语音合成大作业

xuan

1.语音预测模型

（1）对差分方程直接做 Z 变换,$E(z) = S(z) - a_1 z^{-1} S(z) - a_2 z^{-2} S(z)$;

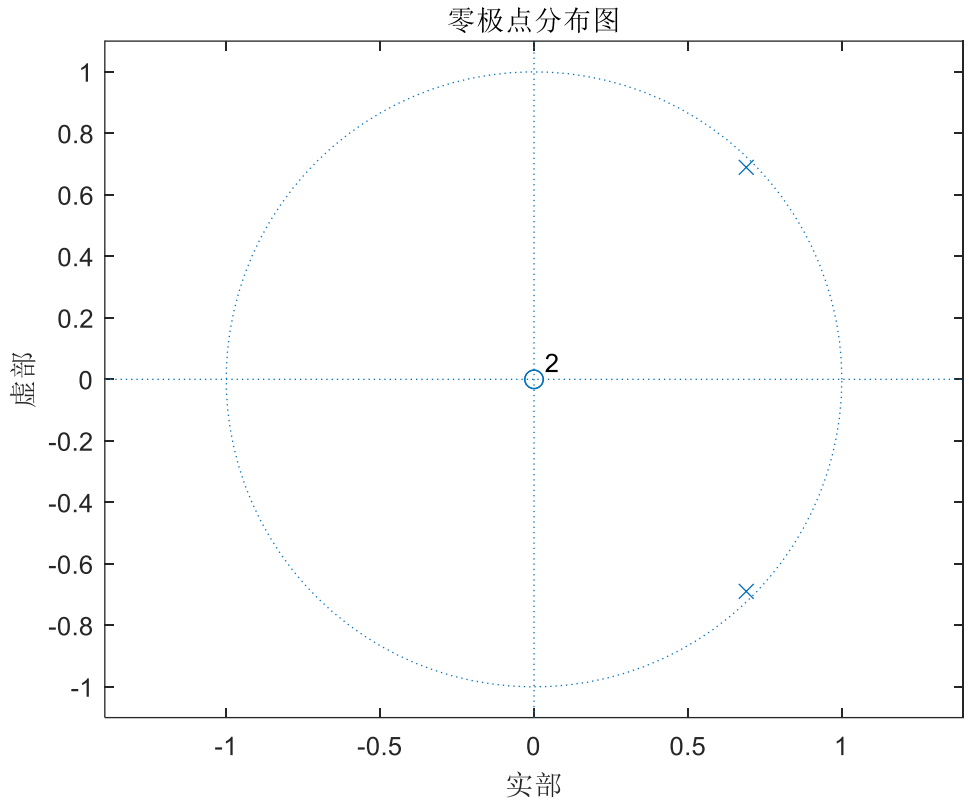
可求得传递函数为$H(z) = \frac{S(z)}{E(z)} = \frac{1}{1 - a_1 z^{-1} - a_2 z^{-2}} = \frac{z^2}{z^2 - a_1 z - a_2}$;再应用 tf 函数，b,a 分别为分子分母的系数降幂排序，得到系统函数

求共振峰频率需用到以下公式$f = \frac{w}{2\pi} = \frac{\Omega}{2\pi T} = \frac{abs(极点幅角)}{2\pi T}$；$T$为采样周期，根据本实验说明中相关信息，采样是以每 10ms 采 80 个点进行的，即T = 0.01/80s;转化为求极点;
求极点可用[r,p] = residuez(b,a)或 p=roots(a);
求得两个极点 p（共轭）分别为 0.6895 + 0.6894i 和 0.6895 - 0.6894i,
代入两个极点的幅角得 f=999.9447

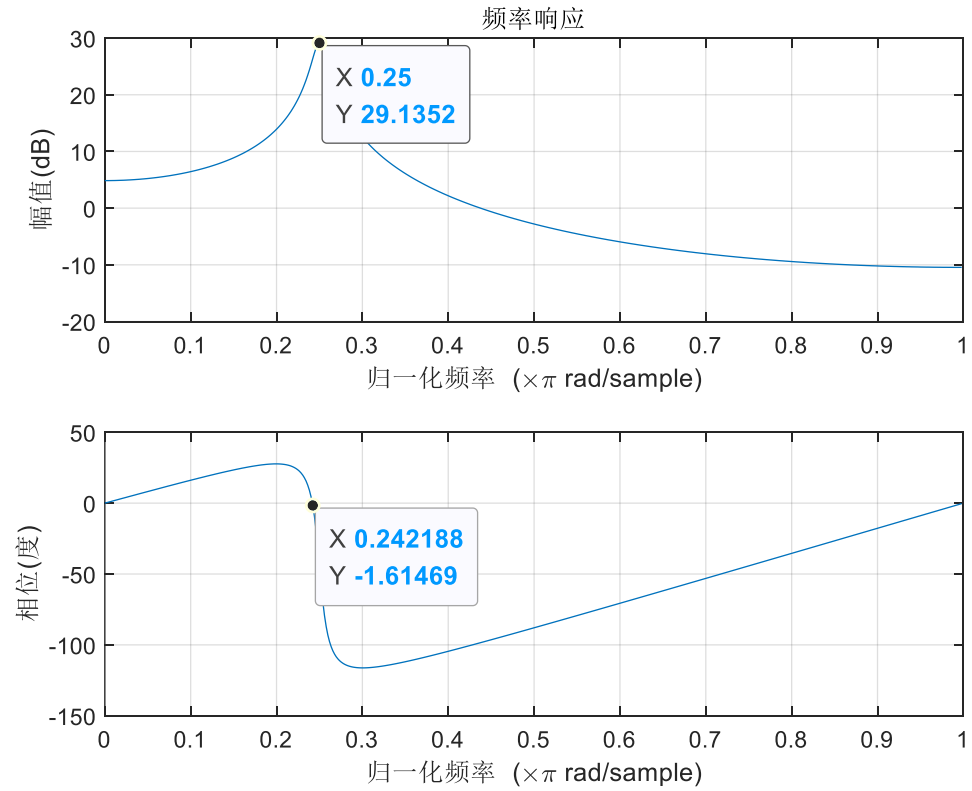
```
sys =

                1
  ------------------------------
  1 - 1.379 z^-1 + 0.9506 z^-2

Sample time: 0.000125 seconds
Discrete-time transfer function.


f =

  999.9447
  999.9447
```

用 zplane（b,a），freqz(b,a)，impz(b,a) 分别绘出零极点图，频率响应和单位样值响应。
用 filter（b,a,x） 绘出单位样值响应。b 为系统函数的分子系数，a 为系统函数的分母系数,x 为单位样值响应，设了 400 个数值。

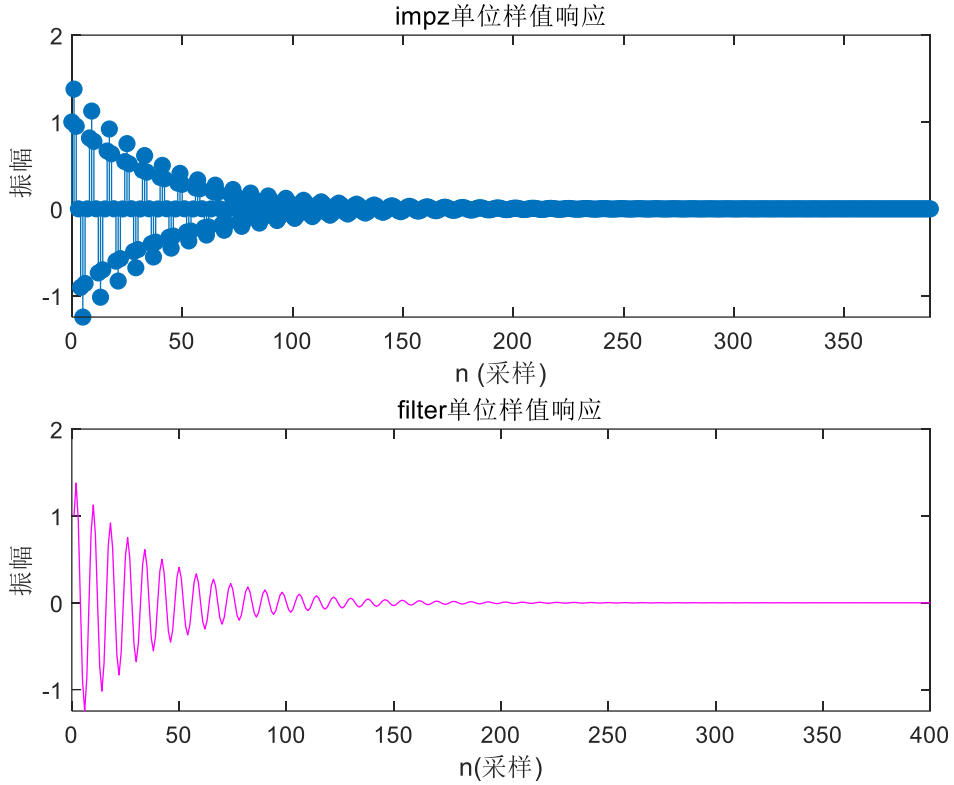
零极点分布图、频率响应、单位样值响应图如下



即原点处有一个二阶零点，在靠近$\pm\frac{\pi}{4}$相位的位置上有一对共轭的一阶极点



从频率响应图看出，在 0.25π rad 的共振峰频率处，有共振峰（幅值最大），同时相位为 0



其中用 x = [1 zeros(1,399)]来产生单位样值激励，399 表示最大仿真时间。由对比图可看出 filter 和 impz 函数绘制的单位样值响应是一样的。

以下题目的代码均在 speechproc.m 中

（2）speechproc 的基本流程：

语音预测模型：即语音信号 s(n) 送入预测滤波器，得到预测残差 e(n)

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

a) 定义相关参数：帧长、窗长、预测系数个数(N)

b) 载入语音信号 s(n)并计算帧数，定义各滤波器的状态、输入和输出

c) 加汉明窗处理语音并计算预测系数（A=[1,-a1,⋯,-a10]）

d). 观测 27 帧时，预测系统的零极点图

即 zplane(b,a),a 为差分方程左侧系数 1，b 为右侧系数 A

e)用 filter 函数,以 s_f 为输入，逐帧计算出激励并存入 exc 中，需保持 filter 状态

f).用 filter 函数和上一步计算出的 exc 逐帧重建语音，存入 s_rec 中，需保持 filter 状态

g) 得到 exc 后,可以计算基音周期 PT 和合成激励的增益 G

h).类比单位样值串生成合成激励 exc_syn，并利用其和 filter 函数逐帧生成合成语音，存入 s_syn 中

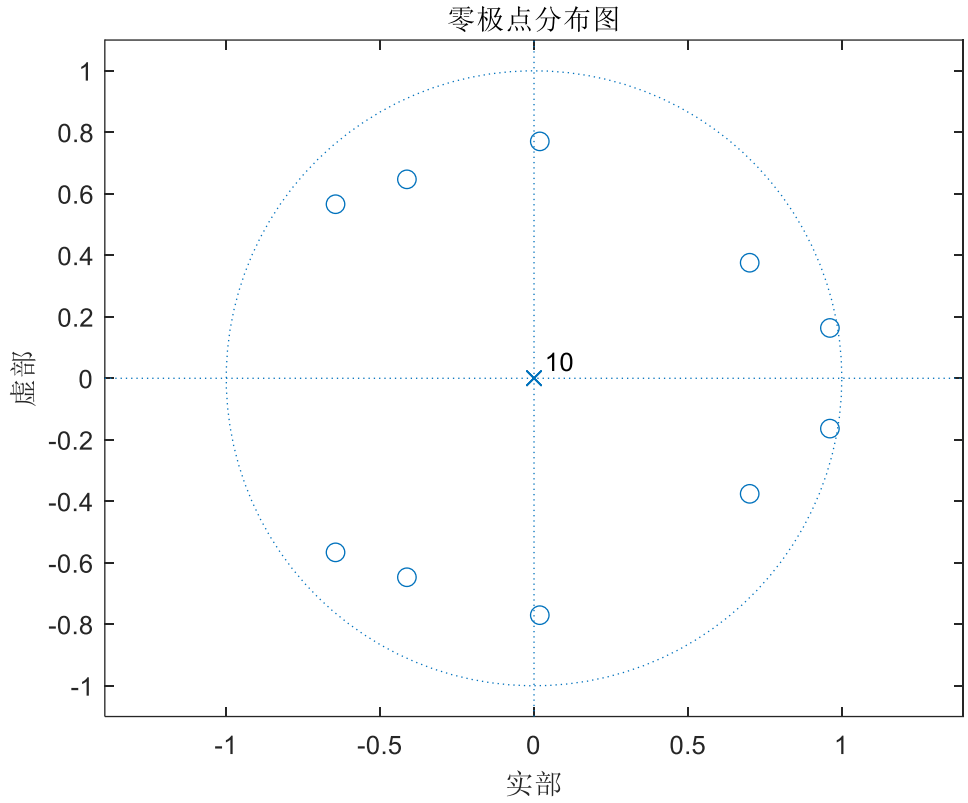
i). 将合成激励的长度增加一倍得到 exc_syn_v，再次利用 filter 函数生成变速不变调的语音，存入 s_syn_v 中

j). 将基音周期减小一半，共振峰频率增加 150Hz，得到合成激励 exc_syn_t，再次利用 filter

函数生成变速不变调的语音，存入 s_syn_t 中
k) 试听语音
l) 保存所有文件
（3）对预测系统，e(n)是输出，s(n)是输入，

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

A 对应差分方程右侧系数，所以用 zplane(A,1)绘制
零极点分布图如下：



原点处为 1 个十阶极点，圆内还有 5 对关于实轴对称的零点。
（4）因为要对每帧都用 filter 计算 激励信号 e(n)，预测模型系数 {ai}是变化的
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。
查阅 matlab 的 help 后,用法是[y,zf] = filter(b,a,x,zi)
zi is the initial conditions for the filter delays;zf is the final conditions of the filter delays。
代码如图

```
% (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_pre, zf_pre] = filter(A, 1, s_f, zi_pre);
%左边的zf_pre为本次循环得到的滤波器的最终状态，需要以其作为下一次循环的filter函数的zi初始状态
%第一次循环的zi为0
zi_pre = zf_pre;
exc((n-1)*FL+1:n*FL) = exc_pre;%将你计算得到的激励写在这里
```

（5）语音重建模型公式是$\hat{s}(n) - \sum_{k=1}^{N} a_k \hat{s}(n - k) = x(n)$ (本实验预测残差 e(n)即为 x(n))
此时 A 对应差分方程左侧系数。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
[srec,zf_rec] = filter(1,A,exc,zi_rec);
%由激励exc得到重建语音srec,同样要维持滤波器状态不变，用的是zi_rec和zf_rec
zi_rec = zf_rec;
s_rec((n-1)*FL+1:n*FL) = srec;%将你计算得到的重建语音写在这里
```

（6）刚开始按照 help 直接用 sound(exc,Fs,16)等听三个声音，发现背景音很嘈杂，查阅 CSDN 的 https://blog.csdn.net/weixin_45770896/article/details/110678692 发现 sound（Y，FS）的参数 Y 中的值需要在-1.0 <= y <= 1.0 的范围内。若超出该范围的值将被裁剪。

```
% 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
```

而从 可知从 PCM 文件读入的音频为 16bit，需要除去 2^15 将其限定在[-1, 1]范围中，否则会有很大噪音。

后来自己改进了 sound 用向量并列表示要听的三个信号，不必加 pause，会自动停顿。

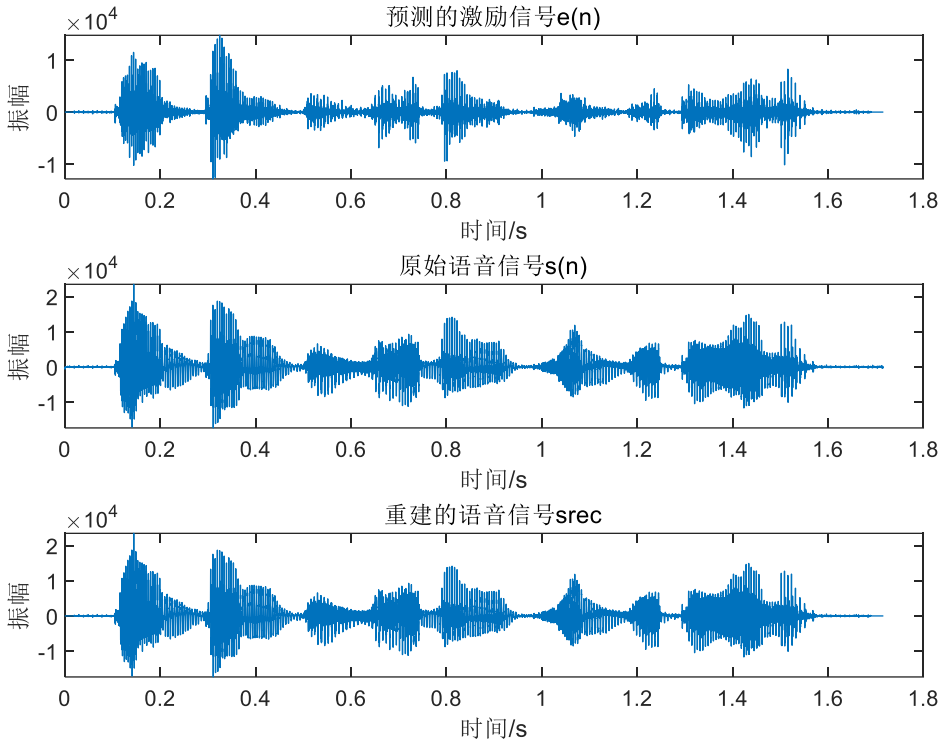
代码如下

```
% (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
Fs = 8000;%采样频率
sound(exc/2^15,Fs);
pause(2);
sound(s/2^15,Fs);
pause(2);
sound(s_rec/2^15,Fs);
```

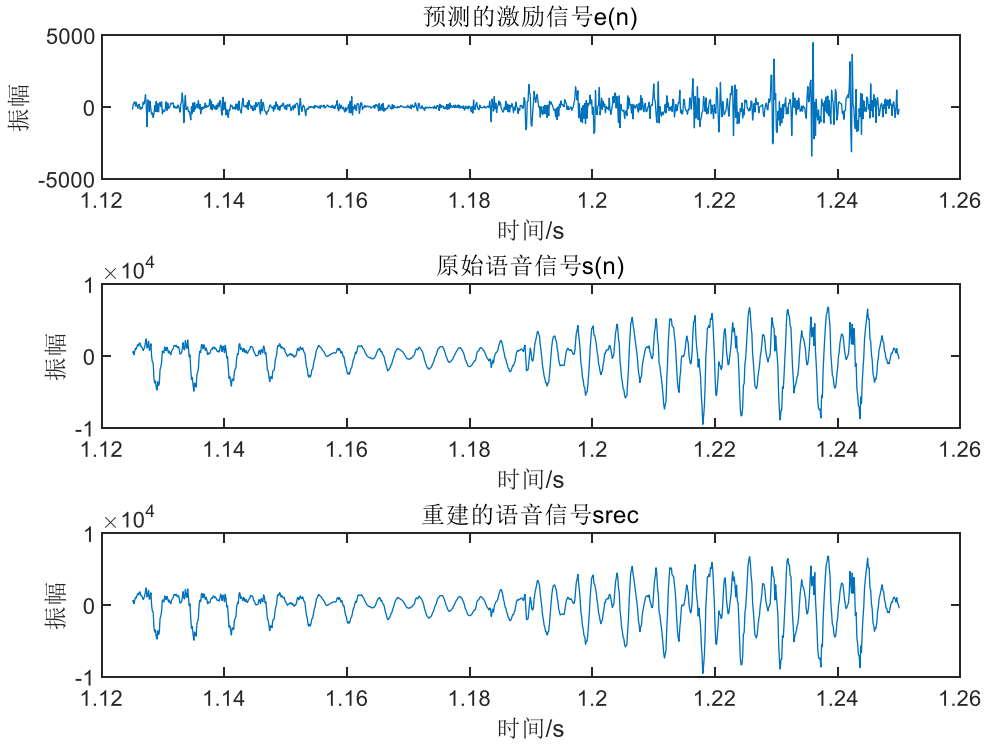
听上去 s(原始语音)和 s_rec（重建语音）基本相同，是清晰饱满的人音；而预测得到的激励性（exc）则比较轻，有种机械音的感觉，也还有一点沙沙的杂音。

非原创等级：借鉴参考。

时域波形如下



选择一小段观察:



可见原始语音和重建语音仍然基本相同，而预测的激励信号噪音更多，频率更快，且振幅更小。

2.语音合成模型

（7）$x(n) = \sum_{k=0}^{NS-1} \delta(n - kN)$

设 Fs 为采样频率，f 为单位样值串的频率，持续时间为 last_time,则

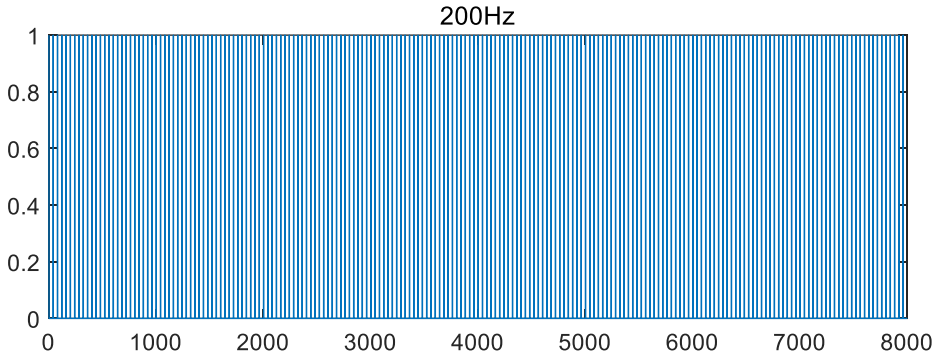
$$N = \frac{F_s}{f}; NS = f * last\_time$$

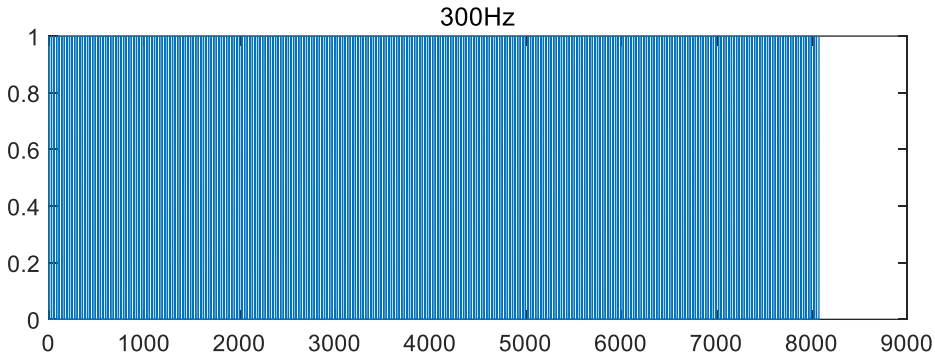
为便捷使用，封装成一个函数，输入为单位样值串的频率、采样频率、信号持续时间，输出为单位样值串。

因为也是放在 speechproc 函数中播放的，而由于 sound 函数并非同步执行，不会阻塞其它指令，故听起来所有声音会重叠，需要 pause 几秒等待上一个音频播放完毕。代码如下

```
function signal = unit_sample(f,last_time,Fs)
%生成特定频率、持续时间和采样频率的单位样值串
% last_time: 信号持续时间
% Fs: 采样频率
% f: 单位样值串的频率
signal = zeros(round(Fs * last_time), 1);
NS = round(last_time * f);
N = round(Fs / f);
k = 0:NS - 1;
signal(k * N + 1) = 1;
end

%试听单位样值串
pause(6);
last_time = 1;
s1 = unit_sample(200,last_time,Fs);
s2 = unit_sample(300,last_time,Fs);
sound([s1;s2],Fs);
```

区别是 300Hz 的信号的音调比 200Hz 的更高

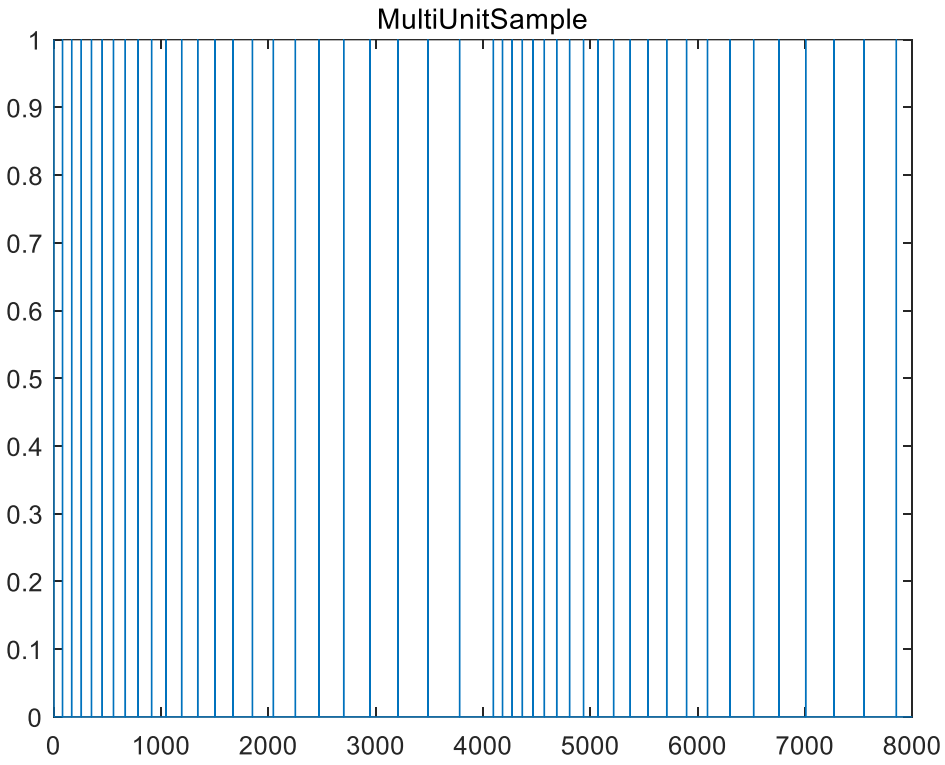




（8）代码如下:

```
%试听单位样值串
pause(2);
last_time = 1;
sound(MultiUnitSample(last_time,Fs),Fs);
```

```
function signal = MultiUnitSample(last_time, Fs)
%生成基音周期随时间变化的单位样值串
 % last_time: 信号持续时间
% Fs: 采样频率
 L = round(Fs * last_time);%信号长度
 signal = zeros(L, 1);
 n = 1;
while n <= L
      signal(n) = 1;
      m = floor(n/80);%10ms的分段
      PT = 80 + 5 * mod(m, 50);
      n = n + PT;
end

end
```

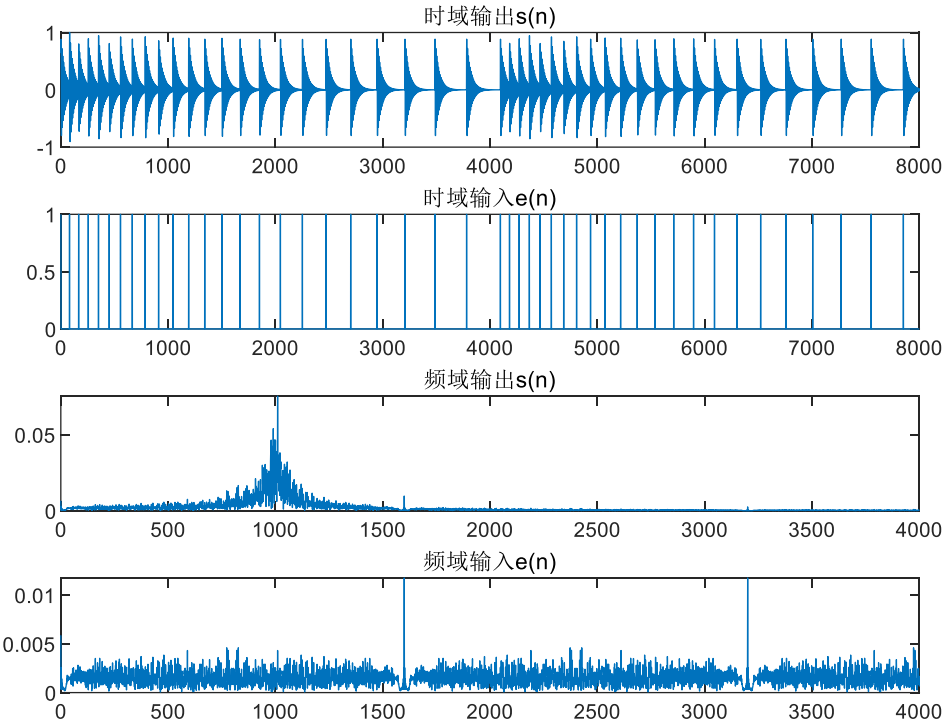


听上去像喷气的声音，有点像电音。

（9）代码如下

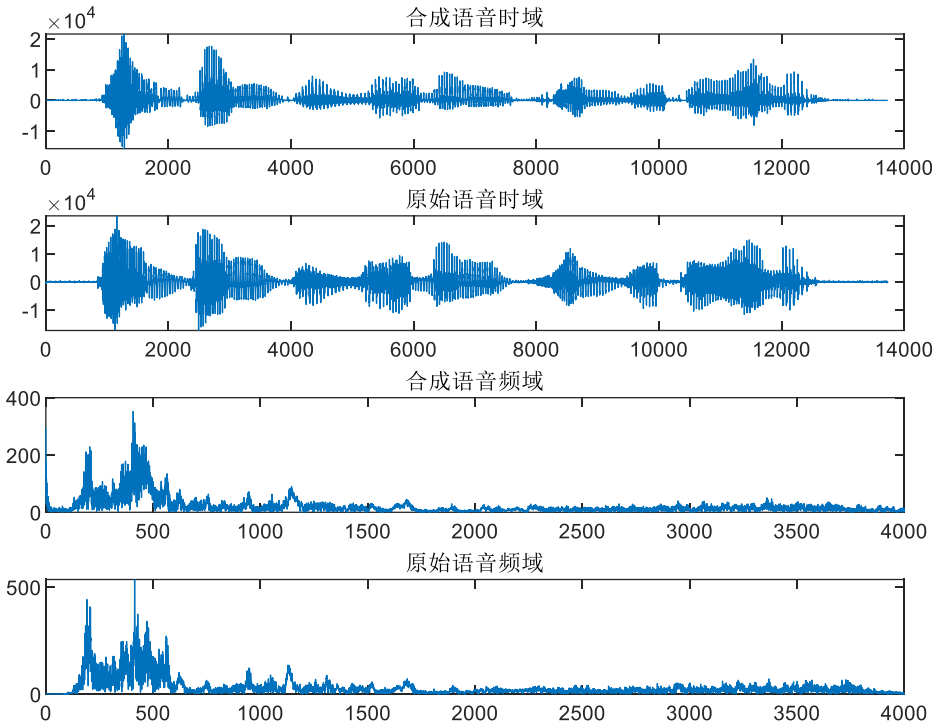
```
%把MultiUnitSample作为输入e(n)输入到ex_1_1.m的滤波器，输出为s(n)
a1 = 1.3789;a2 = -0.9506;%为参数赋值
b = [1, 0, 0];%定义差分方程右侧系数(e(n))
a = [1, -a1, -a2];%定义差分方程左侧系数(s(n))
s_multi = filter(b,a,s3);
pause(3);
sound([s_multi/max(abs(s_multi));s3],Fs);
figure;
subplot(4,1,1);plot(s_multi/max(abs(s_multi)));title('时域输出s(n)');
subplot(4,1,2);plot(s3);title('时域输入e(n)');
subplot(4,1,3);fft_plot(s_multi/max(abs(s_multi)),Fs);title('频域输出s(n)');
subplot(4,1,4);fft_plot(s3,Fs);title('频域输入e(n)');
```

听上去滤波后的 s 更清脆些，没有了 e(n)的刺耳电流音。
绘制的时域和频域波形如图



从时域上看，s(n)像是在 e(n)上叠加了周期指数衰减的包络，使得听起来的电流感减少。
从频域上看，通过滤波器之后的 s(n)能量主要集中在 1000Hz 左右（即滤波器的共振峰频率附近），低频增强了。
（4）合成语音和原始语音基本相同，但原始语音更饱满一些
代码如下：其中 pos 和 zi_syn 只在开头定义常数部分定义，防止重定义以及保证一帧的第一个脉冲是和上一帧的最后一个脉冲去比的。

```
% (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
%仿照1.2.2的（8），不同的是PT是在单帧中是固定的常数
while pos <= n * FL
    exc_syn(pos) = G;
    pos = pos + PT;
end
%exc_syn((n-1)*FL+1:n*FL) =  将你计算得到的合成激励写在这里
%仿照（5），由激励exc_syn得到合成语音s_syn,同样要维持滤波器状态不变，用的是zi_syn和zf_syn
[s_syn1,zf_syn] = filter(1,A,exc_syn((n-1)*FL+1:n*FL),zi_syn);
zi_syn = zf_syn;
s_syn((n-1)*FL+1:n*FL) = s_syn1;%将你计算得到的合成语音写在这里
```

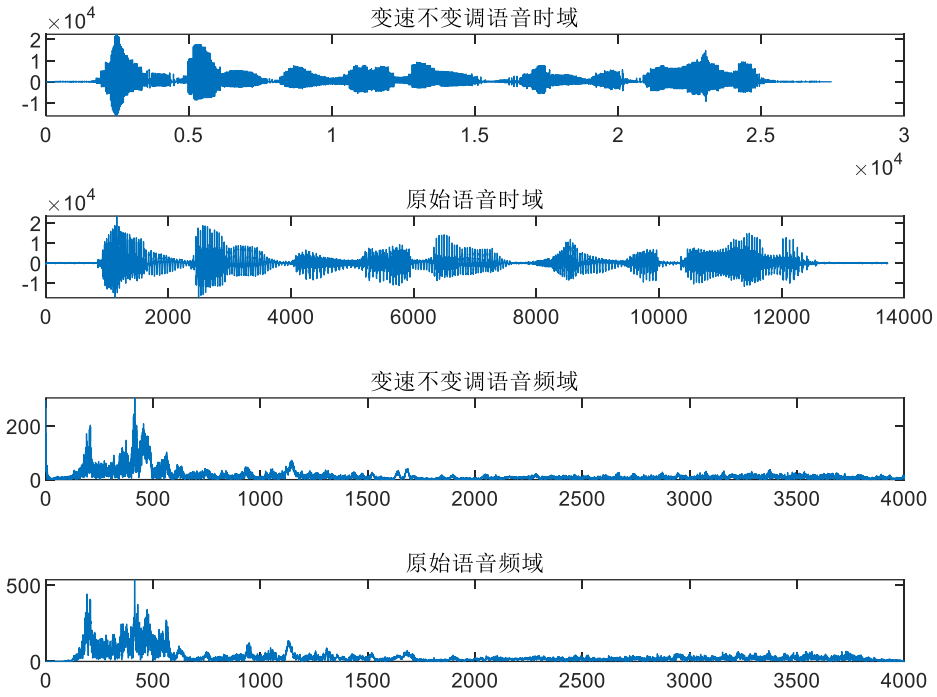


可以看到，时域和频域分布基本相同，所以听起来基本相同

3.变速不变调

听起来感觉声调基本没有变化，而速度慢了一倍，机械感增加了。
代码如下：同理，FL_v=2*FL; pos_v = 2 * FL_v + 1;zi_syn_v = zeros(P, 1);只在定义常数部分初始化

```
% （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。
%仿照（10），不同的是FL变成了FL_v
while pos_v <= n * FL_v
    exc_syn_v(pos_v) = G;
    pos_v = pos_v + PT;
end
[s_syn1_v,zf_syn_v] = filter(1,A,exc_syn_v((n-1)*FL_v+1:n*FL_v),zi_syn_v);
zi_syn_v = zf_syn_v;
s_syn_v((n-1)*FL_v+1:n*FL_v) = s_syn1_v;%将你计算得到的加长合成语音写在这里
% 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
```



从时域看，可以看到变速不变调信号的波形虽然持续时间是原始语音信号的两倍，但波形基本相同，所以音调没有明显变化；
从频域看，与原始语没有太大差别。

4.变调不变速

(1)
由 1.语音预测模型的(1)公式可知，共振峰频率与极点幅角绝对值有关。所以把原极点根据其原幅角正负乘上$e^{i\theta}$*或*$e^{-i\theta}$即可（θ为 150Hz 对应的角度），计算代码如下

```
clear all, close all, clc;
a1 = 1.3789;a2 = -0.9506;%为参数赋值
%为方便起见，把s(n)看做输出，e(n)看做输入
a = [1, -a1, -a2];%定义差分方程左侧系数
b = [1, 0, 0];%定义差分方程右侧系数
[r,p,k] = residuez(b,a);%求出原极点
angle = 150 * 2 * pi / 8000;%θ
p = p .* exp(sign(imag(p)) *1i *  angle);%已知150Hz对应的极点
[b,a] = residue(r,p,k);
a
```

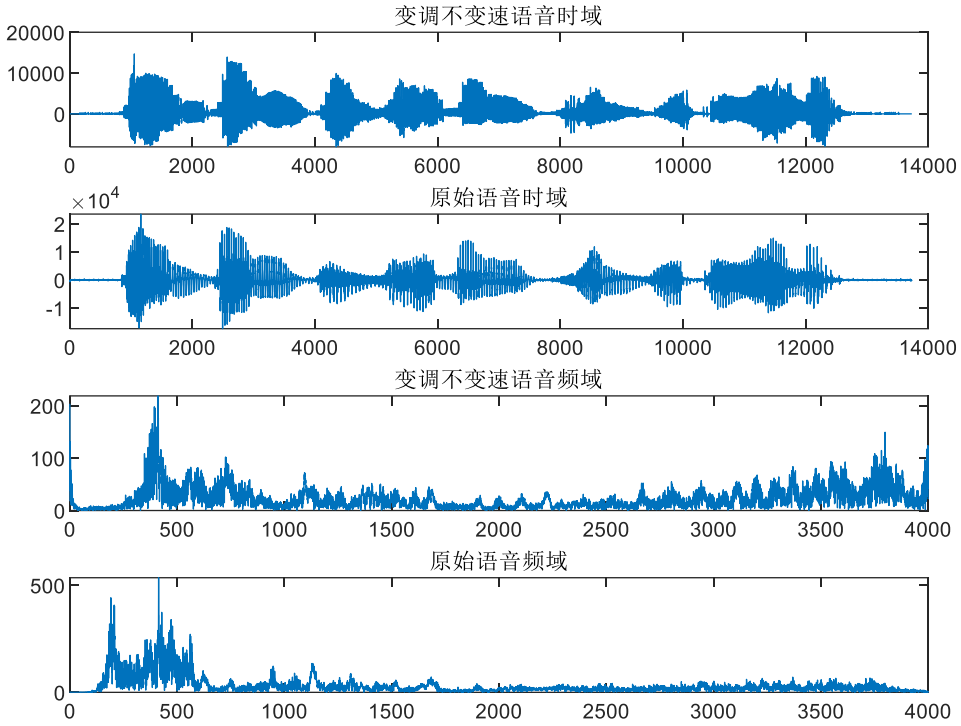
命令行窗口

```
a =

    1.0000   -1.2073    0.9506
```

可得 a1 = 1.2073;a2=-0.9506

(2)代码如下：同理 pos_t 和 zi_syn_t 只在定义常数部分初始化

```
% (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~
while pos_t <= n * FL
    exc_syn_t(pos_t) = G;
    pos_t = pos_t + round(PT/2);
end
[r,p,k] = residuez(1,A);%求出原极点,生成模型中A是差分方程左侧系数
angle = 150 * 2 * pi / 8000;%θ
p = p .* exp(sign(imag(p)) *1i *  angle);%已知150Hz对应的极点
[B_t,A_t] = residue(r,p,k);
[s_syn1_t,zf_syn_t] = filter(1,A_t,exc_syn_t((n-1)*FL+1:n*FL),zi_syn_t);
zi_syn_t = zf_syn_t;
s_syn_t((n-1)*FL+1:n*FL) = s_syn1_t;%将你计算得到的变调合成语音写在这里
```



试听后感觉音调明显更高更尖，但是持续时间和速度与原始语音并没有变化。
从时域看，变调不变速信号的波形则显得更密，所以音调变高。
从频域看，变调不变速信号多出了不少高频分量，所以音调变高。