Matlab 语音处理实验报告

暮月

2020 年 8 月 12 日

目录

环境与依赖说明 本次实验使用的是 Matlab R2020a Update3，主要使用了 XXX。Matlab 设置为使用 utf-8 编码，所有代码文件无特殊情况均为此编码，注释以英文为主。

1 语音预测模型

1.1 滤波器

对滤波器差分方程做 Z 变换：

$$e(n) = s(n) - a_1 s(n-1) - a_2 s(n-2)$$
$$E(z) = S(z) - a_1 z^{-1} S(z) - a_2 z^{-2} S(z)$$

所以传递函数为：

$$H(z)=\frac{S(z)}{E(z)}=\frac{1}{1-a_1z^{-1}-a_2z^{-2}}=\frac{z^2}{z^2-a_1z-a_2}$$

从而共振峰频率为：

$$\begin{aligned}f&=\frac{\omega}{2\pi}=\frac{\Omega}{2\pi T}=\frac{|\angle p_i|}{2\pi T}\\&=\frac{\text{abs}(\text{angle}(p_i))}{2\pi T}\end{aligned}$$

使用`roots([1, -1.3789, 0.9506])`求得两极点为 $0.6895+0.6894i$ 和 $0.6895-0.6894i$ 再代入上式得到分子 Ω 为 0.7854rad，即 0.25πrad。根据说明，取采样间隔为 10ms80 样本点，即 0.125ms，从而得到 $f=\frac{0.7854\text{rad}}{2\pi\cdot 0.125\text{ms}}=1000.00\text{Hz}$。

使用`zplane`、`freqz`、`impz`和`filter`得到相关图像。

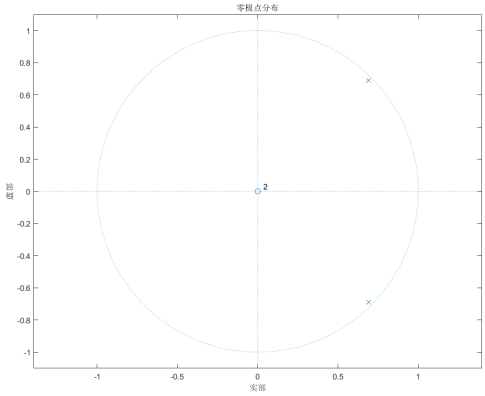


图 1: 滤波器的零极点分布

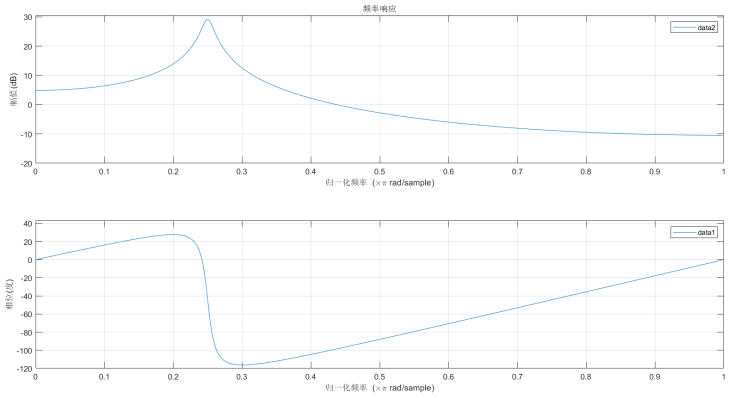
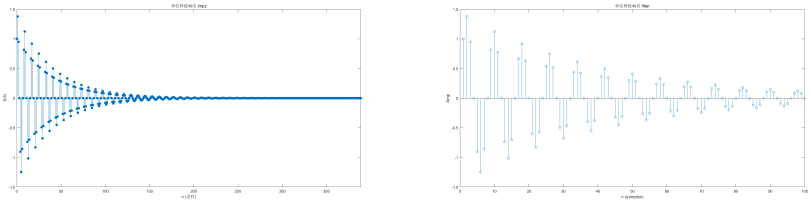


图 2: 滤波器的频率响应

如图1，该滤波器在原点处有一二阶零点，在 $\pm\frac{\pi}{4}$ 靠近单位圆 2 处有一对共轭一阶极点。如图2，该滤波器在 0.25πrad 处（即上方计算的共振峰频率处）有共振峰，同时相位为 0。

该滤波器的单位样值响应如图3。这里由于`filter`的参数只取了 100 个采样点，而不是`impz`默认生成的 400 个采样点，故结果比较稀疏。实际上，两个函数得到的结果完全一致。



(a) impz (b) filter

图 3: 滤波器的单位样值响应

1.2 speechproc 的基本流程

speechproc读取一段音频，对其进行一系列处理后保存处理结果，并将控制权交还调用脚本。流程如下：

1. 定义常数：帧长、窗长、预测系数个数、音频文件
2. 载入语音，并计算相关参数，定义合成信号与滤波器状态等变量
3. 循环处理每帧语音
   (a) 计算预测系数
   (b) 27 帧时，显示预测系统的零极点分布图
   (c) 使用filter计算本帧语音s_f的激励，存入exc
   (d) 使用filter和exc重建语音，存入s_rec
   (e) 计算基音周期PT、合成激励的能量G
   (f) 生成合成激励exc_syn，并合成语音s_syn
   (g) 将合成激励的长度增加一倍，合成语音s_syn_v
   (h) 将基音周期减小一半，将共振峰频率增加 150Hz，合成语音s_syn_t
4. 试听语音
5. 保存语音与激励

1.3 预测系统的零极点分布

预测模型为：

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

注意到这里的输入是 $s(n)$，输出是残差 $e(n)$。故只需将等式右边的系数作为zplane的第一项系数便可以得到预测系统的零极点分布图。添加如下代码：

```
if n == 27
% (3) 在此位置写程序，观察预测系统的零极点图
    figure;
    zplane(A, 1);
```

```
    title('预测系统的零极点分布图');
end
```

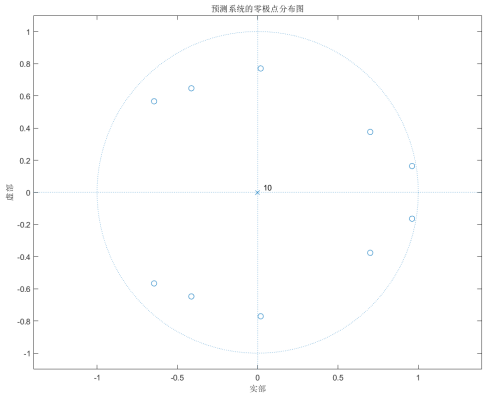


图 4: 预测系统的零极点分布图

可以看到，27 帧时的预测模型有一个 10 阶极点和五对共轭零点。由于其为声道模型的逆系统，此时的声道模型应有一个 10 阶零点和五对共轭极点。

1.4 用 `filter` 计算激励

根据 Matlab 文档，`[y, zf] = filter(b, a, x, zi)`接受系统的分子 b、分母 a、输入 x、滤波器延迟初始条件 zi，返回输出 y 和延迟最终条件 zf。因而使用如下代码存储激励和滤波器状态：

```
[exc((n - 1) * FL + 1 : n * FL), zi_pre] = ...
    filter(A, 1, s_f, zi_pre);
```

1.5 用 `filter` 和 `exc` 重建语音

与上一节类似，注意重建语音与预测模型的零极点交换，故使用如下代码存储重建的语音和滤波器状态：

```
[s_rec((n - 1) * FL + 1 : n * FL), zi_rec] = ...
    filter(1, A, exc((n - 1) * FL + 1 : n * FL), zi_rec);
```

1.6 语音、激励、重建语音的区别

阅读 Matlab 关于音频部分的文档，注意到`sound`接受的第一个参数音频为 $[-1, 1]$ 间实数组成的向量。故应将从 voice.pcm 读入的语音进行量化后使用。

使用 Matlab 确认语音向量的值为 16 位符号数，故除以 32768 进行量化，再使用`sound`，以采样频率 8000Hz 进行播放。代码如下：

```
sound([s; exc; s_rec] / 32768, 8000);
```

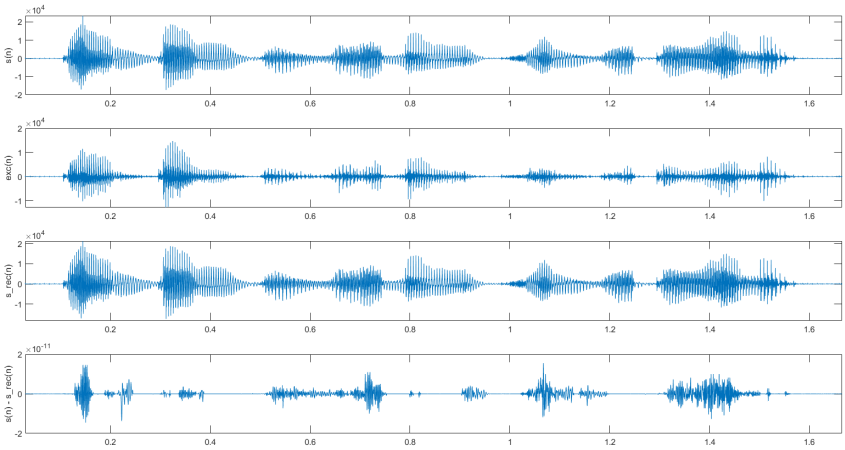


图 5: 语音、激励、重建语音时域

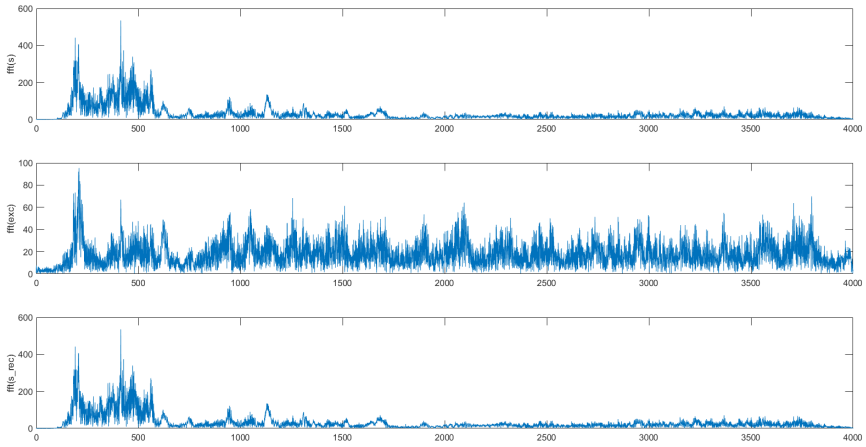


图 6: 语音、激励、重建语音频域

再使用`plot`绘制波形，并仿照 Matlab 关于`fft`的文档生成并绘制语音、激励、重建语音的频域图形。

观察图5中的波形，语音和重建语音几乎没有什么区别，差值普遍在 10^{-11} 以下。转换到频域（图6）后，语音和重建语音肉眼难见区别，相比之下激励各频段的幅度基本一致，没有语音明显的低频分量占比大的现象。使用`sound`播放出的声音，语音和重建语音听不出什么区别，而激励会有较多杂音，听上去有老旧收音机的感觉且声音较小，应是频域图中较多的高频分量而普遍较小的幅度导致。

使用`linkaxes`将语音、激励、重建语音的刻度同步，放大后截取一小段波形如图7。可以看出，激励因为有更多的高频分量，变化更为频繁，这与前面分析一致。语音和重建语音仍然肉眼难见明显区别。

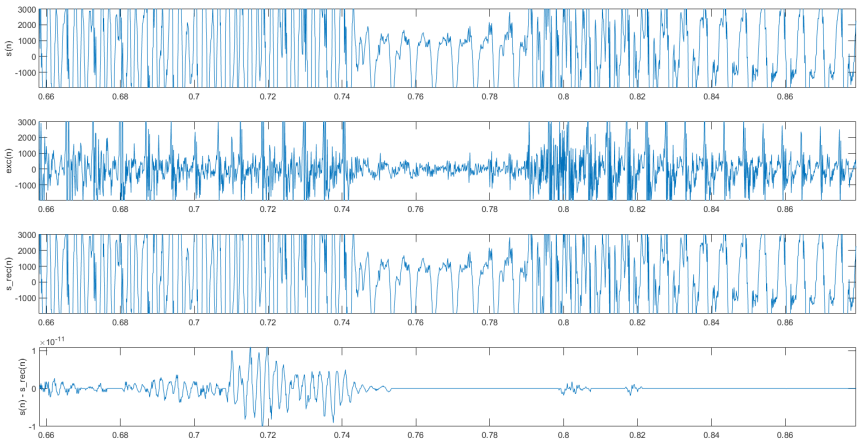


图 7: 语音、激励、重建语音时域局部

2 语音合成模型

2.1 基音周期固定的单位样值串

单位样值串由下式定义：

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

由采样率 8kHz，单位样值串 200Hz，计算得：

$$N = \frac{F_s}{f} = \frac{8000}{200} = 40$$
$$NS = f \cdot t = 200 \times 1 = 200$$

可以使用如下函数实现单位样值串的生成：

```
function signal = unit_samples(sample_freq, freq, duration)
signal = zeros(1, round(sample_freq * duration));
NS = round(freq * duration);
N = round(sample_freq / freq);

i = 0 : NS - 1;
signal(i * N + 1) = 1;
end
```

听起来 300Hz 的单位样值串音调更高，同时播放时是协和的。使用音调矫正器具测量得到，200Hz 约为 $G_3^{\#}$，300Hz 约为 $D_4^{\#}$，正好相差纯五度，同时播放为协和的和声。

2.2 基音周期变化的信号

使用循环实现较为简单，代码如下：

```
function signal = varied_unit_samples(sample_freq, duration)

sig_len = round(sample_freq * duration);

signal = zeros(1, sig_len);

pos = 1;
while pos <= sig_len
    signal(pos) = 1;
    m = ceil(pos / (0.01 * sample_freq));
    PT = 80 + 5 * mod(m, 50);
    pos = pos + PT;
end
end
```

这样实现的问题是每个 10ms 的分界处 PT 会采用前一个 10ms 的值，不过影响较小，应不会对声音产生特别大的影响。另一种实现思路是先求得每一段对应的索引向量 $[1, 1+PT, 1+2PT, \cdots]$，然后将这些索引向量拼接起来，再使用`signal(index_array) = 1`的形式产生信号。

使用下方代码产生一段 10s 的音频并播放：

```
sig = varied_unit_samples(8000, 10);
sound(sig, 8000);
```

这段音频听上去有较强的节奏感和粒子效果，比较接近于“电音”这类音乐的背景伴奏。

2.3 将2.2中信号输入1.1中滤波器

使用下方代码生成并听音频：

```
b = 1;
a = [1, -1.3789, 0.9506];
e = varied_unit_samples(8000, 5);

s = filter(b, a, e);
s = s / max(abs(s));

sound([e, s], 8000);
```

听上去的感觉是滤波之后音调变低，相对不那么刺耳，有点像敲击木鱼和推拉老旧木门的声音。绘制时域和频域波形如图8和图9。

可以看到，滤波将能量集中到 1000Hz 附近，增强了低频，与听的结果一致。注意到 1000Hz 为该滤波器的共振峰频率，符合该滤波器的幅频响应。时域上看是叠加了一个周期性指数衰减的包络，可能因此感觉不那么刺耳。

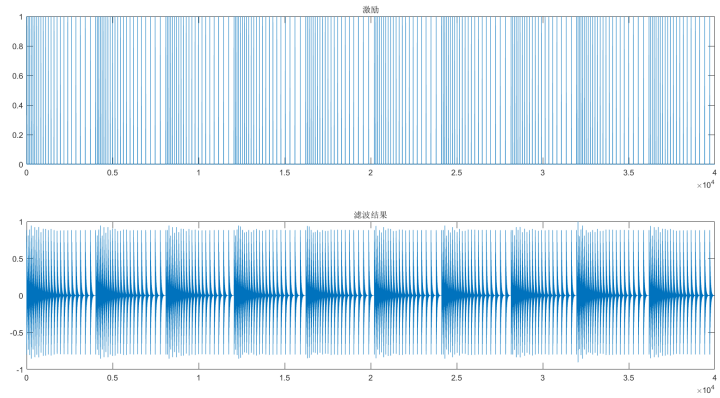


图 8: 基音周期改变激励及滤波结果时域波形

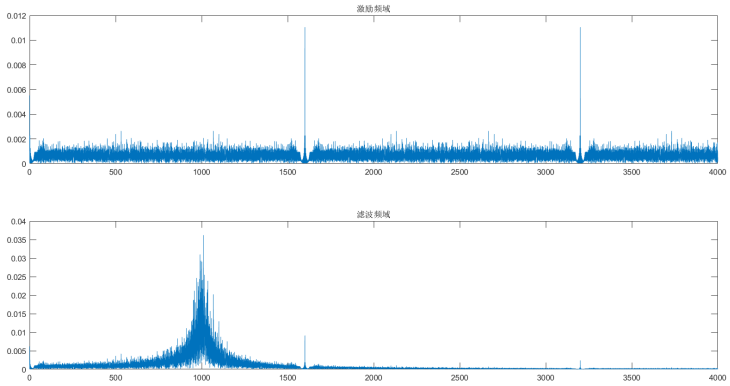


图 9: 基音周期改变激励及滤波结果频域波形

2.4 激励、语音合成

仿照前面章节`filter`的使用，定义滤波器状态`zi_syn`，并使用下方代码生成激励和重建语音：

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

pos = (n - 1) * FL + 1;
while pos <= n * FL
    exc_syn(pos) = G;
    pos = pos + PT;
end
[s_syn((n - 1) * FL + 1 : n * FL), zi_syn] = ...
    filter(1, A, exc_syn((n - 1) * FL + 1 : n * FL), zi_syn);
```

绘制重建语音和原语音的时域如图10，频域如图11。可以看到，重建语音的能量更集中于一系列频点上，或许失去了一些细节。

听上去，重建语音较类似于老电影中机器人说话的声音，其中“灯”和“进”的音调似乎略有上升，并且接近于中文的“阳平”。

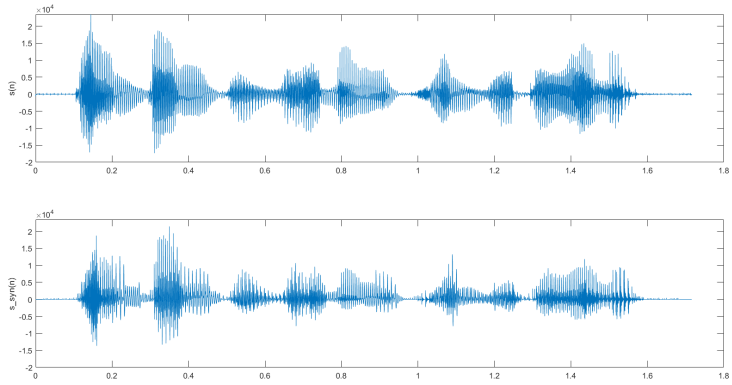


图 10: 语音和重建语音时域波形

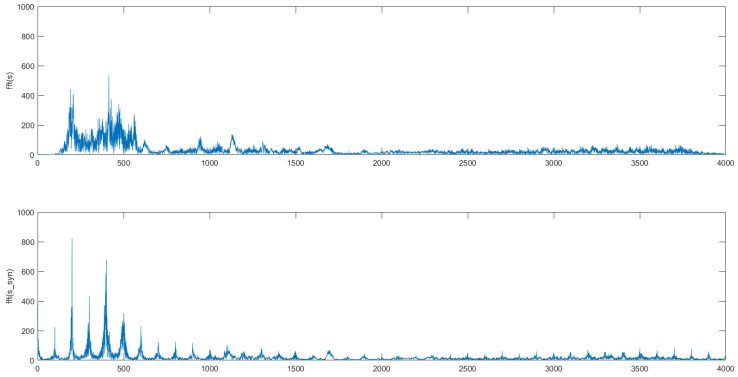


图 11: 语音和重建语音频域波形

3 变速不变调

3.1 激励长度翻倍的语音合成

与节2.4类似，定义滤波器状态、生成激励并重建语音。代码如下：

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

FL_v = FL * 2;
pos_v = (n - 1) * FL_v + 1;
while pos_v <= n * FL_v
    exc_syn_v(pos_v) = G;
    pos_v = pos_v + PT;
end
[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);
```

听上去音调没有变化，而速度变慢了一倍，可以清晰地听到语音中的“颗粒感”，更加类似老电影中机器人说话的声音。

4 变调不变速

4.1 共振峰频率提高 150Hz 的1.1滤波器

回顾节1.1中共振峰频率的计算，只需将共轭极点的辐角绝对值增大，便可以提高共振峰频率。欲求新的极点，可以通过乘以 $e^{\pm i\theta}$ 来改变辐角。下面使用代码求解：

```
b = 1;
a = [1, -1.3789, 0.9506];

rot_angle = 150 * 2 * pi / 8000;

[z, p, k] = tf2zpk(b, a);
p = p .* exp(1i * sign(imag(p)) * rot_angle);

[b, a] = zp2tf(z, p, k);
```

求得 $a_1 = -1.207283861048186$，$a_2 = 0.950600000000000$。将系数放入节1.1的代码，得到相关图形。

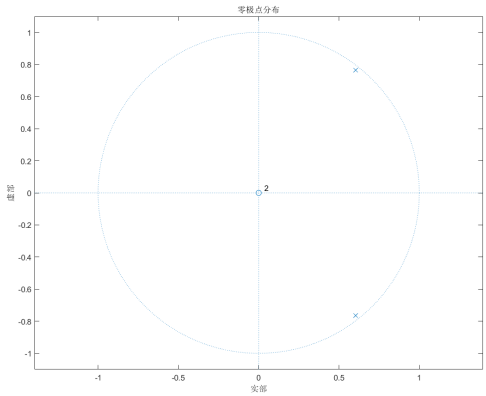


图 12: 共振峰频率增加 150Hz 的滤波器的零极点分布

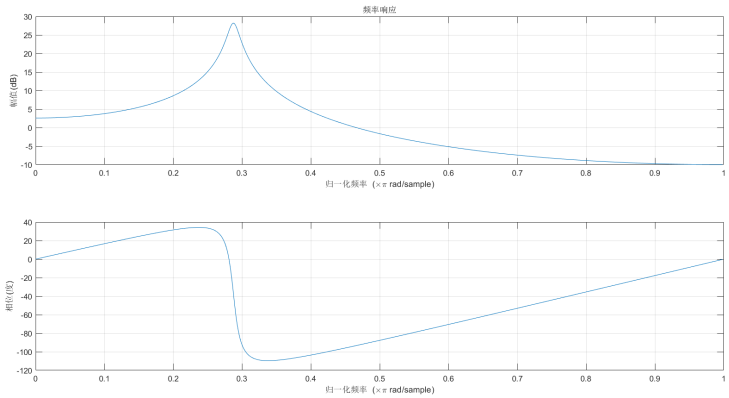


图 13: 共振峰频率增加 150Hz 的滤波器的频率响应

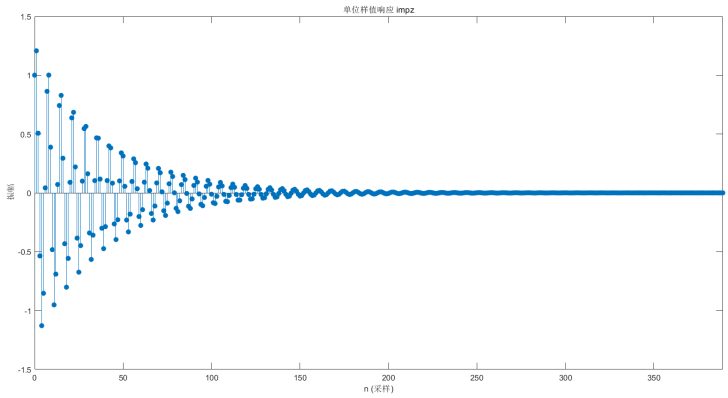


图 14: 共振峰频率增加 150Hz 的滤波器的单位样值响应

由于极点只变化了约 0.12rad，难以从图12中看出变化。1150Hz 对应 0.2875πrad，与图13一致。注意到图14表现出来的单位样值响应没有3对称，而是错落有致，这应与后面变调的语音合成有关。

4.2 减半基音周期并提高共振峰频率的语音合成

与前面实验类似，使用如下代码实现此功能：

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

[z, p, k] = tf2zpk(1, A);
rot_angle = 150 * 2 * pi / 8000;
p = p .* exp(1i * sign(imag(p)) * rot_angle);
[~, A_t] = zp2tf(z, p, k);
pos_t = (n - 1) * FL + 1;
while pos_t <= n * FL
    exc_syn_t(pos_t) = G;
    pos_t = pos_t + round(PT / 2);
end
[s_syn_t((n - 1) * FL + 1 : n * FL), zi_syn_t] = ...
    filter(1, A_t, exc_syn_t((n - 1) * FL + 1 : n * FL), zi_syn_t);
```

听上去音调有了显著的上升，更尖，而速度并未变化，与原语音时长一致。

绘制变速和变调的语音波形，时域见图15，频域见图16。

从图15可以直观地看出变数不变调虽然时长变为了两倍，但波形基本没有变化，这与听上去音调没有变化是相符的。而变调不变速则是时长没有变化，但波形显得更为密集，且形状发生了改变，这也与听上去的感受相符。图16显示出变调不变速明显将能量向高频移动了。值得注意的是，变数不变调的频谱中也有较多高频分量，我认为这是由于激励长度翻倍的处理中引入的高频分量，表现在听的感受上是一些类似“颤音”的背景声。而人耳听起来并不是语音的主要特征，故感受上整体音调是没有变化的。

5 后记

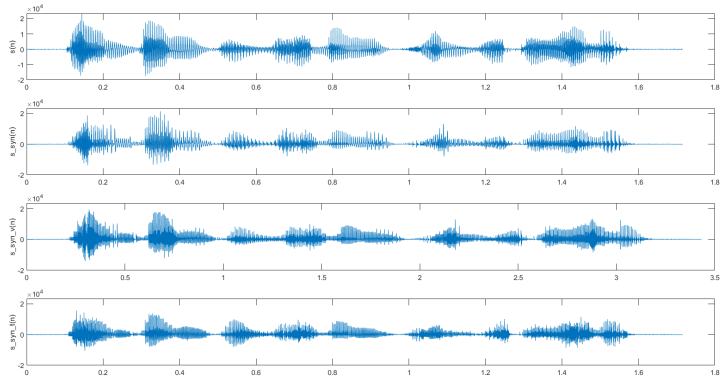


图 15: 重建语音时域波形

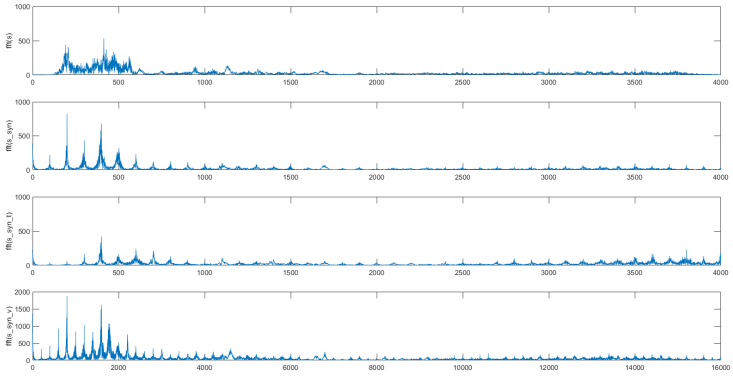


图 16: 重建语音频域波形