第5卷第2期 控制理论与应用 Vol.5, No.2
1988年6月 CONTROL THEORY AND APPLICATIONS June, 1988

一种可以同时辨识模型参数和阶次的辅助变量法

肖德云　　牛绍华

（清华大学自动化系，北京）

摘　要

在用辅助变量法对SISO线性模型进行辨识时，如果将其数据向量和辅助向量重新排列并扩维，利用由此而出现的“移位性质”，则可构成一种新的数据乘积矩阵，其逆矩阵称作信息压缩阵。本文提出一种新的 UDV^{τ} 分解方法，并利用它对这个信息压缩阵进行递推分解，构成了一种能同时进行模型参数和阶次辨识的递推算法。使用这种新的辨识算法，可以减小整个辨识过程的计算量，改善数值计算品质，提高辨识精度。

一、引　言

对于SISO线性差分方程模型来说，当系统噪声为有色噪声时，为得到模型参数的无偏估计，可采用辅助变量法[2]。但是，如果系统的阶次也需要辨识的话，就必须采用阶次搜索的方法，逐一提高模型的阶次，在不同的模型阶次下估计出模型的参数和对应的损失函数值，再利用这些损失函数值，根据F检验或AIC准则等判阶方法来确定模型的阶次。显然，每改变一次阶次都要进行一次参数估计，这就使得整个辨识过程的计算量很大，模型阶次较高时尤为如此。

如果对辅助变量法中的数据向量和辅助向量扩维和重新排列，并利用由此而出现的“移位性质”（Shift Structure），便可以构成一种新的数据乘积矩阵，其逆矩阵称作辅助变量信息压缩阵（Condensed Information Matrix）。它是对文献〔1〕提出的信息压缩阵的扩展。这种信息压缩阵同样包含了模型各阶参数估计值和损失函数（广义）的全部信息。

本文针对辅助变量法中数据乘积矩阵不对称的特点，对传统的UD分解方法[4]进行修改和扩充，提出一种递推的 UDV^{τ} 分解方法，利用它对辅助变量信息压缩阵进行递推分解，构成一种新的递推辨识方法。使用这种算法可在每步递推计算过程中同时获得从1阶到 n 阶（n 是实际过程最大的可能阶次）模型的所有参数估计值和对应的广义损失函数值。根据所得到的各阶广义损失函数值，利用判阶方法可方便地确定模型的阶次。这就避免了传统的阶次搜索过程，从而减小了整个辨识过程的计算量。同时，由

本文于1987年5月15日收到。1987年10月16日收到修改稿。

于采用了 UDV^{τ} 分解技术，该算法还具有较好的数值计算品质，可以获得较高的辨识精度。

二、信息压缩阵

文献[1]提出了信息压缩阵的思想，但它仅适合于数据乘积矩矩阵是对称的情况。本文以辅助变量法为背景，对数据乘积矩矩阵不对称的情况重新构造了信息压缩阵。

设系统以如下的SISO差分方程模型表示

$$z(k)+a_1z(k-1)+\cdots+a_nz(k-n)=b_1u(k-1)+\cdots+b_nu(k-n)+e(k), \qquad (1)$$

其中，$u(k)$和$z(k)$为模型的输入输出变量；a_i和b_i（$i=1,2,\cdots,n$）为模型参数；$e(k)$为零均值随机噪声，它可为白噪声，也可为有色噪声。

定义数据向量和参数向量

$$\begin{cases} h_n(k)=[-z(k-n),u(k-n),\cdots,-z(k-1),u(k-1)]^{\tau}, \\ \theta_n=[a_n,b_n,\cdots,a_1,b_1]^{\tau}, \end{cases} \qquad (2)$$

则（1）式可以写成如下的最小二乘格式

$$z(k)=h_n^{\tau}(k)\theta_n+e(k). \qquad (3)$$

如果在数据向量 $h_n(k)$中加入当前的数据 $z(k)$，即定义

$$\varphi_n(k)=[-z(k-n),u(k-n),\cdots,-z(k-1),u(k-1),-z(k)]^{\tau}, \qquad (4)$$

则有

$$\varphi_n(k)=\begin{bmatrix} h_n(k) \\ -z(k) \end{bmatrix}, \qquad (5)$$

$$h_n(k)=\begin{bmatrix} \varphi_{n-1}(k-1) \\ u(k-1) \end{bmatrix}. \qquad (6)$$

（5）和（6）式体现了数据向量的“移位性质”[2]、[3]。

取辅助向量为

$$h_n^*(k)=[-x(k-n),u(k-n),\cdots,-x(k-1),u(k-1)]^{\tau}, \qquad (7)$$

其中

$$x(k)=h_n^{*\tau}(k)\hat{\theta}_n(k), \qquad (8)$$

$\hat{\theta}_n(k)$为模型参数 θ 的辅助变量估计值。若在辅助向量 $h_n^*(k)$中加入当前的辅助变量值 $x(k)$，即定义

$$\varphi_n^*(k)=[-x(k-n),u(k-n),\cdots,-x(k-1),u(k-1),-x(k)]^{\tau}, \qquad (9)$$

则同样可以得到如下的数据“移位性质”

$$\varphi_n^*(k)=\begin{bmatrix} h_n^*(k) \\ -x(k) \end{bmatrix}, \tag{10}$$

$$h_n^*(k)=\begin{bmatrix} \varphi_{n-1}^*(k-1) \\ u(k-1) \end{bmatrix}. \tag{11}$$

定义下面的数据乘积矩矩阵

$$S_n(k)=\sum_{j=1}^{k}\varphi_n^*(j)\varphi_n^\tau(j),\quad S_{n-1}(k-1)=\sum_{j=0}^{k-1}\varphi_{n-1}^*(j)\varphi_{n-1}^\tau(j), \tag{12}$$

$$R_n(k)=\sum_{j=1}^{k}h_n^*(j)h_n^\tau(j),\quad R_{n-1}(k-1)=\sum_{j=0}^{k-1}h_{n-1}^*(j)h_{n-1}^\tau(j). \tag{13}$$

利用（5）和（10）式的数据“移位性质”，对（12）式进行分解，可以得到

$$S_n(k)=\begin{bmatrix} R_n(k) & -\sum_{j=1}^{k}h_n^*(j)z(j) \\ -\sum_{j=1}^{k}h_n^\tau(j)x(j) & \sum_{j=1}^{k}x(j)z(j) \end{bmatrix}$$

$$=\begin{bmatrix} I_{2n} & 0 \\ -\hat{\theta}_n^{*\tau}(k) & 1 \end{bmatrix}\begin{bmatrix} R_n(k) & 0 \\ 0 & J_n(k) \end{bmatrix}\begin{bmatrix} I_{2n} & 0 \\ -\hat{\theta}_n^\tau(k) & 1 \end{bmatrix}^\tau, \tag{14}$$

其中

$$\begin{cases} \hat{\theta}_n(k)=R_n^{-1}(k)\sum_{j=1}^{k}h_n^*(j)z(j), \\ \hat{\theta}_n^*(k)=[R_n^{-1}(k)]^\tau\sum_{j=1}^{k}h_n(j)x(j), \\ J_n(k)=\sum_{j=1}^{k}x(j)z(j)-\hat{\theta}_n^{*\tau}(k)R_n(k)\hat{\theta}_n(k). \end{cases} \tag{15}$$

明显看出，$\hat{\theta}_n(k)$恰好是模型（3）参数θ_n的辅助变量估计值；$\hat{\theta}_n^*(k)$是辅助模型（8）的参数的辅助变量估计值；$J_n(k)$称作广义损失函数，定义为

$$J_n(k)=\sum_{j=1}^{k}\tilde{z}(j)\tilde{x}(j)=\sum_{j=1}^{k}[z(j)-\hat{z}(j)][x(j)-\hat{x}(j)]$$

$$=\sum_{j=1}^{k}\left\{[z(j)-h_n^\tau(j)\hat{\theta}_n(k)]h_n^{*\tau}(j)[\hat{\theta}_n(k)-\hat{\theta}_n^*(k)]\right\}. \tag{16}$$

同样，利用（6）和（11）式的数据“移位性质”，对（13）式进行分解，可以得到

$$R_n(k)=\begin{bmatrix} I_{2n-1} & 0 \\ \hat{\theta}^{*\tau}_{(n-1)u}(k-1) & 1 \end{bmatrix}\begin{bmatrix} S_{n-1}(k-1) & 0 \\ 0 & J_{(n-1)u}(k-1) \end{bmatrix}$$

$$\cdot\begin{bmatrix} I_{2n-1} & 0 \\ \hat{\theta}^{\tau}_{(n-1)u}(k-1) & 1 \end{bmatrix}^{\tau}, \tag{17}$$

其中

$$\begin{cases} \hat{\theta}_{(n-1)u}(k-1)=S^{-1}_{n-1}(k-1)\sum\limits_{j=0}^{k-1}\varphi^{*}_{n-1}(j)u(j), \\ \hat{\theta}^{*}_{(n-1)u}(k-1)=[S^{-1}_{n-1}(k-1)]^{\tau}\sum\limits_{j=0}^{k-1}\varphi_{n-1}(j)u(j), \\ J_{(n-1)u}(k-1)=\sum\limits_{j=0}^{k-1}u^2(j)-\hat{\theta}^{*\tau}_{(n-1)u}(k-1)S_{n-1}(k-1)\hat{\theta}_{(n-1)u}(k-1). \end{cases} \tag{18}$$

这里，$\hat{\theta}_{(n-1)u}(k-1)$、$\hat{\theta}^{*}_{(n-1)u}(k-1)$和$J_{(n-1)u}(k-1)$并没有明确的物理意义。

将（14）和（17）式的递推关系不断重复下去，并设$C_n(k)=S_n^{-1}(k)$（$C_n(k)$称为信息压缩阵），根据$C_n(k)$的性质，可以得到

$$C_n(k)=U_n(k)D_n(k)V_n^{\tau}(k), \tag{19}$$

其中

$$U_n(k)=$$

$$\left[\begin{array}{c|c|c|c|c|c|c} 1 & \hat{\theta}_{0u}(k-n) & & & & & \\ & 1 & \hat{\theta}_1(k-n+1) & & & & \\ & & 1 & \hat{\theta}_{1u}(k-n+1) & & & \\ & & & 1 & \ddots & & \\ & & & & & \hat{\theta}_{(n-1)u}(k-1) & \\ & \text{\Large 0} & & & & 1 & \hat{\theta}_n(k) \\ & & & & & & 1 \end{array}\right], \tag{20}$$

$$V_n(k)=\begin{pmatrix} 1 & \hat{\theta}^*_{0u}(k-n) & & & & & \\ & 1 & \hat{\theta}^*_{1}(k-n+1) & & & & \\ & & 1 & \hat{\theta}^*_{1u}(k-n+1) & & & \\ & & & 1 & \ddots & & \\ & & & & \ddots & \hat{\theta}^*_{(n-1)u}(k-1) & \\ & 0 & & & & 1 & \hat{\theta}_n(k) \\ & & & & & & 1 \end{pmatrix},$$

(21)

$$D_n(k)=\begin{pmatrix} J_0^{-1}(k-n) & & & & & \\ & J_{0u}^{-1}(k-n) & & & 0 & \\ & & J_1^{-1}(k-n+1) & & & \\ & & & \ddots & & \\ & 0 & & & J_{(n-1)u}^{-1}(k-1) & \\ & & & & & J_n^{-1}(k) \end{pmatrix}.$$

(22)

上式就是辅助变量信息压缩阵的 UDV^{τ} 分解形式。由于 $C_n(k)$ 的各阶顺序主子式均大于零，由〔5〕可知该分解是唯一的。可以看出，$U_n(k)$ 阵中包含了 1 到 n 阶全部的辅助变量法参数估计值，$D_n(k)$ 阵中包含了各阶对应的广义损失函数，$V_r(k)$ 阵中则是辅助模型（8）的参数估计值，当模型的阶次高于系统的真实阶次时，广义损失函数将明显下降。由此可以方便地确定模型的阶次。同时，$D_n(k)$ 阵中的元素还可以用来监视信息压缩阵 $C_n(k)$ 的正定性，只要 $D_n(k)$ 阵中任一元素小于或等于零，则可断定信息压缩阵 $C_n(k)$ 已经失去了正定性。

三、递　推　UDV^{τ}　分　解

文献〔2〕、〔3〕给出数据乘积矩矩阵是对称的递推 UD 分解算法，但它不适用于辅

助变量法，因为辅助变量法的数据乘积矩矩阵是不对称的。然而仿对称的UD分解算法的推导，经过适当的修改，可以推出一种不对称的UD分解算法，记作UDV$^\tau$分解。利用这种新的UD分解算法，对信息压缩阵$C_n(k)$进行递推分解，便可得到一种能同时获得模型参数和阶次的递推辨识算法。

注意到（12）式的定义，有

$$S_n(k)=\sum_{j=1}^{k}\varphi_n^*(j)\,\varphi_n^\tau(j)=S_n(k-1)+\varphi_n^*(k)\,\varphi_n^\tau(k), \tag{23}$$

对$S_n(k)$求逆，并注意到$C_n(k)=S_n^{-1}(k)$，有

$$C_n(k)=\frac{1}{\lambda(k)}\left[C_n(k-1)-C_n(k-1)\varphi_n^*(k)\,\varphi_n^\tau(k)C_n(k-1)\beta^{-1}(k)\right], \tag{24}$$

其中，$\lambda(k)$为遗忘因子，且

$$\beta(k)=\lambda(k)+\varphi_n^\tau(k)C_n(k-1)\varphi_n^*(k)。 \tag{25}$$

设

$$\begin{cases} g_n(k)=D_n(k-1)f_n(k), & g_n^*(k)=D_n(k-1)f_n^*(k), \\ f_n(k)=U_n^\tau(k-1)\varphi_n(k), & f_n^*(k)=V_n^\tau(k-1)\,\varphi_n^*(k), \end{cases} \tag{26}$$

显然有

$$\beta_n(k)=\lambda(k)+f_n^\tau(k)g_n^*(k)=\lambda(k)+g^\tau(k)f_n^*(k). \tag{27}$$

根据（24）和（27）式，并设$C_n(k-1)=U_D(k-1)D_n(k-1)V_n^\tau(k-1)$，则

$$C_n(k)=U_n(k-1)\left[D_n(k-1)-\frac{g_n^*(k)g_n^\tau(k)}{\beta_n(k)}\right]V_n^\tau(k-1)/\lambda(k). \tag{28}$$

令

$$D_n(k-1)-\frac{g_n^*(k)g_n^\tau(k)}{\beta_n(k)}\triangleq\overline{U}_n(k)\overline{D}_n(k)\overline{V}_n^\tau(k), \tag{29}$$

则有

$$\begin{cases} U_n(k)=U_n(k-1)\overline{U}_n(k), \\ V_n(k)=V_n(k-1)\overline{V}_n(k), \\ D_n(k)=\overline{D}_n(k)/\lambda(k)。 \end{cases} \tag{30}$$

仿〔2〕、〔3〕中的推导，经过一系列初等正交变换，可以由$U_n(k-1)$、$D_n(k-1)$和

$V_n^\tau(k-1)$递推求得$U_n(k)$、$D_n(k)$和$V_n^\tau(k)$。最后得到的递推UDV^τ分解算法归纳为：

（1）$f_n(k)=U_n^\tau(k-1)\varphi_n(k)$，$g_n(k)=D_n(k-1)f_n(k)$，

$f_n^*(k)=V_n^\tau(k-1)\varphi_n^*(k)$，$g_n^*(k)=D_n(k-1)f_n^*(k)$。

（2）令$\beta_0(k)=\lambda(k)$，从$j=1$到$N\triangleq 2n+1$，计算（3）～（5）。

（3）$\beta_j(k)=\beta_{j-1}(k)+f_j(k)g_j^*(k)$，

$d_{jj}(k)=\beta_{j-1}(k)d_{jj}(k-1)/\beta_j(k)\lambda(k)$，

$\mu_j(k)=-f_j(k)/\beta_{j-1}(k)$，$\mu_j^*(k)=-f_j^*(k)/\beta_{j-1}(k)$，

$\nu_j(k)=g_j(k)$，$\nu_j^*(k)=g_j^*(k)$，

其中，$d_{jj}(k)$、$f_j(k)$和$g_j(k)$分别是矩阵$D_n(k)$、向量$f_n(k)$和$g_n(k)$对应的元素。

（4）从$i=1$到$j=1$，计算（5）。若$j=1$，返回（3）。

（5）$u_{ij}(k)=u_{ij}(k-1)+\nu_i^*(k)\mu_j(k)$，

$v_{ij}(k)=v_{ij}(k-1)+\nu_i(k)\mu_j^*(k)$，

$\nu_i(k):=\nu_i(k)+v_{ij}(k-1)\nu_j(k)$，

$\nu_i^*(k):=\nu_i^*(k)+u_{ij}(k-1)\nu_j^*(k)$，

其中，$u_{ij}(k)$和$v_{ij}(k)$分别是矩阵$U_n(k)$和$V_n(k)$对应的元素。

至此，我们已经得到了一种基于对信息压缩阵进行UDV^τ分解的新的递推辨识算法。它是一种能同时辨识模型阶次和参数的有效方法。下面以一个仿真例子来验证这一算法。

四、仿 真 结 果

考虑如下SISO差分方程模型

$$\begin{cases} z(k)-z(k-1)+0.18z(k-2)-0.784z(k-3)+0.656z(k-4) \\ \quad =0.115u(k-1)+0.115u(k-3)+e(k), \\ e(k)-1.2e(k-1)+0.7e(k-2)+0.1e(k-3) \\ \quad =v(k)+0.96v(k-1)-0.48v(k-2)+0.3v(k-3), \end{cases}$$

其中，$u(k)$和$z(k)$分别为模型的输入输出变量；$v(k)$为零均值白噪声；输入$u(k)$采用幅值为1.0的5阶PRBS信号；数据长度取1000；遗忘因子取常数1.0。利用本文提出的算法进行辨识，可以得到1至5阶的模型参数估计值，如表1所示。

图1是广义损失函数与模型阶次在不同噪信比下的关系曲线。可以看出，当模型的

表 1　模型参数辨识结果（N/S＝22.8%）

n	$\hat{a}_1$	$\hat{b}_1$	$\hat{a}_2$	$\hat{b}_2$	$\hat{a}_3$	$\hat{b}_3$	$\hat{a}_4$	$\hat{b}_4$	$\hat{a}_5$	$\hat{b}_5$
5	－1.4730	0.1138	0.6578	－0.0547	－0.8728	0.1160	1.0286	－0.0541	－0.3136	0.0014
4	－0.9989	0.1130	0.1795	－0.0012	－0.7806	0.1147	0.6528	0.0003		
3	－1.2764	0.1162	0.3619	－0.0313	－0.7789	0.1098				
2	－1.2514	0.1105	0.2766	－0.0347						
1	－0.9936	0.1139								
真值	－1.0	0.115	0.18	0.0	－0.784	0.115	0.656	0.0	0.0	0.0

阶次估计值等于真实阶次时，广义损失函数会有一个显著的下降。依此可方便地确定模型的阶次。显然，本例的模型阶次可定为4。

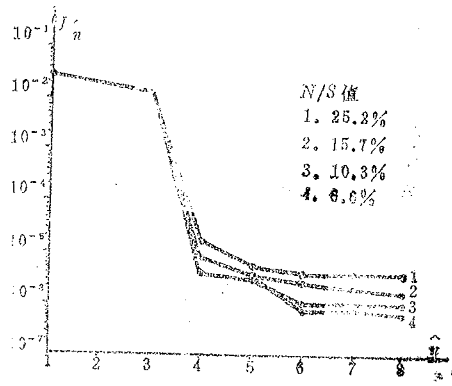

图 1　广义损失函数与阶次的关系

五、结　束　语

本文基于信息压缩阵思想，利用递推 UDV^{τ} 分解，导出一种新的可以同时辨识模型参数和阶次的递推算法。它和原先的辅助变量法比较具有明显的优点，不仅可降低整个辨识过程的计算量，而且改善了数值计算品质，提高了辨识精度。它尤其适用于模型结构未知的过程辨识问题。仿真结果证实这种新的算法是成功的。

参　考　文　献

〔1〕 陈伯成，最小二乘类离线辨识算法的探讨，清华大学自动化系硕士学位论文，（1984）。

〔2〕 方崇智、肖德云，过程辨识，清华大学出版社，北京，（1988）。

〔3〕 Ljung, L. and T. Söderström, Theory and Practice of Recursive Identification, MIT Press, (1983).
〔4〕 Bierman, G. J., Factorization Methods for Discrete Sequential Estimation, Academic Press, (1977).
〔5〕 李庆扬、王能超、易大义，数值分析，华中工学院出版社，武汉，(1982).

An Implementation of the Instrumental Variable Method for Simultaneous Identification of Model Order and Parameters

Xiao Deyun, Niu Shaohua

(Department of Automation, Tsinghua University, Bijing)

Abstract

The paper has presented an implementation of the instrumental variable method which can be used in simultaneous identification of the model order and parameters. By extension and rearrangement of the data vectors in the IV algorithm, a covariance matrix called the Condensed Information Matrix (CIM), which is nonsymmetric, can be constructed. Applying the UDV^{τ} factorization technique to the matrix, the model order and parameters can be obtained simultaneously, This identification method is superior to the conventinal IV method when the model order is high, Our algorithm has better in accuracy and requires less computational efforts.