第 5 章　基于 RLS 算法的数据预测与 MATLAB 实现

递归最小二乘（RLS）算法是一种典型的数据处理方法，由著名学者高斯在 1795 年提出，高斯认为，根据所获得的观测数据来推断未知参数时，未知参数最可能的值是这样一个数据，即它使各项实际观测值和计算值之间的差的平方乘以度量其精度的数值以后的和为最小，这就是著名的最小二乘。递归最小二乘（RLS）算法在信号自适应滤波分析中广泛应用，递归最小二乘（RLS）算法收敛速度快，且对自相关矩阵特征值的分散性不敏感，然而其计算量较大。本章主要内容是研究基于 RLS 进行数据的预测与 MATLAB 实现。

学习目标：

（1）学习和掌握递归最小二乘（RLS）算法原理；

（2）学习和掌握 MATLAB 编程实现 RLS 算法进行数据预测等。

5.1　递归最小二乘（RLS）算法应用背景

在自适应滤波系统中，最陡梯度（LMS）法由于其简便性及易用性得到广泛的应用。但各种最陡梯度（LMS）算法收敛速度较慢，特别是对于非平稳信号的适应性差。究其原因主要是各种最陡梯度（LMS）算法只是简单的用以各时刻的抽头参量等作为该时刻数据块估计，采用平方误差为极小原则，而没有考虑到当前时刻的抽头参量等，来对以往各时刻的数据块做重新估计，即没有采用累加平方误差最小原则（即所谓的最小平方（LS）准则）。

为了克服各种最陡梯度（LMS）算法收敛速度慢，信号非平稳适应性差等缺点，根据最小平方（LS）准则，即在每时刻对所有输入信号作重估，采用 LS 准则。这种方法是在现有的约束条件下，利用了最多的可利用信息的准则，在很大程度上提高了算法收敛速度，并且它也是一种最有效，信号非平稳的适应性能最好的算法之一。按照这样一种改进算法思路，广大学者建立起来的适应非平稳信号处理的方法就是递归最小二乘（RLS：Recursive Least Square）算法，又称为广义 Kalman 自适应算法。

用矩阵来表示 RLS 算法显得较为简便，因此我们首先定义一些向量和矩阵。假定在时刻 t，均衡器的输入信号为 r_t，线性均衡器对于信息符号的估计可以表示为：

$$\hat{I}(t)=\sum_{j=-K}^{K} c_j(t-1)r_{t-j} \tag{5.1}$$

式（5.1）中，$j=0,1,\cdots,(N-1)$，同时定义 $y(t)=v_{t+K}$，则 $\hat{I}(t)$ 变为：

$$\hat{I}(t)=\sum_{j=0}^{N-1}c_j(t-1)y(t-j)=C'_N(t-1)Y_N(t) \tag{5.2}$$

式（5.2）中，$C_N(t-1)$ 和 $Y_N(t)$ 分别为均衡器系数 $c_j(t-1)$，$j=0,1,\cdots,N-1$ 和输入信号 $y(t-j)$，$j=0,1,\cdots,N-1$ 的列向量。

同理，在 DFE 均衡器结构中，均衡器系数 $c_j(t)$，$j=0,1,\cdots,N-1$ 的前 K_1+1 个系数为前向滤波器系数，剩下的 $K_2=N-K_1-1$ 为反馈滤波器系数。用来预测 $\hat{I}(t)$ 的数据为 $r_{t+K_1},\cdots,r_{t+1},\tilde{I}_{t-1},\tilde{I}_{t-K_2}$，其中 $\tilde{I}_{t-j},1\leqslant j\leqslant K_2$ 为判决器先前作出判决的数据。

一般情况下，我们忽略判决器误判的情况，此时 $\tilde{I}_{t-j}=I_{t-j},1\leqslant j\leqslant K_2$。我们在这里也定义 $y(t-j)$ 的取值，具体如下：

$$y(t-j)=\begin{cases}v_{t+K_1-j}(0\leqslant j\leqslant K_1)\\ I_{t+K_1-j}(K_1<j\leqslant N-1)\end{cases} \tag{5.3}$$

因此有，

$$Y_N(t)=[y(t),y(t-1),\cdots,y(t-N+1)]' \tag{5.4}$$

$$Y_N(t)=[r_{t+K_1},\cdots,r_{t+1},r_t,I_{t-1},\cdots,I_{t-K_2}]' \tag{5.5}$$

假定我们的观测向量为 $Y_N(n)$，$n=0,1,\cdots,t$，我们期望得到均衡器的系数向量 $C_N(t)$ 使得均方误差的加权平方和最小。其中误差定义为：

$$\varepsilon(n)=\sum_{n=0}^{t}\lambda^{t-n}\,|\,e_N(n,t)\,|^2 \tag{5.6}$$

$$e_N(n,t)=I(n)-C'_N(t)Y_N(n) \tag{5.7}$$

式（5.6）中，λ 代表遗忘因子，$0<\lambda<1$。为什么引入遗忘因子呢？主要是考虑到信道时变特性，因此需要对过去的情况进行分析与处理。

5.2　RLS 算法基本原理与流程

5.2.1　RLS 算法基本原理

如图 5-1 所示的 M 抽头（M 个权系数）横向滤波器，滤波器输入信号 $u(n)$ 仅有 N 个输入数据，期望响应也仅有 N 个数据。

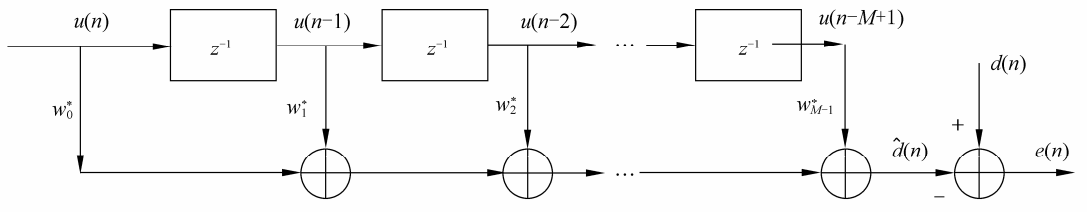

图 5-1　M 抽头权系数的横向滤波器

定义滤波器权向量 w、误差向量 e 和期望响应向量 b 分别为：

$$w=\left[w_0,w_1,\cdots,w_{M-1}\right]^T \tag{5.8}$$

$$e=\left[e(M),e(M+1),\cdots e(N)\right]^H \tag{5.9}$$

$$b=\left[d(M),d(M+1),\cdots,d(N)\right]^H \tag{5.10}$$

定义数据矩阵：

$$\begin{aligned} A^H &=\begin{bmatrix} u(M) & u(M+1) & \cdots & u(N) \end{bmatrix} \\ &=\begin{bmatrix} u(M) & u(M+1) & \cdots & u(N) \\ u(M-1) & u(M) & \cdots & u(N-1) \\ \vdots & \vdots & & \vdots \\ u(1) & u(2) & \cdots & u(N-M+1) \end{bmatrix} \end{aligned} \tag{5.11}$$

$$e=b-\hat{b}=b-Aw$$

横向滤波器的设计原则是，寻找权向量 w 使得误差信号 $e(n)$ 的模的平方和最小。

在此定义代价函数，如式（5.12）所示：

$$J=\sum_{n=M}^{N}\left|e(n)\right|^2=\left\|e\right\|^2=\mathrm{e}^H e \tag{5.12}$$

对式（5.12）求 J 关于 w 的梯度，并令其为零：

$$\nabla J=2\frac{\partial J}{\partial w^*}=-2A^H b+2A^H Aw=0 \tag{5.13}$$

得到确定性正则方程如式（5.14）所示。

$$A^H A\hat{w}=A^H b \tag{5.14}$$

当 $M<N-M+1$ 时，通常方阵 $A^H A$ 是非奇异的，得权向量的最小二乘解：

$$\hat{w}=\left(A^H A\right)^{-1}A^H b \tag{5.15}$$

5.2.2　RLS 算法流程

RLS 算法流程主要有如下几步。

（1）初始化：$P(0)=\delta I\in C^{M\times M}$，$\delta$ 是小的正数，$\hat{w}(0)=0$；遗忘因子 λ 一般取值接近于 1，例如 $\lambda=1$ 或 $\lambda=0.98$ 等。

（2）当 $n=1,2,\cdots,N$ 时，完成如下迭代运算：

$$k(n)=\frac{\lambda^{-1}P(n-1)u(n)}{1+\lambda^{-1}u^H(n)P(n-1)u(n)} \tag{5.16}$$

$$\xi(n)=d(n)-\hat{w}^H(n-1)u(n) \tag{5.17}$$

$$\hat{w}(n)=\hat{w}(n-1)+k(n)\xi^*(n) \tag{5.18}$$

$$P(n)=\lambda^{-1}P(n-1)-\lambda^{-1}k(n)u^H(n)P(n-1) \tag{5.19}$$

（3）更新 $n=n+1$，转（2）。

5.3　RLS 数据线性预测分析与 MATLAB 实现

考虑一阶 AR 模型$u(n)=-0.99u(n-1)+v(n)$的线性预测。假设白噪声$v(n)$的方差为$\sigma_v^2=0.995$。使用抽头数为 *M*=2 的 FIR 滤波器，用 RLS 算法实现$u(n)$的线性预测，选择遗忘因子 λ=0.98 。

产生满足该一阶 AR 模型的样本序列，编程如下：

```
clc,clear,close all                 % 清屏、清工作区、关闭窗口
warning off                         % 消除警告
feature jit off                     % 加速代码执行
N = 1000;                           % 信号观测长度
a1 = 0.99;                          % 一阶 AR 参数
sigma = 0.0731;                     % 加性白噪声方差
v = sqrt(sigma)*randn(N,1);         % 产生 v（n）加性白噪声
u0 = [0];                           % 初始数据
num = 1;                            % 分子系数
den = [1,a1];                       % 分母系数
Zi = filtic(num,den,u0);            % 滤波器的初始条件
un = filter(num,den,v,Zi);          % 产生样本序列 u(n), N x 1 x trials
figure,stem(un),title('随机信号');grid on;
```

运行程序输出图形如图 5-2 所示。

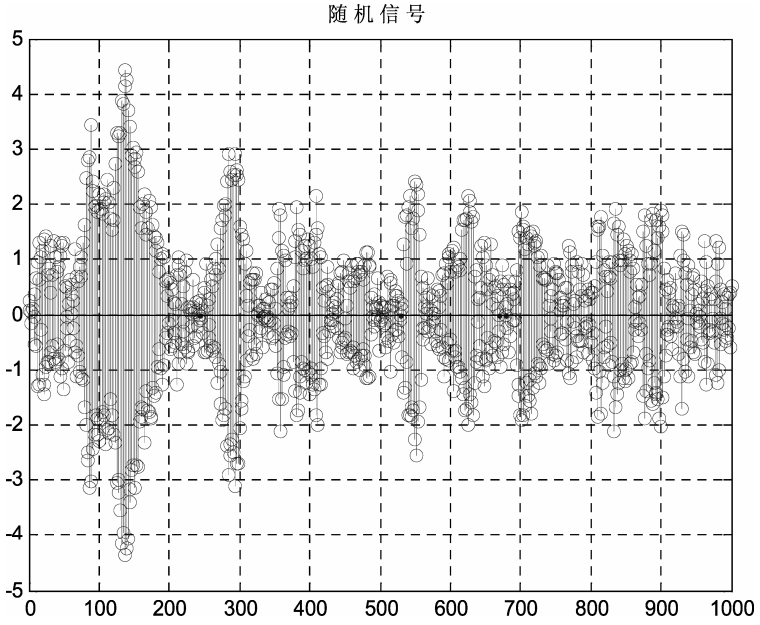

图 5-2　样本序列

重复 100 次，产生样本经由 FIR 滤波器，用 RLS 算法实现$u(n)$的线性预测，RLS 算法迭代运算如下：

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

$$\xi(n)=d(n)-\hat{w}^{H}(n-1)u(n)$$

$$\hat{w}(n)=\hat{w}(n-1)+k(n)\xi^{*}(n)$$

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

其中，$k(n)$为增益矩阵，$\hat{w}(n)$为权向量，$P(n)$为相关矩阵的逆矩阵，$\xi(n)$为先验估计误差。

由此编程如下：

```
clc,clear,close all                 % 清屏、清工作区、关闭窗口
warning off                         % 消除警告
feature jit off                     % 加速代码执行
N = 1000;                           % 信号观测长度
a1 = 0.99;                          % 一阶 AR 参数
sigma = 0.0731;                     % 加性白噪声方差
for kk =1:100
   v = sqrt(sigma)*randn(N,1);      % 产生 v（n）加性白噪声
   u0 = [0];                        % 初始数据
   num = 1;                         % 分子系数
   den = [1,a1];                    % 分母系数
   Zi = filtic(num,den,u0);         % 滤波器的初始条件
   un = filter(num,den,v,Zi);       % 产生样本序列 u(n), N x 1 x trials
%    figure,stem(un),title('随机信号');grid on;
   % 产生期望响应信号和观测数据矩阵
   n0 = 1;                          % 虚实现 n0 步线性预测
   M = 2;                           % 滤波器阶数
   b = un(n0+1:N);                  % 预测的期望响应
   L = length(b);
   un1 = [zeros(M-1,1)',un'];       % 扩展数据
   A = zeros(M,L);
   for k=1:L
      A(:,k) = un1(M-1+k : -1 : k);% 构建观测数据矩阵
   end
   % 应用 RLS 算法进行迭代寻优计算最优权向量
   delta = 0.004;                   % 调整参数
   lamda = 0.98;                    % 遗忘因子
   w = zeros(M,L+1);
   epsilon = zeros(L,1);
   P1 = eye(M)/delta;
   % RLS 迭代算法过程
   for k=1:L
      PIn = P1 * A(:,k);
      denok = lamda + A(:,k)'*PIn;
      kn = PIn/denok;
      epsilon(k) = b(k)-w(:,k)'*A(:,k);
      w(:,k+1) = w(:,k) + kn*conj(epsilon(k));
      P1 = P1/lamda - kn*A(:,k)'*P1/lamda;
   end
```

```
    w1(kk,:) = w(1,:);
    w2(kk,:) = w(2,:);
    MSE = abs(epsilon).^2;
    MSE_P(kk) = mean(MSE);
end
W1 = mean(w1);                    % 取平均值
W2 = mean(w2);                    % 取平均值
figure,plot(1:kk,MSE_P,'r','linewidth',2),title('平均MSE');grid on;
figure,plot(1:length(W1),W1,'r','linewidth',2),title('平均MSE');hold on;
plot(1:length(W2),W2,'b','linewidth',2),title('权值');hold on;
grid on;legend('\alpha1=0','\alpha2=-1')
```

运行程序输出图形如图 5-3 和图 5-4 所示。

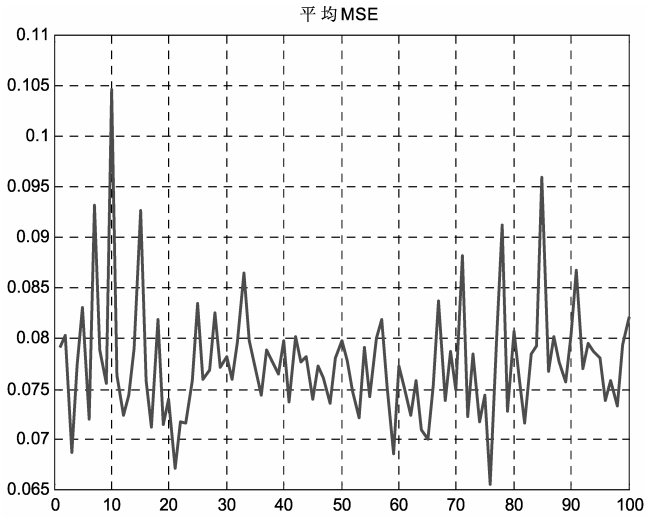

图 5-3　100 次迭代平均均方差曲线

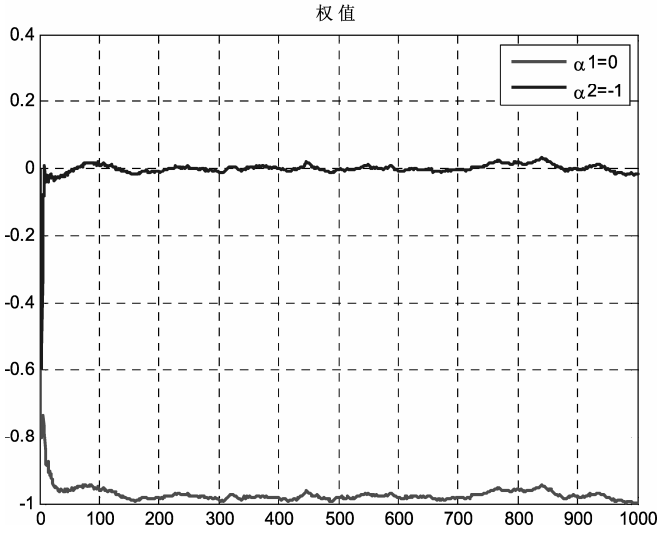

图 5-4　100 次迭代平均权值曲线

由图 5-3 可知，误差比较小，所求权值实际值为 0 和-1，和图 5-4 中曲线非常逼近，因此该算法具有一定的可行性。

5.4　本 章 小 结

自适应信号处理是近 40 年发展起来的信号处理领域的一个新的分支，自适应滤波是统计平稳信号处理和非平稳随机信号处理的主要内容，它具有维纳滤波和卡尔曼滤波的最佳滤波性能，而且不需要先验知识的初始条件。本章主要讲解了基于递归最小二乘（RLS）算法进行数据预测，递归最小二乘（RLS）算法表现收敛速度快，但是计算量比较大等特点。