Least squares identification

Improved algorithm

Xing Chao

Institute of Astronautics, Northwestern Polytechnical University

Contents

- The identification accuracy is higher than the basic least squares estimation method；
- computate easily；
- Asymptotically unbiased estimate ；
- Construction of auxiliary variable matrix。

Principle of auxiliary variable method

$$
\begin{aligned}
\mathrm{Y} &= \Phi\theta + \xi \\
\Phi^{\mathrm{T}}\mathrm{Y} &= \Phi^{\mathrm{T}}\Phi\theta + \Phi^{\mathrm{T}}\xi \\
(\Phi^{\mathrm{T}}\Phi)^{-1}\Phi^{\mathrm{T}}\mathrm{Y} &= (\Phi^{\mathrm{T}}\Phi)^{-1}\Phi^{\mathrm{T}}\Phi\theta + (\Phi^{\mathrm{T}}\Phi)^{-1}\Phi^{\mathrm{T}}\xi \\
(\Phi^{\mathrm{T}}\Phi)^{-1}\Phi^{\mathrm{T}}\mathrm{Y} &= \theta + (\Phi^{\mathrm{T}}\Phi)^{-1}\Phi^{\mathrm{T}}\xi
\end{aligned}
$$

Principle of auxiliary variable method

$$
\begin{aligned}
\mathrm{Y} &= \Phi\theta + \xi \\
\mathrm{Z}^{\mathrm{T}}\mathrm{Y} &= \mathrm{Z}^{\mathrm{T}}\Phi\theta + \mathrm{Z}^{\mathrm{T}}\xi \\
(\mathrm{Z}^{\mathrm{T}}\Phi)^{-1}\Phi^{\mathrm{T}}\mathrm{Y} &= (\mathrm{Z}^{\mathrm{T}}\Phi)^{-1}\Phi^{\mathrm{T}}\Phi\theta + (\mathrm{Z}^{\mathrm{T}}\Phi)^{-1}\mathrm{Z}^{\mathrm{T}}\xi \\
(\mathrm{Z}^{\mathrm{T}}\Phi)^{-1}\Phi^{\mathrm{T}}\mathrm{Y} &= \theta + (\mathrm{Z}^{\mathrm{T}}\Phi)^{-1}\mathrm{Z}^{\mathrm{T}}\xi
\end{aligned}
$$

where：

$$
\begin{aligned}
\mathrm{E}(\mathrm{Z}^{\mathrm{T}}\xi) &= 0 \\
\mathrm{E}(\mathrm{Z}^{\mathrm{T}}\Phi) &= \mathrm{Q}
\end{aligned}
$$

where Q Nonsingular 。

Asymptotically unbiased

$$
\begin{aligned}
\mathrm{E}[\hat{\theta}_{\mathrm{IV}}] &= \mathrm{E}[(\mathrm{Z}^{\mathrm{T}}\Phi)^{-1}\mathrm{Z}^{\mathrm{T}}\mathrm{Y}] \\
&= \mathrm{E}[(\mathrm{Z}^{\mathrm{T}}\Phi)^{-1}\mathrm{Z}^{\mathrm{T}}(\Phi\theta+\xi)] \\
&= \theta + \mathrm{E}[(\mathrm{Z}^{\mathrm{T}}\Phi)^{-1}\mathrm{Z}^{\mathrm{T}}\xi] \\
\lim_{\mathrm{N}\to\infty}\mathrm{E}[\hat{\theta}_{\mathrm{IV}}] &= \theta + \mathrm{E}[(\mathrm{Z}^{\mathrm{T}}\Phi)^{-1}]\mathrm{E}[\mathrm{Z}^{\mathrm{T}}\xi] \\
&= \theta
\end{aligned}
$$

The construction method of auxiliary variable method

- Recursive auxiliary variable parameter estimation method
- Adaptive filtering method
- Pure lag method
- Taly principle method

Recursive auxiliary variable parameter estimation method:Z

$$
\begin{aligned}
\hat{Y} &= Z\hat{\theta} \\
Z &= \begin{bmatrix} -\hat{y}_n & \cdots & -\hat{y}_1 & u_{n+1} & \cdots & u_1 \\ \vdots & & \vdots & \vdots & & \vdots \\ -\hat{y}_{n+N-1} & \cdots & -\hat{y}_N & u_{n+N} & \cdots & u_N \end{bmatrix}
\end{aligned}
$$

Recursive auxiliary variable parameter estimation method:process

- initialize：use basic least squares method to estimate $\hat{\theta}$, let $\mathrm{Z} = \Phi$,
- Recurse：
  - update Z
    $$\hat{\mathrm{Y}} = \mathrm{Z}\hat{\theta}$$
  - compute $\hat{\theta}$
    $$\hat{\theta} = (\mathrm{Z}^{\mathrm{T}}\Phi)^{-1}\mathrm{Z}^{\mathrm{T}}\mathrm{Y}$$
  - iterate until $\hat{\theta}$ converges。

Adaptive filtering method

On the basis of recursive auxiliary variable parameter estimation method ,let:

$$\hat{\theta}_{\mathrm{k}} = (1-\alpha)\hat{\theta}_{\mathrm{k}-1} + \alpha\hat{\theta}_{\mathrm{k}-\mathrm{d}}$$

α: $\in [0.01, 0.1]$

d: $\in [0, 10]$,

Pure lag method

$$\hat{y}_k = u_{k-m}$$

When $d = n$，there is：

$$Z = \begin{bmatrix} -u_0 & \cdots & -u_{1-n} & u_{n+1} & \cdots & u_1 \\ -u_1 & \cdots & -u_{2-n} & u_{n+2} & \cdots & u_2 \\ \vdots & & \vdots & \vdots & \vdots & \\ -u_{N-1} & \cdots & -u_{N-n} & u_{n+N} & \cdots & u_2 \end{bmatrix}$$

Tally principle

If noise ξ_k is refered as output of this model :

$$\xi_k = c(z^{-1})n_k$$

where n_k is uncorrelated ramdom noise with zero mean。
and:

$$c(z^{-1}) = 1 + c_1 z^{-1} + \cdots + c_m z^{-m}$$

then, let:

$$\begin{aligned}
\hat{y}_k &= y_{k-m} \\
Z &= \begin{bmatrix} -y_{n-m} & \cdots & -y_{1-m} & u_{n+1} & \cdots & u_1 \\ -y_{n+1-m} & \cdots & -y_{2-m} & u_{n+2} & \cdots & u_2 \\ \vdots & & \vdots & \vdots & \vdots & \\ -y_{n+N-1-m} & \cdots & -y_{N-m} & u_{n+N} & \cdots & u_2 \end{bmatrix}
\end{aligned}$$

Recursive auxiliary variable method

$$
\begin{aligned}
\hat{\theta}_N &= P_N Z_N^T Y_N \\
P_N &= (Z_N^T \Phi_N)^{-1} \\
\hat{\theta}_{N+1} &= P_{N+1} Z_{N+1}^T Y_{N+1}
\end{aligned}
$$

Recursive auxiliary variable method

$$
\begin{aligned}
\hat{\theta}_N &= P_N Z_N^T Y_N \\
P_N &= (Z_N^T \Phi_N)^{-1} \\
\hat{\theta}_{N+1} &= P_{N+1} Z_{N+1}^T Y_{N+1} \\
P_{N+1} &= \left(\begin{bmatrix} Z_N^T & z_{N+1} \end{bmatrix} \begin{bmatrix} \Phi_N \\ \Psi_{N+1}^T \end{bmatrix} \right)^{-1} \\
&= (P_N^{-1} + z_{N+1} \Psi_{N+1}^T)^{-1} \\
\Psi_{N+1} &= \begin{bmatrix} -y_{n+N} & \cdots & -y_{N+1} & u_{n+N+1} & \cdots & u_{N+1} \end{bmatrix}^T \\
z_{N+1} &= \begin{bmatrix} -\hat{y}_{n+N} & \cdots & -\hat{y}_{N+1} & u_{n+N+1} & \cdots & u_{N+1} \end{bmatrix}^T
\end{aligned}
$$

Recursive auxiliary variable method

By using the inverse lemma of matrix, the recursive formula can be deduced：

$$\begin{aligned}\hat{\theta}_{\mathrm{N+1}} &= \hat{\theta}_{\mathrm{N}} + \mathrm{K}_{\mathrm{N+1}}(\mathrm{y}_{\mathrm{N+1}} - \psi^{\mathrm{T}}_{\mathrm{N+1}}\hat{\theta}_{\mathrm{N}}) \\ \mathrm{P}_{\mathrm{N+1}} &= \mathrm{P}_{\mathrm{N}} - \mathrm{K}_{\mathrm{N+1}}\Psi^{\mathrm{T}}_{\mathrm{N+1}}\mathrm{P}_{\mathrm{N}} \\ \mathrm{K}_{\mathrm{N+1}} &= \mathrm{P}_{\mathrm{N}}\mathrm{z}_{\mathrm{N+1}}(1 + \Psi^{\mathrm{T}}_{\mathrm{N+1}}\mathrm{P}_{\mathrm{N}}\mathrm{z}_{\mathrm{N+1}})^{-1}\end{aligned}$$

- Select initial parameters by reference to recursive least square method
- is sensitive to initial value P_0，it is better to use recursive least squares methods with first 50~100 point，then use auxiliary variable method.

Generalized least square method

- The filtering model is established to whiten the data
- The method is complex and with heavy computation
- The convergence of the iterative algorithm is not proved

Generalized least squares: system model

$$
\begin{aligned}
a(z^{-1})y_k &= b(z^{-1})u_k + \xi_k \\
f(z^{-1}) &= 1 + f_1 z^{-1} + \cdots + f_m z^{-m} \\
\xi_k &= \frac{1}{f(z^{-1})}\varepsilon_k \\
f(z^{-1})\xi_k &= \varepsilon_k \\
\xi_k &= -f_1\xi_{k-1} - \cdots - f_m\xi_{k-m} + \varepsilon_k
\end{aligned}
$$

Generalized least squares: system model

$$
\begin{aligned}
a(z^{-1})f(z^{-1})y_k &= b(z^{-1})f(z^{-1})u_k + \varepsilon_k \\
a(z^{-1})\bar{y}_k &= b(z^{-1})\bar{u}_k + \varepsilon_k \\
\bar{y}_k &= f(z^{-1})y_k \\
&= y_k + f_1 y_{k-1} + \cdots + f_m y_{k-m} \\
\bar{u}_k &= f(z^{-1})u_k \\
&= u_k + f_1 u_{k-1} + \cdots + f_m u_{k-m}
\end{aligned}
$$

Generalized least squares method: noise model parameter estimation

$$
\begin{aligned}
\xi &= \Omega \mathrm{f} + \varepsilon \\
\xi &= \begin{bmatrix} \xi_{n+1} & \xi_{n+2} & \cdots & \xi_{n+N} \end{bmatrix}^T \\
\mathrm{f} &= \begin{bmatrix} \mathrm{f}_1 & \mathrm{f}_2 & \cdots & \mathrm{f}_m \end{bmatrix}^T \\
\varepsilon &= \begin{bmatrix} \varepsilon_{n+1} & \varepsilon_{n+2} & \cdots & \varepsilon_{n+N} \end{bmatrix}^T \\
\Omega &= \begin{bmatrix} -\xi_n & \cdots & -\xi_{n+1-m} \\ -\xi_{n+1} & \cdots & -\xi_{n+2-m} \\ \vdots & & \vdots \\ -\xi_{n+N-1} & \cdots & -\xi_{n+N-m} \end{bmatrix} \\
\hat{\mathrm{f}} &= (\Omega^T \Omega)^{-1} \Omega^T \xi
\end{aligned}
$$

Generalized least squares: process

- initialize，let

$$\hat{f}(z^{-1}) = 1$$

- iterate
  - filtering：

$$\begin{aligned} \bar{y}_k &= \hat{f}(z^{-1})y_k \\ \bar{u}_k &= \hat{f}(z^{-1})u_k \end{aligned}$$

  - Least square estimation ：

$$\hat{\theta} = (\bar{\Phi}^T\bar{\Phi})^{-1}\bar{\Phi}^T\bar{Y}$$

  - residue：

$$\hat{\xi} = Y - \Phi\hat{\theta}$$

  - use residue $\hat{\xi}$ instead of ξ to compute $\hat{f}$：

$$\hat{f} = (\hat{\Omega}^T\hat{\Omega})^{-1}\hat{\Omega}^T e$$

Recursive generalized least squares method

- include resursive estimate of parameter $\hat{\theta}$ and noise model parameter $\hat{\mathrm{f}}$
- The results of offline and recursive calculation are not exactly the same
- process:
  - Initialization, and the initial value is selected by referring to recursive least square
  - filtering, compute new value of $\bar{\mathrm{y}}_{\mathrm{k}}, \bar{\mathrm{u}}_{\mathrm{k}}$
  - compute $\hat{\theta}$ and $\hat{\mathrm{f}}$ by using recursive least square algorithm

Recursive generalized least squares method

- initialize:

$$
\begin{aligned}
\hat{\theta}_0 &= 0 \\
\mathrm{P}_0^{(\theta)} &= \mathrm{c}_1^2\mathrm{I} \\
\hat{\mathrm{f}}_{(0)} &= 0 \\
\mathrm{P}_0^{(\mathrm{f})} &= \mathrm{c}_2^2\mathrm{I}
\end{aligned}
$$

- filtering

$$
\begin{aligned}
\bar{\mathrm{y}}_{\mathrm{N}+1} &= \hat{\mathrm{f}}_{(\mathrm{N})}(\mathrm{z}^{-1})\mathrm{y}_{\mathrm{N}+1} \\
&= \hat{\mathrm{f}}_{(\mathrm{N})}(\mathrm{z}^{-1})\mathrm{y}_{(\mathrm{n}+\mathrm{N}+1)} \\
\bar{\mathrm{u}}_{\mathrm{N}+1} &= \hat{\mathrm{f}}_{(\mathrm{N})}(\mathrm{z}^{-1})\mathrm{u}_{\mathrm{N}+1} \\
&= \hat{\mathrm{f}}_{(\mathrm{N})}(\mathrm{z}^{-1})\mathrm{u}_{(\mathrm{n}+\mathrm{N}+1)}
\end{aligned}
$$

Recursive generalized least squares method

- compute $\hat{\theta}$

$$
\begin{aligned}
\hat{\theta}_{N+1} &= \hat{\theta}_N + K_{N+1}^{(\theta)}(\bar{y}_{N+1} - \bar{\Psi}_{N+1}^T \hat{\theta}_N) \\
K_{N+1}^{(\theta)} &= P_N^{(\theta)} \bar{\Psi}_{N+1} (1 + \bar{\Psi}_{N+1}^T P_N^{(\theta)} \bar{\Psi}_{N+1})^{-1} \\
P_{N+1}^{(\theta)} &= P_N^{(\theta)} - K_{N+1}^{(\theta)} \bar{\Psi}_{N+1}^T P_N^{(\theta)} \\
\bar{\Psi}_{N+1} &= \begin{bmatrix} -\bar{y}_{n+N} & \cdots & -\bar{y}_{N+1} & \bar{u}_{n+N+1} & \cdots & \bar{u}_{N+1} \end{bmatrix}^T
\end{aligned}
$$

Recursive generalized least squares method

- compute residue $\hat{\xi}_{N+1}$

$$\hat{\xi}_{N+1} = y_{N+1} - \Psi_{N+1}\hat{\theta}_{N+1}$$

- compute $\hat{f}$

$$\begin{aligned}
\hat{f}_{N+1} &= \hat{f}_N + K^{(f)}_{N+1}(\hat{\xi}_{N+1} - \hat{\omega}^T_{N+1}\hat{f}_N) \\
K^{(f)}_{N+1} &= P^{(f)}_N \hat{\omega}_{N+1}(1 + \hat{\omega}^T_{N+1} P^{(f)}_N \hat{\omega}_{N+1})^{-1} \\
P^{(f)}_{N+1} &= P^{(f)}_N - K^{(f)}_{N+1}\hat{\omega}^T_{N+1} P^{(f)}_N \\
\hat{\omega}_{N+1} &= \begin{bmatrix} -\hat{\xi}_{n+N} & \cdots & -\hat{\xi}_{n+N+1-m} \end{bmatrix}
\end{aligned}$$

Hsia method

- Alternately solving system model and noise model parameters
- It can be divided into two types: Hsia correction method and Hsia improvement method
- Recursive algorithm can be extended to MIMO system
- There is no need to filter the data repeatedly, so the calculation efficiency is relatively high
- The estimation result is relatively good

Method: record system model

$$
\begin{aligned}
a(z^{-1})y(k) &= b(z^{-1})u(k) + \xi_k \\
\xi_k &= \frac{\varepsilon(k)}{f(z^{-1})} \\
a(z^{-1}) &= 1 + a_1 z^{-1} + \cdots + a_n z^{-n} \\
b(z^{-1}) &= b_0 + b_1 z^{-1} + \cdots + b_n z^{-n} \\
f(z^{-1}) &= 1 + f_1 z^{-1} + \cdots + f_m z^{-m}
\end{aligned}
$$

Method: record system model

$$
\begin{aligned}
a(z^{-1})y(k) &= b(z^{-1})u(k) + \xi_k \\
\xi_k &= \frac{\varepsilon(k)}{f(z^{-1})} \\
a(z^{-1}) &= 1 + a_1 z^{-1} + \cdots + a_n z^{-n} \\
b(z^{-1}) &= b_0 + b_1 z^{-1} + \cdots + b_n z^{-n} \\
f(z^{-1}) &= 1 + f_1 z^{-1} + \cdots + f_m z^{-m} \\
\xi_k &= (1 - f(z^{-1})\xi_k + \varepsilon_k \\
a(z^{-1})y(k) &= b(z^{-1})u(k) + (1 - f(z^{-1}))\xi_k + \varepsilon_k
\end{aligned}
$$

Method: the system model of vector xias

$$
\begin{aligned}
y_N &= y_{(n+N)} \\
&= \Psi_N^T \theta + \omega_N^T f + \varepsilon_N \\
f &= \begin{bmatrix} f_1 & \cdots & f_m \end{bmatrix}^T \\
\Psi_N &= \begin{bmatrix} -y_{(n+N-1)} & \cdots & -y_{(N)} & u_{(n+N)} & \cdots & u_{(N)} \end{bmatrix}^T \\
\omega_N &= \begin{bmatrix} -\xi_{(n+N-1)} & \cdots & -\xi_{(n+N-m)} \end{bmatrix}^T
\end{aligned}
$$

Method: parameters

$$
\begin{aligned}
\begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix} &= \begin{bmatrix} \Psi_1^T & \omega_1^T \\ \vdots & \vdots \\ \Psi_N^T & \omega_N^T \end{bmatrix} \begin{bmatrix} \theta \\ f \end{bmatrix} + \begin{bmatrix} \varepsilon_1 \\ \vdots \\ \varepsilon_N \end{bmatrix} \\
Y &= \begin{bmatrix} \Phi & \Omega \end{bmatrix} \begin{bmatrix} \theta \\ f \end{bmatrix} + \varepsilon \\
\begin{bmatrix} \hat{\theta} \\ \hat{f} \end{bmatrix} &= \begin{bmatrix} \Phi^T\Phi & \Phi^T\Omega \\ \Omega^T\Phi & \Omega^T\Omega \end{bmatrix}^{-1} \begin{bmatrix} \Phi^T Y \\ \Omega^T Y \end{bmatrix}
\end{aligned}
$$

Method: deviation correction method

inverse by ulsing block matrix：

$$
\begin{aligned}
\begin{bmatrix}\hat{\theta}\\ \hat{\mathrm{f}}\end{bmatrix} &= \begin{bmatrix}\Phi^{\mathrm{T}}\Phi & \Phi^{\mathrm{T}}\Omega\\ \Omega^{\mathrm{T}}\Phi & \Omega^{\mathrm{T}}\Omega\end{bmatrix}^{-1}\begin{bmatrix}\Phi^{\mathrm{T}}\mathrm{Y}\\ \Omega^{\mathrm{T}}\mathrm{Y}\end{bmatrix}\\
&= \begin{bmatrix}\mathrm{P_N}\Phi^{\mathrm{T}}\mathrm{Y}-\mathrm{P_N}\Phi^{\mathrm{T}}\Omega\mathrm{D}^{-1}\Omega^{\mathrm{T}}\mathrm{MY}\\ \mathrm{D}^{-1}\Omega^{\mathrm{T}}\mathrm{MY}\end{bmatrix}\\
&= \begin{bmatrix}\hat{\theta}_{\mathrm{LS}}-\mathrm{P_N}\Phi^{\mathrm{T}}\Omega\hat{\mathrm{f}}\\ \mathrm{D}^{-1}\Omega^{\mathrm{T}}\mathrm{MY}\end{bmatrix}\\
\mathrm{P_N} &= (\Phi^{\mathrm{T}}\Phi)^{-1}\\
\mathrm{M} &= \mathrm{I}-\Phi(\Phi^{\mathrm{T}}\Phi)^{-1}\Phi^{\mathrm{T}}\\
\mathrm{D} &= \Omega^{\mathrm{T}}\mathrm{M}\Omega
\end{aligned}
$$

Method: deviation correction method iteration steps

- Initialization: computing basic least squares estimation

$$\hat{\theta} = (\Phi^T\Phi)^{-1}\Phi^T Y$$

- iterate
    - compute residual $\hat{\xi}$ to construct $\hat{\Omega}$

$$\hat{\xi} = Y - \Phi\hat{\theta}$$

    - compute $\hat{f}$ to correct $\hat{\theta}$

$$\begin{aligned} \hat{f} &= D^{-1}\hat{\Omega}^T M Y \\ \hat{\theta} &= \hat{\theta} - (\Phi^T\Phi)^{-1}\Phi^T\hat{\Omega}\hat{f} \end{aligned}$$

Method: the modified method

use $\hat{\theta}$ instead of θ：

$$
\begin{aligned}
\mathrm{Y} &= \begin{bmatrix} \Phi & \Omega \end{bmatrix} \begin{bmatrix} \hat{\theta} \\ \mathrm{f} \end{bmatrix} + \varepsilon \\
&= \Phi\hat{\theta} + \Omega\mathrm{f} + \varepsilon \\
\mathrm{Y} - \Phi\hat{\theta} &= \Omega\mathrm{f} + \varepsilon
\end{aligned}
$$

obtained least squares estimate of f：

$$
\begin{aligned}
\hat{\mathrm{f}} &= (\hat{\Omega}^{\mathrm{T}}\hat{\Omega})^{-1}\hat{\Omega}^{\mathrm{T}}(\mathrm{Y} - \Phi\hat{\theta}) \\
\hat{\theta} &= \hat{\theta} - (\Phi^{\mathrm{T}}\Phi)^{-1}\Phi^{\mathrm{T}}\Omega\hat{\mathrm{f}}
\end{aligned}
$$

Method: the modified method iteration steps

- initialize: Computational basic least squares estimation

$$\hat{\theta} = (\Phi^{\mathrm{T}}\Phi)^{-1}\Phi^{\mathrm{T}}\mathrm{Y}$$

- iterate
  - compute residual $\hat{\xi}$ to construct $\hat{\Omega}$

$$\hat{\xi} = \mathrm{Y} - \Phi\hat{\theta}$$

  - compute $\hat{\mathrm{f}}$ to correct $\hat{\theta}$

$$\begin{aligned} \hat{\mathrm{f}} &= (\hat{\Omega}^{\mathrm{T}}\hat{\Omega})^{-1}\hat{\Omega}^{\mathrm{T}}(\mathrm{Y} - \Phi\hat{\theta}) \\ \hat{\theta} &= \hat{\theta} - (\Phi^{\mathrm{T}}\Phi)^{-1}\Phi^{\mathrm{T}}\hat{\Omega}\hat{\mathrm{f}} \end{aligned}$$

Method: the recursive method

$$
\begin{aligned}
\tilde{\Phi} &= \begin{bmatrix} \Phi & \hat{\Omega} \end{bmatrix} \\
\tilde{\theta} &= \begin{bmatrix} \hat{\theta} \\ \mathrm{f} \end{bmatrix} \\
\tilde{\theta}^{\mathrm{T}}_{\mathrm{N+1}} &= \tilde{\theta}_{\mathrm{N}} + \mathrm{K}_{\mathrm{N+1}}(\mathrm{y}_{\mathrm{N+1}} - \tilde{\Psi}^{\mathrm{T}}_{\mathrm{N+1}}\tilde{\theta}_{\mathrm{N}}) \\
\mathrm{P}_{\mathrm{N+1}} &= \mathrm{P}_{\mathrm{N}} - \mathrm{K}_{\mathrm{N+1}}\tilde{\Psi}^{\mathrm{T}}_{\mathrm{N+1}}\mathrm{P}_{\mathrm{N}} \\
\mathrm{K}_{\mathrm{N+1}} &= \mathrm{P}_{\mathrm{N}}\tilde{\Psi}^{\mathrm{T}}_{\mathrm{N+1}}(1 + \tilde{\Psi}^{\mathrm{T}}_{\mathrm{N+1}}\mathrm{P}_{\mathrm{N}}\tilde{\Psi}_{\mathrm{N+1}})^{-1}
\end{aligned}
$$

其中：

$$
\begin{aligned}
\mathrm{y}_{\mathrm{N}} &= \tilde{\Psi}^{\mathrm{T}}_{\mathrm{N}}\tilde{\theta} + \hat{\varepsilon}_{(\mathrm{n+N})} \\
\tilde{\Psi}_{\mathrm{N}} &= \begin{bmatrix} \Psi^{\mathrm{T}}_{\mathrm{N}} & \hat{\omega}^{\mathrm{T}}_{\mathrm{N}} \end{bmatrix}^{\mathrm{T}} \\
\Psi_{\mathrm{N}} &= \begin{bmatrix} -\mathrm{y}_{(\mathrm{n+N-1})} & \cdots & -\mathrm{y}_{(\mathrm{N})} & \mathrm{u}_{(\mathrm{n+N})} & \cdots & \mathrm{u}_{(\mathrm{N})} \end{bmatrix}^{\mathrm{T}} \\
\hat{\omega}_{\mathrm{N}} &= \begin{bmatrix} \hat{\xi}_{(\mathrm{n+N-1})} & \cdots & \hat{\xi}_{(\mathrm{n+N-m})} \end{bmatrix}^{\mathrm{T}} \\
\hat{\xi}_{\mathrm{k}} &= \mathrm{y}_{\mathrm{k}} - \Psi_{\mathrm{k}}\hat{\theta}
\end{aligned}
$$

Augmented matrix method

- The noise model parameters are extended to the identified parameter vectors
- Simultaneous identification of system parameters and noise parameters
- It is widely used and has a good convergence
- Recursive methods are often used in practical algorithms

Augmented matrix method: system model

$$
\begin{aligned}
a(z^{-1})y(k) &= b(z^{-1})u(k) + c(z^{-1})\varepsilon(k) \\
a(z^{-1}) &= 1 + a_1 z^{-1} + \cdots + a_n z^{-n} \\
b(z^{-1}) &= b_0 + b_1 z^{-1} + \cdots + b_n z^{-n} \\
c(z^{-1}) &= 1 + c_1 z^{-1} + \cdots + c_n z^{-n}
\end{aligned}
$$

Augmented matrix method: system model vector representation

$$
\begin{aligned}
\mathrm{y_N} &= \mathrm{y_{(n+N)}} \\
&= \Psi_{\mathrm{N}}^{\mathrm{T}}\theta + \varepsilon_{\mathrm{(n+N)}} \\
&= \begin{bmatrix}\Psi_{\mathrm{N,(y,u)}}^{\mathrm{T}} & \Psi_{\mathrm{N},\xi}^{\mathrm{T}}\end{bmatrix}\begin{bmatrix}\theta_{\mathrm{(y,u)}} \\ \theta_{\xi}\end{bmatrix} + \varepsilon_{\mathrm{N}} \\
\theta &= \begin{bmatrix}\theta_{\mathrm{(y,u)}} & \theta_{\xi}\end{bmatrix}^{\mathrm{T}} \\
\theta_{\mathrm{(y,u)}} &= \begin{bmatrix}\mathrm{a_1} & \cdots & \mathrm{a_n} & \mathrm{b_0} & \cdots & \mathrm{b_n}\end{bmatrix}^{\mathrm{T}} \\
\theta_{\xi} &= \begin{bmatrix}\mathrm{c_1} & \cdots & \mathrm{c_n}\end{bmatrix}^{\mathrm{T}} \\
\Psi_{\mathrm{N}} &= \begin{bmatrix}\Psi_{\mathrm{N,(y,u)}} & \Psi_{\mathrm{N},\xi}\end{bmatrix}^{\mathrm{T}} \\
\Psi_{\mathrm{N,(y,u)}} &= \begin{bmatrix}-\mathrm{y_{(n+N-1)}} & \cdots & -\mathrm{y_{(N)}} & \mathrm{u_{(n+N)}} & \cdots & \mathrm{u_{(N)}}\end{bmatrix}^{\mathrm{T}} \\
\Psi_{\mathrm{N},\xi} &= \begin{bmatrix}\varepsilon_{\mathrm{(n+N-1)}} & \cdots & \varepsilon_{\mathrm{(N)}}\end{bmatrix}^{\mathrm{T}}
\end{aligned}
$$

Augmented matrix method: Parameter Solving

$$
\begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix} = \begin{bmatrix} \Psi^T_{1,(y,u)} & \Psi^T_{1,\xi} \\ \vdots & \vdots \\ \Psi^T_{N,(y,u)} & \Psi^T_{N,\xi} \end{bmatrix} \begin{bmatrix} \theta_{(y,u)} \\ \theta_\xi \end{bmatrix} + \begin{bmatrix} \varepsilon_1 \\ \vdots \\ \varepsilon_N \end{bmatrix}
$$

$$
Y = \begin{bmatrix} \Phi_{(y,u)} & \Phi_\xi \end{bmatrix} \begin{bmatrix} \theta_{(y,u)} \\ \theta_\xi \end{bmatrix} + \varepsilon
$$

$$
\begin{bmatrix} \hat{\theta}_{(y,u)} \\ \hat{\theta}_\xi \end{bmatrix} = \begin{bmatrix} \Phi^T_{(y,u)}\Phi_{(y,u)} & \Phi^T_{(y,u)}\Phi_\xi \\ \Phi^T_\xi \Phi_{(y,u)} & \Phi^T_\xi \Phi_\xi \end{bmatrix}^{-1} \begin{bmatrix} \Phi^T_{(y,u)} Y \\ \Phi^T_\xi Y \end{bmatrix}
$$

Augmented matrix method: recursive equations

use $\hat{\varepsilon}$ instead of ε:

$$
\begin{aligned}
y_N &= \hat{\Psi}_N^T \hat{\theta} + \hat{\varepsilon}_{(n+N)} \\
\hat{\Psi}_N &= \begin{bmatrix} -y_{(n+N-1)} & \cdots & -y_{(N)} & u_{(n+N)} & \cdots & u_{(N)} & \hat{\epsilon}_N^T \end{bmatrix}^T \\
\hat{\epsilon}_N &= \begin{bmatrix} \hat{\varepsilon}_{(n+N-1)} & \cdots & \hat{\varepsilon}_{(N)} \end{bmatrix}^T
\end{aligned}
$$

Available recurrence formula :

$$
\begin{aligned}
\hat{\theta}_{N+1}^T &= \hat{\theta}_N + K_{N+1}(y_{N+1} - \hat{\Psi}_{N+1}^T \hat{\theta}_N) \\
P_{N+1} &= P_N - K_{N+1} \hat{\Psi}_{N+1}^T P_N \\
K_{N+1} &= P_N \hat{\Psi}_{N+1}^T (1 + \hat{\Psi}_{N+1}^T P_N \hat{\Psi}_{N+1})^{-1}
\end{aligned}
$$