MATLAB 综合实验之语音合成

实验报告

学号：2014011216

姓名：刘 前

班级：无 47

日期：2016 年 7 月 16 日

一、语音预测模型

(1) 给定 $e(n)=s(n)-a_1s(n-1)-a_2s(n-2)$。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。

[滤波器传递函数]

本合成模型的差分方程为

$$e(n)=s(n)-a_1s(n-1)-a_2s(n-2)$$

两边同时作 z 变换，可得

$$E(z)=S(z)-a_1z^{-1}S(z)-a_2z^{-2}S(z)$$

整理得，传递函数

$$H(z)=\frac{1}{1-a_1z^{-1}-a_2z^{-2}}$$

零极点图，频率响应和单位样值响应均由 MATLAB 相关函数实现，代码如下：

[代码]

§§ex1_1.m

```
clear all,close all,clc;

a=[1,-1.3789,0.9506];          %定义输入信号系数
b=[1];                         %定义输出信号系数
%共振峰频率
[r,p,k]=residuez(b,a);         %求解极点
fs=8000;                       %采样频率取 8000Hz
OMG=abs(angle(p));             %Ω=ωT=ω/fs
fp=OMG*fs/(2*pi)               %f=ω/(2*pi)
%用 zplane,freq,impz 分别绘出零极点分布图，频率响应和单位样值响应。
zplane(b,a);                   %zplane 作零极点分布图
figure;                        %生成新图框
freqz(b,a);                    %freq 作频率响应
figure;                        %生成新图框
impz(b,a);                     %impz 作单位样值响应
set(gca,'XLim',[0,200]);       %修改 x 范围
%用 filter 绘出单位样值响应
figure;                        %生成新图框
n=[0:200]';                    %生成时间点
x=(n==0);                      %以单位样值序列作为激励信号
imp=filter(b,a,x);             %filter 单位样值响应
stem(n,imp);                   %作出单位样值响应
```

```
%比较 impz 和 filter 绘出的单位样值响应是否相同
figure;                    %生成新图框
impz(b,a);                 %用 impz 作出单位样值响应
set(gca,'XLim',[0,200]);   %修改 x 范围
hold on;                   %不擦除上图
stem(n,imp,'k-');          %用 filter 作出单位样值响应
```

a. 共振峰频率：

共振峰频率公式的推导：

系统的极点为 $p=|p_i|e^{\pm j\Omega}$，在时域冲激响应中的贡献是

$$A|p_i|^n\cos(\Omega nt+\varphi)。$$

其中，共振峰频率 f 与极点的弧度 Ω 之间的关系为：

$$f=\frac{\omega}{2\pi}=\frac{\Omega}{T_s}\frac{1}{2\pi}=\frac{\Omega f_s}{2\pi}$$

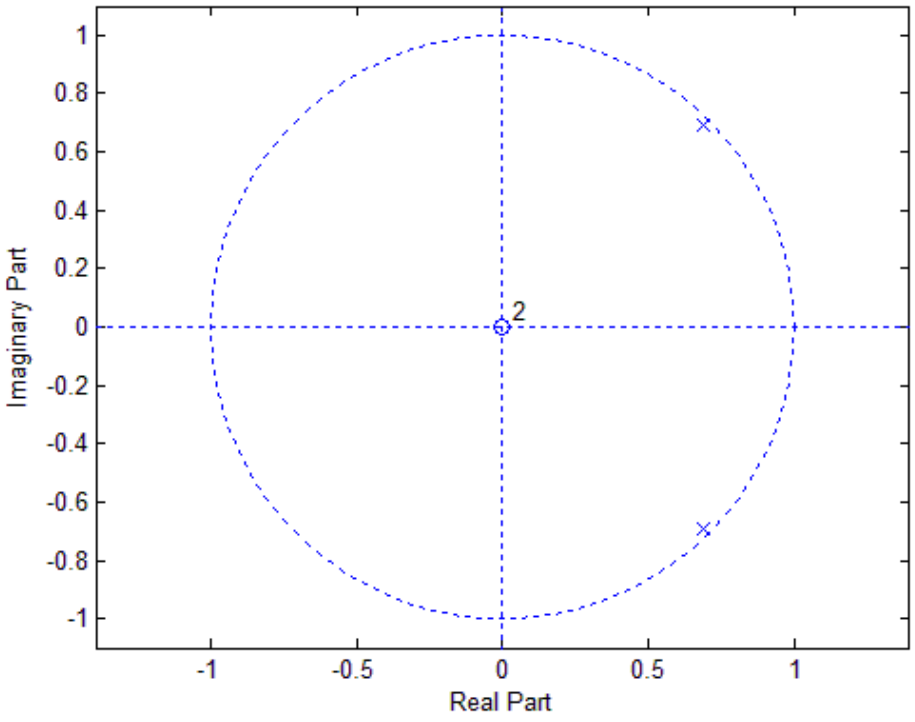
Matlab 运算结果：

```
fp =

  999.9447
  999.9447
```

可得，合成模型的共振峰频率 999.9447Hz。

b. *zplane* 函数作出零极点分布图

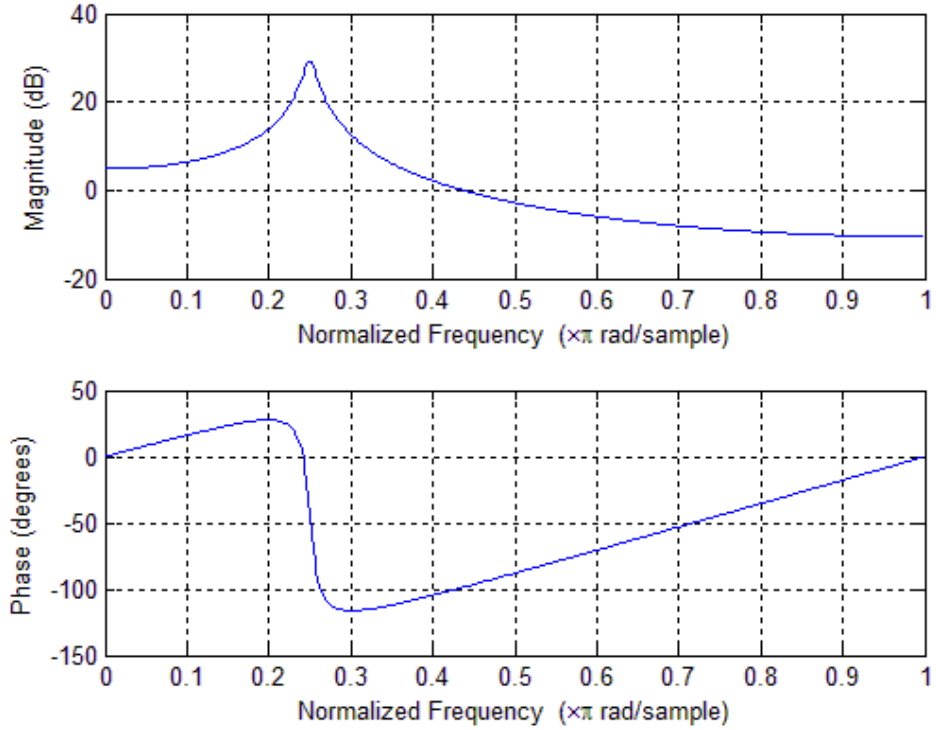


由零极点分布图可得出结论：

零点：原点处为二阶零点；

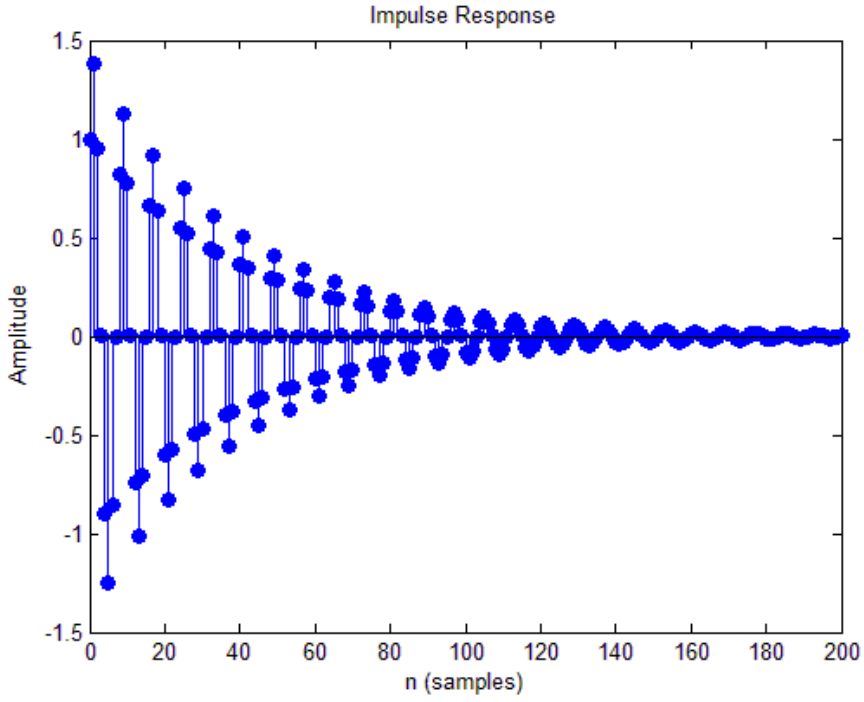
极点：单位圆上有一对共轭极点，均为一阶极点。

c. 用 *freqz* 函数得到频率响应

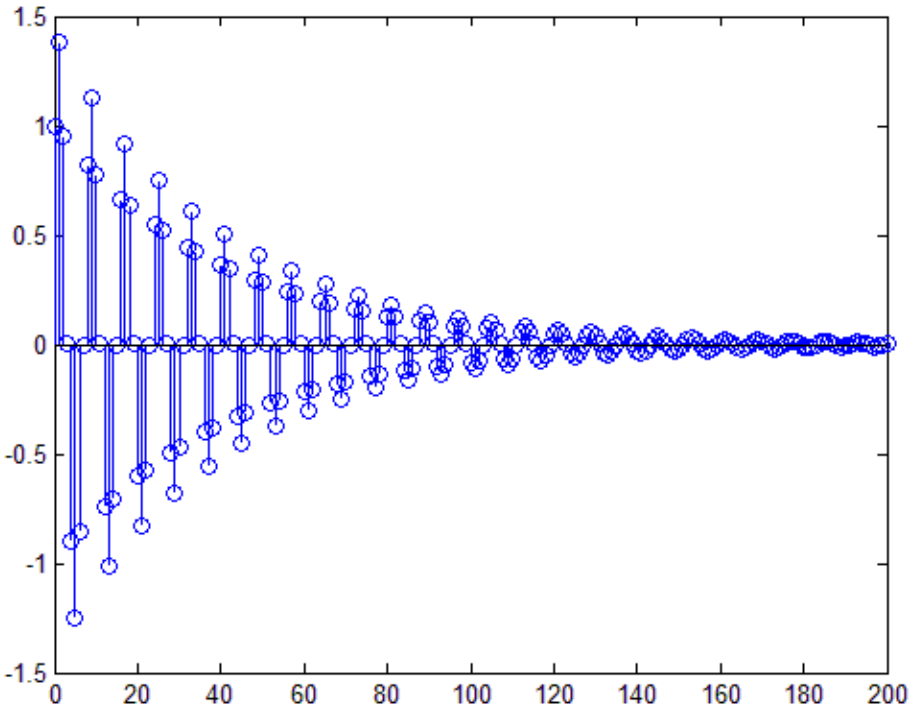


由频率响应曲线可判断出该合成模型属于带通系统。

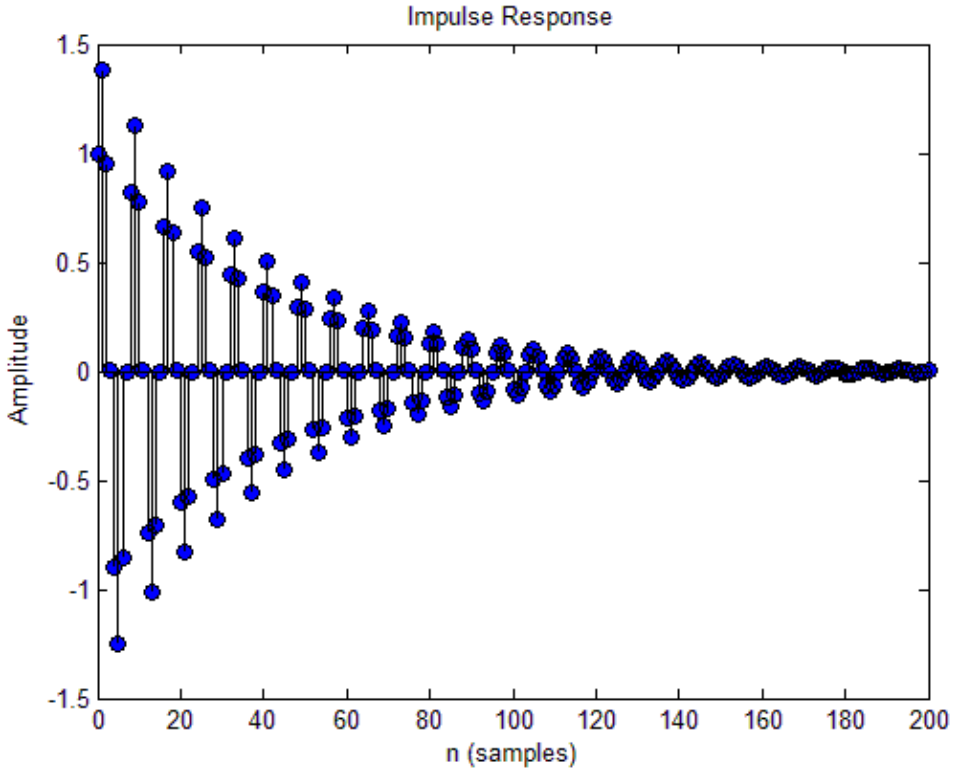
d. 用 *impz* 函数得到单位样值响应



e. 用 *filter* 函数得到单位样值响应



f. *impz* 和 *filter* 所得单位样值响应比较



蓝色是用 impz 得到的单位样值响应；
黑色是用 filter 得到的单位样值响应。
通过比较，可以得出：两种方法得到的单位样值响应完全相同。

其实上，直接使用 impz 函数求冲激响应和使用 filter 函数求解本质一样。

(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。

[分析]

阅读 speechproc.m 程序可以看出程序主要分为以下部分：

1. 基本变量定义；
2. 依次处理每帧语音：
   ① Hamming 窗加权；
   ② 线性预测法计算预测系数
   ③ 输入预测信号至预测滤波器，得到激励信号
   ④ 输入激励信号至重建滤波器，得到重建语音信号
4. 由合成激励得到合成语音信号
5. 变速不变调情况下由合成激励得到合成语音信号
6. 变调不变速情况下由合成激励得到合成语音信号
7. 保存所有相关语音文件。

(3) 运行该程序到 27 帧时停住，用（1）中的方法观察零极点图。

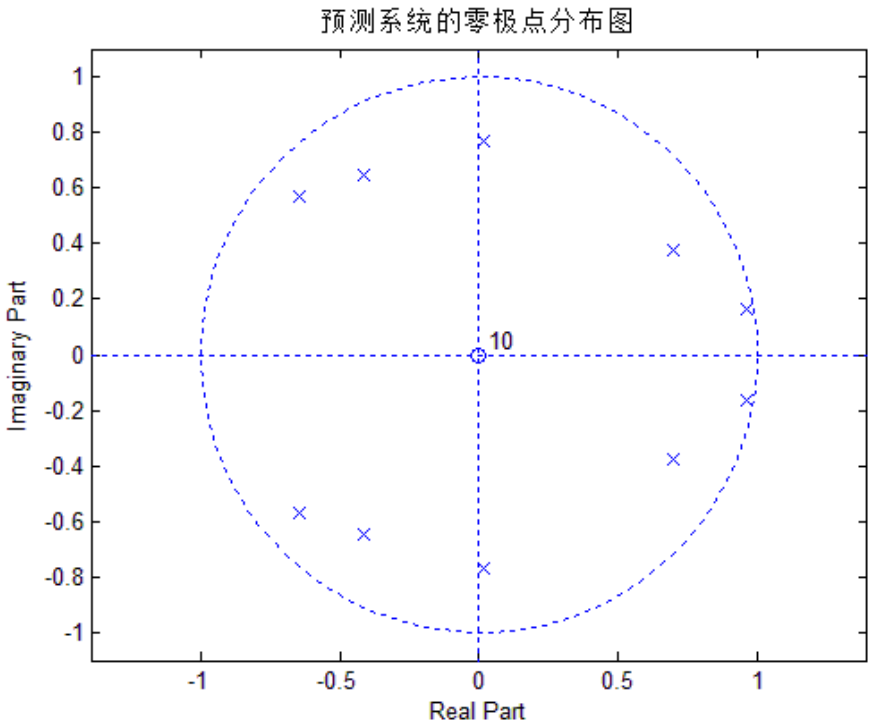
[分析]

系统的零极点分布由差分方程的输入输出系数决定。对于预测系统，输入系数即为预测系数向量 A，输出系数为 1。

[代码]

```
if n == 27
% (3) 在此位置写程序，观察预测系统的零极点图
    a_pre=A;                    %定义输入信号系数（预测系数）
    b_pre=[1];                  %定义输出信号系数（1）
    zplane(b_pre,a_pre);        %zplane 作出零极点分布图
    title('预测系统的零极点分布图');      %作图
end
```

[预测系统的零极点分布图]



(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{ai}，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。

[分析]

系数变化时连续滤波，需维持滤波器的状态不变，利用 filter 的 zi 和 zf 参数，即调用下面形式的 filter 函数：

$$[y, zf] = filter(b, a, x, zi)$$

其中，a, b 分别表示系统对应差分方程左侧和右侧的系数；zi 和 zf 分别表示系统的初始状态和终止状态。

为了计算激励信号，可将每帧语音信号 s_f 作为输入，在原系统函数倒数的作用下即可得到激励信号 en，因而参数(b, a)对应(A, 1)。为维持滤波器状态不变，zi_pre=zf_pre;

[代码]

```
% (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), zf_pre]=filter(A,1,s_f,zi_pre);      %filter 函数得到激励
zi_pre=zf_pre;                                              %保持滤波器状态不变
```

(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{ai}，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。

[分析]

由(4)中所得的激励信号 en，用 filter 函数重建语音 s_rec。同样用函数：

$$[y, zf] = filter(b, a, x, zi)$$

为维持滤波器状态，zi=zf=zi_pre；与(4)中不同，由激励信号计算输出，参数(b, a)对应(1, A)。

[代码]

```
% (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),zf_rec]=filter(1,A,exc((n-1)*FL+1:n*FL),zi_rec);
  zi_rec=zf_rec;                                          %保持滤波器状态不变
```

(6) 在循环结束后添加程序：用 sound 试听(5)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。

[分析]

使用 sound 函数分别试听三个信号。为了避免同时播放造成的混杂，每两个信号之间使用 pause 信号加以停顿。

[Matlab 代码]

```
% (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
    subplot(3,1,1); plot(s);          %完整的 s(n)信号
    subplot(3,1,2); plot(exc);        %完整的 e(n)信号
    subplot(3,1,3); plot(s_rec);      %完整的 s^(n)信号
    figure;
    sound(s);                         %播放直接从 voice.pcm 载入的语音信号 s(n)
    subplot(3,1,1);                   %子图 1
    plot(s(0.3*L:0.5*L));             %选取完整语音的 0.3~0.5 部分
    title('s(n)信号');                %设置标题
    pause(2);                         %停顿 2s,便于区分声音
    sound(exc);                       %播放之前得到的激励信号 e(n)
    subplot(3,1,2);                   %子图 2
    plot(exc(0.3*L:0.5*L));           %选取完整语音的 0.3~0.5 部分
    title('e(n)信号');                %设置标题
    pause(2);                         %停顿 2s,便于区分声音
    sound(s_rec);                     %播放由激励信号 e(n)恢复的语音信号 s^(n)
    subplot(3,1,3);                   %子图 3
    plot(s_rec(0.3*L:0.5*L));         %选取完整语音的 0.3~0.5 部分
    title('s''(n)信号');              %设置标题
```

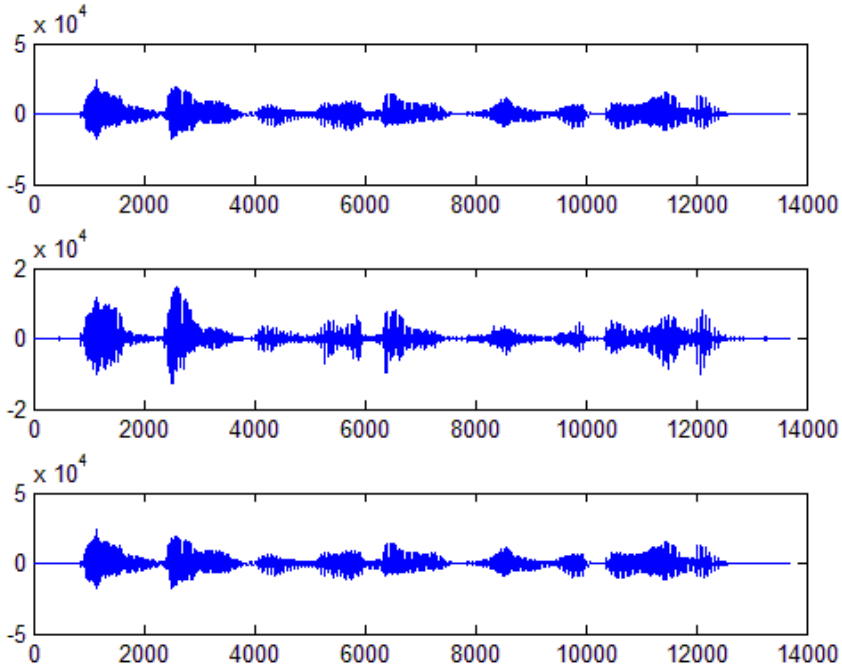
[播放效果]

s(n)信号，即 s 变量，播放声音比较清晰，但有一定杂音；

e(n)信号，即 exc 变量，能听语音，但是杂音很大；

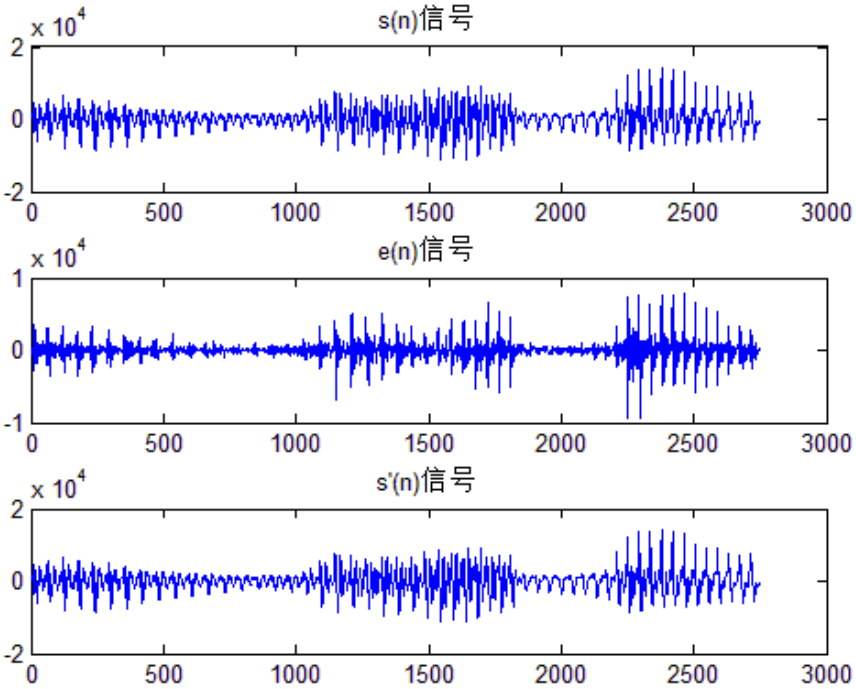
$\hat{s}(n)$ 信号，即 s_rec 变量，播放声音比较清晰，与 s(n)比较相似。

[完整的信号]



[选择一小段对比]

选择 0.3~0.5 倍总时长部分，比较三个信号。



比较：

根据上面得到的三个信号，可以得出与试听相同的的结论：

s(n)和 $\hat{s}(n)$ 信号较为接近，e(n)具有和 s(n)、$\hat{s}(n)$ 类似的大致趋势，但变化变化非常急剧，可以看出高频噪声较多。

二、语音合成模型

(7) 生成一个 8kHz 抽样的持续 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) 。

[分析]

因为该单位样值\串的频率为 200Hz，因而 NS=200。

抽样频率为 8kHz，持续 1s 中 $\delta(n)$ 信号有 8000 个，所以 NS=8000/200=40。

事实上，根据 $x(n)$ 展开式更容易理解：

$$x(n)=\delta(n)+\delta(n-40)+\delta(n-2\times40)+...+\delta(n-199\times40)$$

[代码]

```
clear all,close all,clc;

%单位样值串频率为 200Hz
fs=8000;                          %采样频率为 8000Hz
f_delta=200;                      %单位样值串频率为 200Hz
num=floor(fs/f_delta);            %num 为采样点的间隔
f_200=(mod([1:fs],num)==0)+0;     %选取 num 的倍数的点进行采样
                                  %加 0 可以将逻辑型变量转换为整数型
sound(f_200,8000);                %以 8kHz 采样频率播放
pause(1.1);                       %停顿 1.1s,将两种语音分开
%单位样值串频率为 300Hz
fs=8000;                          %采样频率为 8000Hz
f_delta=300;                      %单位样值串频率为 300Hz
num=floor(fs/f_delta);            %num 为采样点的间隔
f_300=(mod([1:fs],num)==0)+0;     %选取 num 的倍数的点进行采样
                                  %加 0 可以将逻辑型变量转换为整数型
sound(f_300,8000);                %以 8kHz 采样频率播放
```

[试听效果]

频率为 300Hz 的信号音调比较明显地高于频率为 200Hz 的信号，无其他差别。

(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 10ms 长的若干个段，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

$$PT = 80 + 5mod\left(m, 50\right)$$

其中 PT 表示基音周期，m 表示段序号。生成 1s 的上述信号并试听。（提示：用循环逐段实现，控制每个段内每个脉冲和前一个脉冲的间隔为本段的 PT 值，注意每个段内的第一个脉冲要和上一（或多）个的最后一个脉冲去比。）

[分析]

第 m 段的基音周期为 $PT = 80 + 5mod\left(m, 50\right)$，每段 10ms，在 8kHz 的采样频率下，原来一共 8000 个采样点。

[代码]

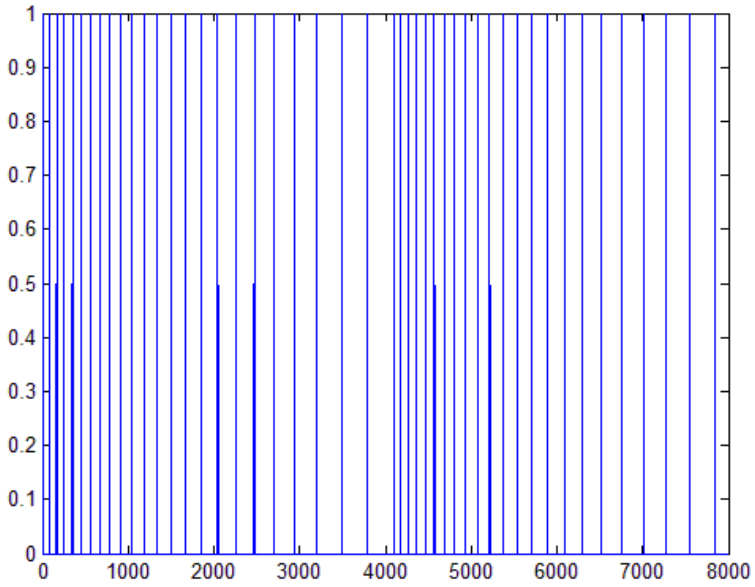
§§ex2_2.m

```
clear all,close all,clc;

n=1;                        %起始采样点标号，原来共 8000 个采样点
m=1;                        %段序号
PT=80;                      %初始化基音周期
sp=zeros(1,8000);           %经过处理后的 1s 的信号 sp
while n>0&&n<=8000
    sp(n)=1;                %脉冲点
    m=floor(n/80);          %段序号
    PT=80+5*mod(m,50);      %基音周期
    n=n+PT;                 %每段内每个脉冲个和前一脉冲
                            %的间隔为本段的 PT 值
end
sound(sp,8000);             %以 8kHz 采样频率播放
%plot(sp);
```

[试听效果]

经过试听，声音分为较为明显的两段。用 plot(sp)作出下图，可以明显看出，每段的基音周期不同，且声音中间会发生比较明显的变化。



(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。

[分析]

已知激励信号和系统的差分方程

$$e(n)=s(n)-1.3789s(n-1)+0.9506s(n-2)$$

用 filter 函数即可得到响应信号。

[代码]

§§ex2_3.m

```
clear all,close all,clc;

n=1;                        %起始采样点标号，原来共 8000 个采样点
m=1;                        %段序号
PT=80;                      %初始化基音周期
sp=zeros(1,8000);           %经过处理后的 1s 的信号 sp
while n>0&&n<=8000
    sp(n)=1;                %脉冲点
    m=floor(n/80);          %段序号
    PT=80+5*mod(m,50);      %基音周期
    n=n+PT;                 %每段内每个脉冲个和前一脉冲
                            %的间隔为本段的 PT 值
end
sound(sp,8000);             %以 8kHz 采样频率播放
pause(1.1);                 %停顿 1.1s 便于分别
a=[1];                      %定义输入信号系数
b=[1, -1.3789, 0.9506];     %定义输出信号系数
sn=filter(b,a,sp);          %用 filter 由输入信号得到输出 s(n)
sound(sn,8000);             %以 8kHz 采样频率播放
```

[试听效果]

比较 e(n)和得到的 s(n)信号，可以听出两者都分为较为明显的两段，但 s(n)的音调比 e(n)的音调高。

(10) 重改 speechproc.m 程序。利用每一帧已经计算得到的基音周期和（2）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。

[分析]

由本部分第(2)题处理基音周期的方法，能够得到增益后的合成激励信号。再运用 filter 函数，将每帧合成激励信号通过合成滤波器处理得到每帧的合成语音。最后即可得到完整的合成语音。

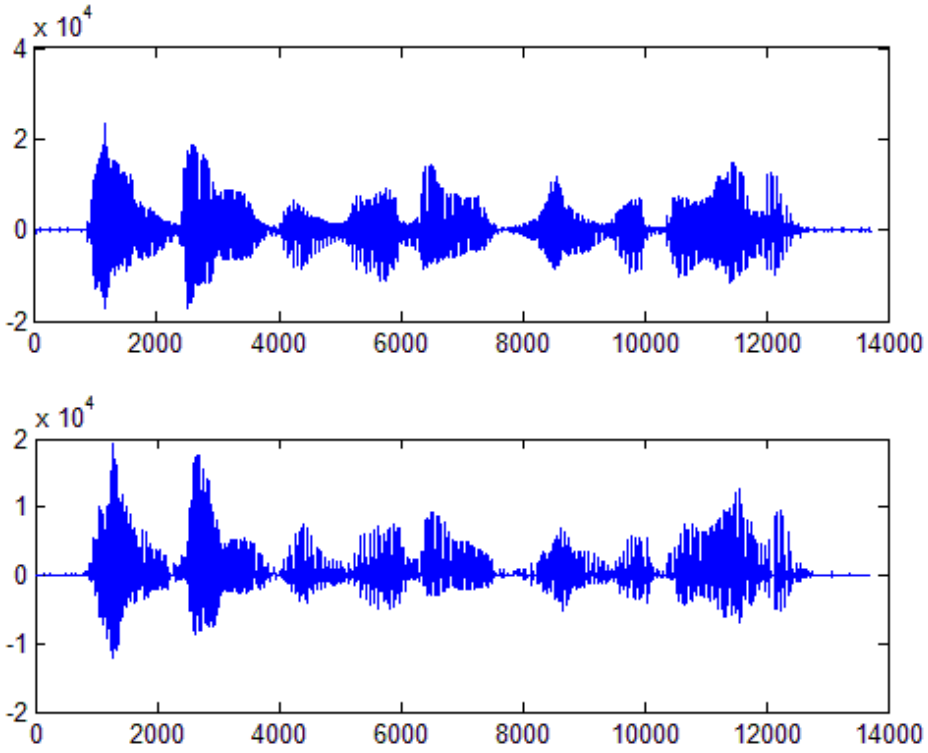
[代码]

```
% (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
    pos_syn=2*FL+1;              %由于从第 3 帧开始处理，
                                 %n=3 时，语音信号从 2*FL+1 到 n*FL
    while(pos_syn<=n*FL)         %利用 9.2.2（2）中的基音周期处理方法
      exc_syn(pos_syn)=G;        %G 为计算得到的增益
      pos_syn=pos_syn+PT;        %控制每段内每个脉冲和前一个脉冲的
                                 %间隔为本段的 PT 值
    end
    Gxn=exc_syn((n-1)*FL+1:n*FL);              %Gxn 即 exc_syn 为合成的激励信号
    [sn_syn,zf_syn]=filter(1,A,Gxn,zi_syn);    %A 为预测系数
    zi_syn= zf_syn;                            %保持滤波器状态不发生改变
    s_syn((n-1)*FL+1:n*FL)=sn_syn;             %合成语音~s(n)
```

[试听效果]

在循环结束后，增加 sound(s_syn, 8000) 命令，即可试听合成语音 $\tilde{s}(n)$。合成语音能够分辨出语音的具体文字信息，但掺杂了较多噪声，与从 voice.pcm 直接读取的原始语音相比清楚度相对较低。

下图是合成语音与原始语音信号的对比：



三、变速不变调

(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 10ms 的一帧变成了 20ms 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。

[分析]

每帧 10ms 变为 20ms，帧长从原来的 FL=80，变为 2*FL=160。

因此，语音总长度变为原来的 2 倍；与(10)类似，只需将帧长改变。

[代码]

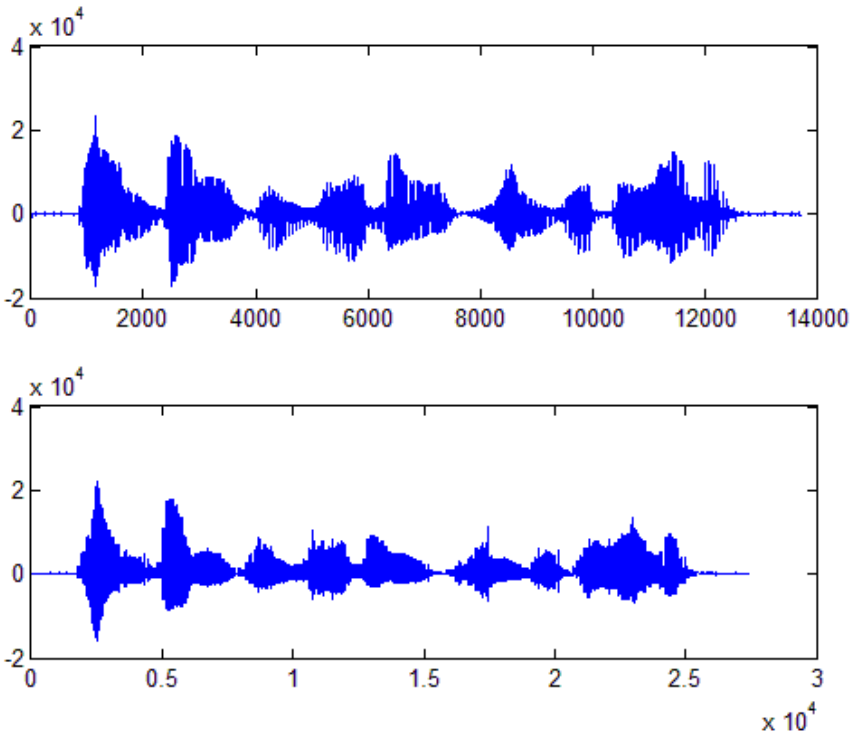
```
% (11) 不改变基音周期和预测系数，将合成激励的长度增加一倍，再作为 filter
%  的输入得到新的合成语音，听一听是不是速度变慢了，但音调没有变。
    pos_syn_v=2*FL+1;                %由于从第 3 帧开始处理，n=3 时，
                                     %语音信号从 2*FL+1 到 n*FL
    while(pos_syn_v<=2*n*FL)         %利用 9.2.2（2）中的基音周期处理；
                                     %语音总长度为原来的 2 倍
        exc_syn_v(pos_syn_v)=G;      %G 为计算得到的增益
        pos_syn_v=pos_syn_v+PT;      %同样控制每段内每个脉冲和
                                     %前一个脉冲的间隔为本段的 PT 值
    end
```

```
Gxn_t=exc_syn_v(2*(n-1)*FL+1:2*n*FL);          %Gxn_v 即 exc_syn_v 为
                                               %合成的激励信号
[sn_syn_v,zf_syn_v]=filter(1,A,Gxn_t,zi_syn_v);  %A 为预测系数
zi_syn_v= zf_syn_v;                  %保持滤波器状态不发生改变
s_syn_v(2*(n-1)*FL+1:2*n*FL)=sn_syn_v;          %合成语音 s_syn_v
```

[试听效果]

使用原始抽样速度播放时，速度明显变慢，时间长度是原来的 2 倍，但是语音的音调基本没有变化，仅从试听角度来说与原始语音音调相同，实现了“变速不变调”的效果。

下图是合成语音与原始语音信号的对比：



四、变调不变速

(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？

[分析]

共振峰频率改变，即极点位置改变，通过公式推导求解出新的极点，即可由 zp2tf 函数，得到新的系统差分方程的系数。

以下是公式推导：

已知(1)中系统的极点为 $p=|p_i|e^{\pm j\Omega}$，在时域冲激响应中的贡献是

$$A|p_i|^n\cos(\Omega nt+\varphi)。$$

其中，共振峰频率 f 与极点的弧度 Ω 之间的关系为：

$$f=\frac{\omega}{2\pi}=\frac{\Omega}{T_s}\frac{1}{2\pi}=\frac{\Omega f_s}{2\pi}$$

其中 f_s 为抽样频率。

设共振峰原来频率为 f_0，提高 150Hz 之后为 f_i 的极点为 p'，极点对应的弧度变化量为 Δ_Ω

由此可得关系式：

$$f_0=\frac{\Omega f_s}{2\pi} \qquad ①$$

$$f_i=\frac{(\Omega+\Delta_\Omega)f_s}{2\pi}=f_0+150Hz \qquad ②$$

由①②整理可得，

$$\Delta_\Omega=\frac{2\pi\times150Hz}{f_s}$$

所以，新的极点为

$$p'=|p_i|e^{\pm j(\Omega+\Delta_\Omega)}=pe^{\pm j\Delta_\Omega}$$

[代码]

§§ex4_1.m

```
clear all,close all,clc;

a=[1,-1.3789,0.9506];          %定义输入信号系数
b=[1];                         %定义输出信号系数
[z,p,k]=tf2zp(b,a);            %求解 9.2.1(1)极点
fs=8000;                       %采样频率取 8000Hz
delta_omg=2*pi*150/fs.*sign(angle(p));   %推导的公式中的±△Ω
pn=p.*exp(1i*delta_omg);                 %得到两个共轭的新极点
[B,A]=zp2tf(z,pn,k);                     %用 zp2tf 函数得到系数矩阵 A,B
```

[结果]

```
B =

     0     0     1


A =

    1.0000   -1.2073    0.9506
```

共振峰频率提高 150Hz 后，a_1=1.2073，a_2=－0.9506

(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。

[分析]

将每帧的基音周期减小一半，即可重新合成出激励信号，再由（1）中的增加共振峰频率的方法，得到新特性的合成模型，将激励信号通过该系统模型，即可得到新的合成语音。

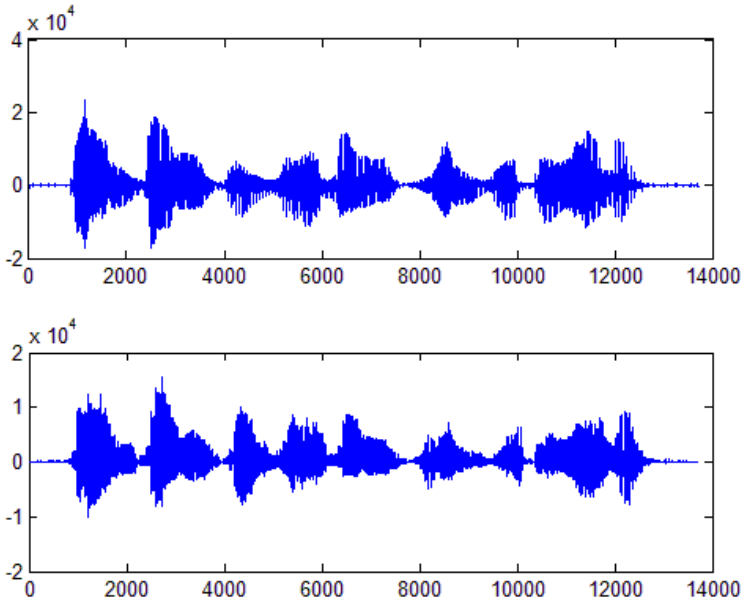
[代码]

```
[z,p,k]=tf2zp(1,A);                    %求解 9.2.1(1)极点
fs=8000;                               %采样频率取 8000Hz
delta_omg=2*pi*150*sign(angle(p))/fs;          %推导的公式中的±△Ω
pn=p.*exp(1i*delta_omg);                       %得到两个共轭的新极点
[Bc,Ac]=zp2tf(z,pn,k);      %用 zp2tf 函数得到系数矩阵 A,B
pos_syn_t=2*FL+1;           %从第 3 帧开始处理，语音信号从 2*FL+1 到 n*FL
while(pos_syn_t<=n*FL)              %基音周期处理；语音总长度为原来的 2 倍
    exc_syn_t(pos_syn_t)=G;         %G 为计算得到的增益
    pos_syn_t=pos_syn_t+round(PT/2);           %每段内每个脉冲和
                                               %前一脉冲间隔为本段 PT
end
Gxn_t=exc_syn_t((n-1)*FL+1:n*FL);    %Gxn_t 即 exc_syn_t 为合成的激励信号
[sn_syn_t,zf_syn_t]=filter(Bc,Ac,Gxn_t,zi_syn_t);    %A 为预测系数
zi_syn_t= zf_syn;                   %保持滤波器状态不发生改变
s_syn_t((n-1)*FL+1:n*FL)=sn_syn_t;             %合成语音 s_syn_t
```

[试听效果]

将基音周期减小一半，并且将共振峰频率增加 150Hz 后，语音的频率明显升高（音调变高），但语音速度没有改变，实现了“变调不变速”的效果。

下图是合成语音与原始语音信号的对比：



实验总结

在本次 MATLAB 综合实验中，原始语音 voice.pcm 包含一定成分的噪音，对最终的实验结果产生了一定的影响（音效均有不同程度杂音）。

通过本次实验，首先了解到了语音合成的背景知识，然后通过在实验练习的一步步引导下，从基本的语音预测模型分析到最后实现语音变速不变调、变调不变速的效果，对语音信号的处理有了基本的掌握，对 MATLAB 强大的信号处理能力也有了更直接地体会。

附录

上面相关部分只附加了主要代码，其中一些变量的初始化等之前略写，因而在附录部分粘贴已实现所有功能的 speechproc.m。

[代码]

```
function speechproc()

    % 定义常数
    FL = 80;                        % 帧长
    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);        % 重建语音的滤波器状态
    zi_syn = zeros(P,1);        % 合成语音的滤波器状态
    zi_syn_v = zeros(P,1);      % 变速不变调的合成语音的滤波器状态
    zi_syn_t = zeros(P,1);      % 变调不变速的合成语音的滤波器状态

    % 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
    % 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);           % 合成语音
    % 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
```

```
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
        a_pre=A;                   %定义输入信号系数（预测系数）
        b_pre=[1];                 %定义输出信号系数（1）
        zplane(b_pre,a_pre);       %zplane 作出零极点分布图
        title('预测系统的零极点分布图');
    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),zf_pre]=filter(A,1,s_f,zi_pre);
    zi_pre=zf_pre;                 % 保持滤波器状态


% (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),zf_rec]=filter(1,A,exc((n-1)*FL+1:n*FL),zi_rec);
    zi_rec=zf_rec;                 % 保持滤波器状态


    % 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
    pos_syn=2*FL+1;                %由于从第 3 帧开始处理，n=3 时
                                   %语音信号从 2*FL+1 到 n*FL
    while(pos_syn<=n*FL)           %利用 9.2.2（2）中的基音周期处理方法
        exc_syn(pos_syn)=G;        %G 为计算得到的增益
        pos_syn=pos_syn+PT;        %控制每段内每个脉冲和前一个脉冲
                                   %的间隔为本段的 PT 值
    end
```

```
        Gxn=exc_syn((n-1)*FL+1:n*FL); %Gxn 即 exc_syn 为合成的激励信号
        [sn_syn,zf_syn]=filter(1,A,Gxn,zi_syn);      %A 为预测系数
        zi_syn=zf_syn;                  %保持滤波器状态不变
        s_syn((n-1)*FL+1:n*FL)=sn_syn;  %合成语音~s(n)

% (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。

        pos_syn_v=2*FL+1;               %由于从第 3 帧开始处理，n=3 时，
                                        %语音信号从 2*FL+1 到 n*FL
        while(pos_syn_v<=2*n*FL)        %利用(8)中的基音周期处理；
                                        %语音总长度为原来的 2 倍
            exc_syn_v(pos_syn_v)=G;     %G 为计算得到的增益
            pos_syn_v=pos_syn_v+PT;     %同样控制每段内每个脉冲和
                                        %前一个脉冲的间隔为本段的 PT 值
        end
        Gxn_t=exc_syn_v(2*(n-1)*FL+1:2*n*FL);  %Gxn_v 即 exc_syn_v 为
                                               %合成的激励信号
        [sn_syn_v,zf_syn_v]=filter(1,A,Gxn_t,zi_syn_v);      %A 为预测系数
        zi_syn_v= zf_syn_v;             %保持滤波器状态不发生改变
        s_syn_v(2*(n-1)*FL+1:2*n*FL)=sn_syn_v; %合成语音 s_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～
        [z,p,k]=tf2zp(1,A);             %求解 9.2.1(1)极点
        fs=8000;                        %采样频率取 8000Hz
        delta_omg=2*pi*150*sign(angle(p))/fs;    %推导的公式中的±△Ω
        pn=p.*exp(1i*delta_omg);                 %得到两个共轭的新极点
        [Bc,Ac]=zp2tf(z,pn,k);          %用 zp2tf 函数得到系数矩阵 A,B
        pos_syn_t=2*FL+1;    %从第 3 帧开始处理，语音信号从 2*FL+1 到 n*FL
        while(pos_syn_t<=n*FL)          %基音周期处理；语音总长度为原来的 2 倍
            exc_syn_t(pos_syn_t)=G; %G 为计算得到的增益
            pos_syn_t=pos_syn_t+round(PT/2);
                                        %每段内每个脉冲和前一脉冲间隔为本段 PT
        end
        Gxn_t=exc_syn_t((n-1)*FL+1:n*FL);
                                        %Gxn_t 即 exc_syn_t 为合成的激励信号
        [sn_syn_t,zf_syn_t]=filter(Bc,Ac,Gxn_t,zi_syn_t);    %A 为预测系数
        zi_syn_t=zf_syn_t;              %保持滤波器状态不变
        s_syn_t((n-1)*FL+1:n*FL)=sn_syn_t;                  %合成语音 s_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
    figure;
    subplot(3,1,1); plot(s);          %完整的 s(n)信号
    subplot(3,1,2); plot(exc);        %完整的 e(n)信号
    subplot(3,1,3); plot(s_rec);      %完整的 s^(n)信号

    figure;
    sound(s);                         %试听播放直接从 voice.pcm 载入的语音信号 s(n)
    subplot(3,1,1);                   %子图 1
    plot(s(0.3*L:0.5*L));             %选取完整语音的 0.3~0.5 部分
    title('s(n)信号');                %设置标题
    pause(2);                         %停顿 2s,便于区分声音
    sound(exc);                       %试听之前得到的激励信号 e(n)
    subplot(3,1,2);                   %子图 2
    plot(exc(0.3*L:0.5*L));           %选取完整语音的 0.3~0.5 部分
    title('e(n)信号');                %设置标题
    pause(2);                         %停顿 2s,便于区分声音
    sound(s_rec);                     %试听由激励信号 e(n)恢复的语音信号 s^(n)
    subplot(3,1,3);                   %子图 3
    plot(s_rec(0.3*L:0.5*L));         %选取完整语音的 0.3~0.5 部分
    title('s''(n)信号');              %设置标题

    pause(2);
    sound(s_syn);                     %试听(10)的合成语音
  %figure;subplot(2,1,1);plot(s);subplot(2,1,2);plot(s_syn);
    pause(2);
    sound(s_syn_v);                   %试听(11)变速不变调的合成语音
  %figure;subplot(2,1,1);plot(s);subplot(2,1,2);plot(s_syn_v);
    pause(4);
    sound(s_syn_t);                   %试听(13)变调不变速的合成语音
  %figure;subplot(2,1,1);plot(s);subplot(2,1,2);plot(s_syn_t);


    % 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);
return

% 写语音到 PCM 文件中
function writespeech(filename,s)
	fid = fopen(filename,'w');
	fwrite(fid, s, 'int16');
	fclose(fid);
return

% 计算一段语音的基音周期，不要求掌握
function PT = findpitch(s)
	[B, A] = butter(5, 700/4000);
	s = filter(B,A,s);
	R = zeros(143,1);
	for k=1:143
		R(k) = s(144:223)'*s(144-k:223-k);
	end
	[R1,T1] = max(R(80:143));
	T1 = T1 + 79;
	R1 = R1/(norm(s(144-T1:223-T1))+1);
	[R2,T2] = max(R(40:79));
	T2 = T2 + 39;
	R2 = R2/(norm(s(144-T2:223-T2))+1);
	[R3,T3] = max(R(20:39));
	T3 = T3 + 19;
	R3 = R3/(norm(s(144-T3:223-T3))+1);
	Top = T1;
	Rop = R1;
	if R2 >= 0.85*Rop
		Rop = R2;
		Top = T2;
	end
	if R3 > 0.85*Rop
		Rop = R3;
		Top = T3;
	end
	PT = Top;
return
```