语音合成大作业

郭一隆 无36 2013011189

原创性声明

完全原创

语音预测模型

1. 给定

$$e(n) = s(n) - a_1 s(n-1) - a_2 s(n-2)$$

   i. 假设$e(n)$是输入信号, $s(n)$是输出信号, 则上述滤波器的传递函数为

$$H(z) = \frac{z^2}{z^2 - a_1 z - a_2}$$

   ii. 如果$a_1 = 1.3789, a_2 = -0.9506$, 则利用`[Z,P,K]=tf2zp(b,a)`或`roots(poly)`等函数可求出系统极点为

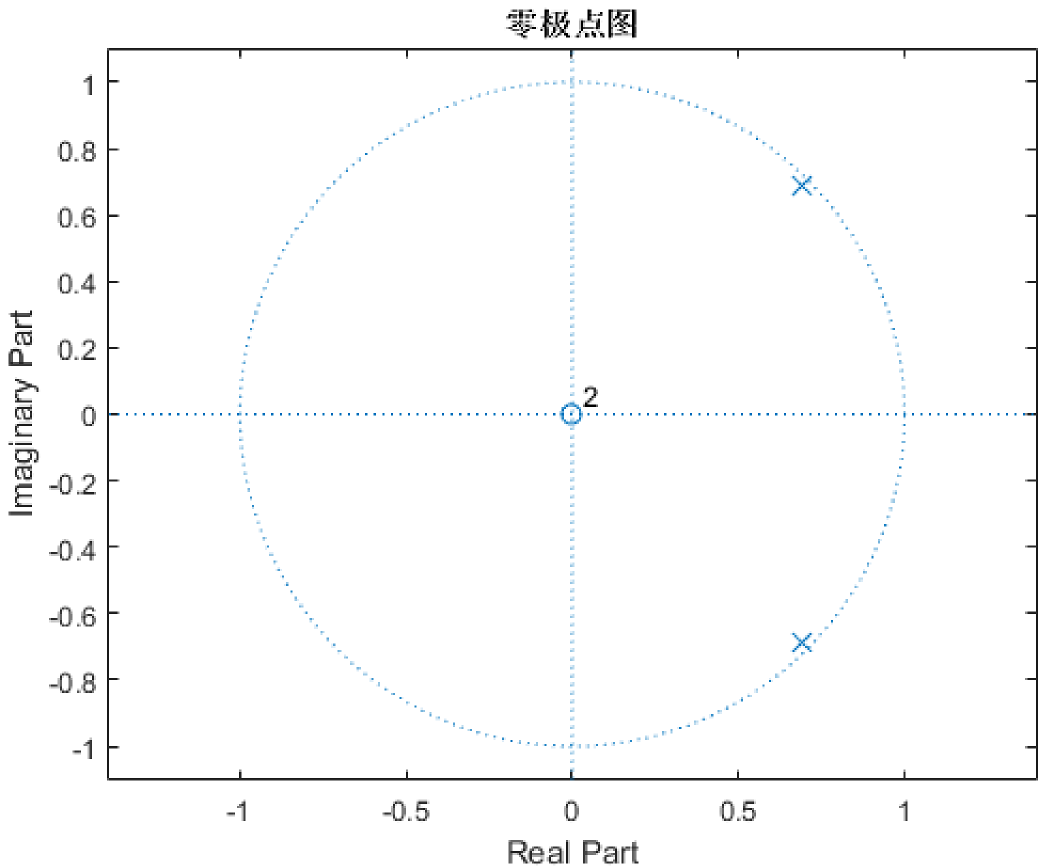
$$p_{1,2} = 0.6895 \pm 0.6894j = 0.9750e^{\pm j0.7854}$$

   根据共振峰频率的定义

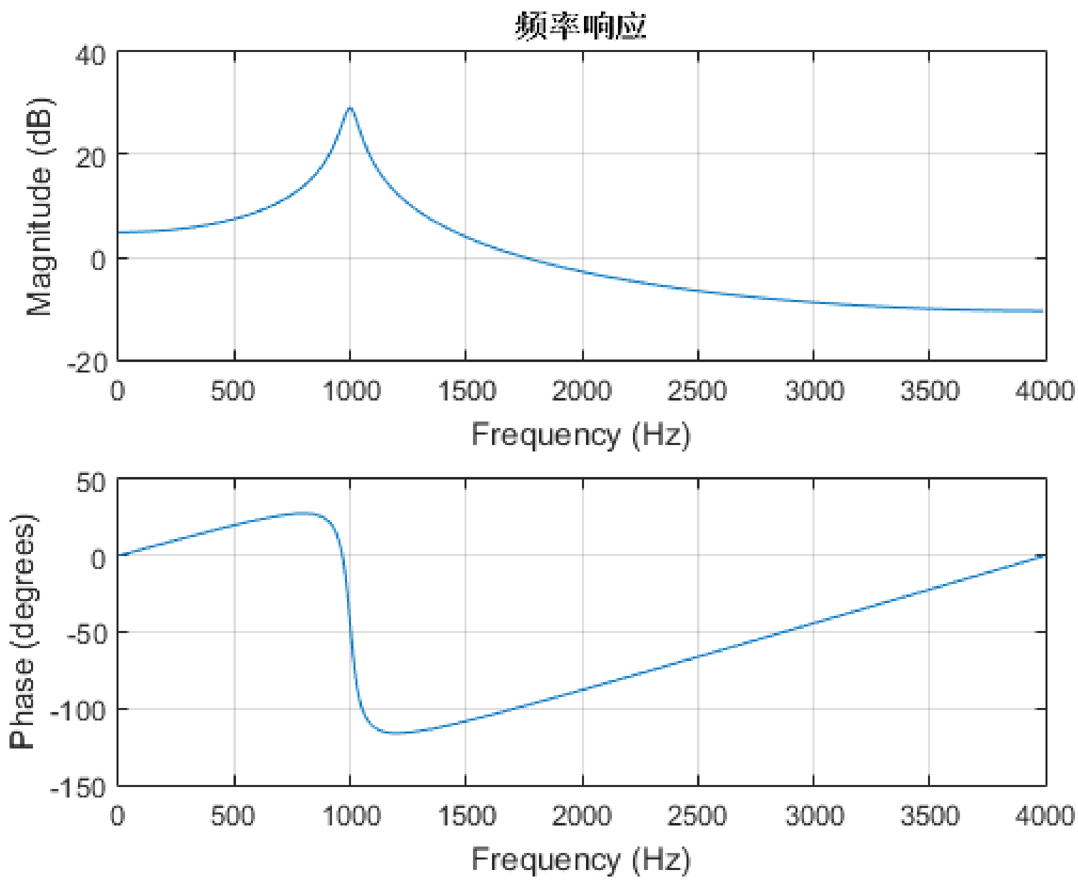
$$f = \frac{\omega}{2\pi} = \frac{\Omega}{2\pi T} = \frac{\Omega f_s}{2\pi} = \frac{Arg(p)}{2\pi} f_s$$

   取$f_s = 8000\text{Hz}$, 求得共振峰频率(Formant Frequency) $f_f = 999.94\text{Hz}$

   iii. 绘制零极点图, `zplane(Z,P)`



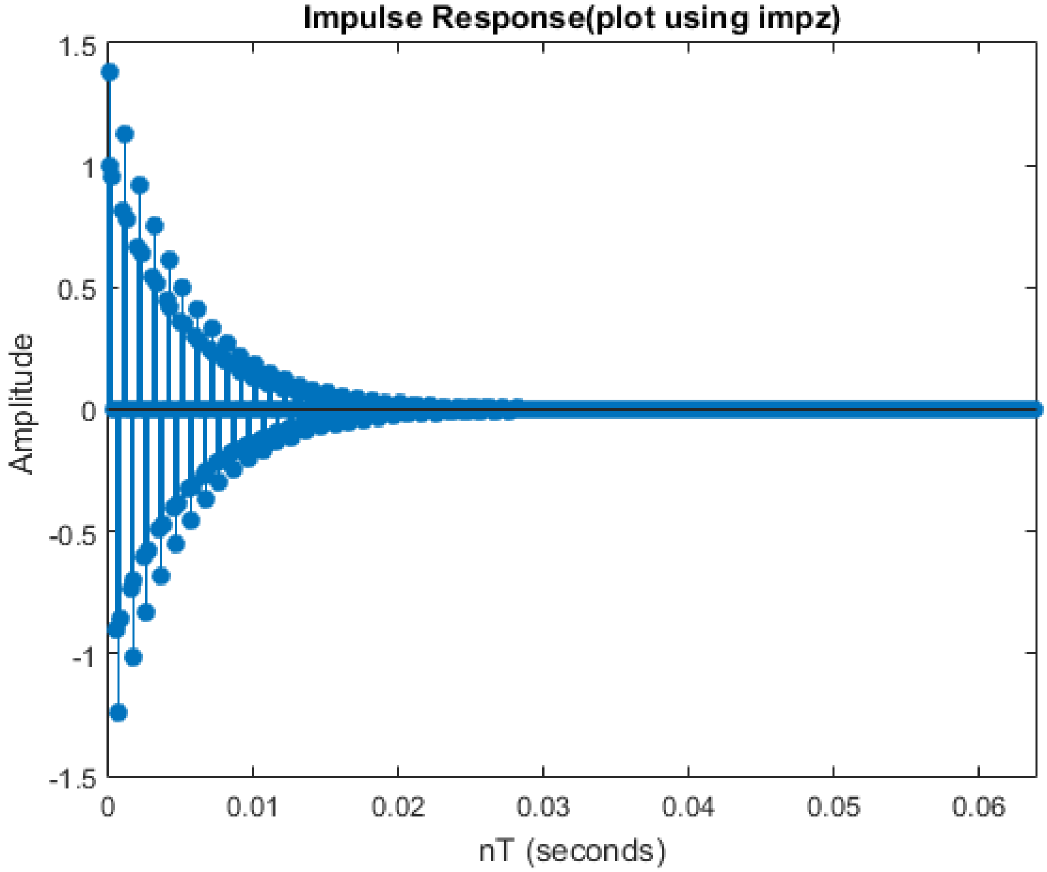
iv. 绘制频率响应，`freqz(b,a,512,fs)`



从频率响应图中也可直观地看出共振峰频率在1000Hz附近

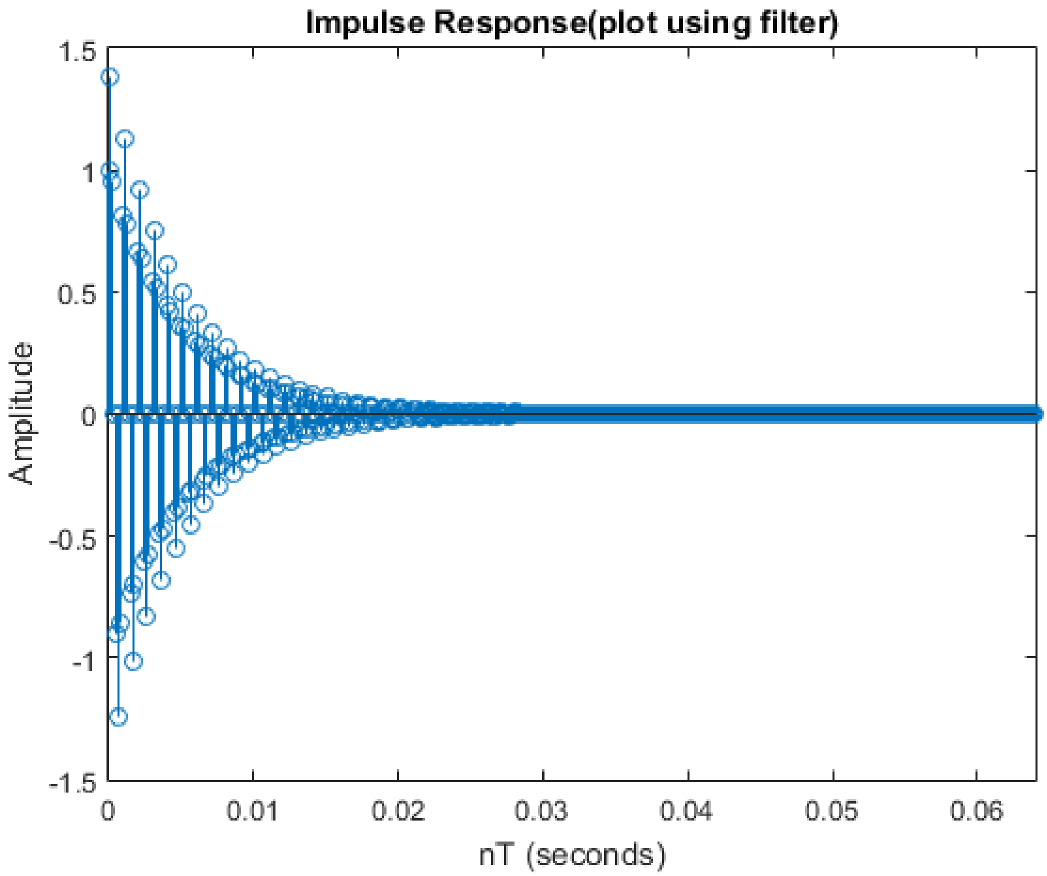
v. 绘制单位样值响应

- `impz(b,a,512,fs)`



- `filter`

```
x = zeros(1,512);
x(1) = 1;
y = filter(b,a,x);
stem(0:1/fs:511/fs,y);
```



与**impz**画出的结果相同

2. 阅读speechproc.m

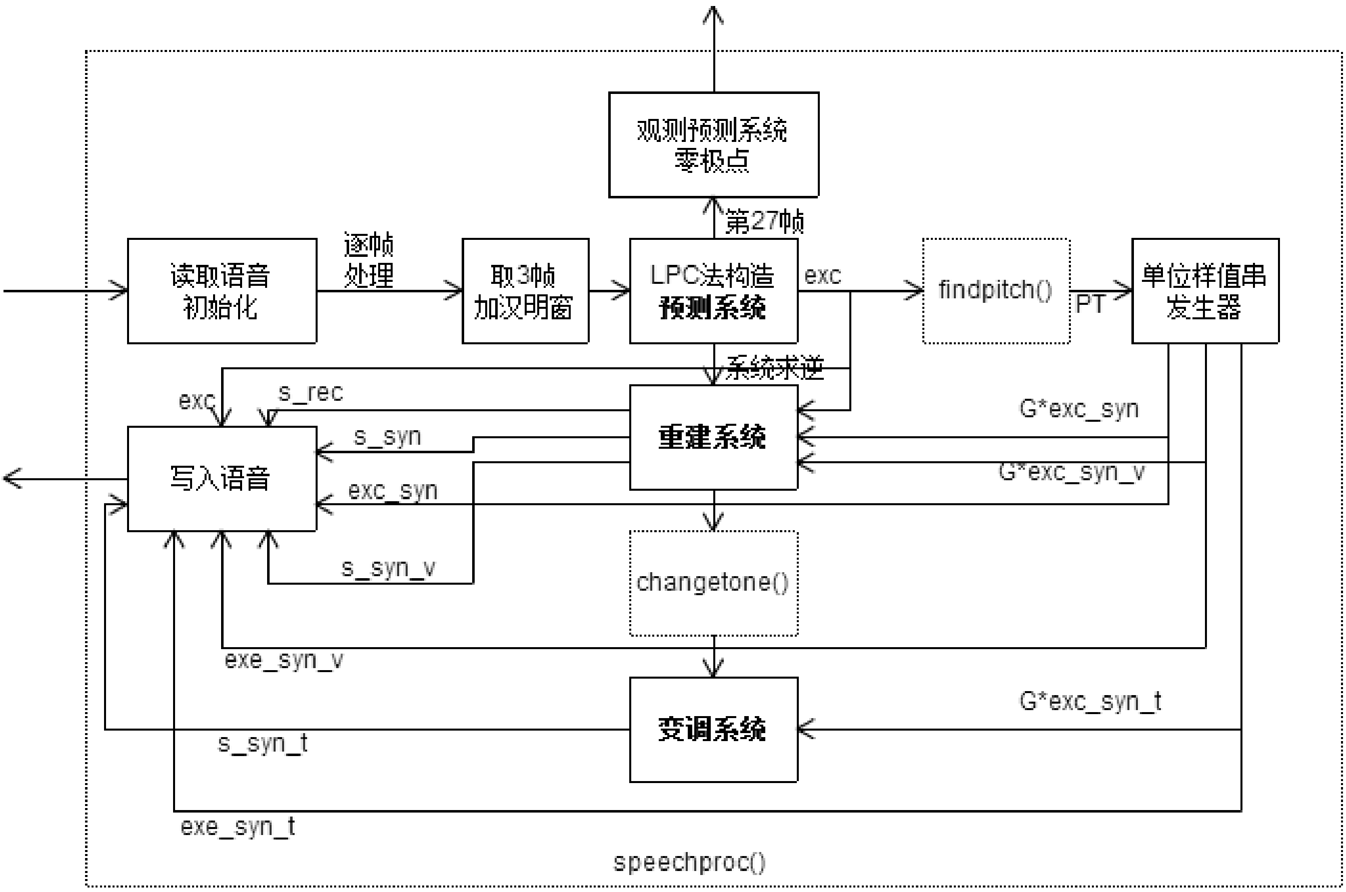
```
function speechproc()

    % 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);
    % 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

        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

        % (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) = ... 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
        % s_syn((n-1)*FL+1:n*FL) = ...   将你计算得到的合成语音写在这里

        % (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。


        % 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
        % 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

        % (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～


        % 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
        % 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
```

```
    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


    % 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
```

speechproc函数的系统框图



3. 在 `speechproc` 运行至第27帧时观察预测系统的零极点图

预测系统

$$e(n) = s(n) - \sum_{k=1}^{N} a_k s(n-k)$$

其中$s(n)$为输入，$e(n)$为输出，则系统函数

$$H_{pre}(z) = \frac{z^N - \sum_{k=1}^{N} a_k z^{N-k}}{z^N}$$

取预测系数个数 $N = P = 10$

- `lpc` 函数

    ```
    [A,E] = lpc(X,N)
    % A = [1 A(2) A(3) ... A(N+1)]
    ```

    系数A_i 使得

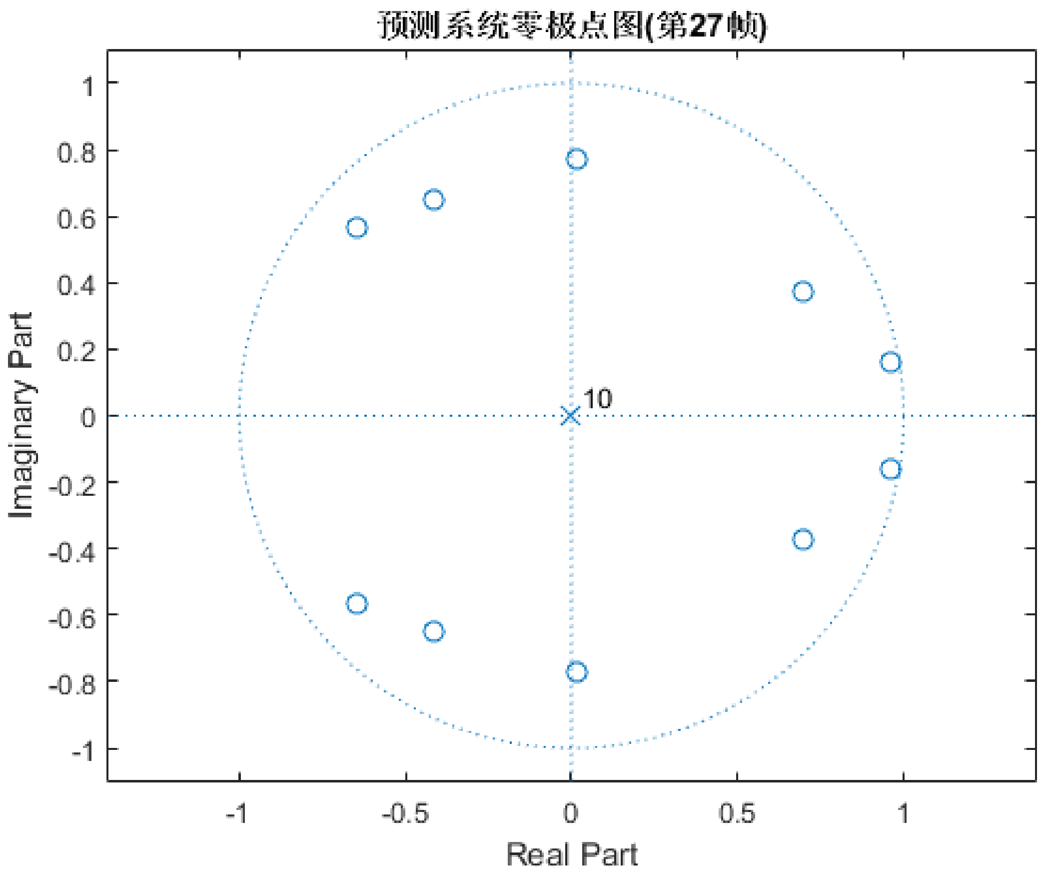
    $$err(n) = X(n) - \sum_{k=1}^{N}(-A_{k+1}X(n-k))$$

    取到最小值, 于是可知

    $$A = [1, -a_1, -a_2, \ldots, -a_N]$$

在 `speechproc` 中相应位置插入代码:

```
if n == 27
% (3) 在此位置写程序，观察预测系统的零极点图
    [z,p,~] = tf2zp(A,[1,zeros(1,P)]);
    zplane(z,p);
    title('预测系统零极点图(第27帧)');
end
```



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

```
% (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
[Y,zi_pre] = filter(A,[1,zeros(1,P)],s_f,zi_pre);   % keep state: zi_pre
exc((n-1)*FL+1:n*FL) = Y;
% exc((n-1)*FL+1:n*FL) = ... 将你计算得到的激励写在这里
```

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

   对于语音重建模型

$$\hat{s}(n) = x(n) + \sum_{k=1}^{N} a_k \hat{s}(n-k)$$

   输入为$x(n)$, 输出为$\hat{s}(n)$, 则系统函数

$$H_{rec}(z) = \frac{1}{H_{pre}(z)} = \frac{z^N}{z^N - \sum_{k=1}^{N} a_k z^{N-k}}$$

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

6. 对比归一化(normalize.m)后的$e(n)$, $s(n)$以及$\hat{s}(n)$信号

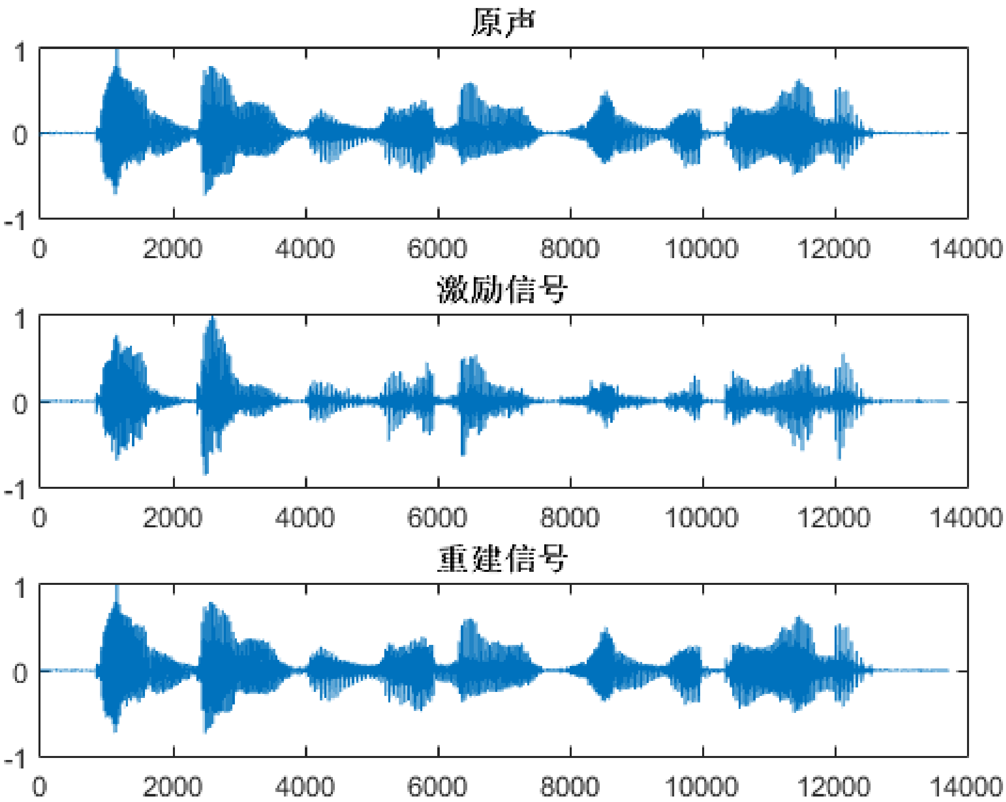
```
% normalization
s = normalize(s);
exc = normalize(exc);
s_rec = normalize(s_rec);

sound([s;exc;s_rec],8000);
figure(2);
subplot(3,1,1);plot(s);title('原声');
subplot(3,1,2);plot(exc);title('激励信号');
subplot(3,1,3);plot(s_rec);title('重建信号');
figure(3);
plot(s,'k');axis([6400 6500 -1 1]);hold on
plot(exc,'r');
plot(s_rec);hold off;
legend('原声','激励信号','重建信号');title('片段对比');
```

   听觉感受

   - $s(n)$和$\hat{s}(n)$听不出区别, 清晰明了

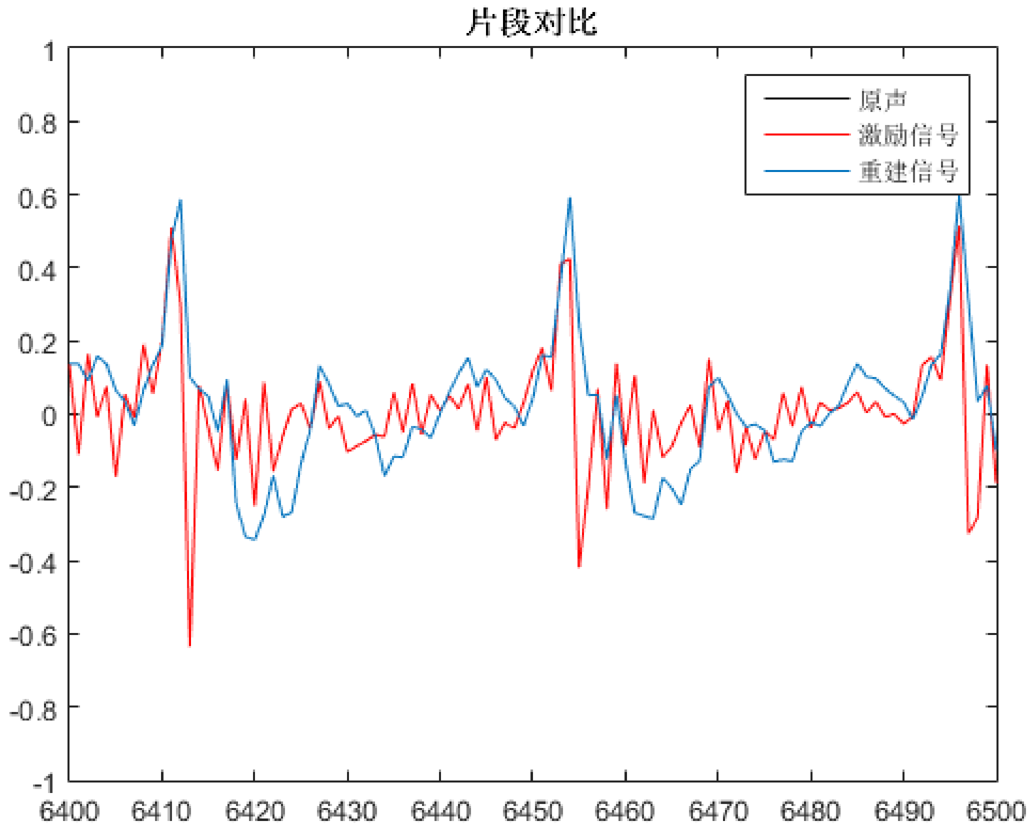
- $e(n)$ 有杂音, 但语音内容基本可以辨认

整体波形



- $s(n)$ 和 $\hat{s}(n)$ 波形包络基本一致, $e(n)$ 则不同
- 三者波形包络基本可以反映音节出现的位置

局部波形



- $s(n)$和$\hat{s}(n)$波形完全重合(黑色的$s(n)$被蓝色的$\hat{s}(n)$覆盖)
- $e(n)$变化更陡峭剧烈, 而$s(n)$和$\hat{s}(n)$的变化相对缓慢
- $e(n)$的局部峰值基本对应$s(n)$和$\hat{s}(n)$的局部峰值

语音合成模型

1. **(练习7)** 生成一个$f_s = 8000\text{Hz}$抽样的持续时间$T = 1\text{s}$的数字信号, 该信号是一个频率为$f = 200\text{Hz}$的单位样值"串", 即

$$x(n) = \sum_{i=0}^{NS-1} \delta(n - iN)$$

则式中$N = \frac{f_s}{f} = 40, NS = Tf = 200$

单位样值"串"即每隔一定间隔N有一脉冲(幅度为1), 其余位置取值为0, 则可以通过以下方法生成$x(n)$ (impulsestring.m)

```
function x = impulsestring(T,f,fs)
%Generate impulse string
%输入:
%   <double>T: 信号长度, 单位(秒)
%   <double>f: 冲激串频率
%   <double>fs: 采样频率
```

```
%输出:
%   <column vector>x: length(x)==round(T*fs), sum(x)==round(T*f)

x = zeros(round(T*fs),1);    % initialize x(n)
NS = round(T*f);             % NS
for i=0:NS-1
    x(round(i*fs/f)+1) = 1;    % x(k) = 1 if (k-i*N == 0) else 0
end

end
```

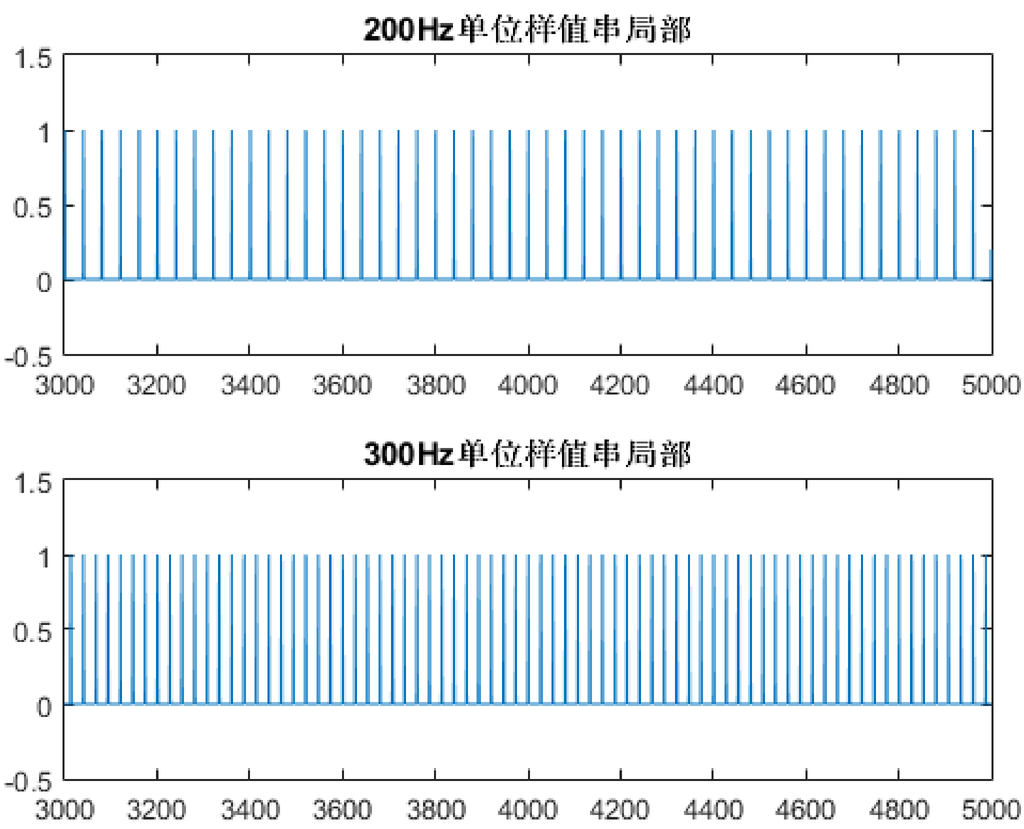
注1: 另一种生成方法是先利用`N=round(fs/f)`将`N`求出, 再以`N`为步长产生冲激, 但这样的结果不精确, 如当$314 \leq f \leq 326$时, 求得的N均为25, 产生的单位样值串频率均为320Hz, 误差在2%左右

注2: 上述方法产生的信号, 虽然相邻两冲激之间的间隔不是恒定(有± 1的浮动)的, 但从整体上看, 生成信号的频率更接近需求

试听

```
>> sound(impulsestring(1,200,fs),fs);
>> sound(impulsestring(1,300,fs),fs);
```

- 略刺耳
- 300Hz单位样值串音调更高



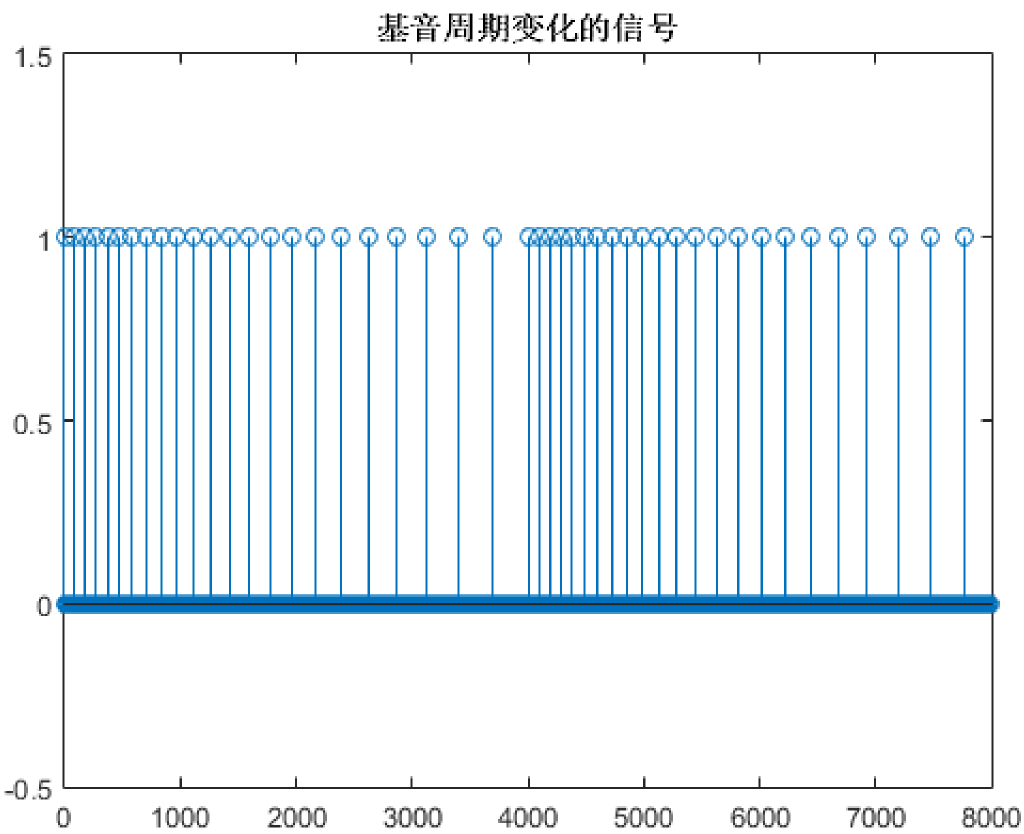
2. **(练习8)** 真实语音信号的基音周期总是随时间变化的. 我们首先将信号分成若干个10毫秒长的段, 假设每个段内基音周期固定不变, 但段和段之间则不同, 具体为

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

其中PT表示基因周期, m表示段序号, 相邻两脉冲的间隔由前一个脉冲所在的段序号决定

生成时长为1秒钟的上述信号(pitchtest.m):

```
x = zeros(8000,1);
cursor = 1;
m = 1;          % index of slice
while m <= 100
    x(cursor) = 1;
    cursor = cursor + 80 + 5*mod(m,50);     % next cursor
    m = ceil(cursor/80);                    % locate next cursor
end
figure;
stem(0:8000-1,x);
```



试听

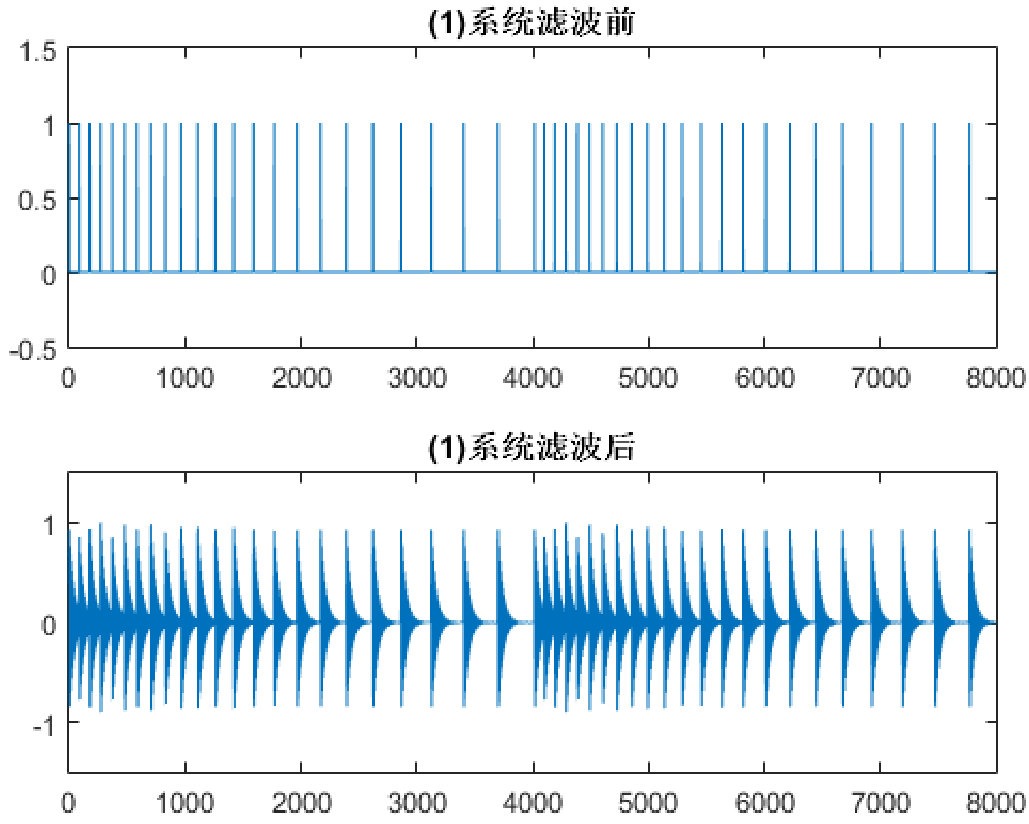
- 生成的信号被均分为重复的两段, 每段内音调逐渐变化
- 不好听, 很压抑, 好像也不能称之为颤音, 这个声音让我想到拉链的声音, 频率随时间变化的特征也相似, 姑且称之为拉链音

3. **(练习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$s(n)$

```
y = normalize(filter(b,a,x));
sound(y,fs);
```

试听

- 经过滤波器后音色发生显著变化, 变得更清脆了, 不那么刺耳
- 经过滤波器后音调似乎变低沉了一些



4. **(练习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$Gx(n)$, 用 `filter` 函数将$Gx(n)$送入合成滤波器得到合成语音$\tilde{s}(n)$

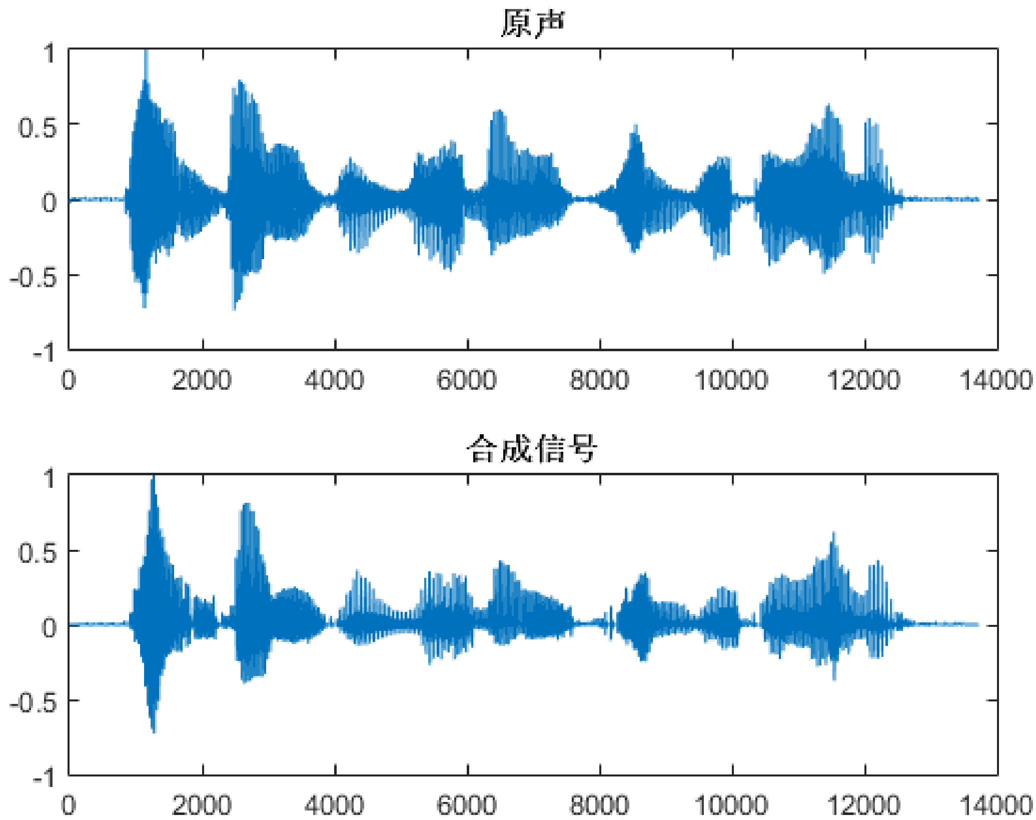
```
% (10) 在此位置写程序，生成合成激励，并用激励和filter函数产生生成合成语音

if n == 3                   % first loop
    cursor = (n-1)*FL+1;    % initialize cursor
    m = n;                  % initialize m
end

while m == n                % cursor still point into current frame
    exc_syn(cursor) = 1;
    cursor = cursor + PT;   % next cursor
    m = ceil(cursor/FL);    % locate next cursor
end
```

```
s_syn((n-1)*FL+1:n*FL) = filter([1,zeros(1,P)],A,...
    G*exc_syn((n-1)*FL+1:n*FL));

% exc_syn((n-1)*FL+1:n*FL) = ... 将你计算得到的合成激励写在这里
% s_syn((n-1)*FL+1:n*FL) = ...   将你计算得到的合成语音写在这里
```

试听

- 可以听出合成信号语音的内容
- 有些单薄，略有杂音

波形比较



- 波形包络基本相同
- `y<0` 部分的包络似乎不太吻合

变速不变调

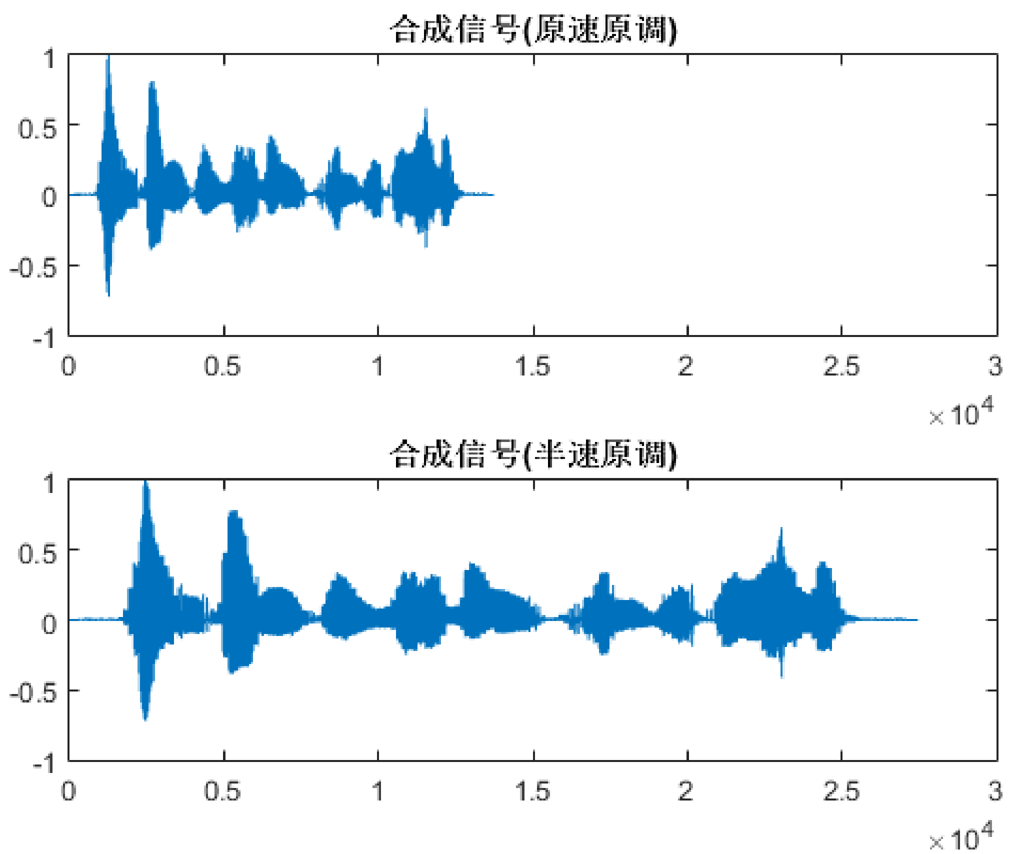
1. **(练习11)** 仿照**(10)**重改speechproc.m程序，只不过将**(10)**中合成激励的长度增加一倍，即原来10毫秒的一帧变成了20毫秒一帧，再用同样的方法合成出语音 `s_syn_v`

```
% (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。
```

```
FL_v = 2*FL;
if n == 3                   % first loop
    cursor_v = (n-1)*FL_v+1;    % initialize cursor
    m_v = n;                  % initialize m
end

while m_v == n                % cursor still point into current frame
    exc_syn_v(cursor_v) = 1;
    cursor_v = cursor_v + PT;   % next cursor
    m_v = ceil(cursor_v/FL_v);    % locate next cursor
end

s_syn_v((n-1)*FL_v+1:n*FL_v) = filter([1,zeros(1,P)],A,...
    G*exc_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
% 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
```

试听

- 在合成信号 `s_syn` 的基础上，`s_syn_v` 速度变为原来的一半，而音调没有变化



变调不变速

1. **(练习12)** 重新考察**(1)**中的系统

$$e(n) = s(n) - a_1 s(n-1) - a_2 s(n-2)$$

$$a_1 = 1.3789, a_2 = -0.9506$$

$$p_{1,2} = 0.6895 \pm 0.6894j = 0.9750e^{\pm j0.7854}$$

$$f_f = 999.94\text{Hz}$$

而共振峰频率

$$f_f = \frac{Arg(p)}{2\pi} f_s$$

因此需通过旋转极点(改变幅角)改变共振峰频率

定义 `rotatez` 函数用以旋转复数(rotatez.m)

```
function zr = rotatez(z,rad)
%Rotate complex number z by rad counterclockwisely

zr = z*exp(rad*1j);

end
```

则共振峰频率变化量为

$$\Delta f = \frac{\Delta Arg(p)}{2\pi} f_s$$

定义 `changetone` 函数(changetone.m)

```
function A = changetone(a,df,fs)
%A = changetong(a,df,fs)
%输入:
%   <vector>a: 传递函数的分母, 用于求极点
%   <double>df: 频率变化量
%   <double>fs: 采样频率
%输出:
%   <row vector>A: 变频后新系统传递函数的分母

p = roots(a);     % get poles
for k=1:length(p)
    p(k) = rotatez(p(k),sign(rem(angle(p(k)),pi))*2*pi*df/fs);     % rotate
    % use rem() in case angle equals pi
end
A = poly(p);     % get poly

end
```

```
>> a_t = changetone(a,150,fs)
```

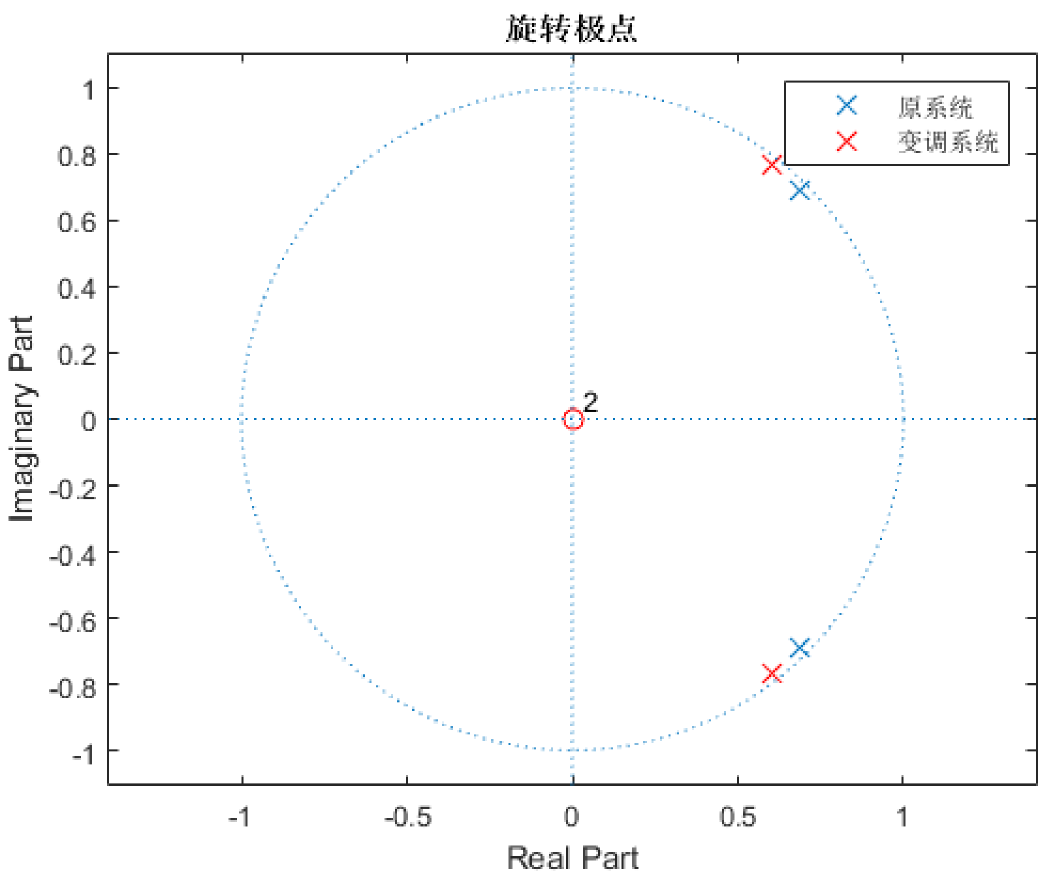
```
a_t =

    1.0000   -1.2073    0.9506
```

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

对比系统前后变化

- 绘制零极点图 (zplane_changetone.m)

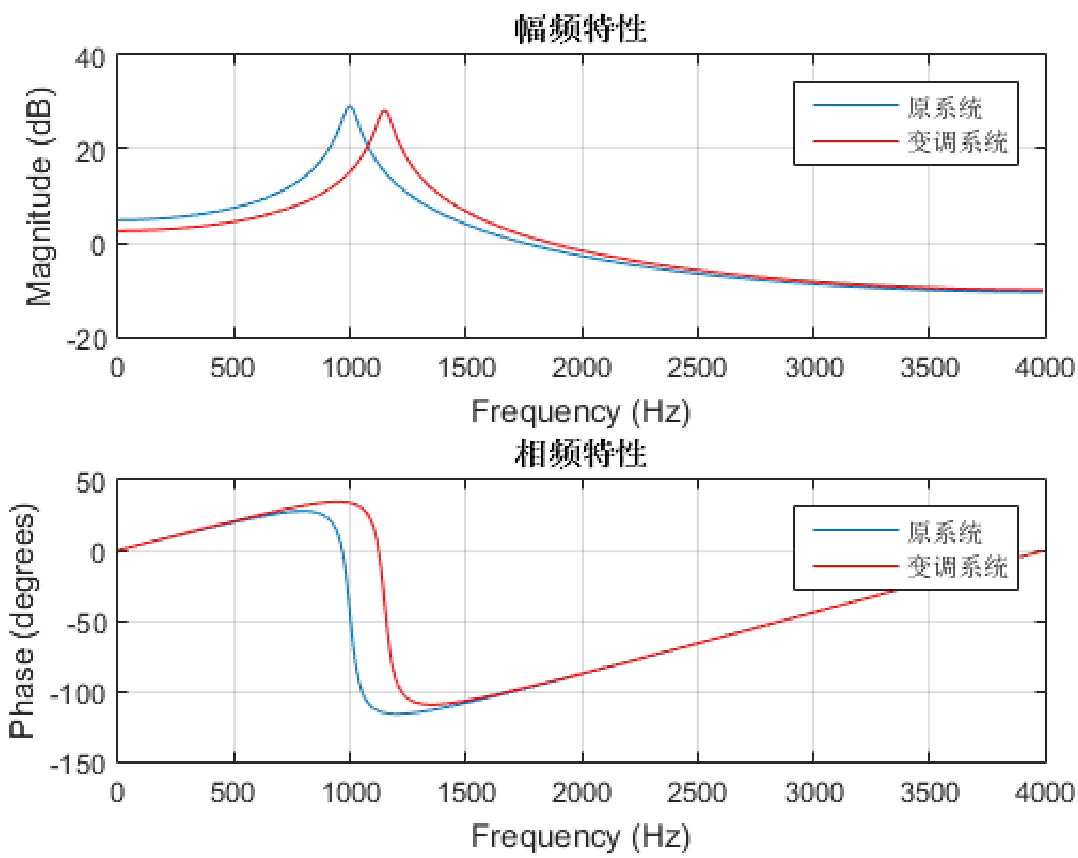
  ```
  [Z,P,~]=tf2zp(b,a);
  [Z_t,P_t,~]=tf2zp(b,a_t);
  figure;
  [Hz1,Hp1,Hl1] = zplane(Z,P);
  hold on;
  [Hz2,Hp2,Hl2] = zplane(Z_t,P_t);
  hold off;
  xlim([-1.4 1.4]);
  set(findobj(Hz2, 'Type', 'line'), 'Color', 'r')
  set(findobj(Hp2, 'Type', 'line'), 'Color', 'r')
  legend([Hp1,Hp2],'原系统','变调系统');
  title('旋转极点');
  ```

  极点幅角绝对值变大, 共振峰频率增大

- 绘制频率响应 (freqz_changetone.m)

```
[H,F]=freqz(b,a,512,fs);
[H_t,F_t]=freqz(b,a_t,512,fs);

figure;
subplot 211
plot(F,20*log10(abs(H)));hold on;
grid on;
xlabel('Frequency (Hz)');ylabel('Magnitude (dB)');
plot(F,20*log10(abs(H_t)),'r');hold off;
legend('原系统','变调系统');
title('幅频特性');

subplot 212
plot(F,angle(H));hold on;
grid on;
hold off;
plot(F_t,angle(H)/pi*180);hold on;
grid on;
xlabel('Frequency (Hz)');ylabel('Phase (degrees)');
plot(F_t,angle(H_t)/pi*180,'r');hold off;
legend('原系统','变调系统');
title('相频特性');
```



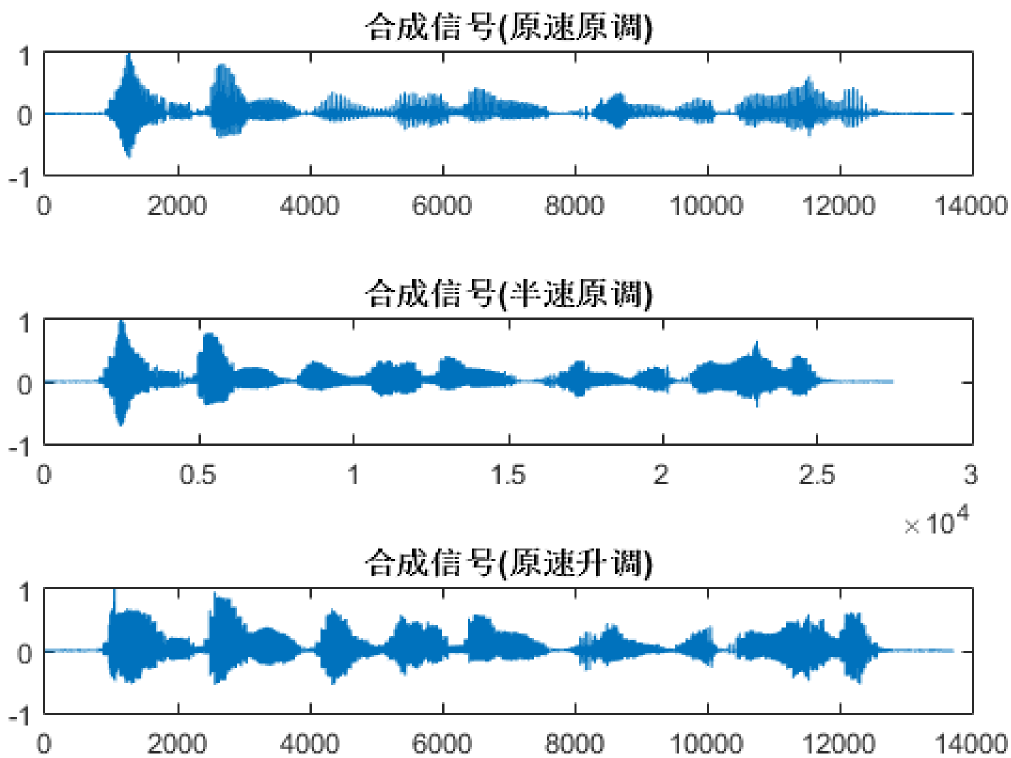
变调后系统共振峰频率变为**1150Hz**

2. **(**练习**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

```
% (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_t = round(PT/2);
if n == 3                   % first loop
    cursor_t = (n-1)*FL+1;    % initialize cursor
    m_t = n;                  % initialize m
end

while m_t == n                % cursor still point into current frame
    exc_syn_t(cursor_t) = 1;
    cursor_t = cursor_t + PT_t;   % next cursor
    m_t = ceil(cursor_t/FL);    % locate next cursor
end

A_t = changetone(A,150,8000);   % ff += 150

s_syn_t((n-1)*FL+1:n*FL) = filter([1,zeros(1,P)],A_t,...
    G*exc_syn_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
% 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
```

试听

- 由男声变为女声



- 波形包络也有变化，原因是改变了基音周期，激励信号$e(n)$变化了

- 时长未改变

变速变调(探究)

1. 容易发现, 变速变调可以完美地结合在一起, 新建 `speechproc1(rv,rt,df)` (speechproc1.m) 实现变速变调功能

```
function H = speechproc1(rv,rt,df)
%speechproc1(rv,rt,df)
%输入:
%   <double>rv: 调速比
%   <double>rt: 调(tiao)调(diao)比, ratio_tone
%   <double>df: 共振峰频率改变量
%输出文件

    % 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);
    % 合成滤波器
    exc_syn = zeros(ceil(L/rv),1);   % 合成的激励信号（脉冲串）
    s_syn = zeros(ceil(L/rv),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

        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
        [Y,zi_pre] = filter(A,[1,zeros(1,P)],s_f,zi_pre);   % keep state
        exc((n-1)*FL+1:n*FL) = Y;
        % exc((n-1)*FL+1:n*FL) = ... 将你计算得到的激励写在这里

        % 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）


        % 变速变调
        FL_v = round(FL/rv);        % change velocity
        PT_t = round(PT/rt);        % change tone
        A_t = changetone(A,df,8000);% change predict sys

        if n == 3                   % first loop
            cursor = (n-1)*FL_v+1;    % initialize cursor
            m = n;                  % initialize m
        end

        while m == n                % cursor still point into current frame
            exc_syn(cursor) = 1;
            cursor = cursor + PT_t;   % next cursor
            m = ceil(cursor/FL_v);    % locate next cursor
        end

        s_syn((n-1)*FL_v+1:n*FL_v) = filter([1,zeros(1,P)],A_t,...
            G*exc_syn((n-1)*FL_v+1:n*FL_v));

    end

    % 保存所有文件
    writespeech('exc_syn_c.pcm',exc_syn);     % _c means Combine both velocity and
    writespeech('syn_c.pcm',s_syn);

    % normalization
    s = normalize(s);
    s_syn = normalize(s_syn);

    sound(s_syn,8000);
    H = plot(s_syn);title('"电灯比油灯进步多了"');

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
```

2. 参数测试

为了方便地进行 `rv rt df` 三个参数的调整，不妨写一个简陋的 `GUI` (gui.m)

```
function varargout = gui(varargin)
% GUI MATLAB code for gui.fig
%      GUI, by itself, creates a new GUI or raises the existing
%      singleton*.
%
%      H = GUI returns the handle to a new GUI or the handle to
%      the existing singleton*.
%
%      GUI('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in GUI.M with the given input arguments.
%
%      GUI('Property','Value',...) creates a new GUI or raises the
```

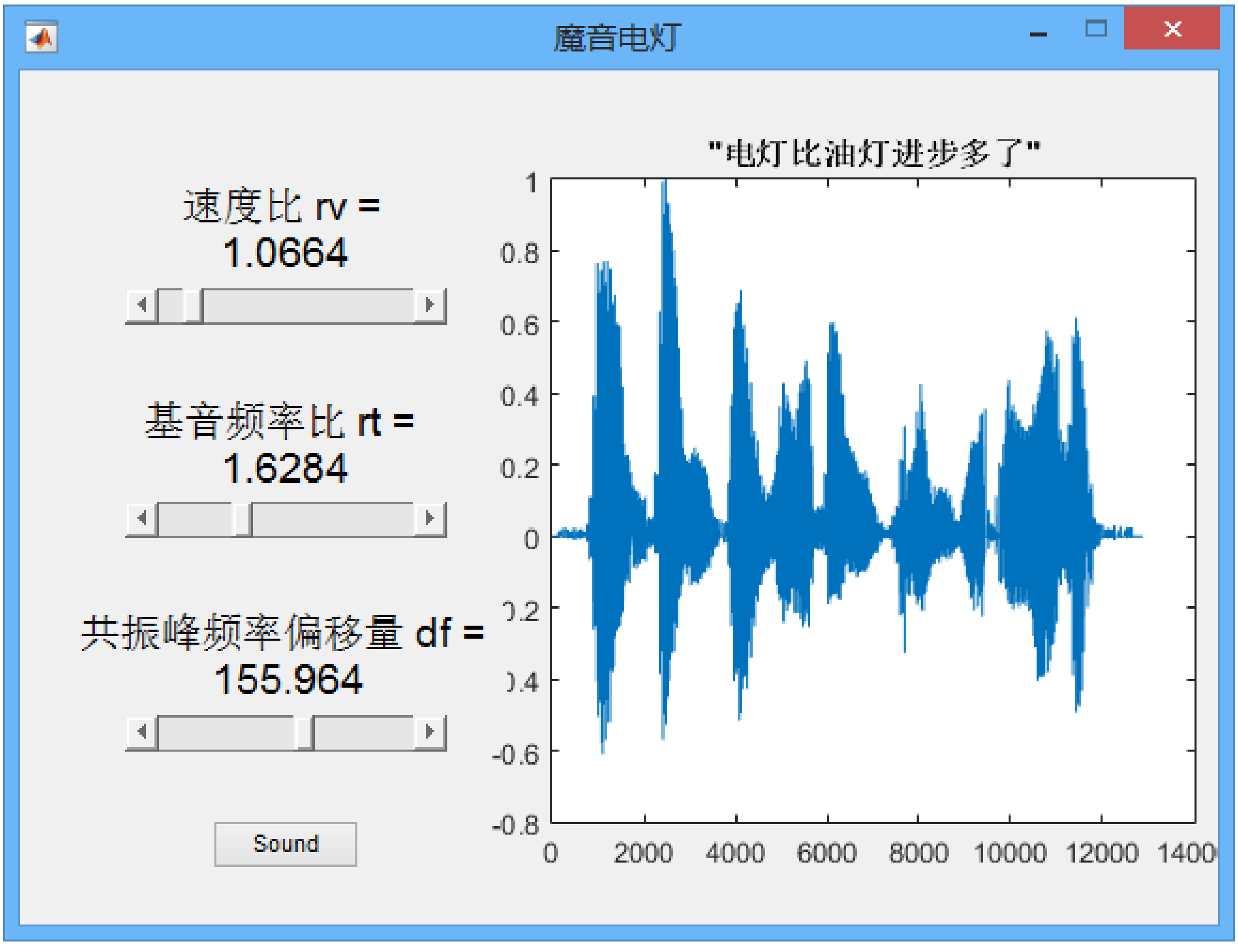
```
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before gui_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to gui_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help gui

% Last Modified by GUIDE v2.5 31-Jul-2015 22:09:11

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @gui_OpeningFcn, ...
                   'gui_OutputFcn',  @gui_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before gui is made visible.
function gui_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to gui (see VARARGIN)

% Choose default command line output for gui
handles.output = hObject;

% Initialize
handles.rv = 1;
handles.rt = 1;
handles.df = 0;

% Display text
```

```
handles.text_rv.String = ['速度比 rv = ',num2str(handles.rv)];
handles.text_rt.String = ['基音频率比 rt = ',num2str(handles.rt)];
handles.text_df.String = ['共振峰频率偏移量 df = ',num2str(handles.df)];

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes gui wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = gui_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on slider movement.
function slider_rv_Callback(hObject, eventdata, handles)
% hObject    handle to slider_rv (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider

handles.rv = get(hObject,'Value');
handles.text_rv.String = ['速度比 rv = ',num2str(handles.rv)];

guidata(hObject,handles);

% --- Executes during object creation, after setting all properties.
function slider_rv_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider_rv (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end


% --- Executes on slider movement.
function slider_rt_Callback(hObject, eventdata, handles)
% hObject    handle to slider_rt (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider

handles.rt = get(hObject,'Value');
handles.text_rt.String = ['基音频率比 rt = ',num2str(handles.rt)];

guidata(hObject,handles);

% --- Executes during object creation, after setting all properties.
function slider_rt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider_rt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end


% --- Executes on slider movement.
function slider_df_Callback(hObject, eventdata, handles)
% hObject    handle to slider_df (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider

handles.df = get(hObject,'Value');
handles.text_df.String = ['共振峰频率偏移量 df = ',num2str(handles.df)];

guidata(hObject,handles);

% --- Executes during object creation, after setting all properties.
function slider_df_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider_df (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end


% --- Executes on button press in button_sound.
function button_sound_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to button_sound (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

handles.axes1 = speechproc1(handles.rv,handles.rt,handles.df);

guidata(hObject,handles);
```

效果如下



- 拖动滑块即可改变参数
- 按下 `Sound` 键可以听到声音，并绘制出波形

写在最后

- 语音合成作业相比于音乐合成简单不少(因为最复杂的求基音周期已经实现好了)，熟练掌握了 `filter` 函数的用法
- 由于处理是逐帧进行的，因此必须保持滤波器的状态，若不保持滤波器状态，虽然合成语音内容还能听得出，但是会伴随显著的噪声，影响合成质量
- 简单了解了**GUI**的原理，实现了简单的**GUI**程序