MATLAB 综合实验之语音合成*

聂浩　无 31　2013011280

2015 年 8 月 16 日

1　语音预测模型

1.1　给定 $e(n) = s(n) - a_1 s(n-1) - a2_s(n-2)$ 假设 e(n) 是输入信号，s(n) 是输出信号，上述滤波器的传递函数是什么？如果 $a_1 = 1.3789, a_2 = -0.9506$ ，上述合成模型的共振峰频率是多少？用 zplane ，freqz ，impz 分别绘出零极点图，频率响应和单位样值响应。用 filter 绘出单位样值响应，比较和 impz 的是否相同。

答：传递函数是

$$V(s) = \frac{1}{1 - a_1 z^{-1} - a_2 z^{-2}} = \frac{1}{1 - 1.3789 z^{-1} + 0.9506 z^{-2}}$$

共振峰频率为

$$f = \frac{\Omega}{2\pi T} = 999.947 Hz$$

.

zplane 绘制零极点图如图1,freqz 绘制频率响应图如图2,impz 和 filter 绘制单位样值响应如图3. 从图中可以看出，impz 和 filter 绘制的图像是一致的。

代码如下 (a21.m):

```
clear;clc;close all;
a1=1.3789;a2=-0.9506;
b=1;
n=0:99;
a=[1 -a1 -a2];
zplane(b,a);
title('pole/zero')
figure
freqz(b,a);
figure
impz(b,a,100);
```

*所有的.m 文件均采用 utf8 编码，windows 版 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

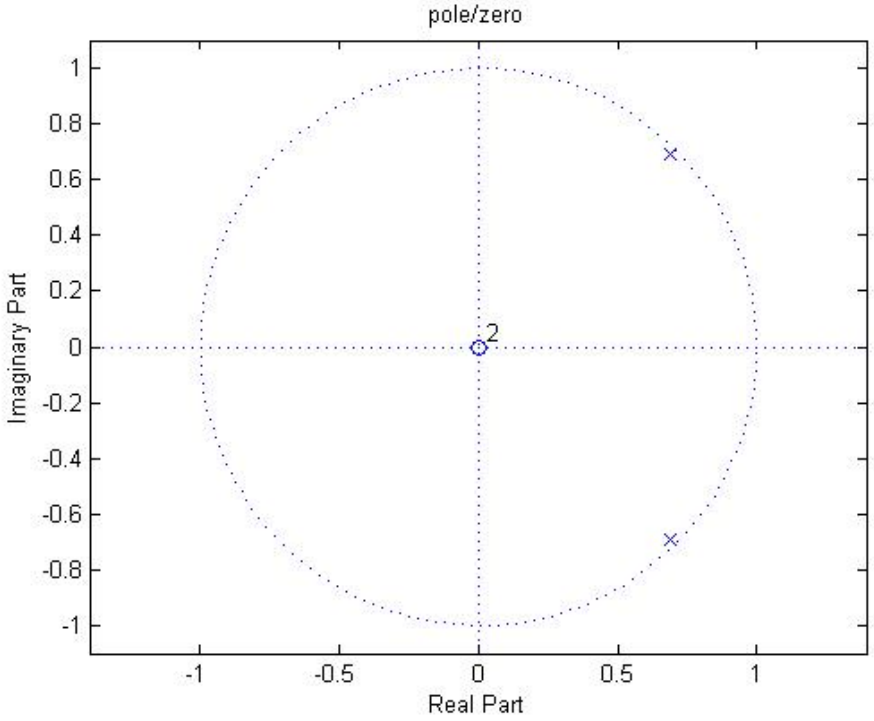

图 1: 零极点图

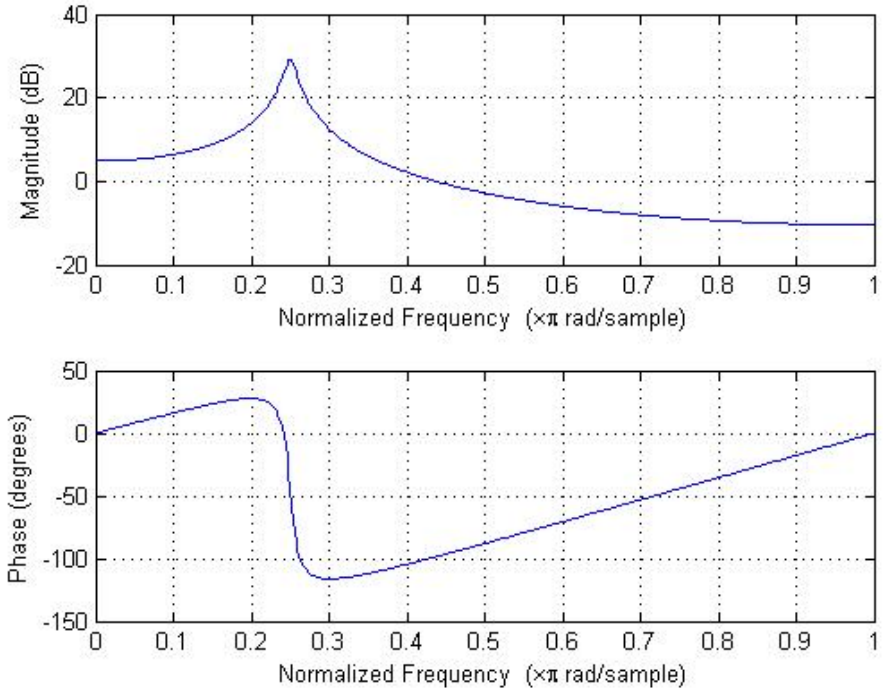

图 2: 频率响应图

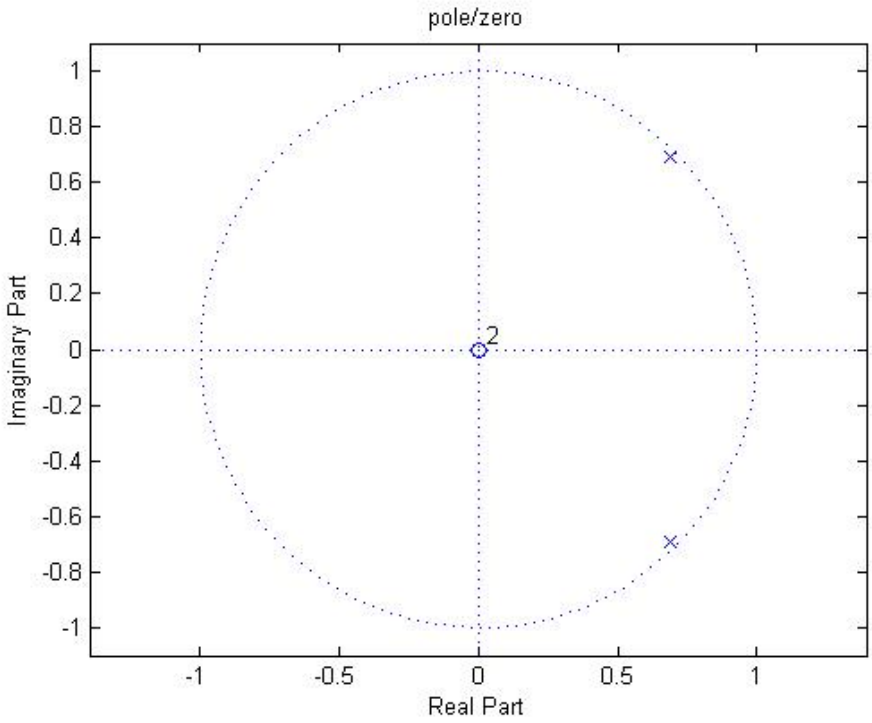

图 3: 单位样值响应图

```
h=filter(b,a,(n==0));
hold on;
plot(n,h,'r*');
legend('impz','filter');
[z,p,~]=tf2zp(b,a);
f=angle(p)/(2*pi/8000)
```

1.2 阅读 speechproc.m 程序，理解基本流程。程序中已经完成了语音分帧、加窗、线性预测、和基音周期提取等功能。注意：不要求掌握线性预测和基音周期提取的算法原理。

以 8KHz 的采样频率，10ms(80 个点) 为一段，对从第三段起的每一段用 haming 窗加窗后进行处理。

1.3 运行该程序到 27 帧时停住，用（1）中的方法观察零极点图。

零极点图如图4

代码如下:

```
if n == 27
```

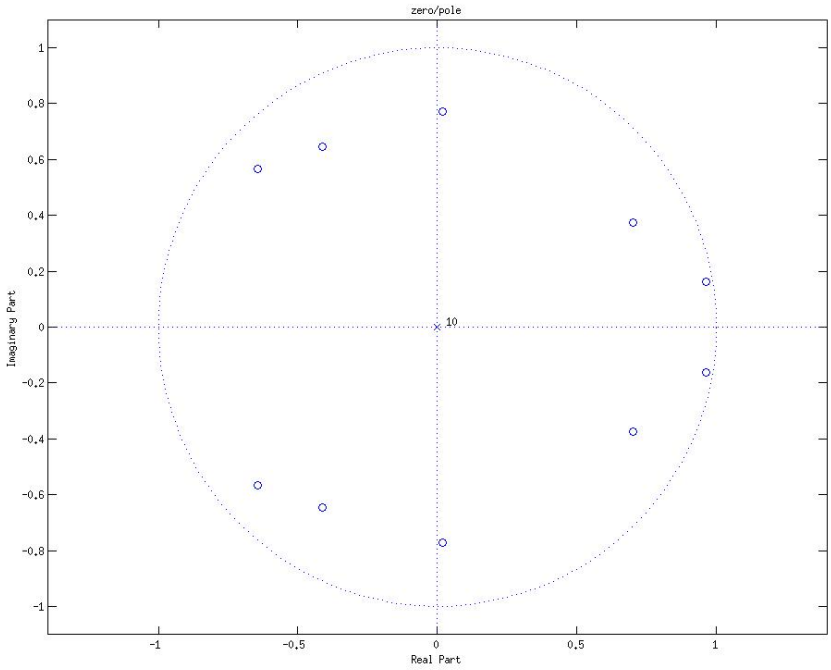

图 4: 单位样值响应图

```
% (3) 在此位置写程序，观察预测系统的零极点图
zplane(A)
title('zero/pole');
end
```

1.4 在循环中添加程序：对每帧语音信号 s(n) 和预测模型系数 $\{a_i\}$，用 filter 计算激励信号 e(n) 。注意：在系数变化的情况下连续滤波，需维持滤波器的状态不变，要利用 filter 的 zi 和 zf 参数。

代码如下:

```
s_f = s((n-1)*FL+1:n*FL);
% 本帧语音，下面就要对它做处理
% (4) 在此位置写程序，用filter函数s_f计算激励，注意保持滤波器状态
% exc((n-1)*FL+1:n*FL) = ...将你计算得到的激励写在这里
[exc((n-1)*FL+1:n*FL),zi_pre]=filter(A,1,s_f,zi_pre);
```

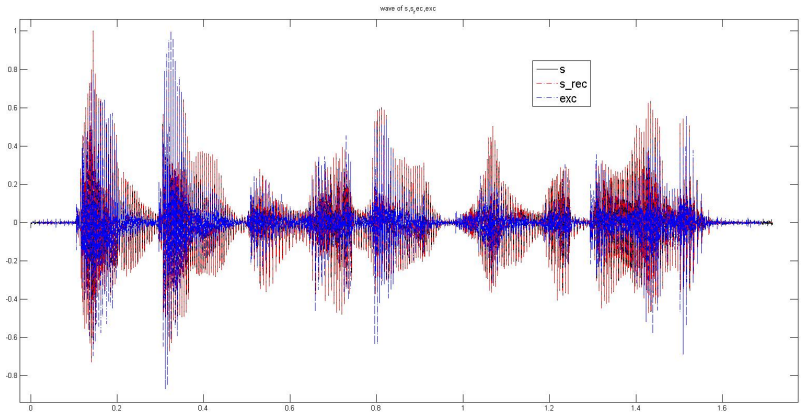

图 5: 波形图

1.5 完善 speechproc.m 程序，在循环中添加程序：用你计算得到的激励信号 e(n) 和预测模型系数 $\{a_i\}$，用 filter 计算重建语音 $\hat{s}(n)$ 。同样要注意维持滤波器的状态不变。

这里需要反过来用滤波器，代码如下:

```
% (5) 在此位置写程序，用filter函数
%和exc重建语音，注意保持滤波器状态
[s_rec((n-1)*FL+1:n*FL),zi_rec]=filter(1,A,exc((n-1)*FL+1:n*FL),
    zi_rec);
% s_rec((n-1)*FL+1:n*FL) =...将你计算得到的重建语音写在这里
```

1.6 在循环结束后添加程序：用 sound 试听（6）中的 e(n) 信号，比较和 s(n) 以及 $\hat{s}(n)$ 信号有何区别。对比画出三个信号，选择一小段，看看有何区别

答: s(n) 和 $\hat{s}(n)$ 无法从听力上区分，e(n) 听起来声音比较嘈杂，这是 e(n) 中的随机噪声所产生的影响。[1]。但仍能从 e(n) 中分辨出语音内容，可见 e(n) 中还是保留了不少 s(n) 的信息。

从波形图图5可以看出，除了开始和结束两个小节，s(n) 和 $\hat{s}(n)$ 是一致的，而 e(n) 比 s(n) 要差不少且波形并不一致。从 0.2 到 0.3s 的局部波形图6也可以得出这样的结论。

[1] $e(n) = s(n) - \sum_{k=1}^{N} a_k s(n-k)$ 为残差，由于差分的影响，e(n) 与 s(n) 之间的差是不断变化的

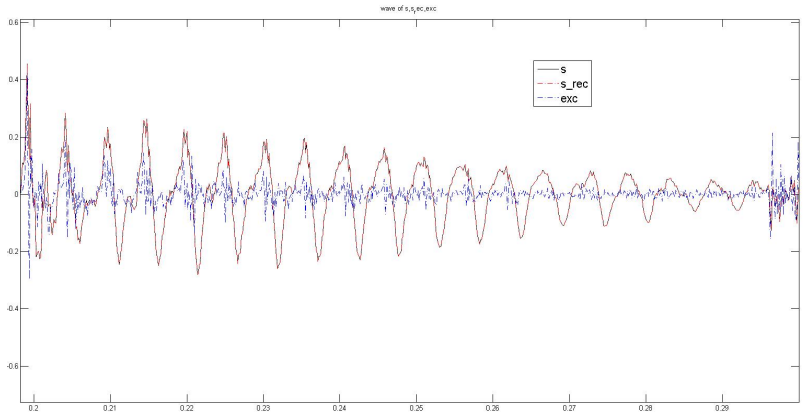

图 6: 0.2 到 0.3s 波形图

2 语音合成模型

2.7 生成一个 8kHz 抽样持续 1 秒钟的数字信号，该信号是一个频率为 200Hz 的单位样值“串”，即 $x(n)=\sum_{i=0}^{NS-1}\delta(n-iN)$ 考虑该信号的 N 和 NS 分别为何值？用 sound 试听这个声音信号。再生成一个 300Hz 的单位样值“串”并试听，有何区别？事实上，这个信号将是后面要用到的以基音为周期的人工激励信号 e(n) 。

答：NS 应当为 200，N 应为 40。300Hz 的音听起来音调要高一些。和之前音乐合成的正弦波相比，这两个音要更尖锐一些。

代码如下 (g200.m):

```
n=0:7999;
f1=200;
f2=300;
x1=double(mod(n,round(8000/f1))==0);
x2=double(mod(n,round(8000/f2))==0);

sound(x1,8000);
pause(1);
sound(x2,8000);
```

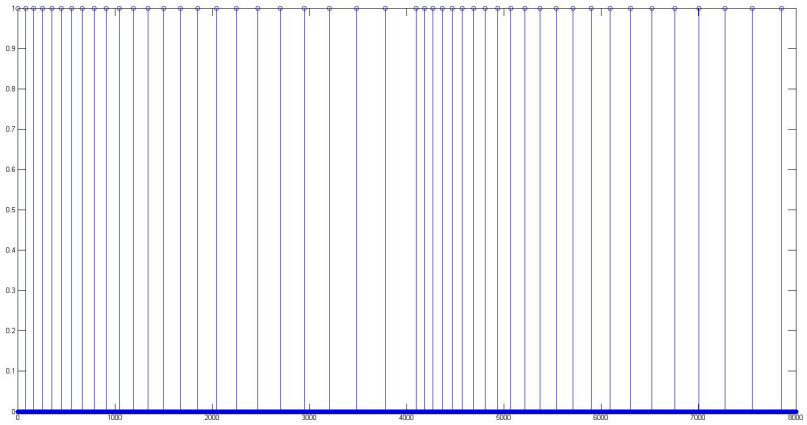

图 7: 变周期 e(n)

2.8 真实语音信号的基音周期总是随着时间变化的。我们首先将信号分成若干个 10 毫秒长的段，假设每个段内基音周期固定不变，但段和段之间则不同，具体为 $PT = 80 + 5mod(m, 50)$ 其中 PT 表示基音周期，m 表示段序号。生成 1 秒钟的上述信号并试听。（提示：用循环逐段实现，控制相邻两个脉冲的间隔为其中某个脉冲所在段的 PT 值。）

生成的波形如图7，听起来像是撕布声。

代码如下:

```
clear;close all;clc;
x=zeros(1,8000);
i=1;
while i<8000
    x(i)=1;
    i=i+ 80 + 5*mod(fix(i/80),50);
end
stem(x);
sound(x,8000);
```

2.9 用 filter 将（8）中的激励信号 e(n) 输入到（1）的系统中计算输出 s(n)，试听和 e(n) 有何区别。

这里听起来就像是给 8 中的 e(n) 信号加了一些混响，有一种在 e(n) 在盒子中发声的感觉，波形如图8。

代码如下

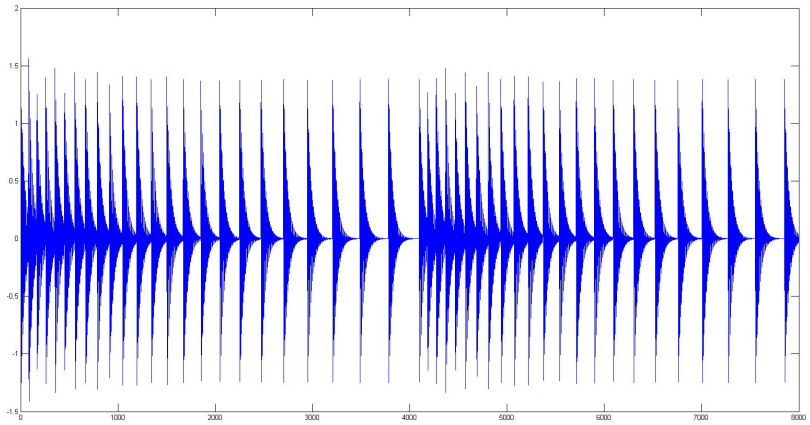

图 8: 处理生成的 s(n)

```
clear;close all;clc;
x=zeros(1,8000);
i=1;
while i<8000
    x(i)=1;
    i=i+ 80 + 5*mod(fix(i/80),50);
end
b=1;a1=1.3789;a2=-0.9506;
a=[1 -a1 -a2];
s=filter(b,a,x);
plot(s);
sound(s,8000);
```

2.10 重改 speechproc.m 程序。利用每一帧已经计算得到的基音周期和（8）的方法，生成合成激励信号 Gx(n)（G 是增益），用 filter 函数将 Gx(n) 送入合成滤波器得到合成语音 $\tilde{s}(n)$ 。试听和原始语音有何差别。

这里的的 $\tilde{s}(n)$ 听起来在每两个字之间有一种模糊的感觉，就像风吹在麦克风上一样。这应该是基频在这些时刻快速改变造成的。

代码如下:

```
% (10) 在此位置写程序，生成合成激励，并用激励和 filter 函数产生合成语音
% exc_syn((n-1)*FL+1:n*FL) = ... 将你计算得到的合成激励写在这里
tmp=(n-1)*FL+1:n*FL;
exc_syn((n-1)*FL+1:n*FL)= G*(mod(tmp,PT)==0);
```

```
% s_syn((n-1)*FL+1:n*FL) = ...     将你计算得到的合成语音写在这里
[s_syn((n-1)*FL+1:n*FL),zi_syn]=filter(1,A,exc_syn((n-1)*FL+1:n*FL),
    zi_syn);
```

在最前部还应该加上 zi_syn 的定义

```
zi_syn = zeros(P,1);
```

3 变速不变调

3.11 仿照（10）重改 speechproc.m 程序，只不过将（10）中合成激励的长度增加一倍，即原来 10 毫秒的一帧变成了 20 毫秒一帧，再用同样的方法合成出语音来，如果你用原始抽样速度进行播放，就会听到慢了一倍的语音，但是音调基本没有变化。

这样的到的语音长为原来的两倍，但是音调并未变化，这与直接慢放（resample 增加采样点）的结果完全不一样。代码如下：

```
% (11) 不改变基音周期和预测系数，将合成激励的长度增加一倍，再作为
    filter
% 的输入得到新的合成语音，听一听是不是速度变慢了，但音调没有变。
tmp_syn_v=(n-1)*FL_v+1:n*FL_v;
% exc_syn_v((n-1)*FL_v+1:n*FL_v) = ...
    将你计算得到的加长合成激励写在这里
exc_syn_v((n-1)*FL_v+1:n*FL_v)= G*(mod(tmp_syn_v,PT)==0);
% s_syn_v((n-1)*FL_v+1:n*FL_v) = ...
    将你计算得到的加长合成语音写在这里
[s_syn_v((n-1)*FL_v+1:n*FL_v),zi_syn_v]=filter(1,A,exc_syn_v((n-1)*
    FL_v+1:n*FL_v),zi_syn_v);
```

在最前部还应该加上 FL_v 和 zi_syn_v 的定义

```
FL_v=2*FL;
zi_syn_v = zeros(P,1);
```

4 变调不变速

4.12 重新考察（1）中的系统，将其共振峰频率提高 150Hz 后的 a1 和 a2 分别是多少？

$$a_1 = 1.2073, a_2 = -0.9506$$

代码如下：

```
%# -*- coding: utf8 -*-
clear;clc;close all;
a1=1.3789;a2=-0.9506;
b=1;
n=0:99;
a=[1 -a1 -a2];
[z,p,k]=tf2zp(b,a);
%角度为正频率加155
%为负减155
p=p.*exp(2*pi/8000*150*  (2* (angle(p)>0) -1) *1j);
[b,a]=zp2tf(z,p,k)
```

4.13 仿照（10）重改 speechproc.m 程序，但要将基音周期减小一半，将所有的共振峰频率都增加 150Hz ，重新合成语音，听听是何感受。

听起来调子高了不少，但音频时长并未改变。然而并不太像真正的女声，有很强的假声的感。

代码如下:

```
% (13) 将基音周期减小一半，将共振峰频率增加150Hz，重新合成语音，
    听听是啥感受～
PT=fix(PT/2);
[z,p,k]=tf2zp(1,A);
%角度为正频率加150
%为负减150
p=abs(p).*exp(( angle(p)+2*pi/8000*150*(2* (angle(p)>0) -1) )*1j);
[B,A]=zp2tf(z,p,k);
tmp_t=(n-1)*FL+1:n*FL;
% exc_syn_t((n-1)*FL+1:n*FL) = ... 将你计算得到的变调合成激励写在这里
exc_syn_t((n-1)*FL+1:n*FL)= G*(mod(tmp_t,PT)==0);
% s_syn_t((n-1)*FL+1:n*FL) = ...    将你计算得到的变调合成语音写在这里
[s_syn_t((n-1)*FL+1:n*FL),zi_syn_t]=filter(B,A,exc_syn_t((n-1)*FL+1:n
    *FL),zi_syn_t);
```

在最前部还应该加上 zi_syn_t 的定义

```
zi_syn_t = zeros(P,1);
```

5 总结

本次试验更多的是利用已有的数学模型在给定框架下进行扩展，与音乐合成实验相比发散性更少一些，最终的实现也较为单一。感觉本次实验最大的收获是对于语音合成、语音处理的进一步理解，是对于 matlab 实际应用的一种练习.

6 speechproc.m 完成版

```
%# -*- coding:utf8 -*-
function speechproc()

    % 定义常数
    FL = 80;                    % 帧长
    FL_v=2*FL;
    WL = 240;                   % 窗长
    P = 10;                     % 预测系数个数
    s = readspeech('voice.pcm',100000);             % 载入语音s
    L = length(s);              % 读入语音长度
    FN = floor(L/FL)-2;         % 计算帧数
    % 预测和重建滤波器
    exc = zeros(L,1);           % 激励信号（预测误差）
    zi_pre = zeros(P,1);        % 预测滤波器的状态
    s_rec = zeros(L,1);         % 重建语音
    zi_rec = zeros(P,1);
    % 合成滤波器
    exc_syn = zeros(L,1);       % 合成的激励信号（脉冲串）
    s_syn = zeros(L,1);         % 合成语音
    zi_syn = zeros(P,1);
    % 变调不变速滤波器
    exc_syn_t = zeros(L,1);     % 合成的激励信号（脉冲串）
    s_syn_t = zeros(L,1);       %
    zi_syn_t = zeros(P,1);
    % 变速不变调滤波器（假设速度减慢一倍）
    exc_syn_v = zeros(2*L,1);   % 合成的激励信号（脉冲串）
    s_syn_v = zeros(2*L,1);     % 合成语音
    zi_syn_v = zeros(P,1);
    hw = hamming(WL);           % 汉明窗

    % 依次处理每帧语音
```

```
for n = 3:FN

    % 计算预测系数（不需要掌握）
    s_w = s(n*FL-WL+1:n*FL).*hw;      %汉明窗加权后的语音
    [A,E] = lpc(s_w, P);              %用线性预测法计算P个预测系数
                                      % A是预测系数，E
                                          会被用来计算合成激励的能量

    if n == 27
    % (3) 在此位置写程序，观察预测系统的零极点图
        zplane(A)
        title('zero/pole');
    end

    s_f = s((n-1)*FL+1:n*FL);         % 本帧语音，
        下面就要对它做处理

    % (4) 在此位置写程序，用filter函数s_f计算激励，
        注意保持滤波器状态
    % exc((n-1)*FL+1:n*FL) = ... 将你计算得到的激励写在这里
    [exc((n-1)*FL+1:n*FL),zi_pre]=filter(A,1,s_f,zi_pre);

    % (5) 在此位置写程序，用filter函数和exc重建语音，
        注意保持滤波器状态

    [s_rec((n-1)*FL+1:n*FL),zi_rec]=filter(1,A,exc((n-1)*FL+1:n*
        FL),zi_rec);
    % s_rec((n-1)*FL+1:n*FL) = ... 将你计算得到的重建语音写在这里

    % 注意下面只有在得到exc后才会计算正确
    s_Pitch = exc(n*FL-222:n*FL);
    PT = findpitch(s_Pitch);      % 计算基音周期PT（不要求掌握）
    G = sqrt(E*PT);               % 计算合成激励的能量G（不要求掌握）


    % (10) 在此位置写程序，生成合成激励，并用激励和filter
        函数产生合成语音

    % exc_syn((n-1)*FL+1:n*FL) = ...
```

```
        将你计算得到的合成激励写在这里
tmp=(n-1)*FL+1:n*FL;
exc_syn((n-1)*FL+1:n*FL)= G*(mod(tmp,PT)==0);

% s_syn((n-1)*FL+1:n*FL) = ...
        将你计算得到的合成语音写在这里
[s_syn((n-1)*FL+1:n*FL),zi_syn]=filter(1,A,exc_syn((n-1)*FL
    +1:n*FL),zi_syn);
% (11) 不改变基音周期和预测系数，将合成激励的长度增加一倍，
        再作为filter
% 的输入得到新的合成语音，听一听是不是速度变慢了，
        但音调没有变。


tmp_syn_v=(n-1)*FL_v+1:n*FL_v;

% exc_syn_v((n-1)*FL_v+1:n*FL_v) = ...
        将你计算得到的加长合成激励写在这里
 exc_syn_v((n-1)*FL_v+1:n*FL_v)= G*(mod(tmp_syn_v,PT)==0);
% s_syn_v((n-1)*FL_v+1:n*FL_v) = ...
        将你计算得到的加长合成语音写在这里
 [s_syn_v((n-1)*FL_v+1:n*FL_v),zi_syn_v]=filter(1,A,exc_syn_v
     ((n-1)*FL_v+1:n*FL_v),zi_syn_v);
% (13) 将基音周期减小一半，将共振峰频率增加150Hz ，
        重新合成语音，听听是啥感受~
PT=fix(PT/2);
[z,p,k]=tf2zp(1,A);
%角度为正频率加150
%为负减150
p=abs(p).*exp((   angle(p)+2*pi/8000*150*   (2* (angle(p)>0)
    -1)   )*1j);
[B,A]=zp2tf(z,p,k);
tmp_t=(n-1)*FL+1:n*FL;
% exc_syn_t((n-1)*FL+1:n*FL) = ...
        将你计算得到的变调合成激励写在这里
exc_syn_t((n-1)*FL+1:n*FL)= G*(mod(tmp_t,PT)==0);
% s_syn_t((n-1)*FL+1:n*FL) = ...
        将你计算得到的变调合成语音写在这里
[s_syn_t((n-1)*FL+1:n*FL),zi_syn_t]=filter(B,A,exc_syn_t((n
```

```
            -1)*FL+1:n*FL),zi_syn_t);
end

    % (6) 在此位置写程序，听一听s，exc和s_rec有何区别，解释这种区别
    % 后面听语音的题目也都可以在这里写，不再做特别注明
    soundsc(s,8000);
    pause(L/8000);
    soundsc(s_rec,8000);
    pause(L/8000);
    soundsc(exc,8000);
    pause(L/8000);
    soundsc(s_syn,8000);
    pause(L/8000);
    soundsc(s_syn_v,8000);
    pause(2*L/8000);
    soundsc(s_syn_t,8000);
   t=(1:L)/8000;
    figure;
    plot(t,s/max(s),'k');
    hold on;
    plot(t,s_rec/max(s_rec),'r-.',t,exc/max(exc),'b-.');
    legend('s','s\_rec','exc');
    title('wave of s,s_rec,exc');
    % 保存所有文件
    writespeech('exc.pcm',exc);
    writespeech('rec.pcm',s_rec);
    writespeech('exc_syn.pcm',exc_syn);
    writespeech('syn.pcm',s_syn);
    writespeech('exc_syn_t.pcm',exc_syn_t);
    writespeech('syn_t.pcm',s_syn_t);
    writespeech('exc_syn_v.pcm',exc_syn_v);
    writespeech('syn_v.pcm',s_syn_v);
return

% 从PCM文件中读入语音
function s = readspeech(filename, L)
    fid = fopen(filename, 'r');
    s = fread(fid, L, 'int16');
    fclose(fid);
```

```
130 return
131
132 % 写语音到PCM文件中
133 function writespeech(filename,s)
134     fid = fopen(filename,'w');
135     fwrite(fid, s, 'int16');
136     fclose(fid);
137 return
138
139 % 计算一段语音的基音周期，不要求掌握
140 function PT = findpitch(s)
141 [B, A] = butter(5, 700/4000);
142 s = filter(B,A,s);
143 R = zeros(143,1);
144 for k=1:143
145     R(k) = s(144:223)'*s(144-k:223-k);
146 end
147 [R1,T1] = max(R(80:143));
148 T1 = T1 + 79;
149 R1 = R1/(norm(s(144-T1:223-T1))+1);
150 [R2,T2] = max(R(40:79));
151 T2 = T2 + 39;
152 R2 = R2/(norm(s(144-T2:223-T2))+1);
153 [R3,T3] = max(R(20:39));
154 T3 = T3 + 19;
155 R3 = R3/(norm(s(144-T3:223-T3))+1);
156 Top = T1;
157 Rop = R1;
158 if R2 >= 0.85*Rop
159     Rop = R2;
160     Top = T2;
161 end
162 if R3 > 0.85*Rop
163     Rop = R3;
164     Top = T3;
165 end
166 PT = Top;
167
168 return
```