此，它是最小均方误差意义下的最优滤波器。此时，失调 $\mathcal{M}=0$，即滤波器不存在任何失调。由此可以看出，失调 $\mathcal{M}$ 实际上是一个衡量滤波器偏离最优滤波器的测度。只要剩余输出能量不等于零，便称滤波器存在失调。通常希望自适应滤波器的失调越小越好，这取决于滤波器的设计和滤波器所处的环境（例如，滤波器希望跟踪的信号的非平稳度）。

下面分析非平稳度 α 与自适应滤波器的失调之间的关系。对于一个由 MMSE 准则设计的滤波器来说，其最小均方误差 $J_{\min}$ 等于测量噪声的方差，即有

$$J_{\min}=\sigma_v^2 \tag{4.5.67}$$

另一方面，由 Malkov 模型公式 (4.5.61) 可以看出，过程噪声向量 $\boldsymbol{\omega}(n)$ 实际上就是滤波器权误差向量，即 $\boldsymbol{\omega}(n)\approx \boldsymbol{w}_{\text{opt}}(n+1)-\boldsymbol{w}_{\text{opt}}(n)=\boldsymbol{\epsilon}(n)$，因为式 (4.5.61) 中的系数 a 非常接近于 1。这表明 n 时刻的过程噪声向量 $\boldsymbol{\omega}(n)$ 的相关矩阵 $\boldsymbol{Q}(n)=\mathrm{E}\{\boldsymbol{\omega}(n)\boldsymbol{\omega}^{\mathrm{H}}(n)\}\approx\mathrm{E}\{\boldsymbol{\epsilon}(n)\boldsymbol{\epsilon}^{\mathrm{H}}(n)\}=\boldsymbol{K}(n)$。将这一关系代入式 (4.5.59)，立即得到自适应滤波器 n 时刻的剩余输出能量

$$\eta_{\text{ex}}(n)\approx \mathrm{tr}[\boldsymbol{RQ}]$$

若自适应滤波器不存在失调，则其稳态剩余输出能量 $J_{\text{ex}}=\eta_{\text{ex}}(\infty)=\mathrm{tr}[\boldsymbol{RQ}]$。即最小均方误差滤波器与最小能量滤波器等价。若自适应滤波器存在失调，则其稳态剩余输出能量 $J_{\text{ex}}>\mathrm{tr}[\boldsymbol{RQ}]$。即是说，

$$J_{\text{ex}}>\mathrm{tr}[\boldsymbol{RQ}] \tag{4.5.68}$$

将式 (4.5.67) 和式 (4.5.68) 代入式 (4.5.66)，则有

$$\mathcal{M}>\frac{\mathrm{tr}[\boldsymbol{RQ}]}{\sigma_v^2}=\alpha^2 \tag{4.5.69}$$

换句话说，自适应滤波器的失调 $\mathcal{M}$ 是时变系统的非平稳度的平方值的上界。

以上分析给出下面的结论 [86]：

- 对于慢时变系统，由于其非平稳度 α 小，因此，自适应滤波器可以跟踪时变系统的变化。
- 若时变系统的变化太快，以致于非平稳度 α 大于 1，那么在这样的情况下，由自适应滤波器造成的失调 $\mathcal{M}>1$，即失调将超过 100%。这意味着，自适应滤波器将不可能跟踪这种快时变系统的变化。

4.6 RLS 自适应算法

这一节将把最小二乘法推广为一种自适应算法，其目的是设计自适应的横向滤波器，使得在已知 $n-1$ 时刻横向滤波器抽头权系数的情况下，能够通过简单的更新，求出 n 时刻的滤波器抽头权系数。这样一种自适应的最小二乘算法称为递推最小二乘算法，简称 RLS 算法。

4.6.1 RLS 算法

与一般的最小二乘方法不同，这里考虑一种指数加权的最小二乘方法。顾名思义，在这种方法里，使用指数加权的误差平方和作为代价函数，即有

$$J(n)=\sum_{i=0}^{n}\lambda^{n-i}|\epsilon(i)|^2 \tag{4.6.1a}$$

式中，加权因子 $0<\lambda<1$ 称作遗忘因子，其作用是对离 n 时刻越近的误差加比较大的权重，而对离 n 时刻越远的误差加比较小的权重。换句话说，λ 对各个时刻的误差具有一定的遗忘作用，故称之为遗忘因子。从这个意义上讲，$\lambda=1$ 相当于各时刻的误差被“一视同仁”，即无任何遗忘功能，或具有无穷记忆功能。此时，指数加权的最小二乘方法退化为一般的最小二乘方法。反之，若 $\lambda=0$，则只有现时刻的误差起作用，而过去时刻的误差完全被遗忘，不起任何作用。在非平稳环境中，为了跟踪变化的系统，这两个极端的遗忘因子值都是不适合的。

式 (4.6.1a) 中的估计误差定义为

$$\epsilon(i)=d(i)-\boldsymbol{w}^{\mathrm{H}}(n)\boldsymbol{u}(i) \tag{4.6.1b}$$

式中 $d(i)$ 代表 i 时刻的期望响应。在期望响应不能已知的情况下，可取滤波器的实际输出直接作为期望响应 $d(i)$。注意，式 (4.6.1b) 中的抽头权向量为 n 时刻的权向量 $\boldsymbol{w}(n)$，而不是 i 时刻的权向量 $\boldsymbol{w}(i)$，其理由如下：在自适应更新过程中，滤波器总是越来越好，这意味着，对于任何时刻 $i\leqslant n$ 而言，估计误差的绝对值 $|\epsilon(i)|=|d(i)-\boldsymbol{w}^{\mathrm{H}}(n)\boldsymbol{u}(i)|$ 总是比 $|e(i)|=|d(i)-\boldsymbol{w}^{\mathrm{H}}(i)\boldsymbol{u}(i)|$ 小。因此，由 $\epsilon(i)$ 构成的代价函数 $J(n)$ 总是比由 $e(i)$ 构成的代价函数 $\tilde{J}(n)$ 小，故代价函数 $J(n)$ 比 $\tilde{J}(n)$ 更合理。根据定义，$\epsilon(i)$ 称为滤波器在 i 时刻的后验估计误差，而 $e(i)$ 则称为 i 时刻的先验

估计误差。因此，加权误差平方和的完整表达式为

$$J(n)=\sum_{i=0}^{n}\lambda^{n-i}|d(i)-\boldsymbol{w}^{\mathrm{H}}(n)\boldsymbol{u}(i)|^2 \tag{4.6.2}$$

它是 $\boldsymbol{w}(n)$ 的函数。由 $\dfrac{\partial J(n)}{\partial \boldsymbol{w}}=0$，易得 $\boldsymbol{R}(n)\boldsymbol{w}(n)=\boldsymbol{r}(n)$，其解为

$$\boldsymbol{w}(n)=\boldsymbol{R}^{-1}(n)\boldsymbol{r}(n) \tag{4.6.3}$$

式中

$$\boldsymbol{R}(n)=\sum_{i=0}^{n}\lambda^{n-i}\boldsymbol{u}(i)\boldsymbol{u}^{\mathrm{H}}(i) \tag{4.6.4}$$

$$\boldsymbol{r}(n)=\sum_{i=0}^{n}\lambda^{n-i}\boldsymbol{u}(i)d^{*}(i) \tag{4.6.5}$$

式 (4.6.3) 表明，指数加权最小二乘问题 (4.6.1) 的解 $\boldsymbol{w}(n)$ 再一次为 Wiener 滤波器。下面考虑它的自适应更新。

根据定义式 (4.6.4) 和式 (4.6.5)，易得递推估计公式：

$$\boldsymbol{R}(n)=\lambda\boldsymbol{R}(n-1)+\boldsymbol{u}(n)\boldsymbol{u}^{\mathrm{H}}(n) \tag{4.6.6a}$$

$$\boldsymbol{r}(n)=\lambda\boldsymbol{r}(n-1)+\boldsymbol{u}(n)d^{*}(n) \tag{4.6.6b}$$

对式 (4.6.6a) 使用著名的矩阵求逆引理，又可得逆矩阵 $\boldsymbol{P}(n)=\boldsymbol{R}^{-1}(n)$ 的递推公式：

$$\begin{aligned}\boldsymbol{P}(n)&=\frac{1}{\lambda}\left[\boldsymbol{P}(n-1)-\frac{\boldsymbol{P}(n-1)\boldsymbol{u}(n)\boldsymbol{u}^{\mathrm{H}}(n)\boldsymbol{P}(n-1)}{\lambda+\boldsymbol{u}^{\mathrm{H}}(n)\boldsymbol{P}(n-1)\boldsymbol{u}(n)}\right]\\&=\frac{1}{\lambda}\left[\boldsymbol{P}(n-1)-\boldsymbol{k}(n)\boldsymbol{u}^{\mathrm{H}}(n)\boldsymbol{P}(n-1)\right]\end{aligned} \tag{4.6.7}$$

式中 $\boldsymbol{k}(n)$ 称为增益向量，定义为

$$\boldsymbol{k}(n)=\frac{\boldsymbol{P}(n-1)\boldsymbol{u}(n)}{\lambda+\boldsymbol{u}^{\mathrm{H}}(n)\boldsymbol{P}(n-1)\boldsymbol{u}(n)} \tag{4.6.8}$$

利用式 (4.6.7) 不难证明：

$$\begin{aligned}\boldsymbol{P}(n)\boldsymbol{u}(n)&=\frac{1}{\lambda}\left[\boldsymbol{P}(n-1)\boldsymbol{u}(n)-\boldsymbol{k}(n)\boldsymbol{u}^{\mathrm{H}}(n)\boldsymbol{P}(n-1)\boldsymbol{u}(n)\right]\\&=\frac{1}{\lambda}\left\{\left[\lambda+\boldsymbol{u}^{\mathrm{H}}(n)\boldsymbol{P}(n-1)\boldsymbol{u}(n)\right]\boldsymbol{k}(n)-\boldsymbol{k}(n)\boldsymbol{u}^{\mathrm{H}}(n)\boldsymbol{P}(n-1)\boldsymbol{u}(n)\right\}\\&=\boldsymbol{k}(n)\end{aligned} \tag{4.6.9}$$

另一方面，由式 (4.6.3) 又有

$$\begin{aligned}\boldsymbol{w}(n) =& \boldsymbol{R}^{-1}(n)\boldsymbol{r}(n) = \boldsymbol{P}(n)\boldsymbol{r}(n)\\ =& \frac{1}{\lambda}\left[\boldsymbol{P}(n-1) - \boldsymbol{k}(n)\boldsymbol{u}^{\mathrm{H}}(n)\boldsymbol{P}(n-1)\right]\left[\lambda\boldsymbol{r}(n-1) + d^*(n)\boldsymbol{u}(n)\right]\\ =& \boldsymbol{P}(n-1)\boldsymbol{r}(n-1) + \frac{1}{\lambda}d^*(n)\left[\boldsymbol{P}(n-1)\boldsymbol{u}(n) - \boldsymbol{k}(n)\boldsymbol{u}^{\mathrm{H}}(n)\boldsymbol{P}(n-1)\boldsymbol{u}(n)\right] -\\ & \boldsymbol{k}(n)\boldsymbol{u}^{\mathrm{H}}(n)\boldsymbol{P}(n-1)\boldsymbol{r}(n-1)\end{aligned}$$

代入式 (4.6.9) 后，上式可写作

$$\boldsymbol{w}(n) = \boldsymbol{w}(n-1) + d^*(n)\boldsymbol{k}(n) - \boldsymbol{k}(n)\boldsymbol{u}^{\mathrm{H}}(n)\boldsymbol{w}(n-1)$$

经化简后，得

$$\boldsymbol{w}(n) = \boldsymbol{w}(n-1) + \boldsymbol{k}(n)e^*(n) \tag{4.6.10}$$

式中

$$e(n) = d(n) - \boldsymbol{u}^{\mathrm{T}}(n)\boldsymbol{w}^*(n-1) \tag{4.6.11a}$$

$$= d(n) - \boldsymbol{w}^{\mathrm{H}}(n-1)\boldsymbol{u}(n) \tag{4.6.11b}$$

为先验估计误差。

综上所述，可以得到 RLS 直接算法如下。

算法 4.6.1 (RLS 直接算法)

步骤 1 初始化：$\boldsymbol{w}(0) = \boldsymbol{0}$，$\boldsymbol{P}(0) = \delta^{-1}\boldsymbol{I}$，其中 δ 是一个很小的值。

步骤 2 更新：$n = 1, 2, \cdots$

$$e(n) = d(n) - \boldsymbol{w}^{\mathrm{H}}(n-1)\boldsymbol{u}(n)$$

$$\boldsymbol{k}(n) = \frac{\boldsymbol{P}(n-1)\boldsymbol{u}(n)}{\lambda + \boldsymbol{u}^{\mathrm{H}}(n)\boldsymbol{P}(n-1)\boldsymbol{u}(n)}$$

$$\boldsymbol{P}(n) = \frac{1}{\lambda}\left[\boldsymbol{P}(n-1) - \boldsymbol{k}(n)\boldsymbol{u}^{\mathrm{H}}(n)\boldsymbol{P}(n-1)\right]$$

$$\boldsymbol{w}(n) = \boldsymbol{w}(n-1) + \boldsymbol{k}(n)e^*(n)$$

RLS 算法的应用需要初始值 $\boldsymbol{P}(0) = \boldsymbol{R}^{-1}(0)$。在非平稳情况下，此初始值由下式决定：

$$\boldsymbol{P}(0) = \boldsymbol{R}^{-1}(0) = \left[\sum_{i=-n_0}^{0}\lambda^{-i}\boldsymbol{u}(i)\boldsymbol{u}^{\mathrm{H}}(i)\right]^{-1} \tag{4.6.12}$$

因此，相关矩阵的表达式 (4.6.5a) 变作

$$\boldsymbol{R}(n)=\sum_{i=1}^{n}\lambda^{n-i}\boldsymbol{u}(i)\boldsymbol{u}^{\mathrm{H}}(i)+\boldsymbol{R}(0) \tag{4.6.13}$$

由于 λ 的遗忘作用，自然希望 $\boldsymbol{R}(0)$ 在式 (4.6.13) 中起的作用很小。考虑到这一点，不妨用一个很小的单位矩阵来近似 $\boldsymbol{R}(0)$，即

$$\boldsymbol{R}(0)=\delta\boldsymbol{I},\quad \delta \text{ 是一很小的正数} \tag{4.6.14}$$

因此，$\boldsymbol{P}(0)$ 的初始值由下式给出：

$$\boldsymbol{P}(0)=\delta^{-1}\boldsymbol{I},\quad \delta \text{ 是一很小的正数} \tag{4.6.15}$$

这就是为什么在算法 4.6.1 中初始值取 $\boldsymbol{P}(0)=\delta^{-1}\boldsymbol{I}$ (其中 δ 很小) 的理由。

δ 的值越小，相关矩阵初始值 $\boldsymbol{R}(0)$ 在 $\boldsymbol{R}(n)$ 的计算中所占比重越小，这是我们所希望的；反之，$\boldsymbol{R}(0)$ 的作用就会突现出来，这是应该避免的。δ 的典型取值为 $\delta=0.01$ 或更小。一般情况下，取 $\delta=0.01$ 与 $\delta=10^{-4}$ 时，RLS 算法给出的结果并没有明显的区别，但是取 $\delta=1$ 将严重影响 RLS 算法的收敛速度及收敛结果，这一点是在应用 RLS 算法时必须注意的。

4.6.2 RLS 算法与 Kalman 滤波算法的比较

考虑一特殊的“无激励”动态模型：

$$\boldsymbol{x}(n+1)=\lambda^{-1/2}\boldsymbol{x}(n) \tag{4.6.16}$$

$$y(n)=\boldsymbol{u}^{\mathrm{H}}(n)\boldsymbol{x}(n)+v(n) \tag{4.6.17}$$

式中 $\boldsymbol{x}(n)$ 为模型的状态向量，$y(n)$ 为一标量的观测值或参考信号，$\boldsymbol{u}^{\mathrm{H}}(n)$ 为观测矩阵，且 $v(n)$ 表示一标量白噪声过程，它具有零均值和单位方差。模型参数 λ 是个正的实常数。由式 (4.6.16) 容易看出：

$$\boldsymbol{x}(n)=\lambda^{-n/2}\boldsymbol{x}(0) \tag{4.6.18}$$

式中 $\boldsymbol{x}(0)$ 是状态向量的初始值。将式 (4.6.18) 代入式 (4.6.17)，并使用共同项 $\boldsymbol{x}(0)$ 表示各个时刻的观测值，则有

$$\left.\begin{aligned} y(0) &= \boldsymbol{u}^{\mathrm{H}}(0)\boldsymbol{x}(0) + v(0) \\ y(1) &= \lambda^{-1/2}\boldsymbol{u}^{\mathrm{H}}(1)\boldsymbol{x}(0) + v(1) \\ &\vdots \\ y(n) &= \lambda^{-n/2}\boldsymbol{u}^{\mathrm{H}}(n)\boldsymbol{x}(0) + v(n) \end{aligned}\right\} \tag{4.6.19}$$

或等价写作

$$\left.\begin{aligned} y(0) &= \boldsymbol{u}^{\mathrm{H}}(0)\boldsymbol{x}(0) + v(0) \\ \lambda^{1/2}y(1) &= \boldsymbol{u}^{\mathrm{H}}(1)\boldsymbol{x}(0) + \lambda^{1/2}v(1) \\ &\vdots \\ \lambda^{n/2}y(n) &= \boldsymbol{u}^{\mathrm{H}}(n)\boldsymbol{x}(0) + \lambda^{n/2}v(n) \end{aligned}\right\} \tag{4.6.20}$$

从 Kalman 滤波的观点看，方程组 (4.6.20) 表示无激励动态模型的随机特性.

与 Kalman 滤波器使用随机模型不同，RLS 算法则采用确定模型，即期望信号 (也称参考信号) 可以用线性回归模型表示为

$$\left.\begin{aligned} d^*(0) &= \boldsymbol{u}^{\mathrm{H}}(0)\boldsymbol{w}_{\mathrm{o}} + e_{\mathrm{o}}^*(0) \\ d^*(1) &= \boldsymbol{u}^{\mathrm{H}}(1)\boldsymbol{w}_{\mathrm{o}} + e_{\mathrm{o}}^*(1) \\ &\vdots \\ d^*(n) &= \boldsymbol{u}^{\mathrm{H}}(n)\boldsymbol{w}_{\mathrm{o}} + e_{\mathrm{o}}^*v(n) \end{aligned}\right\} \tag{4.6.21}$$

式中 $\boldsymbol{w}_{\mathrm{o}}$ 表示模型的未知参数向量，$\boldsymbol{u}(n)$ 为输入向量，而 $e_{\mathrm{o}}(n)$ 为观测噪声.

若令 Kalman 滤波器中状态向量的初始值等于 RLS 算法中的抽头权向量，即

$$\boldsymbol{x}(0) = \boldsymbol{w}_{\mathrm{o}} \tag{4.6.22}$$

则很容易看出，RLS 算法的确定模型 (4.6.21) 与 Kalman 滤波算法的特殊随机模型 (4.6.20) 等价的条件是下面的一一对应关系：

$$y(n) = \lambda^{-n/2}d^*(n) \tag{4.6.23}$$

$$v(n) = \lambda^{-n/2}e_{\mathrm{o}}^*(n) \tag{4.6.24}$$

该式左边为状态空间模型的参数，而该式右边为线性回归模型的参数.

总结以上分析，可以得出如下结论：RLS 自适应算法使用的确定性线性回归模型是 Kalman 滤波算法的一种特殊的无激励的状态空间模型.

这一等价关系是 Sayed 与 Kailaith 于 1994 年建立的 [168]。

表 4.6.1 综合了 Kalman 滤波算法和 RLS 算法之间各个变量的对应关系 [86]。

表 4.6.1 Kalman 滤波算法与 RLS 滤波算法之间的变量对应关系

Kalman 算法		RLS 算法	
参数名称	变量	变量	参数名称
初始状态向量	$\boldsymbol{x}(0)$	$\boldsymbol{w}_o$	抽头权向量
状态向量	$\boldsymbol{x}(n)$	$\lambda^{-n/2}\boldsymbol{w}_o$	指数加权的抽头权向量
参考 (观测) 信号	$y(n)$	$\lambda^{-n/2}d^*(n)$	期望响应
观测噪声	$v(n)$	$\lambda^{-n/2}e_o^*(n)$	测量误差
一步预测的状态向量	$\hat{\boldsymbol{x}}(n+1\|y_1,\cdots,y_n)$	$\lambda^{-n/2}\hat{\boldsymbol{w}}(n)$	抽头权向量的估计
状态预测误差的相关矩阵	$\boldsymbol{K}(n)$	$\lambda^{-1}\boldsymbol{P}(n)$	输入向量相关矩阵的逆矩阵
Kalman 增益	$\boldsymbol{g}(n)$	$\lambda^{-1/2}\boldsymbol{k}(n)$	增益向量
新息	$\alpha(n)$	$\lambda^{-n/2}\xi^*(n)$	先验估计误差
初始条件	$\hat{\boldsymbol{x}}(1)=\boldsymbol{0}$	$\hat{\boldsymbol{w}}(0)=\boldsymbol{0}$	初始条件
	$\boldsymbol{K}(0)$	$\delta^{-1}\boldsymbol{P}(0)$	

4.6.3 RLS 算法的统计性能分析

由于 Wiener 滤波器的测量误差 $e_{\text{opt}}(n)=d(n)-\boldsymbol{w}_{\text{opt}}^{\text{H}}\boldsymbol{u}(n)$ 具有最小的均方值，因此期望响应 $d(n)$ 可写作

$$d(n)=e_{\text{opt}}(n)+\boldsymbol{w}_{\text{opt}}^{\text{H}}\boldsymbol{u}(n) \tag{4.6.25}$$

上式称为期望响应 $d(n)$ 的线性回归模型，$M\times 1$ 权向量 $\boldsymbol{w}_{\text{opt}}$ 表示模型的回归参数向量。由式 (4.6.11) 及式 (4.6.25)，消去 $d(n)$，即可将先验估计误差表示为

$$\begin{aligned} e(n) &= e_{\text{opt}}(n)-\left[\boldsymbol{w}(n-1)-\boldsymbol{w}_{\text{opt}}\right]^{\text{H}}\boldsymbol{u}(n) \\ &= e_{\text{opt}}(n)-\boldsymbol{\epsilon}^{\text{H}}(n-1)\boldsymbol{u}(n) \end{aligned} \tag{4.6.26}$$

式中

$$\boldsymbol{\epsilon}(n-1)=\boldsymbol{w}(n-1)-\boldsymbol{w}_{\text{opt}} \tag{4.6.27}$$

表示 $n-1$ 时刻的实际抽头权向量与最优 Wiener 滤波器抽头权向量之差，简称权误差向量。

考虑先验估计误差的均方值即均方估计误差

$$\xi(n) = \mathrm{MSE}(w(n)) = \mathrm{E}\{|e(n)|^2\} \tag{4.6.28}$$

将式 (4.6.26) 代入式 (4.6.28)，并加以整理，得

$$\begin{aligned}\xi(n) =& \mathrm{E}\{|e_{\mathrm{opt}}(n)|^2\} + \mathrm{E}\left\{\boldsymbol{u}^{\mathrm{H}}(n)\boldsymbol{\epsilon}(n-1)\boldsymbol{\epsilon}^{\mathrm{H}}(n-1)\boldsymbol{u}(n)\right\} - \\ & \mathrm{E}\left\{e_{\mathrm{opt}}(n)\boldsymbol{u}^{\mathrm{H}}(n)\boldsymbol{\epsilon}(n-1)\right\} - \mathrm{E}\left\{\boldsymbol{\epsilon}^{\mathrm{H}}(n-1)\boldsymbol{u}(n)e_{\mathrm{opt}}^{*}(n)\right\}\end{aligned} \tag{4.6.29}$$

下面具体分析式 (4.6.29) 右边各项的值。

(1) 该式右边第一项表示最优 Wiener 滤波器的均方误差，它是所有滤波器所能具有的最小均方误差，记作

$$\xi_{\min} = \mathrm{E}\{|e_{\mathrm{opt}}(n)|^2\} \tag{4.6.30a}$$

(2) 计算式 (4.6.29) 右边的第二项，易得

$$\begin{aligned}\mathrm{E}\left\{\boldsymbol{u}^{\mathrm{H}}(n)\boldsymbol{\epsilon}(n-1)\boldsymbol{\epsilon}^{\mathrm{H}}(n-1)\boldsymbol{u}(n)\right\} &= \mathrm{E}\left\{\mathrm{tr}\left[\boldsymbol{u}^{\mathrm{H}}(n)\boldsymbol{\epsilon}(n-1)\boldsymbol{\epsilon}^{\mathrm{H}}(n-1)\boldsymbol{u}(n)\right]\right\} \\ &= \mathrm{E}\left\{\mathrm{tr}\left[\boldsymbol{u}(n)\boldsymbol{u}^{\mathrm{H}}(n)\boldsymbol{\epsilon}(n-1)\boldsymbol{\epsilon}^{\mathrm{H}}(n)\right]\right\} \\ &= \mathrm{tr}\left[\mathrm{E}\left\{\boldsymbol{u}(n)\boldsymbol{u}^{\mathrm{H}}(n)\boldsymbol{\epsilon}(n-1)\boldsymbol{\epsilon}^{\mathrm{H}}(n)\right\}\right] \\ &= \mathrm{tr}\left[\mathrm{E}\left\{\boldsymbol{u}(n)\boldsymbol{u}^{\mathrm{H}}(n)\right\}E\left\{\boldsymbol{\epsilon}(n-1)\boldsymbol{\epsilon}^{\mathrm{H}}(n-1)\right\}\right] \\ &= \mathrm{tr}\left[\boldsymbol{R}\boldsymbol{K}(n-1)\right]\end{aligned} \tag{4.6.30b}$$

式中，$\boldsymbol{R} = \mathrm{E}\{\boldsymbol{u}(n)\boldsymbol{u}^{\mathrm{H}}(n)\}$ 是滤波器输入的相关矩阵，而 $\boldsymbol{K}(n-1) = \mathrm{E}\{\boldsymbol{\epsilon}(n-1)\boldsymbol{\epsilon}^{\mathrm{H}}(n-1)\}$ 为 $n-1$ 时刻的权误差相关矩阵。

(3) 由于 $n-1$ 时刻的权误差向量 $\boldsymbol{\epsilon}(n-1)$ 与 n 时刻的输入向量 $\boldsymbol{u}(n)$、测量误差 $e_{\mathrm{opt}}(n)$ 统计独立，故式 (4.6.29) 右边第三项为

$$\mathrm{E}\left\{e_{\mathrm{opt}}(n)\boldsymbol{u}^{\mathrm{H}}(n)\boldsymbol{\epsilon}^{\mathrm{H}}(n-1)\right\} = \mathrm{E}\left\{e_{\mathrm{opt}}(n)\boldsymbol{u}^{\mathrm{H}}(n)\right\}\mathrm{E}\left\{\boldsymbol{\epsilon}(n-1)\right\}$$

又由正交性原理知，测量误差 $e_{\mathrm{opt}}(n)$ 与输入向量 $\boldsymbol{u}(n)$ 的所有元素正交，即 $\mathrm{E}\{e_{\mathrm{opt}}(n)\cdot\boldsymbol{u}^{\mathrm{H}}(n)\} = 0$，故

$$\mathrm{E}\left\{e_{\mathrm{opt}}(n)\boldsymbol{u}^{\mathrm{H}}(n)\boldsymbol{\epsilon}^{\mathrm{H}}(n-1)\right\} = 0 \tag{4.6.30c}$$

(4) 同理，有

$$\begin{aligned}\mathrm{E}\left\{\boldsymbol{\epsilon}^{\mathrm{H}}(n-1)\boldsymbol{u}(n)e_{\mathrm{opt}}^{*}(n)\right\} &= \mathrm{E}\left\{\boldsymbol{\epsilon}^{\mathrm{H}}(n-1)\right\}\mathrm{E}\left\{\boldsymbol{u}(n)e_{\mathrm{opt}}^{*}(n)\right\} \\ &= 0\end{aligned} \tag{4.6.30d}$$

将式 (4.6.30a) ~ 式 (4.6.30d) 代入式 (4.6.29), 得

$$\xi(n) = \xi_{\min} + \mathrm{tr}\,[\boldsymbol{R}\boldsymbol{K}(n-1)] \tag{4.6.31}$$

由此可求出剩余均方误差

$$\xi_{\mathrm{ex}}(n) = \xi(n) - \xi_{\min} = \mathrm{tr}\,[\boldsymbol{R}\boldsymbol{K}(n-1)] \tag{4.6.32}$$

及稳态或渐近剩余均方误差

$$\xi_{\mathrm{ex}}(\infty) = \lim_{n\to\infty} \mathrm{tr}\,[\boldsymbol{R}\boldsymbol{K}(n-1)] \tag{4.6.33}$$

实际测量的剩余均方误差 $\xi_{\mathrm{ex}}(n)$ 相对于迭代时间 n 的变化曲线称为 RLS 算法的学习曲线, 它是一条随时间衰减的曲线, 表示 RLS 算法的收敛性能 (速率与稳态剩余均方误差)。

4.6.4 快速 RLS 算法

业已证明 [117],[35],[36],[48], RLS 直接算法中的 Kalman 增益向量可以利用"快速"方式更新, 从而使得 RLS 算法可以快速实现。快速 RLS 算法的关键是恰当利用数据矩阵的移不变性质. 为此, 考虑数据向量 $\boldsymbol{x}_M(n) = [u(n), u(n-1), \cdots, u(n-M+1)]^{\mathrm{T}}$ 增加一阶后的数据向量 $\boldsymbol{x}_{M-1}(n) = [u(n), u(n-1), \cdots, u(n-M)]^{\mathrm{T}}$。显然, 它有两种不同的分块形式:

$$\boldsymbol{x}_{M+1}(n) = \begin{bmatrix} \boldsymbol{x}_M(n) \\ u(n-M) \end{bmatrix} = \begin{bmatrix} u(n) \\ \boldsymbol{x}_{M-1}(n-1) \end{bmatrix} \tag{4.6.34}$$

利用这两种分块形式, 即可得到增阶后的自相关矩阵 $\boldsymbol{R}_{M+1}(n)$ 的恰当分块。根据这些分块, n 时刻的 Kalman 增益向量 $\boldsymbol{c}_M(n)$ 便可以借助增阶后的向量 $\boldsymbol{c}_{M+1}(n)$ 由 $\boldsymbol{c}_{M-1}(n-1)$ 获得。总的更新机理可表示如下:

$$\begin{array}{ccccc} & \boldsymbol{a}_M(n-1) & & \boldsymbol{b}_M(n-1) & \\ & \downarrow & & \downarrow & \\ \boldsymbol{c}_{M-1}(n-1) & \longrightarrow & \boldsymbol{c}_{M+1}(n) & \longrightarrow & \boldsymbol{c}_M(n) \\ & \downarrow & & \downarrow & \\ & \boldsymbol{a}_M(n) & & \boldsymbol{b}_M(n) & \end{array} \tag{4.6.35}$$

其中, 辅助向量 $\boldsymbol{a}_M(n)$ 和 $\boldsymbol{b}_M(n)$ 分别表示前向和后向最小二乘预测器, 它们是在式 (4.5.1) 中分别令 $y(n) = u(n+1)$ 和 $y(n) = u(n-M)$ 之后的 FIR 横向滤波器。

算法 4.6.2 (稳定化的快速 RLS 算法)[78]

步骤 1 初始化: $\boldsymbol{w}_M(0) = 0$, $\boldsymbol{c}_M(0) = 0$, $\boldsymbol{a}_M(0) = [1, 0, \cdots, 0]^{\mathrm{T}}$

$$\boldsymbol{b}_M(0)=[0,\cdots,0,1]^{\mathrm{T}},\ \alpha_M(0)=1,\ \alpha_M^{\mathrm{f}}(0)=\lambda^M\alpha_M^{\mathrm{b}}(0),\ \alpha_M^{\mathrm{b}}(0)=\delta>0$$

步骤 2 $n=1,2,\cdots$

$$e_M^{\mathrm{f}}(n)=u(n)+\boldsymbol{a}_M^{\mathrm{T}}(n-1)\boldsymbol{x}_M(n-1) \tag{4.6.36}$$

$$\varepsilon_M^{\mathrm{f}}(n)=e_M^{\mathrm{f}}(n)/\alpha_M(n-1) \tag{4.6.37}$$

$$\boldsymbol{a}_M(n)=\boldsymbol{a}_M(n-1)-\boldsymbol{c}_M(n-1)\varepsilon_M^{\mathrm{f}}(n) \tag{4.6.38}$$

$$\alpha_M^{\mathrm{f}}(n)=\lambda\alpha_M^{\mathrm{f}}(n-1)+e_M^{\mathrm{f}}(n)\varepsilon_M^{\mathrm{f}}(n) \tag{4.6.39}$$

$$k_{M+1}(n)=\lambda^{-1}\alpha_M^{\mathrm{f}}(n-1)e_M^{\mathrm{f}}(n) \tag{4.6.40}$$

$$\boldsymbol{c}_{M+1}(n)=\begin{bmatrix}0\\ \boldsymbol{c}_M(n)\end{bmatrix}+\begin{bmatrix}1\\ \boldsymbol{a}_M(n-1)\end{bmatrix}k_{M+1}(n-1) \tag{4.6.41}$$

$$\boldsymbol{c}_{M+1}=\begin{bmatrix}\boldsymbol{d}_M(n)\\ d_{M+1}(n)\end{bmatrix} \tag{4.6.42}$$

$$e_M^{\mathrm{b}}(n)=\lambda\alpha_M^{\mathrm{b}}(n-1)d_{M+1}(n) \tag{4.6.43}$$

$$\tilde{e}_M^{\mathrm{b}}(n)=u(n-M)+\boldsymbol{b}_M^{\mathrm{T}}(n-1)\boldsymbol{x}_M(n) \tag{4.6.44}$$

$$\Delta^{\mathrm{b}}(n)=\tilde{e}_M^{\mathrm{b}}(n)-e_M^{\mathrm{b}}(n) \tag{4.6.45}$$

$$\hat{e}_{M,i}^{\mathrm{b}}(n)=\tilde{e}_M^{\mathrm{b}}(n)+\sigma_i\Delta^{\mathrm{b}}(n),\ i=1,2,3 \tag{4.6.46}$$

$$\boldsymbol{c}_M(n)=\boldsymbol{d}_M(n)-\boldsymbol{b}_M(n-1)d_{M+1}(n) \tag{4.6.47}$$

$$\alpha_{M+1}(n)=\alpha_M(n-1)-k_{M+1}(n)e_M^{\mathrm{f}}(n) \tag{4.6.48}$$

$$\alpha_M(n)=\alpha_{M+1}(n)+d_{M+1}(n)\hat{e}_{M,1}^{\mathrm{b}}(n) \tag{4.6.49}$$

$$\tilde{\alpha}_M(n+1)-1+\boldsymbol{c}_M^{\mathrm{T}}(n)\boldsymbol{x}_M(n) \tag{4.6.50}$$

$$\Delta^{\alpha}(n)=\tilde{\alpha}_M(n)+\sigma\Delta^{\alpha}(n) \tag{4.6.51}$$

$$\hat{\alpha}_M(n)=\tilde{\alpha}_M(n)+\sigma\Delta^{\alpha}(n) \tag{4.6.52}$$

$$\varepsilon_{M,i}^{\mathrm{b}}(n)=\hat{e}_{M,i}^{\mathrm{b}}(n)/\hat{\alpha}_M(n),\ i=2,3 \tag{4.6.53}$$

$$\boldsymbol{b}_M(n)=\boldsymbol{b}_M(n-1)-\boldsymbol{c}_M(n)\varepsilon_{M,2}^{\mathrm{b}}(n) \tag{4.6.54}$$

$$\alpha_M^{\mathrm{b}}(n)=\lambda\alpha_M^{\mathrm{b}}(n-1)+\hat{e}_{M,3}^{\mathrm{b}}(n)\varepsilon_{M,3}^{\mathrm{b}}(n) \tag{4.6.55}$$

$$e_M(n)=y(n)-\boldsymbol{w}^{\mathrm{T}}(n-1)\boldsymbol{x}_M(n) \tag{4.6.56}$$

$$\varepsilon_M(n)=e_M(n)/\hat{\alpha}_M(n) \tag{4.6.57}$$

$$\mu(n)=\alpha_M(n)/[1-\rho\hat{\alpha}_M(n)] \tag{4.6.58}$$

$$\boldsymbol{w}_M(n)=\boldsymbol{w}_M(n-1)+\mu(n)\boldsymbol{c}_M(n)\varepsilon_M(n) \tag{4.6.59}$$