语音处理

目录

第 2 章 回声消除（AEC ）原理及实现

2.1 回声消除原理

回声消除的基本原理是使用一个自适应滤波器对未知的回声信道 ω 进行参数辨识，根据扬声器信号与产生的多路回声的相关性为基础，建立远端信号模型，模拟回声路径，通过自适应算法调整，使其冲击响应和真实回声路径相逼近。然后将麦克风接收到的信号减去估计值，即可实现回声消除功能。

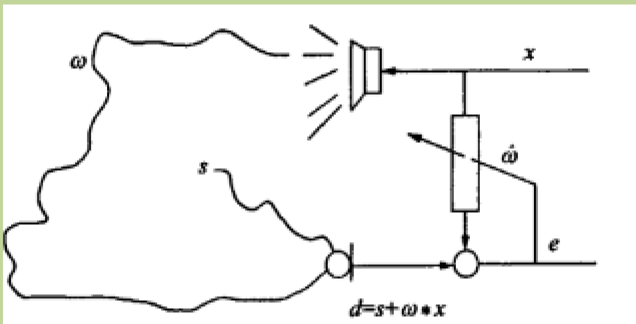

图 2.1 回声消除原理图

$$echo = x * \omega \tag{2.1.1}$$

$$d = s + echo \quad 2.1.2$$

$$\hat{y} = x * \hat{\omega} \quad 2.1.3$$

$$e = d - \hat{y} \quad 2.1.4$$

式中，ω 是回声通道的时域冲击响应函数；x 是远端语音；echo 是所得回声；s 是近端说话人语音，d 为麦克风采集到的信号；$\hat{y}$ 为对回声信号的估计值；e 为误差。在电话、视频会议中这里的 x 通信另一端的语音信号，而在机器语音识别中，这里的 x 则指机器自身发出的声音。

为了消除较长时间的回声，需要 FIR 滤波器的阶数尽量的大。时域计算诸多不便，使用频域分块自适应滤波算法。

2.2 维纳滤波

均方误差（MSE， Mean Square Error）,对于离散时间系统，可定义期待响应 d_k 为一个希望自适应系统的输出 y_k 与之相接近的信号，k 为采样时刻。

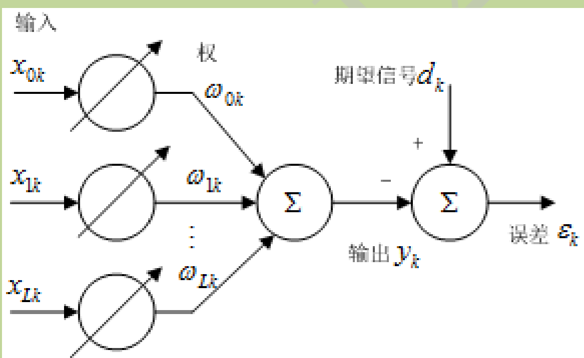

图 2.2 MSE 自适应系统

根据图 2.2，可以求得误差信号：

$$\varepsilon_k = d_k - y_k \quad 2.2.1$$

自适应线性组合器输出：

$$y_k = W_k^T X_k \quad 2.2.2$$

其中：

$$X_k = [x_{0k} x_{1k} \ldots x_{Lk}], W_k = [w_{0k} w_{1k} \ldots w_{Lk}]^T$$

分别为自适应系统在 k 时刻的输入信号向量和权向量，系统的均方误差为：

$$E(|\varepsilon_k|^2) = E[(d_k - y_k)^*(d_k - y_k)] = E(|d_k|^2) + \boldsymbol{W}_k^H E[\boldsymbol{X}_k^* \boldsymbol{X}_k^T] \boldsymbol{W}_k - 2\,\mathrm{Re}\{\boldsymbol{W}_k^T E[d_k^* \boldsymbol{X}_k]\} \quad 2.2.3$$

$$R = E[X^* X^T] = \begin{bmatrix} E[x_0^* x_0] & E[x_0^* x_1] & \cdots & E[x_0^* x_L] \\ E[x_1^* x_0] & E[x_1^* x_1] & \cdots & E[x_1^* x_L] \\ \vdots & \vdots & \ddots & \vdots \\ E[x_L^* x_0] & E[x_L^* x_1] & \cdots & E[x_L^* x_L] \end{bmatrix} \qquad 2.2.4$$

定义期待响应和输入信号之间的互相关向量为：

$$P = E[d^* X] = \begin{bmatrix} d^* x_0 \\ \vdots \\ d^* x_L \end{bmatrix} \qquad 2.2.5$$

将式 2.2.3 简化成下式：

$$\xi(w) = E(|d_k|^2) + W_k^H R W_k - 2\,\mathrm{Re}\{W_k^T P\} \qquad 2.2.5$$

理想情况下 $E\left(|\varepsilon_k|^2\right)$ 等于零，这时估计值等于观测值，如果不能达到理想，则应该是越小越好，这样估计值和观测值最接近。

对 2.2.5 求偏导数，得：

$$\nabla = \frac{\partial}{\partial W}[\xi(W)] = 2RW - 2P^* \qquad 2.2.6$$

最佳权向量处的梯度值为零，于是：

$$\nabla = 2RW_{opt} - 2P^* = 0 \qquad 2.2.7$$

最小均方误差输出情况下的最佳权向量 W_{opt} 满足维纳-霍夫方程：

$$W_{opt} = R^{-1} P^* \qquad 2.2.8$$

2.3 LMS 算法

$$\varepsilon_k = d_k - X_k^T W_k \qquad 2.3.1$$

式中，X_k 为输入样本向量，使用单次采样数据 $|\varepsilon_k|^2$ 来代替均方误差 ξ_k，这样其梯度估计可表示为如下形式：

$$\begin{aligned} \hat{\nabla}_k &= \frac{\partial}{\partial W_k}|\varepsilon_k|^2 = \frac{\partial}{\partial W_k}\left[|d_k|^2 + W_k^H X_k^* X_k^T W_k - 2\,\mathrm{Re}(d_k^* X_k^T W_k)\right] \\ &= 2X_k^* X_k^T W_k - 2dX_k^* = -2\varepsilon_k X_k^* \end{aligned} \qquad 2.3.2$$

基于最速下降法的权向量迭代如下：

$$W_{k+1} = W_k - \mu\hat{\nabla}_k = W_k + 2\mu\varepsilon_k X_k^* \qquad 2.3.4$$

其中 μ 是步长因子，$0<\mu<\frac{1}{\lambda_{\max}}$，$\lambda_{\max}$ 是 R_{XX} 的最大特征值。$W(k)$ 收敛于 W_{opt} 由比值 $d=\frac{\lambda_{\max}}{\lambda_{\min}}$ 决定，该比值叫做谱动态范围。大的 d 值意味着较长的时间才能收敛到最佳权值。

该算法用在语音增强的加性噪声消除功能上时，其工程实践并不完全按照式 2.3.1 意义来实现。

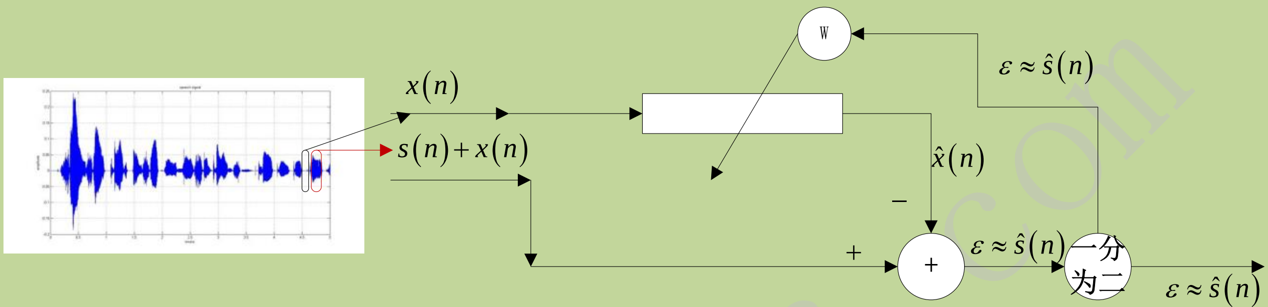

图 LMS 算法在语音增强中使用方法

在语音增强中，其目的是获得纯净的语音信号 $s(n)$，即上图中的最后输出信号，输入信号有两种，一种是带噪的语音信号 $s(n)+x(n)$，另一种是只有噪声的输入 $x(n)$，在没有人说话的情况下的输入信号，就仅仅是噪声输入。这里要使得噪声估计 $\hat{x}(n)$ 非常接近 $x(n)$，这样 $\varepsilon=\sum_{n} s(n)+x(n)-\hat{x}(n)$，这时如果 ε^2 最小，则可以知道估计出的 ε 最接近 $s(n)$。

上述过程可以概述如下：

A) 首先获取到噪声输入 $x(n)$，并存储下来，以 64 或者 128 点为总长度不断刷新存储噪声输入。

B) 采集带噪声的语音信号 $s(n)+x(n)$。

C) 用采集带噪语音信号减去估计到的噪声信号 $s(n)+x(n)-\hat{x}(n)$。

D) 用 C 的输出作为误差，调节噪声权向量 W。

MATLAB 实现具体包括如下三个部分：

```
% Loop over input vector
for ii = 1:length(signal_with_noise)
   % Update buffer
   noise_buf = obj.update_buf(noise_buf, noise(ii));//输入噪声估计
   % Filter this sample with current coefficient values
   filter_output = obj.data_filter(coefs, noise_buf);//通过权向量估计 x̂(n)
```

% Compute error

err = signal_with_noise(ii) - filter_output; //相减得到$\hat{s}(n)$

% Update coefficients

coefs = obj.update_coefs(coefs, noise_buf, obj.filter_params.step_size, obj.filter_params.leakage, err); //用$\hat{s}(n)$调节权向量

% Build output vector

dout(ii) = err; //存储输入信号的估计值$\hat{s}(n)$

2.3.1 NLMS

输入信号较大时，会遇到梯度噪声放大的问题，使得能量低的信号算法收敛速度较慢。将输入信号按照自身的平均能量进行归一化处理，即得到归一化 NLMS 算法。设输入带噪信号可表示为：$x(n)$，其迭代算法的 NLMS 表示公式如下：

$$\boldsymbol{W}_{n+1} = \boldsymbol{W}_n - \mu\hat{\nabla}_n = \boldsymbol{W}_n + \frac{\mu}{N}\frac{e(n)\boldsymbol{x}(n)}{\hat{\sigma}_x^2(n)}$$

其中$\hat{\sigma}_x^2(n)=\frac{1}{N}\sum_{n=0}^{N-1}x^2(n-k)$，其中 N 是噪声消除器和回波抵消器的长度，（常取 512 或者 1024）；μ 是步长因子。当$\hat{\sigma}_x^2(k)$较小时，$\frac{\mu}{\hat{\sigma}_x^2(n)}$的值可能较大，这时迭代算法变成如下形式：

$$\boldsymbol{W}_{n+1} = \boldsymbol{W}_n - \mu\hat{\nabla}_n = \boldsymbol{W}_n + \frac{\mu}{N}\frac{e(n)\boldsymbol{x}(n)}{\sigma + \hat{\sigma}_x^2(n)}$$

其计算过程如下：

参数： M=抽头系数（即 FIR 滤波器长度）

μ 自适应常数

$0<\mu<2\frac{E\left[|\boldsymbol{x}(n)|^2\right]E\left[|\boldsymbol{\varepsilon}(n)|^2\right]}{E\left[|\boldsymbol{e}(n)|^2\right]}$，其中$E\left[|\boldsymbol{\varepsilon}(n)|^2\right]=E\left[\left|\boldsymbol{W}_{opt}-\hat{\boldsymbol{W}}(n)\right|^2\right]$，是权向量均方偏差，$\boldsymbol{W}_{opt}$是最优维纳解，$\hat{\boldsymbol{W}}(n)$是第 n 次迭代中得到的估值。$E\left[|\boldsymbol{x}(n)|^2\right]$是带噪输入信号的功率，$E\left[|\boldsymbol{e}(n)|^2\right]$是误差信号功率。

初始化：

如果知道抽头权向量$\hat{\boldsymbol{W}}(n)$的先验知识，则用其来初始化$\hat{\boldsymbol{W}}(0)$，否则令$\hat{\boldsymbol{W}}(0)=\boldsymbol{0}$

数据：

A）给定的 $\boldsymbol{X}(n)$ =第 n 时刻 $M \times 1$ 抽头输入向量，

$d(n)$ =第 n 时间步的期望响应

B）要计算的：$\hat{\boldsymbol{W}}(n+1)$=第 n+1 步抽头权向量的估计

计算：

对 n=0,1,2，…计算

$$e(n)=d(n)-\hat{\boldsymbol{W}}^{H}(n)\boldsymbol{X}(n)$$

$$\boldsymbol{W}_{n+1} = \boldsymbol{W}_{n} - \mu\hat{\nabla}_{n} = \boldsymbol{W}_{n} + \frac{\mu}{N}\frac{e(n)\boldsymbol{x}(n)}{\hat{\sigma}_{x}^{2}(n)}$$

2.3.2 SE-LMS

Signed-errorLMS（SE-LMS）算法将误差 $e(n)$ 用 $[-1 \quad 0 \quad 1]$ 这三个量化值来代替，如果误差大于 0，则将 $e(n)$ 赋值为 1，其它类推。这时式 2.3.4 退化成如下形式：

$$\boldsymbol{W}_{k+1} = \boldsymbol{W}_{k} - \mu\hat{\nabla}_{k} = \boldsymbol{W}_{k} + 2\mu\boldsymbol{X}_{k}^{*}sign(e[n])$$

该算法在加快运算速度的同时简化了电路结构，μ 设置成 2 的指数时，通过移位就可以实现这里的乘法操作。降低了硬件实现的复杂度。

2.3.2 SD-LMS

Signal-dependent LMS 算法和 SE-LMS 很相似，误差也是只取 $[-1 \quad 0 \quad 1]$ 这三个值，不同的是，其选择是以采样到的误差信号为参考的，如果 $x(n)$ 大于 0，则 $e(n)$ 用 1 代替，依次类推。

$$\boldsymbol{W}_{k+1} = \boldsymbol{W}_{k} - \mu\hat{\nabla}_{k} = \boldsymbol{W}_{k} + 2\mu sign(x[n])$$

2.3.2 SS-LMS

Sign-sign LMS 算法既考虑输入信号的符号又考虑误差的信号。

$$\boldsymbol{W}_{k+1} = \boldsymbol{W}_{k} - \mu\hat{\nabla}_{k} = \boldsymbol{W}_{k} + 2\mu sign(\boldsymbol{x}[n])sign(e[n])$$

2.3.2 LLMS

Leaky LMS 算法减轻了系数溢出问题。其不仅考虑了均方误差 $e^{2}(n)$ 也考虑了滤波器系数。其权向量更新方程如下：

$$\boldsymbol{W}_{k+1} = (1-\mu\alpha)\boldsymbol{W}_{k} - \mu\hat{\nabla}_{k} = \boldsymbol{W}_{k} + 2\mu e(n)\boldsymbol{x}(n)$$

2.3.2 LNLMS

Leaky NLMS 是归一化的 LLMS 算法。

$$\boldsymbol{W}_{n+1} = (1-\mu\alpha)\boldsymbol{W}_n - \mu\hat{\nabla}_n = (1-\mu\alpha)\boldsymbol{W}_n + \frac{\mu}{N}\frac{e(n)\boldsymbol{x}(n)}{\hat{\sigma}_x^2(n)}$$

2.4 块自适应滤波

2.4.1 块自适应滤波器

计算过程如下，对参考信号 x 分段并做 FFT 变换，分别对各段数据做频域滤波，累加后做 FFT 反变换，并只取后 L(L 是原始信号的分段后的长度)点为有效的线性卷积结果，得到的是估计信号，将估计信号从回声信号中去除，得残差信号。计算子带步长，调整各段滤波器系数。这一过程表示如下图。

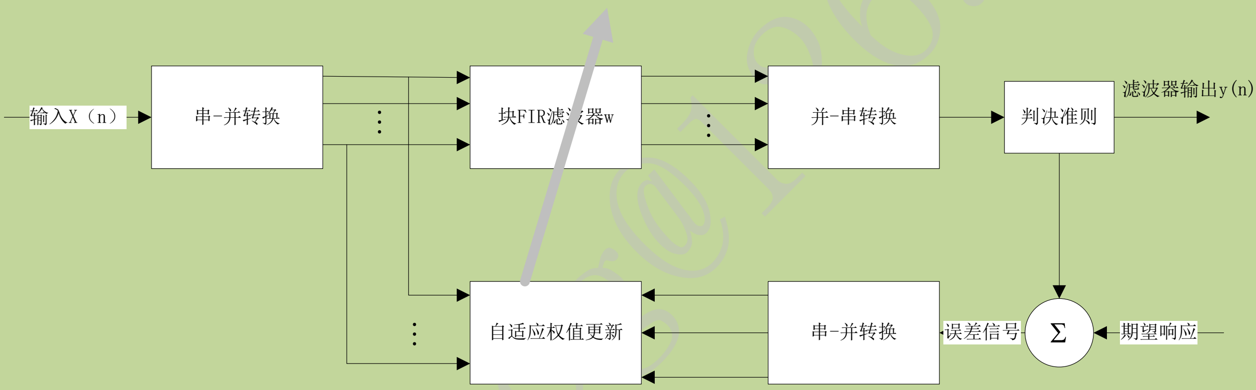

图 2.2 块自适应滤波器

设 n 时刻输入序列 $x(n)$ 如下：

$$X(n) = [x(n), x(n-1), \ldots, x(n-M+1)]^T \qquad 2.4.1$$

对应于长度为 M 的 FIR 滤波器在 n 时刻的抽头权向量为：

$$\hat{W}(n) = [\hat{w}_0(n), \hat{w}_1(n-1), \ldots, \hat{w}_{M-1}(n)]^T \qquad 2.4.2$$

根据 FIR 滤波器原理：

$$y(n) = x(n)\times\hat{w}_0(n) + x(n-1)\times\hat{w}_1(n) + \ldots + x(n-M+1)\times\hat{w}_{M-1}(n) \qquad 2.4.3$$

用向量可以表示成如下：

$$y(n) = X(n)^T\hat{W}(n) \qquad 2.4.4$$

下面对 x(n)进行分块，设 k 表示块下标，它与原始样值时间 n 的关系为：

$$n = kL + i, i = 0,1,\ldots,L-1; k = 1,2,\ldots \qquad 2.4.5$$

其中 L 是块的长度。第 k 块的数据为 $\{X(kL+i)\}_{i=0}^{L-1}$,其矩阵表示形式如下：

$$A^T(k)=\left[x(kL),x(kL+1),\ldots,x(kL+L-1)\right] \qquad 2.4.6$$

将滤波器对输入块 $A(k)$ 的响应表示如下：

$$y(kL+i)=\hat{W}^T(k)A(k)=\sum_{j=0}^{M-1}\hat{w}_j(k)x(kL+i-j),i=0,1,\ldots,L-1 \qquad 2.4.7$$

设 $d(kL+i)$ 表示期望信号，误差信号表示如下：

$$e(kL+i)=d(kL+i)-y(kL+i) \qquad 2.4.8$$

考虑滤波器长度 M=3，块长度 L=3,其三个相邻的数据块是 k-1，k， k+1，则

k-1 滤波结果如下：

$$\begin{vmatrix} y(3k-3) \\ y(3k-2) \\ y(3k-1) \end{vmatrix} = \begin{vmatrix} x(3k-3) & x(3k-4) & x(3k-5) \\ x(3k-2) & x(3k-3) & x(3k-4) \\ x(3k-1) & x(3k-2) & x(3k-3) \end{vmatrix} \begin{vmatrix} w_0(k-1) \\ w_1(k-1) \\ w_2(k-1) \end{vmatrix} \qquad 2.4.9$$

k 块滤波结果如下：

$$\begin{vmatrix} y(3k) \\ y(3k+1) \\ y(3k+2) \end{vmatrix} = \begin{vmatrix} x(3k) & x(3k-1) & x(3k-2) \\ x(3k+1) & x(3k) & x(3k-1) \\ x(3k+2) & x(3k+1) & x(3k) \end{vmatrix} \begin{vmatrix} w_0(k) \\ w_1(k) \\ w_2(k) \end{vmatrix} \qquad 2.4.10$$

k+1 块滤波器结果如下：

$$\begin{vmatrix} y(3k+3) \\ y(3k+4) \\ y(3k+5) \end{vmatrix} = \begin{vmatrix} x(3k+3) & x(3k+2) & x(3k+1) \\ x(3k+4) & x(3k+3) & x(3k+2) \\ x(3k+5) & x(3k+4) & x(3k+3) \end{vmatrix} \begin{vmatrix} w_0(k+1) \\ w_1(k+1) \\ w_2(k+1) \end{vmatrix} \qquad 2.4.11$$

上面的数据矩阵是托伯利兹矩阵，主对角线元素都相同。

2.4.2 块 LMS

权向量调整公式如下：

（权向量的调整）=（步长参数）*（抽头输入向量）*（误差信号）

因为在块 LMS 算法中误差信号随抽样速率而变，对于每一个数据块，我们有不同的用于自适应过程的误差信号值。因此，每一个块的抽头权向量更新公式如下：

$$\hat{w}(k+1)=\hat{w}(k)+\mu\sum_{i=0}^{L-1}x(kL+i)e(kL+i) \qquad 2.4.12$$

其梯度向量的估计如下：

$$\hat{\nabla}(k)=-\frac{2}{L}\sum_{i=0}^{L-1}x(kL+i)e(kL+i) \qquad 2.4.13$$

$\hat{\nabla}(k)$ 的无偏估计如下：

$$\hat{w}(k+1) = \hat{w}(k) - \frac{1}{2}\mu_B \hat{\nabla}(k) \quad 2.4.14$$

2.4.3 块 LMS 算法收敛性

由于时间平均的缘故，它具有估计精度随快长度增加而大幅提高的特性。然而，长度的增加会导致其收敛速度进一步减慢。后文的快速 LMS 算法加速了这一过程。

- 平均时长数

$$\tau_{mse,av} = \frac{1}{2\mu_B \lambda_{av}} \quad 2.4.15$$

其中 λ_{av} 是输入自相关矩阵 $R = E\left[x(n)x(n)^T\right]$

上式中为了使零阶公式成立，μ_B 必须小于 $1/\lambda_{\max}$ ，其中 $\lambda_{\max}$ 是相关矩阵的最大特征值。

- 失调

$$t\,\upsilon = \frac{\mu_B}{2L} tr[R] \quad 2.4.16$$

$tr[B]$ 是相关矩阵的迹。

2.4.4 块长的选择

设滤波器长度 M 和块长度 L 的关系有三种可能：

1. L=M，从计算的复杂性上看，最佳

2. L<M，有降低延迟的好处。

3. L>M,将产生自适应过程冗余运算。

2.5 FLMS

FLMS（Fast LMS）的基本思想是将时域块 LMS 放到频域来计算。利用 FFT 算法在频域上完成滤波器系数的自适应。快速卷积算法用重叠相加法和重叠存储法。重叠相加法是将长序列分成大小相等的短片段，分别对各个端片段做 FFT 变换，再将变换重叠的部分相加构成最终 FFT 结果，重叠存储法在分段时，各个短的段之间存在重叠，对各个段进行 FFT 变换，最后将 FFT 变换得结果直接相加即得最终变换结果。当块的大小和权值个数相等时，运算效率达到最高。

根据重叠存储方法，将滤波器 M 个抽头权值用等个数的零来填补，并采用 N 点 FFT 进行计算，其中 $N = 2M$ ，因此，N*1 的向量：

$$\hat{\boldsymbol{W}}(k)=FFT\begin{bmatrix}\hat{\boldsymbol{w}}(k)\\ \boldsymbol{0}\end{bmatrix} \qquad 2.5.1$$

表示 FFT 补零后的系数，抽头权向量为$\hat{\mathbf{w}}(k)$。值得注意的是频域权向量$\hat{\boldsymbol{W}}(k)$的长度是时域权向量$\hat{\mathbf{w}}(k)$长度的两倍。相应的令：

$$X(k)=diag\left\{FFT\left[\underset{K-1,block}{x(kM-M),\ldots,x(kM-1),}\quad \underset{K,block}{x(kM),\ldots,x(kM+M-1),}\right]\right\} \qquad 2.5.2$$

表示对输入数据的两个相继子块进行傅里叶变换得到一个$N\times N$ 对角阵。

将重叠存储法应用于 2.4.7 得。

$$\begin{aligned} y^T(k)&=\left[y(kM),y(kM+1),...,y(kM+M-1)\right]\\ &=IFFT\left[\boldsymbol{X}(k)\hat{\boldsymbol{W}}(k)\right],lastM \end{aligned} \qquad 2.5.3$$

是 2.5.3 只有最后 M 个元素被保留，因为最前面的 M 个元素是循环卷积的结果。

设第 K 块的$M\times 1$ 期望响应和误差信号分别如下：

$$\boldsymbol{d}(k)=\left[d(kM),d(kM+1),\ldots,d(kM+M-1)\right]^T \qquad 2.5.4$$

$$\boldsymbol{e}(k)=\left[e(kM),e(kM+1),\ldots,e(kM+M-1)\right]^T=\boldsymbol{d}(k)-\boldsymbol{y}(k) \qquad 2.5.5$$

根据式 2.5.3，可将$\boldsymbol{e}(k)$变换到频域，即

$$E(k)=FFT\begin{bmatrix}\boldsymbol{0}\\ \boldsymbol{e}(k)\end{bmatrix} \qquad 2.5.6$$

则在更新权值的相关矩阵如下：

$$\boldsymbol{\Phi}(k)=\sum_{i=0}^{L-1}x(kL+i)e(kL+i)=IFFT\left[\boldsymbol{X}^T(k)\boldsymbol{E}(k)\right]\text{ ，的最前面 M 个元素} \qquad 2.5.7$$

则抽头的更新过程在频域中的表现如下：

$$\hat{\boldsymbol{W}}(k+1)=\hat{\boldsymbol{W}}(k)+\mu FFT\begin{bmatrix}\Phi(k)\\ \boldsymbol{0}\end{bmatrix} \qquad 2.5.8$$

2.6 MDF 自适应权值调整

时域解

对于 N 阶 NLMS 算法，其误差调节向量如下式：

$$e(n)=d(n)-\hat{y}(n)=d(n)-\sum_{k=0}^{N-1}\hat{w}_k(n)x(n-k) \qquad 2.6.1$$

权值更新如下：

$$\begin{aligned}\hat{w}_k\left(n+1\right)&=\hat{w}_k\left(n\right)+\mu\frac{e\left(n\right)x^*\left(n-k\right)}{\sum_{i=0}^{N-1}\left|x\left(n-i\right)\right|^2}\\&=\hat{w}_k\left(n\right)+\mu\frac{\left(d\left(n\right)-\sum_i\hat{w}_i\left(n\right)x\left(n-i\right)\right)x^*\left(n-k\right)}{\sum_{i=0}^{N-1}\left|x\left(n-i\right)\right|^2}\end{aligned}\tag{2.6.2}$$

其中 $x\left(n\right)$ 是参考信号，$\hat{w}_k\left(n\right)$ 是 n 时刻和步长 μ 的权值更新。假设滤波后的误差为 $\sigma_k\left(n\right)=\hat{w}_k\left(n\right)-w_k\left(n\right)$ ，$d\left(n\right)=v\left(n\right)+\sum_k\hat{w}_k\left(n\right)x\left(n-k\right)$,则误差的迭代关系如下：

$$\delta_k\left(n+1\right)=\delta_k\left(n\right)+\mu\frac{\left(v\left(n\right)-\sum_i\delta_i\left(n\right)x\left(n-i\right)\right)x^*\left(n-k\right)}{\sum_{i=0}^{N-1}\left|x\left(n-1\right)\right|^2}\tag{2.6.3}$$

在每一次调节中，滤波器的误差估计为 $\Lambda\left(n\right)=\sum_k\delta_k^*\left(n\right)\delta_k\left(n\right)$ ，展开后得如下形式：

$$\Lambda\left(n+1\right)=\sum_{k=0}^{N-1}\left|\delta_k\left(n\right)+\mu\frac{\left(v\left(n\right)-\sum_i\delta_i\left(n\right)x\left(n-i\right)\right)x^*\left(n-k\right)}{\sum_{i=0}^{N-1}\left|x\left(n-i\right)\right|^2}\right|^2\tag{2.6.4}$$

如果 $x\left(n\right)$ 和 $v\left(n\right)$ 是不相关的白噪声信号，则下式：

$$E\left\{\Lambda\left(n+1\right)|\Lambda\left(n\right),x\left(n\right)\right\}=\Lambda\left(n\right)\left[1-\frac{2\mu}{N}+\frac{\mu^2}{N}+\frac{\mu^2\sigma_v^2}{\Lambda\left(n\right)\sum_{i=0}^{N-1}\left|x\left(n-i\right)\right|^2}\right]\tag{2.6.5}$$

可以通过求解 $\partial E\left\{\Lambda\left(n+1\right)\right\}/\partial\mu=0,\Lambda\neq0$：

$$\frac{-2}{N}+\frac{2\mu}{N}+\frac{2\mu\sigma_v^2}{\Lambda\left(n\right)\sum_{i=0}^{N-1}\left|x\left(n-1\right)\right|^2}=0\tag{2.6.6}$$

求解后得到最优步长：

$$\mu_{opt}\left(n\right)=\frac{1}{1+\frac{\sigma_v^2}{\Lambda\left(n\right)\left(1/N\right)\sum_{i=0}^{N-1}\left|x\left(n-i\right)\right|^2}}\tag{2.6.7}$$

期望 $\Lambda\left(n\right)\left(1/N\right)\sum_{i=0}^{N-1}\left|x\left(n-i\right)\right|^2$ 等于剩余回声的方差 $\sigma_r^2\left(n\right)$ ，如果剩余回声的方差值等于 0，则步长因子等于 1，$r\left(n\right)=y\left(n\right)-\hat{y}\left(n\right)$ ，则有输出信号的方差是：

$$\sigma_e^2\left(n\right)=\sigma_v^2\left(n\right)+\sigma_r^2\left(n\right)\tag{2.6.8}$$

这样可以求得这种情况下的最优步长因子为：

$$\mu_{opt}\left(n\right)\approx\frac{\sigma_r^2\left(n\right)}{\sigma_e^2\left(n\right)}\tag{2.6.9}$$

则最优步长因子如下：

$$\hat{\mu}_{opt}(n)=\min\left(\frac{\hat{\sigma}_r^2(n)}{\hat{\sigma}_e^2(n)},1\right) \quad 2.6.10$$

当 $\Lambda(n)\approx\frac{\sigma_v^2}{\sigma_x^2\left(\frac{2}{\mu}-1\right)}$ 时，式 2.6.5 的迭代将停止（滤波器系数不在更新，$E\{\Lambda(n+1)\}=\Lambda(n)$ ）。将 2.6.9 带入 2.6.10 得到在滤波器系数不更新情况下的剩余回声：

$$\sigma_r^2(n)\approx\min\left(\frac{1}{2}\hat{\sigma}_r^2(n),\sigma_v^2(n)\right) \quad 2.6.11$$

频域解

和时域相比，频域可以使步长因子 $\mu(k,l)$ 按频域划分，$Y(k,l)$ 和 $E(k,l)$ 分别是频域中的记号，其和时域中的 $\hat{y}(n)$ 和 $e(n)$ 是对等的关系。k 是频域索引，l 是帧索引。和 2.6.9 类似，可得频域步长因子如下：

$$\mu_{opt}(k,l)\approx\frac{\sigma_r^2(k,l)}{\sigma_e^2(k,l)} \quad 2.6.12$$

假设滤波器有一个和频谱无关的泄露（滤波器的误差）系数 $\eta(l)$ ，这将得到：

$$\hat{\sigma}_r^2(k,l)=\hat{\eta}(l)\hat{\sigma}_{\hat{Y}}^2(k,l) \quad 2.6.13$$

$\eta(l)$ 实际上是滤波器的回声返回损失增强（ERLE）。

为了让步长因子调节的更快，使用瞬时估计，$\hat{\sigma}_y(k,l)=|Y(k,l)|^2$ 和 $\hat{\sigma}_e(k,l)=|E(k,l)|^2$，这将使 2.6.10 步长因子调整变为下式：

$$\hat{\mu}_{opt}(k,l)=\min\left(\hat{\eta}(l)\frac{|\hat{Y}(k,l)|^2}{|E(k,l)|^2},\mu_{\max}\right) \quad 2.6.14$$

$\mu_{\max}$ 是小于等于 1 的数，以确保滤波器稳定。

WebRTC AEC 算法

AEC 算法：

AEC 算法主要包括如下几个重要的模块：

1 回声延迟估计

2 NLMS（normalize least mean square）最小均方误差

3 NLP（）非线性滤波

4 CNG 舒适噪声产生，一般经典 aec 算法还包括双端检测（DT）

回声延迟估计

回声延迟长短对回声抵消器的性能有较大影响，过长的的滤波器抽头会带来较大的延迟，并且语音信号是短时平稳信号，过长的滤波器抽头也不适合短时平稳信号特征。

基于相关的延迟算法。该算法的主要思想是：

设 1 表示有说话音，0 表无说话声（或者很弱的说话声），参考端（远端）信号 $x(t)$ 和接收端信号可能的组合方式如下：

(0, 0); (0, 1); (1, 0); (1, 1)

webrtc 默认(1, 0)和（0，1）是不可能发生的。设在时间间隔 p 上，即 $p=1,2,...,p$,频带 q, $q=1,2,...,Q$,输入信号 x 加窗后的功率谱用 $X_{w(p,q)}$ 表示，其角标表示其加了窗函数。对每个频带的功率谱设定一个门限 $X_{w(p,q)}$,

如果 $X_w(p,q) >= X_w(p,q)threshold$,则 $X_w(p,q)=1$；

如果 $X_w(p,q) \; < \; X_w(p,q)threshold$,则 $X_w(p,q)=0$；

同理，对于信号 $y(t)$,加窗信号功率谱 $Y_W(p,q)$ 和门限 $Y_W(p,q)threshold$,

如果 $Y_w(p,q) \; >= \; Y_w(p,q)threshold$，则 $Y_w(p,q)$=1；

如果 $Y_w(p,q) < \; Y_w(p,q)threshold$，则 $Y_w(p,q)$=0；考虑到实际处理的方便，在 webrtc 的 c 代码中，将经过 fft 变换后的频域功率谱分为 32 个子带，这样每个特定子带 $X_w(p,q)$ 的值可以用 1 个比特来表示，总共需要 32 个比特，只用一个 32 位数据类型就可以表示。

2）NLMS 归一化最小均方自适应算法

LMS/NLMS/RLS 都是经典自适应滤波算法，设远端信号为 $x(n)$,近端信号为 $d(n)$，则误差信号 $e(n)=d(n)-w(n)x(n)$,此处'表示转置，因为信号一般使用列向量表示的。NULMS 对滤波器的系数更新使用变步长方法，即步长 $u=u0/\left(gamma+x'(n)*x(n)\right)$;其中 u0 为更新步长因子，gamma 是稳定因子，则滤波器系数更新方程为 $W(n+1)=W(n)+u*e(n)*x(n)$;

3)NLP(非线性滤波)

webrtc 采用了维纳滤波器，此处只给出传递函数的表达式，设估计的语音信号的功率谱为 $P_s(\omega)$,噪声的功率谱为 $P_n(\omega)$，则滤波器的传递函数为 $H(\omega)=P_s(\omega)/\left(P_s(\omega)+P_n(\omega)\right)$。

4）CNG（舒适噪声产生）

首先生成在[0, 1]上均与分布随机噪声矩阵，再用噪声的功率谱开方后去调制噪声的幅度。

fullaec 算法需要注意两点：

1）延迟要小，因为算法默认滤波器长度是分为 12 块，每块 64 点，按照 8000 采样率，也就是 12×8ms=96ms 的数据，超过这个长度就处理不了了。

2）延迟抖动要小，因为算法是默认 10 块也计算一次参考数据的位置（即滤波器能量最大的那一块），所以如果抖动很大的话，找参考数据不准确的，这样回声就消除不掉了。