ID6
ARMAX Identification

6.20 PEM IDENTIFICATION OF MULTIVARIABLE ARMAX MODELS

The procedure outlined in ID6.18 to estimate the parameters of a multivariable ARMAX model requires, as in the MISO case, two steps and it is difficult to evaluate how the errors in the estimate of $Q(z)$ and $P(z)$ influence the estimate of $R(z)$. Moreover this procedure is not based on the minimization of a cost function but relies on the asymptotic unbiasedness of IV estimates. This procedure is however useful to obtain an estimate of the process that can be refined by minimizing the prediction error of predictor (3.18.6). To implement a PEM procedure it is necessary to define a scalar cost function; denote, to this purpose, with $\hat{y}(t,\theta)$ the prediction associated with the parameterization θ and with $\varepsilon(t,\theta)$ the associated prediction error

$$\varepsilon(t,\theta) = y(t) - \hat{y}(t,\theta). \tag{6.20.1}$$

The sample covariance matrix of the errors (6.20.1) is

$$R_e(\theta) = \frac{1}{L-\nu_M}\sum_{t=\nu_M+1}^{L}\varepsilon(t,\theta)\,\varepsilon^T(t,\theta); \tag{6.20.2}$$

prediction error methods compute the parameters that minimize a suitable scalar function, $V(R_e(\theta))$, of the positive definite matrix (6.20.2). The choice $V(R_e(\theta)) = \det R_e(\theta)$ corresponds to the Maximum Likelihood method if the process belongs to the model set and $w(t)$ is Gaussian. A more appealing choice from the computational standpoint is

$$V\big(R_e(\theta)\big) = \mathrm{tr}\,\big(W\,R_e(\theta)\big) \tag{6.20.3}$$

where W is a positive definite weighting matrix. A nice choice for W is $W = [\mathrm{cov}\,w(t)]^{-1}$ but the covariance matrix of the noise is, in general, not known. It is however possible to approximate the process with an high–order auxiliary ARX model and $W^{-1} = \mathrm{cov}\,w(t)$ with the sample covariance matrix of its residuals. Assuming

the elements of $w(t)$ as mutually independent, the loss function (6.20.3) corresponds to

$$V\big(R_e(\theta)\big) = \text{tr } R_e(\theta) \tag{6.20.4}$$

after scaling the output sequences by means of the inverses of the standard deviations of the residuals of the auxiliary ARX model. Assuming this operation as performed, the cost function to be minimized is, modulo the multiplication by a constant,

$$V\big(R_e(\theta)\big) = \sum_{t=\nu_M+1}^{L} \varepsilon^T(t,\theta)\,\varepsilon(t,\theta). \tag{6.20.5}$$

Performing a shift to rewrite (6.20.5) in the simpler form

$$V(\theta) = \sum_{t=1}^{N} \varepsilon^T(t,\theta)\,\varepsilon(t,\theta), \tag{6.20.6}$$

where $N = L - \nu_M$, and denoting by $\psi(t,\theta)$ the $(\ell \times m)$ matrix

$$\psi(t,\theta) = -\frac{\partial \varepsilon^T(t,\theta)}{\partial \theta}, \tag{6.20.7}$$

the Gauss–Newton algorithm consists in relation

$$\theta^{k+1} = \theta^k + \left[\sum_{t=1}^{N} \psi(t,\theta^k)\,\psi^T(t,\theta^k)\right]^{-1} \sum_{t=1}^{N} \psi(t,\theta^k)\,\varepsilon(t,\theta^k) \tag{6.20.8}$$

where θ^k denotes the estimate of θ after the k–th step. Introducing the $(Nm \times \ell)$ matrix

$$H_\psi(\theta^k) = \begin{bmatrix} \psi^T(1,\theta^k) \\ \vdots \\ \psi^T(N,\theta^k) \end{bmatrix} \tag{6.20.9}$$

and the vector

$$\varepsilon^\circ(\theta^k) = \begin{bmatrix} \varepsilon(1,\theta^k) \\ \vdots \\ \varepsilon(N,\theta^k) \end{bmatrix}, \tag{6.20.10}$$

it is possible to rewrite (6.20.8) in the form

$$\theta^{k+1} = \theta^k + \big[\, H_\psi^T(\theta^k)\, H_\psi(\theta^k) \,\big]^{-1} H_\psi^T(\theta^k)\,\varepsilon^\circ(\theta^k) \tag{6.20.11}$$

or in the usual variation

$$\theta^{k+1} = \theta^k + \Delta_k \big[\, H_\psi^T(\theta^k)\, H_\psi(\theta^k) \,\big]^{-1} H_\psi^T(\theta^k)\,\varepsilon^\circ(\theta^k) \tag{6.20.12}$$

where the scalar $\Delta_k \leq 1$ allows a better convergence control. The implementation of the algorithm requires thus the computation of $\psi(t,\theta)$; making reference to model (6.17.8), straightforward differentiation gives, assuming $\varepsilon(t) = w(t)$, i.e. a parametrization near to the true one,

$$\tilde{S}(z)^* \frac{\partial \varepsilon(t)}{\partial \alpha_{ijk}} = \frac{\partial \tilde{P}(z)^*}{\partial \alpha_{ijk}} \big(y(t) - \varepsilon(t)\big) = \frac{\partial \tilde{P}(z)^*}{\partial \alpha_{ijk}} \hat{y}(t) \tag{6.20.13}$$

where

$$\frac{\partial \tilde{P}(z)^*}{\partial \alpha_{ijk}} \hat{y}(t) = \begin{bmatrix} 0 \\ \vdots \\ -\hat{y}_j(t+k+\Delta\nu_i-1) \\ \vdots \\ 0 \end{bmatrix} \leftarrow i \tag{6.20.14}$$

so that

$$\frac{\partial \varepsilon(t)}{\partial \alpha_{ijk}} = -\hat{y}_{ij}^F(t+k+\Delta\nu_i-1) \tag{6.20.15}$$

where $\hat{y}_{ij}^F(t)$ denotes the output of the filter

$$\tilde{S}(z)^* \, \hat{y}_{ij}^F(t) = v(t) \tag{6.20.16}$$

driven by the input

$$v(t) = \begin{bmatrix} 0 \\ \vdots \\ \hat{y}_j(t) \\ \vdots \\ 0 \end{bmatrix} \leftarrow i. \tag{6.20.17}$$

Similar considerations show that

$$\frac{\partial \varepsilon(t)}{\partial \beta_{ijk}} = -u_{ij}^F(t+k+\Delta\nu_i-1) \tag{6.20.18}$$

where $u_{ij}^F(t)$ denotes the output of the filter

$$\tilde{S}(z)^* \, u_{ij}^F(t) = v(t) \tag{6.20.19}$$

driven by the input

$$v(t) = \begin{bmatrix} 0 \\ \vdots \\ u_j(t) \\ \vdots \\ 0 \end{bmatrix} \leftarrow i. \tag{6.20.20}$$

Finally

$$\frac{\partial \varepsilon(t)}{\partial \gamma_{ijk}} = -\varepsilon_{ij}^F(t + k + \Delta\nu_i - 1) \tag{6.20.21}$$

where $\varepsilon_{ij}^F(t)$ denotes the output of the filter

$$\tilde{S}(z)^* \, \varepsilon_{ij}^F(t) = v(t) \tag{6.20.22}$$

driven by the input

$$v(t) = \begin{bmatrix} 0 \\ \vdots \\ \varepsilon_j(t) \\ \vdots \\ 0 \end{bmatrix} \leftarrow i. \tag{6.20.23}$$

The requested result is thus

$$\psi(t,\theta) = \tag{6.20.24}$$

$$\begin{bmatrix} y_{11}^F(t+\Delta\nu_1) \ \ldots \ y_{11}^F(t+\nu_1+\Delta\nu_1-1) \mid \ldots \mid y_{1m}^F(t+\Delta\nu_1) \ \ldots \ y_{1m}^F(t+\nu_{1m}+\Delta\nu_1-1) \mid \\ \ldots \\ \mid y_{m1}^F(t+\Delta\nu_m) \ \ldots \ y_{m1}^F(t+\nu_{m1}+\Delta\nu_m-1) \mid \ldots \mid y_{mm}^F(t+\Delta\nu_m) \ \ldots \\ \qquad \ldots \ y_{mm}^F(t+\nu_m+\Delta\nu_m-1) \mid \\ \mid u_{11}^F(t+\Delta\nu_1) \ \ldots \ u_{11}^F(t+\nu_1+\Delta\nu_1-1) \mid \ldots \mid u_{1p}^F(t+\Delta\nu_1) \ \ldots \ u_{1p}^F(t+\nu_1+\Delta\nu_1-1) \mid \\ \ldots \\ \mid u_{m1}^F(t+\Delta\nu_m) \ \ldots \ u_{m1}^F(t+\nu_m+\Delta\nu_m-1) \mid \ldots \mid u_{mp}^F(t+\Delta\nu_m) \ \ldots \\ \qquad \ldots \ u_{mp}^F(t+\nu_m+\Delta\nu_m-1) \mid \\ \mid \varepsilon_{11}^F(t+\Delta\nu_1) \ \ldots \ \varepsilon_{11}^F(t+\nu_1+\Delta\nu_1-1) \mid \ldots \mid \varepsilon_{1m}^F(t+\Delta\nu_1) \ \ldots \ \varepsilon_{1m}^F(t+\nu_1+\Delta\nu_1-1) \mid \\ \ldots \\ \mid \varepsilon_{m1}^F(t+\Delta\nu_m) \ \ldots \ \varepsilon_{m1}^F(t+\nu_m+\Delta\nu_m-1) \mid \ldots \mid \varepsilon_{mm}^F(t+\Delta\nu_m) \ \ldots \\ \qquad \ldots \ \varepsilon_{mm}^F(t+\nu_m+\Delta\nu_m-1) \end{bmatrix}^T.$$

Remark 6.20.1 – The computation of (6.20.24) requires $2m^2 + mp$ filterings using filters (6.20.16), (6.20.19) and (6.20.22) with inputs (6.20.17), (6.20.20) and (6.20.23) obtained from the entries of $\hat{y}(t)$, $u(t)$ and $\varepsilon(t)$.

Remark 6.20.2 – The PEM procedure that has been described does not contain any explicit stability constraint on the models. When the initial model is stable, loss function (6.20.6) assures also the stability of the final model. Stability violations can however

occasionally occur for different reasons; it is possible to recover from these situations returning to the last stable model and using a lower value of Δ_k in (6.20.12). Note that the stability of the model concerns not only $\tilde{P}(z)$ but also $\tilde{S}(z)$ which describes the filters (6.20.16), (6.20.19) and (6.20.22).

SECTIONS MODULES QUESTIONS HOME PAGE
PREV. MODULE FAQ TUTOR NEXT MODULE