연세경제연구 제Ⅱ권, 제2호
Yonsei Economic Studies, Vol. Ⅱ, No. 2 (September 1995)

A Levinson-Type Prediction Algorithm of a Single-Input Single-Output System

Chul Eung Kim and ByoungSeon Choi

A Levinson-type algorithm is presented to calculate the linear predictions of a single-input single-output system recursively.

Ⅰ. Introduction

Consider a single-input single-output system. The input process $\{x_t\}$ and the output process $\{y_t\}$ are real valued and wide-sense stationary, which have zero-means and finite second-order moments. Assume that the processes are not purely deterministic. Denote by $\mathcal{L}(k)$ the linear space spanned by $x_k, x_{k-1}, \cdots, x_0$. Also, define the predictor $\hat{y}_k$ as the projection of y_k on $\mathcal{L}(k)$, which is characterized by the orthogonality relations

$$E(y_k - \hat{y}_k)x_j = 0 \qquad j = 0, 1, \cdots, k \tag{1}$$

where E denotes the expectation. Since $\hat{y}_k$ is the projection of y_k on $\mathcal{L}(k)$, there exist the coefficients $v_{k,0}, v_{k,1}, \cdots, v_{k,k}$ satisfying

$$\hat{y}_k = v_{k,0}\, x_k + v_{k,1}\, x_{k-1} + \cdots + v_{k,k}\, x_0 \tag{2}$$

Define the autocovariance function (ACVF) of $\{x_t\}$ the ACVF of $\{y_t\}$ and the cross-covariance function (CCVF) of $\{x_t\}$ and $\{y_t\}$ by

$$\sigma_{xx}(j) = Cov(x_{t+j}, x_t) \qquad j = 0, \pm 1, \cdots$$

Department of Applied Statistics, Yonsei University, Seoul, 120-749, Korea.

$$\sigma_{yy}(j)=Cov(y_{t+j},\ y_t) \qquad j=0,\ \pm 1,\ \cdots$$
$$\sigma_{yx}(j)=\sigma_{xy}(-j)=Cov(y_{t+j},\ x_t) \qquad j=0,\ \pm 1,\ \cdots$$

Equations (1) and (2) imply

$$\sigma_{yx}(j)=\sum_{m=0}^{k} v_{k,m}\ \sigma_{xx}(j-m) \qquad j=0,\ 1,\ \cdots,\ k \qquad (3)$$

When $\{\sigma_{xx}(j)\}$ and $\{\sigma_{yx}(j)\}$ are given, the coefficients $v_{k,0},\ v_{k,1},\ \cdots,\ v_{k,k}$ can be obtained by solving the simultaneous equations (3). As mentioned in [1, p. 379], the simultaneous equations (3) are cumbersome to solve if the traditional methods like the Gauss elimination are applied. The purpose of this correspondence is to present a computationally efficient Levinson-type algorithm to solve the simultaneous systems. So can be easily obtained the predictors $\{\hat{y}_k \mid k=0,\ 1,\ \cdots\}$.

II. A Levinson-Type Algorithm

For $j=1,\ 2,\ \cdots,\ k$ and $k=0,\ 1,\ 2,\ \cdots$, let $l_{k,j}$ be the solution of the simultaneous equations:

$$\sigma_{xx}(j)=\sum_{m=1}^{k} l_{k,m}\ \sigma_{xx}(j-m) \qquad j=1,\ 2,\ \cdots,\ k$$

which are the Yule-Walker equations of the input process $\{x_t\}$. Since $\{x_t\}$ is not purely deterministic, the solution $\{l_{k,1},\ l_{k,2},\ \cdots,\ l_{k,k}\}$ exist uniquely. Define two variances of the prediction errors as

$$K_k=E(y_t-\sum_{j=0}^{k} v_{k,j}\ x_{t-j})^2$$
$$\lambda_k=E(x_t-\sum_{j=1}^{k} l_{k,j}\ x_{t-j})^2$$

Then,

$$K_k=\sigma_{yy}(0)-\sum_{j=0}^{k} v_{k,j}\ \sigma_{yx}(j)$$
$$\lambda_k=\sigma_{xx}(0)-\sum_{j=1}^{k} l_{k,j}\ \sigma_{xx}(j)$$

Define a $(k+1)$-dimensional symmetric Toeplitz matrix and two associated vectors by

$$\sum_{xx}(k)=\begin{bmatrix} \sigma_{xx}(0) & \sigma_{xx}(-1) & \cdots & \sigma_{xx}(-k) \\ \sigma_{xx}(1) & \sigma_{xx}(0) & \cdots & \sigma_{xx}(-k+1) \\ \vdots & \vdots & \vdots & \vdots \\ \sigma_{xx}(k) & \sigma_{xx}(k-1) & \cdots & \sigma_{xx}(0) \end{bmatrix}$$

$$\sigma_{yx}(k)=[\sigma_{yx}(0),\ \sigma_{yx}(1),\ \cdots,\ \sigma_{yx}(k)]^t$$

$$v(k)=(v_{k,0},\ v_{k,1},\ \cdots\ v_{k,k})^t$$

Also, define a $(k+1)\times(k+1)$ lower triangular matrix $L(k)$ and a $(k+1)\times(k+1)$ diagonal matrix $\Lambda(k)$ by

$$L(k)=\begin{bmatrix} l_{0,0} & 0 & 0 & 0 \\ l_{1,1} & l_{1,0} & 0 & 0 \\ l_{2,2} & l_{2,1} & l_{2,0} & 0 \\ \vdots & \vdots & \vdots & \vdots \\ l_{k,k} & l_{k,k-1} & l_{k,k-2} & l_{k,0} \end{bmatrix}$$

where $l_{k,0}=-1,\ (k=0,\ 1,\ \cdots)$ and

$$\Lambda(k)=\begin{bmatrix} \lambda_0 & 0 & 0 & \cdots & 0 \\ 0 & \lambda_1 & 0 & \cdots & 0 \\ 0 & 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & \lambda_k \end{bmatrix}$$

Our problem is to solve

$$v(k)=\sum_{xx}^{-1}(k)\ \sigma_{yx}(k) \tag{4}$$

for $k=\ 0,\ 1,\ \cdots$. It is known [2, p. 81] that the inverse of $\sum_{xx}(k)$ can be decomposed as follows.

Lemma 1. For $k=0,\ 1,\ 2,\ \cdots,\ \sum_{xx}^{-1}(k)$ can be uniquely factored into a product of an upper triangular, a diagonal and a lower triangular matrices as follows.

$$\sum_{xx}^{-1}(k)=L(k)^t\ \Lambda^{-1}(k)\ L(k)$$ ▨

A Levinson-type algorithm based on Lemma 1 is presented to calculate

$\{v_{k,0}, v_{k,1}, \cdots, v_{k,k}, K_k \mid k=0, 1, \cdots\}$ as follows.

Algorithm 1. *A Levinson-Type Algorithm*

Initial values for recursion:

$$v_{0,0} = \frac{\sigma_{yx}(0)}{\sigma_{xx}(0)}$$

$$\lambda_0 = \sigma_{xx}(0)$$

$$K_0 = \sigma_{yy}(0) - v_{0,0}\,\sigma_{yx}(0)$$

$$l_{1,1} = \frac{\sigma_{xx}(1)}{\sigma_{xx}(0)}$$

$$\lambda_1 = \sigma_{xx}(0)(1-l_{1,1}^2)$$

For $k = 1, 2, \cdots$,

$$l_{k+1,k+1} = \frac{1}{\lambda_k}[\sigma_{xx}(k+1) - l_{k,1}\,\sigma_{xx}(k) - \cdots - l_{k,k}\,\sigma_{xx}(1)]$$

$$v_{k,k} = \frac{1}{\lambda_k}[\sigma_{yx}(k) - l_{k,1}\,\sigma_{yx}(k-1) - \cdots - l_{k,k}\,\sigma_{yx}(0)]$$

$$\lambda_{k+1} = \lambda_k(1-l_{k+1,k+1}^2)$$

$$K_k = K_{k-1} - v_{k,k}^2\lambda_k$$

For $j = 1, 2, \cdots, k$

$$l_{k+1,j} = l_{k,j} - l_{k+1,k+1}\,l_{k,k+1-j}$$

$$v_{k,j-1} = v_{k-1,j-1} - v_{k,k}\,l_{k,k+1-j}$$

▨

Derivation of Algorithm 1. The formulae about $\{l_{k,j}\}$ and $\{\lambda_k\}$ consist of the Levinson-Durbin algorithm. See, e.g., [3, pp. 182~186]. For completeness they are derived as follows.

Let

$$\sigma_{xx}(1;k) = [\sigma(1), \sigma(2), \cdots, \sigma(k)]^t$$

$$l(k) = (l_{k,1}, l_{k,2}, \cdots, l_{k,k})^t$$

$$z(k) = (l_{k,k}, l_{k,k-1}, \cdots, l_{k,1})^t$$

Then,

$$l(k) = \sum_{xx}^{-1}(k-1)\,\sigma_{xx}(1;k)$$

Also, $L(k)$, $\Lambda(k)$ and $\sigma_{xx}(1;k+1)$ can be decomposed as

$$L(k)=\begin{bmatrix} L(k-1) & 0 \\ z(k)' & -1 \end{bmatrix}$$

$$\Lambda(k)=\begin{bmatrix} \Lambda(k-1) & 0 \\ 0' & \lambda_k \end{bmatrix}$$

$$\sigma_{xx}(1;k+1)=\begin{bmatrix} \sigma_{xx}(1;k) \\ \sigma_{xx}(k+1) \end{bmatrix}$$

Lemma 1 implies

$$\begin{aligned} &\sum_{xx}(k) \\ &=\begin{bmatrix} L(k-1)' & z(k) \\ 0' & -1 \end{bmatrix}\begin{bmatrix} \Lambda^{-1}(k-1) & 0 \\ 0' & 1/\lambda_k \end{bmatrix}\begin{bmatrix} L(k-1) & 0 \\ z(k)' & -1 \end{bmatrix} \\ &=\begin{bmatrix} \sum_{xx}^{-1}(k-1)+z(k)\,z(k)'/\lambda_k & -z(k)/\lambda_k \\ z(k)'/\lambda_k & 1/\lambda_k \end{bmatrix} \end{aligned}$$

Therefore,

$$\begin{aligned} &l(k+1) \\ &=\sum_{xx}^{-1}(k)\ \sigma_{xx}(1;k+1) \\ &=\begin{bmatrix} l(k)-z(k)\ \{\sigma_{xx}(k+1)-z(k)'\ \sigma_{xx}(1;k)\}/\lambda_k \\ \{\sigma_{xx}(k+1)-z(k)'\ \sigma_{xx}(1;k)\}/\lambda_k \end{bmatrix} \end{aligned}$$

Thus,

$$\begin{aligned} &l_{k+1,\,k+1}=\{\sigma_{xx}(k+1)-l_{k,1}\ \sigma_{xx}(k)-\cdots-l_{k,k}\ \sigma_{xx}(1)\}/\lambda_k \\ &l_{k+1,\,j}=l_{k,j}-l_{k+1,\,k+1}\ l_{k,\,k+1-j}, \qquad j=1,\ 2,\ \cdots,\ k \end{aligned}$$

The recursive formula for $\{\lambda_k\}$ can be derived as follows.

$$\begin{aligned} &\lambda_{k+1} \\ &=\sigma_{xx}(0)-\sum_{j=1}^{k} l_{k+1,\,j}\ \sigma_{xx}(j)-l_{k+1,\,k+1}\ \sigma_{xx}(k+1) \\ &=\sigma_{xx}(0)-\sum_{j=1}^{k}(l_{k,\,j}-l_{k+1,\,k+1}\,l_{k,\,k+1-j})\sigma_{xx}(j)-l_{k+1,\,k+1}\ \sigma_{xx}(k+1) \\ &=\{\sigma_{xx}(0)-\sum_{j=1}^{k} l_{k,\,j}\ \sigma_{xx}(j)\} \\ &\quad-l_{k+1,\,k+1}\ \{\sigma_{xx}(k+1)-\sum_{j=1}^{k} l_{k,\,k+1-j}\sigma_{xx}(j)\} \end{aligned}$$

$$= \lambda_k (1 - l_{k+1,\,k+1}^2)$$

Decompose $\sigma_{yx}(k)$ as

$$\sigma_{yx}(k) = \begin{bmatrix} \sigma_{yx}(k-1) \\ \sigma_{yx}(k) \end{bmatrix}$$

Applying the same method as before, it can be shown

$$\begin{aligned} & v(k) \\ &= \sum_{xx}^{-1}(k)\ \sigma_{yx}(k) \\ &= \begin{bmatrix} \sum_{xx}^{-1}(k-1) + z(k)\,z(k)'/\lambda_k & -z(k)/\lambda_k \\ -z(k)'/\lambda_k & 1/\lambda_k \end{bmatrix} \begin{bmatrix} \sigma_{yx}(k-1) \\ \sigma_{yx}(k) \end{bmatrix} \\ &= \begin{bmatrix} v(k-1) - z(k)\{\sigma_{yx}(k) - z(k)'\ \sigma_{yx}(k-1)\}/\lambda_k \\ \{\sigma_{yx}(k) - z(k)'\ \sigma_{yx}(k-1)\}/\lambda_k \end{bmatrix} \end{aligned}$$

Thus,

$$v_{k,k} = \frac{1}{\lambda_k}\{\sigma_{yx}(k) - l_{k,1}\ \sigma_{yx}(k-1) - \cdots - l_{k,k}\ \sigma_{yx}(0)\}$$

$$v_{k,j} = v_{k-1,j} - v_{k,k}\ l_{k,k-j}, \qquad j = 0,\ 1,\ \cdots,\ k-1$$

The recursive formula for $\{K_k\}$ can be derived as follows.

$$\begin{aligned} & K_k \\ &= \sigma_{xx}(0) - \sum_{j=0}^{k-1} v_{k,j}\ \sigma_{yx}(j) - v_{k,k}\ \sigma_{yx}(k) \\ &= \sigma_{yy}(0) - \sum_{j=0}^{k-1} (v_{k-1,j} - v_{k,k}\, l_{k,k-j})\sigma_{yx}(j) - v_{k,k}\ \sigma_{yx}(k) \\ &= \{\sigma_{yy}(0) - \sum_{j=0}^{k-1} v_{k-1,j}\ \sigma_{yx}(j)\} - v_{k,k}\ \{\sigma_{yx}(k) - \sum_{j=0}^{k-1} l_{k,k-j}\ \sigma_{yx}(j)\} \\ &= K_{k-1} - \lambda_k\ v_{k,k}^2 \end{aligned}$$

It completes the derivation.

III. Comment

In a single-input single-output linear system, the input series $\{x_t\}$ and the output

series $\{y_t\}$ are related through an infinite-order linear transfer function, LTF(∞), model,

$$y_t = \sum_{j=0}^{\infty} v_j x_{t-j} + n_t$$

where $\{n_t\}$ is the noise process that is independent of the input series $\{x_t\}$. The coefficients $\{v_0, v_1, \cdots\}$ are named the impulse response weights. In practical situation, it is impossible to estimate all the coefficients of the LTF(∞) model, because the number of observations is finite. Thus, it is approximated by a linear transfer function model of order k, LTF(k) model,

$$y_t = v_{k,0}\, x_t + v_{k,1}\, x_{t-1} + \cdots v_{k,k}\, x_{t-k} + n_{k,t} \tag{5}$$

where $\{x_t\}$ and $\{n_{k,t}\}$ are independent. Clearly the normal equations to estimate $v_{k,0}$, $v_{k,2}$, $\cdots$, $v_{k,k}$ by the least squares method satisfy the orthogonality conditions (1) and the variance of n_{t}, is x_k. To approximate the LTF(∞) model, it is necessary to select the order k of the LTF(k) model. In literature some penalty function methods such as FPE, AIC, CAT, BIC and HQC are popularly used to determine the order [2, pp. 43~74]. To utilize these methods, it is required to estimate K_k for $k=0, 1, \cdots$. When the observations are given, $\{K_k \mid k=0, 1, \cdots\}$ can be recursively estimated by substituting the sample ACVF and the sample CCVF for the ACVF and the CCVF, respectively, in Algorithm 1.

Since $\sum_{xx}(k)$ is ill-conditioned matrix [4], the least squares method bears a multicollinearity problem. Thus, the ridge solution may give a better estimate [5], which is intrinsically to solve the equation

$$v_\alpha(k) = (\sum_{xx}(k) + \alpha I)^{-1}\, \sigma_{yx}(k) \tag{6}$$

where I is an identity matrix and $\alpha(\geq 0)$ is the shrinkage factor. When $\alpha > 0$, the ridge estimates of $v_{k,1}$, $v_{k,2}$, $\cdots$, $v_{k,k}$ are biased but tend to be more stable than the ordinary least squares estimates utilizing the normal equations (4). Equation (6) can be solved for $k=0, 1, \cdots$ by replacing $\sigma_{xx}(0)$ with $\sigma_{xx}(0) + \alpha$ in Algorithm 1.

❖ *REFERENCES* ❖

1. Box, G. E. P. and G. M. Jenkins, *Time Series Analysis; Forecasting and Con-*

trol, (Revised Edition), Holden-Day, San Francisco, 1976.

2. Choi, B. S., *ARMA Model Identification*, Springer-Verlag, New York, 1992.
3. Caines, P. E., *Linear Stochastic Systems*, Wiley, New York, 1988.
4. Cybenko, G., "The Numerical Stability of the Levinson-Durbin Algorithm for Toeplitz Systems of Equations," *SIAM Journal of Science and Statistical Computing*, Vol. 1, 1980, pp. 303~310.
5. Edlund, P. O., "On Identification of Transfer Function Models by Biased Regression Methods," *Journal of Statistics and Computational Simulation*, Vol. 31, 1989, pp. 131~148.