基于 MATLAB 的语音信号去噪方法

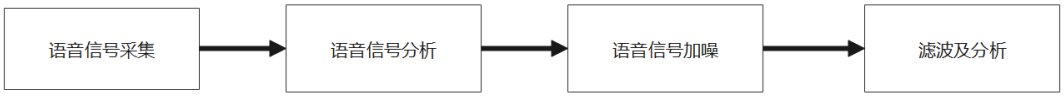
班级：22 通信 2 班　学号：2215404064　姓名：王旭

1 摘要

语言是人们获得各类有效信息的主要途径，而语音是语言的表现形式。语音在一定程度上可影响人们的生活，因此，语音信号的研究对科学领域和人们日常生活具有一定的研究价值和意义。噪声广泛存在于生活，信号在传输过程中不可避免会受到各类噪声的干扰，降低了信号的可读性，一定程度上影响了人们所接收信息的准确性，因此对携带噪声的信号进行去噪处理十分必要。人们提出了各种语音去噪方法。其中，可以使用低通滤波器对噪音进行滤波。常用的有滤波器有 FIR 有限脉冲响应滤波器和 IIR 无限脉冲响应滤波器。本文将利用 MATLAB 作为平台，使用 FIR 和 IIR 滤波器实现对语音信号的滤波和分析，并且分析两者的优缺点和适用范围。

[关键词]：语音信号，滤波去噪， FIR， IIR

[流程图]：



2 语音信号的处理

2.1语音信号的选取

选取 clean.wav 为原始语音信号，该信号为一段长度为 6s 的音频。

2.2噪声的添加

使用 audioread 函数读取音频文件。使用 randn 函数生成与音频信号相同长度的白噪音信号。添加白噪音到音频信号将生成的白噪音信号与原始音频信号相加。

Matlab 代码见附录 1。

添加白噪音后的音频命名为 noise.wav，与原始语音信号相比，声音刺耳根本不能分辨出原始语音信号。

3 基于 matlab 仿真及结果分析

3.1语音去噪的方法

对于语音去噪现在已经有一些比较成熟的方法，如小波变换法、小波包变换法，以及滤波器法。利用小波的方法去噪设计过程相对繁琐，因此，选用滤波器方法来实现去噪功能。数字滤波器若按照冲激响应来分类可以分为 FIR 滤波器和 IIR 滤波器两种。其中 FIR 为有限脉冲响应滤波器，IIR 滤波器是无限脉冲响应滤波器。

3.2FFR 滤波器

3.2.1 原理介绍

FIR 滤波器是数字滤波器的一种，相比于模拟滤波器，其滤波精度高，稳定，可随时修改，不需要考虑阻抗等问题，可以实现特殊要求的幅频特性，同时满足一定条件的 FIR 滤波器具有线性相位，因此 FIR 滤波器在实际工程中得到了较为广泛的应用，例如图像处理、通信以及雷达等。

一个 N 阶的 FIR 滤波器输出公式 y(n) 如下：

$$y(n)=\sum_{k=0}^{N-1}h(k)x(n-k) \tag{1}$$

式 1 中 h(k)为滤波器的系数，x(n-k)为 x(n)延时 k 个周期。

系统的传输函数 H(z)可表示成公式 2：

$$H(z)=\sum_{k=0}^{N-1}h(k)z^{-n}=h(0)+h(1)z^{-1}+\cdots+h(N-1)z^{-(N-1)} \tag{2}$$

从式(1)看出：滤波过程主要是一组特定的系数与信号完成卷积的过程。从式(2)看出，在有限的 Z 平面内它有 N-1 个零点同时其 N-1 个极点全部位于 Z=0 中，因此 FIR 滤波器也被称为全零点滤波器，是一个单位脉冲响应有限长的稳定系统。

FIR 滤波器在系数满足一定条件的情况下，它的相频特性是线性的，可以有效的保留信号的相位信息，因此线性相位的 FIR 滤波器在实际工程中有着较为广泛的应用。

3.2.2 Matlab 仿真

代码见附录 2。

3.2.3 结果分析

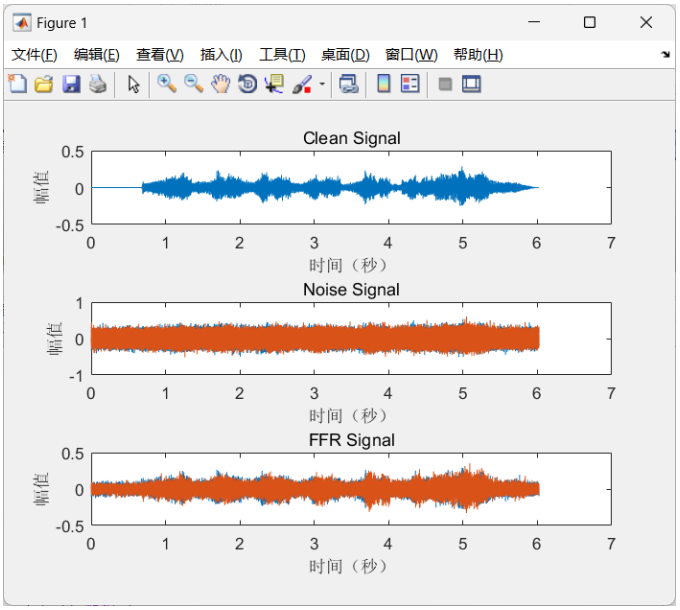


图 3.2-1 FIR 滤波结果

可以看出 FIR 低通滤波器可以消除噪音信号的高频部分，滤波后能够恢复一些初始信号中的语音信息。

3.2.4 缺点分析

FIR 滤波器具有一些显著的优点，但同时也存在一些缺点。相比于 IIR 滤波器，FIR 滤波器的设计通常更为复杂，尤其是在需要满足特定频率响应特性时。虽然 FIR 滤波器可以设计为具有线性相位特性，但它们的选择性可能不如某些类型的 IIR 滤波器。同时，FIR 滤波器通常需要更多的乘法和累加操作，尤其是在使用直接形式实现时。这意味着在实时应用中，可能需要更快的处理器或更多的计算资源。为了达到所需的性能，FIR 滤波器可能需要较长的滤波器系数，这会增加计算量和延迟。最后，FIR 滤波器的性能对系数的精度非常敏感，系数的量化误差可能会显著影响滤波器的性能。

尽管存在这些缺点，FIR 滤波器由于其线性相位特性和稳定性，仍然在许多应用中被广泛使用，尤其是在那些对信号的相位特性有严格要求的场合。

3.3 IIR 滤波器

3.3.1 原理介绍

IIR 滤波器相比 FIR，是一种在运算效率方面非常高效的滤波器，相比于 FIR 滤波器，在实现同等的滤波效果时，IIR 滤波器所需的阶数要比 FIR 滤波器低很多。FIR 滤波器的阶数一方面取决于通带与阻带之间的增益，一方面取决于滤波精度。

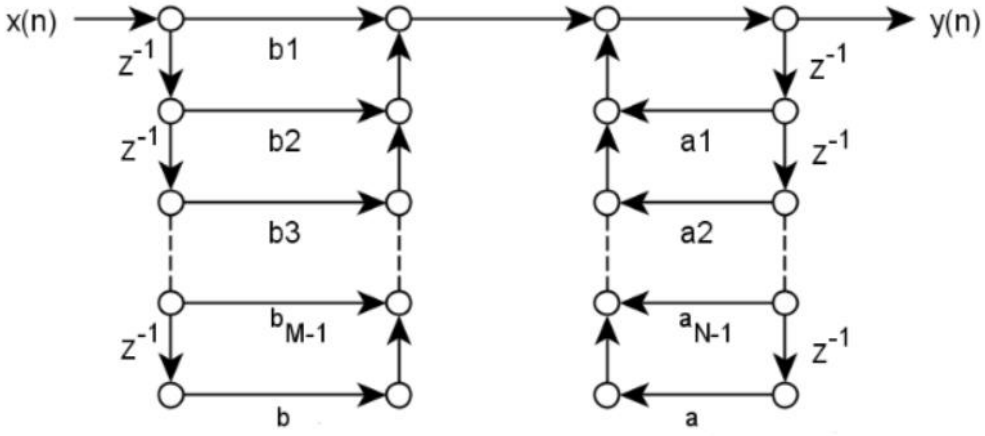


图 3.3-1 IIR 滤波器原理

在时域中对于上述流图，用时域描述即为：

$$y(n) = b_0x(n) + b_1x(n-1) + \cdots + b_Mx(n-M) - a_1y(n-1) - \cdots - a_Ny(n-N) \tag{3}$$

Z 变换传递函数为：

$$H(z) = \frac{Y(z)}{X(z)} = \frac{b_0 + b_1z^{-1} + \cdots + b_Mz^{-M}}{1 + a_1z^{-1} + \cdots + a_Nz^{-N}} \tag{4}$$

上述数字滤波器，如果从编程的角度来看，x(n-1)表示上一次的信号，可能是来自的上次采样，而 y(n-1)则为上一次滤波器的输出值，对应就比较好理解 x(n-N)就表示前第 n 次输入样本信号，而 y(n-M)则为前第 M 次滤波器的输出。

3. 3. 2 Matlab 仿真

代码见附录 3。

3.3.3 结果分析

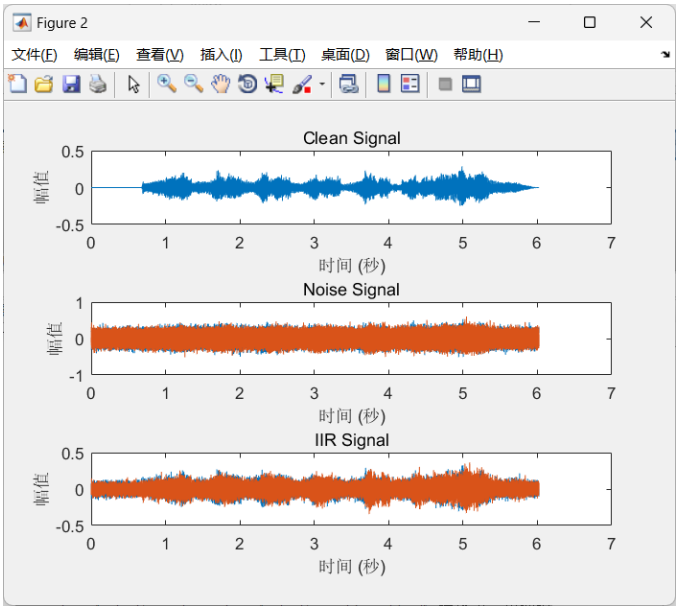


图 3.3-2 IIR 滤波器滤波结果

可以看出 IIR 低通滤波器可以消除噪音信号的高频部分，滤波后能够恢复一些初始信号中的语音信息。

3.3.4 缺点分析

IIR 滤波器的设计和分析可能涉及到稳定性问题和更复杂的数学处理。而且 IIR 滤波器在设计时必须特别注意防止出现不稳定的情况。不稳定的 IIR 滤波器会导致输入信号的无限放大，这可能对系统造成严重影响。同时，IIR 滤波器在时域上会引入反馈，这可能导致不稳定或者有振铃的响应。由于运算中的舍入处理，IIR 滤波器可能会产生微弱的寄生振荡，这是由于反馈环路中误差的不断累积。所以 IIR 滤波器的性能对系数的精度非常敏感，系数的微小变化可能会对滤波器的性能产生较大影响。

IIR 数字滤波器的相位特性不好控制，对相位要求较高时，可能需要加入额外的相位校准网络。

4 总结

对于语音降噪处理，可以采用 FIR 滤波器和 IIR 滤波器。针对不同的情况有不同的选择，具体来讲 IIR 滤波器可以用相对较少的阶数来达到滤波的目的，而且可以根据模拟滤波器原型来设计相对工作量较小。FIR 滤波器由于为有限单位脉冲响应，因此较为稳定。根据语音信号的特点，一般在研究语音信号的时候不考虑它的相位问题，而且 IIR 滤波器的设计思路较 IIR 滤波器来讲较为简单，因此选择两种滤波器的时候建议选择 IIR 低通滤波器。滤波器对语音信号进行去噪处理应该对语音信号和噪声的频谱进行十分精确的分析，这样才能

出较准确的滤波器参数，也就是说，利用数字滤波器的方法完全可以达到对语音信号去噪或降噪的目的。

5 参考文献

[1]结硕,张黎.基于 MATLAB 的有噪声的语音信号处理[J].电声技术,2021,45(07):7-9+17.DOI:10.16311/j.audioe.2021.07.003.
[2]韩长军.基于 MATLAB 的语音信号去噪方法应用[J].辽东学院学报(自然科学版),2017,24(01):72-77.DOI:10.14168/j.issn.1673-4939.2017.01.14.

附录 1：噪音产生与添加 matlab 代码

```
[audioSignal, fs] = audioread('clean.wav');

% 生成白噪音
noise = randn(size(audioSignal));

% 调整白噪音幅度
noise = noise * 0.1;

% 添加白噪音到音频信号
noisySignal = audioSignal + noise;

% 确保信号不会超出有符号整数的范围
noisySignal = min(max(noisySignal, -1), 1);

% 保存新的音频文件
audiowrite('noise.wav', noisySignal, fs);
```

附录 2：FIR 滤波器 matlab 代码

```
cleanAudioFilePath = 'clean.wav';
noiseAudioFilePath = 'noise.wav';
```

```
3.
4. % 读取未加噪声的音频文件
5. [cleanAudioSignal, fs] = audioread(cleanAudioFilePath);
6.
7. % 确保音频信号是单声道
8. if isvector(cleanAudioSignal)
9.     cleanAudioSignal = cleanAudioSignal';
10.else
11.    cleanAudioSignal = cleanAudioSignal(:, 1); % 选择第一个
   通道
12.end
13.
14.% 读取加噪声的音频文件
15.[noisyAudioSignal, fs] = audioread(noisyAudioFilePath);
16.
17.
18.% 设计 FIR 滤波器参数
19.n = 31; % 滤波器阶数
20.cutoff = 3000; % 截止频率，假设我们只保留 3kHz 以下的频率
21.b = fir1(n, cutoff/(fs/2)); % 使用窗函数法设计 FIR 滤波器
22.
23.% 应用 FIR 滤波器到加噪声的音频信号
24.filteredNoisySignal = filter(b, 1, noisyAudioSignal);
25.
26.% 绘制未加噪声的音频信号、加噪声的音频信号和滤波后的音频信号
27.timeVector = (0:length(noisyAudioSignal)-1) / fs;
28.
29.% 创建图形窗口
30.figure;
31.subplot(3,1,1); % 第一个子图
32.plot(timeVector, cleanAudioSignal);
33.title('Clean Signal');
34.xlabel('时间（秒）');
35.ylabel('幅值');
36.
37.subplot(3,1,2); % 第二个子图
38.plot(timeVector, noisyAudioSignal);
39.title('Noise Signal');
40.xlabel('时间（秒）');
41.ylabel('幅值');
42.
43.subplot(3,1,3); % 第三个子图
44.plot(timeVector(1:length(filteredNoisySignal)), filteredNoi
   sySignal);
```

```
45.title('FFR Signal');
46.xlabel('时间（秒）');
47.ylabel('幅值');
48.
49.
50.% 保存滤波后的音频
51.outputFilePath = 'FIR clean.wav';
52.audiowrite(outputFilePath, filteredNoisySignal, fs);
```

附录 3：IIR 滤波器 matlab 代码

```
1. cleanAudioFilePath = 'clean.wav';
2. noisyAudioFilePath = 'noise.wav';
3.
4. % 读取未加噪声的音频文件
5. [cleanAudioSignal, fs] = audioread(cleanAudioFilePath);
6.
7. % 确保音频信号是单声道
8. if isvector(cleanAudioSignal)
9.     cleanAudioSignal = cleanAudioSignal';
10.else
11.    cleanAudioSignal = cleanAudioSignal(:, 1); % 选择第一个
   通道
12.end
13.
14.% 读取可能加噪声的音频文件
15.[noisyAudioSignal, fs] = audioread(noisyAudioFilePath);
16.
17.% 设计 IIR 滤波器参数
18.n = 2; % 滤波器阶数，对于巴特沃斯滤波器，n 通常为偶数
19.Wn = 3000 / (fs / 2); % 归一化截止频率
20.Rp = 1; % 通带波动
21.Rs = 60; % 阻带衰减
22.
23.% 使用巴特沃斯滤波器设计 IIR 滤波器
24.[B, A] = butter(n, Wn, 'low');
25.
26.% 应用 IIR 滤波器到加噪声的音频信号
27.filteredNoisySignal = filter(B, A, noisyAudioSignal);
28.
29.% 绘制未加噪声的音频信号、加噪声的音频信号和滤波后的音频信号
30.timeVector = (0:length(noisyAudioSignal)-1) / fs; % 时间向量
31.
32.figure;
33.subplot(3,1,1); % 第一个子图
```

```
plot(timeVector, cleanAudioSignal);
title('Clean Signal');
xlabel('时间 (秒)');
ylabel('幅值');

subplot(3,1,2); % 第二个子图
plot(timeVector, noisyAudioSignal);
title('Noise Signal');
xlabel('时间 (秒)');
ylabel('幅值');

subplot(3,1,3); % 第三个子图
plot(timeVector(1:length(filteredNoisySignal)), filteredNoisySignal);
title('IIR Signal');
xlabel('时间 (秒)');
ylabel('幅值');

% 保存滤波后的音频
outputFilePath = 'IIR clean.wav';
audiowrite(outputFilePath, filteredNoisySignal, fs);
```