然动听自然了很多，还能自己写首歌，matlab真是厉害👍。
2. 看着还有时间就尝试做了第二部分，但是真的真的好难好难好难，刚开始还知道咋回事，越往后越看不明白，看了别人的代码和讲解才略知一二，学到精妙之处还忍不住拍手称赞，但是到了第三问就觉得莫名其妙，磕磕绊绊的学了一会，却不深知其法，只恍惚感觉之后的时间会让我成长，现在只是第一步，害，于是就到这了。
3. 总而言之，这次的实验很精妙，有一定的趣味之后，却给人以一种退却之感，只能自责自己学的不精，但是老实说，老师课上讲的还是很精妙的！平时将作业也能将我想的很复杂的地方以简单的方法写出，TA师兄也很认真负责！刚开始没有好好写作业被发现后，每次作业都认真做了，不过不得其法，但还是很钦佩老师和TA的敬业与认真，信号与系统是一门神奇的课程，选它不后悔！最后感谢你们这一学期在学业上对我们的帮助，祝老师和TA师兄新年快乐！