MODELING, IDENTIFICATION AND CONTROL, 1984, VOL. 5, NO. 2, 71–101
doi:10.4173/mic.1984.2.1

Lattice form recursive linear least square algorithms[†]

DONALD M. WIBERG[‡], FRED BASKIN[§]
and R. D. LINDSAY[§]

Keywords: *lattices, least squares, recursive estimation, digital filtering, signal processing, communications, control systems, noise cancelling, spectral estimation, signal identification, channel equalization, adaptive arrays.*

There are many practical applications of lattice form recursive linear least square algorithms (called lattices for short) in signal processing, communications, and control systems. The goal of this tutorial is to help practicing engineers to decide if lattices are appropriate for their particular projects.

First, a definition of the linear least square problem is given and lattices are described in a non-mathematical way. Next, the advantages and disadvantages of using lattices are discussed. Many applications of lattices are presented, as examples of the types of problems in which lattices can be useful. The rather complicated lattice equations are then given, and general properties of lattices are discussed. The tutorial concludes with the results of a number of simulations.

1. Introduction

Fast and stable computation of the solution to linear least square problems is needed in many applications. The recently developed lattice form algorithms are fast, numerically stable, and recursive in both time and system order. In this tutorial scattered material concerning lattice form recursive linear least square algorithms has been assembled in a manner to emphasize practical application. Computer simulations are given that show the utility and also the limitations of the lattice algorithms.

Lattice form algorithms minimize the sum of squared errors recursively both in time t and in system order N. Because of the recursive nature in N, the number of arithmetic operations required to implement the lattice algorithms increases linearly with N, compared with the quadratic increase with N of other previous algorithms.

The utility of the lattice algorithms is enhanced because their mathematical lattice structure can be physically realized by micro-electronics. Chips implementing a lattice algorithm are easily designed. Furthermore, lattice algorithms can be extended to the vector case to handle multiple inputs and multiple outputs.

Present practical applications of lattice algorithms include speech generation and recognition, spectral estimation, geophysical data analysis, adaptive antenna arrays, adaptive channel equalization, noise cancelling, and recursive parameter estimation.

The analytic theory behind lattices is not complex. Lattices are equivalent to the recursive least square algorithm (Goodwin and Payne 1977) in dynamic effect, and are merely a re-organization of the manner in which the equations are solved. Therefore presentation of the lattice equations is postponed until § 3, and even there proof is relegated to the references.

Received 22 November 1983.

† This work was done at the Aerospace Corporation, El Segundo, California 90009, U.S.A., and at the Institute for Teknisk Kybernetikk, NTH, Trondheim, Norway.

‡ Pro temp. Inst. for Teknisk Kybernetikk, NTH, Tondheim, Norway.

§ The Aerospace Corp., El Segundo, California 90009, U.S.A.

1.1. *The linear least square problem*

Lattice form algorithms solve the linear least square problem. This problem can be stated in the following way. Given two data sequences $\{x(0), x(1), \ldots, x(t)\}$ and $\{y(0), y(1), \ldots, y(t)\}$, find constants $c_{(0)}, c_{(1)}, \ldots, c_{(N)}$ to minimize the sum of squares $S(t, N)$ where

$$S(t, N) = \sum_{\tau=0}^{t} [x(\tau) - c_{(0)}y(\tau) - c_{(1)}y(\tau-1) - \ldots - c_{(N)}y(\tau-N)]^2 \qquad (1.1.1)$$

Lattices can also solve the linear least square problem involving only one data sequence $\{y(0), y(1), \ldots, y(t)\}$. In this case the problem is to find constants $c_{(1)}, c_{(2)}, \ldots, c_{(N)}$ to minimize the sum of squares $S(t, N)$ where

$$S(t, N) = \sum_{\tau=0}^{t} [y(\tau) - c_{(1)}y(\tau-1) - c_{(2)}y(\tau-2) - \ldots - c_{(N)}y(\tau-N)]^2 \qquad (1.1.2)$$

Vector-valued lattice form algorithms can be extended to solve the vector-valued version of the above two problems. For this case $\boldsymbol{x}(t)$ and $\boldsymbol{y}(t)$ are p-dimensional vectors, i.e.

$$\boldsymbol{x}(\tau) = \begin{pmatrix} x_{(1)}(\tau) \\ x_{(2)}(\tau) \\ \vdots \\ x_{(p)}(\tau) \end{pmatrix} \quad \text{and} \quad \boldsymbol{y}(\tau) = \begin{pmatrix} y_{(1)}(\tau) \\ y_{(2)}(\tau) \\ \vdots \\ y_{(p)}(\tau) \end{pmatrix} \qquad (1.1.3)$$

the constants $C_{(0)}, C_{(1)}, \ldots, C_{(N)}$ are $p \times p$ matrices, and the sum of squares $S(t, N)$ is

$$S(t, N) = \sum_{\tau=0}^{t} \|\boldsymbol{x}(\tau) - C_{(0)}\boldsymbol{y}(\tau) - C_{(1)}\boldsymbol{y}(\tau-1) - \ldots - C_{(N)}\boldsymbol{y}(\tau-1)\|^2 \qquad (1.1.4)$$

in which the notation $\|\boldsymbol{x}\|$ denotes the Euclidean length of the vector $\boldsymbol{x}$, i.e.

$$\|\boldsymbol{x}\|^2 = \boldsymbol{x}^T\boldsymbol{x} \qquad (1.1.5)$$

where $\boldsymbol{x}^T$ is the transpose of the vector $\boldsymbol{x}$.

The goal of the vector-valued version of the one data sequence linear least square problem is to minimize the sum of squares $S(t, N)$ where

$$S(t, N) = \sum_{\tau=0}^{t} \|\boldsymbol{y}(\tau) - C_{(1)}\boldsymbol{y}(\tau-1) - C_{(2)}\boldsymbol{y}(\tau-2) - \ldots - C_{(N)}\boldsymbol{y}(\tau-N)\|^2 \qquad (1.1.6)$$

The case where the dimension of $\boldsymbol{x}(t)$ is not equal to the dimension of $\boldsymbol{y}(t)$ can be reduced to the above vector-valued case with equal dimension by adding zeros as remaining components to the vector with the lesser dimension.

By differentiating eqn. (1.1.1) with respect to each $c_{(n)}$ for $n = 1, \ldots, N$, the solution to the linear least square problem can be shown (Goodwin and Payne 1977) to be equivalent to the solution of the normal equation

$$R\boldsymbol{c} = \sum_{\tau=0}^{t} Y(\tau)x(\tau) \qquad (1.1.7)$$

where

$$Y(\tau) = [y(\tau), y(\tau-1), \ldots, y(\tau-N)]^T \qquad (1.1.8)$$

$$\boldsymbol{c} = [c_{(0)}, c_{(1)}, \ldots, c_{(N)}]^T \qquad (1.1.9)$$

and

$$R = \sum_{\tau=0}^{t} Y(\tau) Y^T(\tau) \tag{1.1.10}$$

Therefore, the lattice form algorithms solve normal equations as well as the linear least square problem.

Finally, lattice form algorithms can be modified to solve the weighted linear least square problem, that is, the linear least square problem in which the sum of squares $S(t, N)$ includes a weighting constant λ, where $0 < \lambda \leqslant 1$, as follows:

$$S(t, N) = \sum_{\tau=0}^{t} \lambda^{t-\tau}[x(\tau) - c_{(0)}y(\tau) - c_{(1)}(\tau-1) - \ldots - c_{(N)}y(\tau-N)]^2 \tag{1.1.11}$$

Although the lattice form algorithm can be included as part of algorithms that solve a broader class of problems, lattice form algorithms can be used directly to solve only the linear least square problems of the form of eqns. (1.1.1), (1.1.2), (1.1.4), (1.1.6) or (1.1.11) and no others.

1.2. *Qualitative description of lattice algorithms*

Lattice form recursive linear least square algorithms minimize the sum of squares $S(t, N)$ recursively in time. In other words, given the solution to the linear least square problem at time t, i.e. given the optimum values of $c_{(0)}(t), c_{(1)}(t), \ldots, c_{(N)}(t)$, suppose two new data points $x(t+1)$ and $y(t+1)$ are added to the data sets. The time recursive problem is then to find new optimum values $c_{(0)}(t+1), c_{(1)}(t+1), \ldots, c_{(N)}(t+1)$ to minimize $S(t+1, N)$ by somehow using the previous solution to advantage.

Lattice form recursive linear least square algorithms also minimize the sum of squares $S(t, N)$ recursively in system order. In other words, at a fixed time t, given the solution to the linear least square problem for system order N, i.e. given the optimum values of $c_{N(0)}(t), c_{N(1)}(t), \ldots, c_{N(N)}(t)$, find new optimum values $c_{N+1(0)}(t), c_{N+1(1)}(t), \ldots, c_{N+1(N)}(t), c_{N+1(N+1)}(t)$ to minimize $S(t, N+1)$ by somehow using the previous solution to advantage.

The recursive nature in both system order and time is shown in Fig. 1, which also illustrates a reason for calling the algorithm lattice form.

The optimal values of the constants $c_{(1)}, c_{(2)}, \ldots, c_{(N)}$ are not found directly in the lattice algorithm. However, the optimal values of $c_{(1)}, c_{(2)}, \ldots, c_{(N)}$ can be computed from other, related, constants $k_{(1)}, k_{(2)}, \ldots, k_{(N)}$ called reflection coefficients, which are directly computed in the lattice algorithm. There are lattice algorithms that recursively compute the constants $c_{(1)}, c_{(2)}, \ldots, c_{(N)}$ after computation of the reflection coefficients. However, for brevity these lattices are not considered in this tutorial, and the interested reader is referred to Porat *et al.* (1982). The other directly computed variables are forward and backward residuals $\epsilon_N(t)$ and $\zeta_N(t)$ defined by

$$\epsilon_N(t) = y(t) - \hat{c}_{N(1)}(t)y(t-1) - \hat{c}_{N(2)}(t)y(t-2) - \ldots - \hat{c}_{N(N)}(t)y(t-N) \tag{1.2.1}$$

$$\zeta_N(t) = y(t-N) - \hat{c}'_{N(1)}(t)y(t-N+1) - \ldots - \hat{c}'_{N(N)}(t)y(t) \tag{1.2.2}$$

where $\hat{c}'_{N(1)}, \hat{c}'_{N(2)}, \ldots, \hat{c}'_{N(N)}$ constants are the solution of the related (backwards to eqn. (1.1.2)) problem of $S'(t, N)$ where

$$S'(t, N) = \sum_{\tau=0}^{t} [y(\tau-N) - c'_{N(1)}y(\tau-N+1) - \ldots - c'_{N(N)}y(\tau)]^2 \tag{1.2.3}$$

D 2

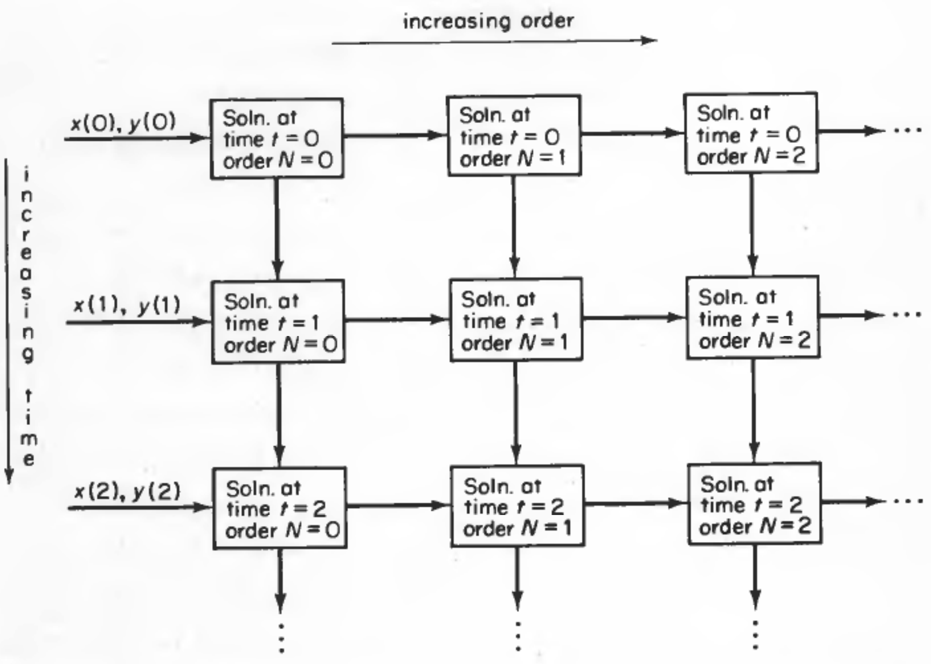
increasing order
increasing time
$x(0), y(0)$
$x(1), y(1)$
$x(2), y(2)$
Soln. at time $t = 0$ order $N = 0$
Soln. at time $t = 0$ order $N = 1$
Soln. at time $t = 0$ order $N = 2$
Soln. at time $t = 1$ order $N = 0$
Soln. at time $t = 1$ order $N = 1$
Soln. at time $t = 1$ order $N = 2$
Soln. at time $t = 2$ order $N = 0$
Soln. at time $t = 2$ order $N = 1$
Soln. at time $t = 2$ order $N = 2$

Figure 1. A lattice structure arises from recursiveness in both time and system order.

As a result of the mathematical derivation of the lattice equations, only forward and backward residuals are needed to propagate the estimation procedure. Therefore, the contents of the boxes shown in Fig. 1 are simply given as in Fig. 2.

There are also auxiliary equations for updating the reflection coefficients $k_N^{\epsilon}(t)$ and $k_N^{\zeta}(t)$. The auxiliary equations are essentially four more recursion relations to compute the reflection coefficients.

Figure 2 shows the simplest lattice form, that of the single data sequence (1.1.2). For the case of two data sequences (1.1.1), only the forward residual for the x-process is needed in addition to the forward and backward residuals ϵ and ζ of the y-process. Therefore, the computational load of problem (1.1.1) is only about one and one-half

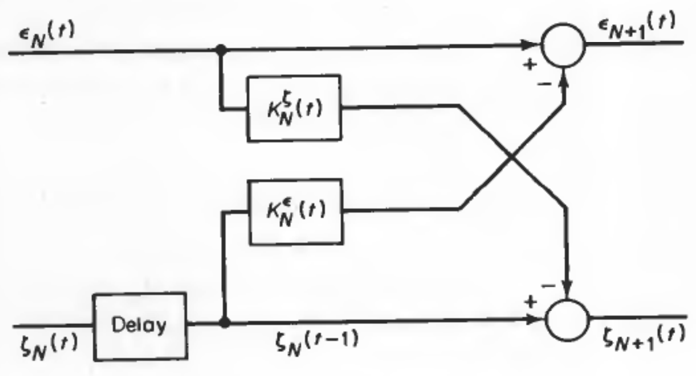

Figure 2. Segment of the lattice algorithm for the solution at time t and order N.

times that of problem (1.1.2). In the vector version (1.1.6), the variables shown in Fig. 2 become vectors and matrices. Finally, there is virtually no increase in computational complexity to incorporate a weighting constant λ as in (1.1.11).

1.3. *Advantages and disadvantages of lattices*

The main reasons for using lattice form algorithms are the following: (1) with the exception of very low order problems, they require fewer computations to solve the recursive linear least square problem than other algorithms that are presently used; (2) they are numerically stable; (3) they are easily implemented on micro-electronic chips; (4) if the input is sampled, they make efficient use of the inter-sample time to obtain as high an order estimate as possible; and (5) they can be incorporated easily into their algorithms that solve a broader class of problems, such as α-stationary (Levi-Ari and Kailath 1981) and parameter estimation (Åström and Mayne 1982) problems. We shall now discuss each of these five main reasons in the order given.

At the present time, Householder (Goodwin and Payne 1977) and other algorithms that are numerically stable are used to solve the non-recursive linear least square problem. For use in a recursive manner, the Y vector of eqn. (1.1.8) and the R matrix of eqn. (1.1.10) must be recomputed for each time point, which is not feasible for a large number of time points. Therefore, it has been necessary to use the recursive least square method (Goodwin and Payne 1977) which is not in lattice form. Unfortunately, that algorithm requires the solution of an N dimensional matrix Riccati equation for which $(N+1)N/2$ coupled non-linear recurrence equations are required. In contrast, the lattice algorithm requires $6N$ non-linear recurrence equations. Consequently, for a system of order $N=30$, the Riccati equations requires 465 recurrence equations, whereas the lattice requires only 180. Therefore, lattice algorithms are used to decrease the number of computations required for high order solutions to the linear least squares problem.

Lattice form algorithms are input/output stable because they describe the stable physical process of the reflection of waves travelling through layered media (see § 4). Numerical stability often results from this input/output stability, and the lattice has been proven to be numerically stable (Ljung and Ljung 1984). Another form of the lattice algorithm exists, called the normalized lattice, but it has been proven biased (Samson and Reddy 1983) and our simulations indicate that it is numerically unstable. Therefore the normalized lattice is not considered this tutorial.

The lattice structure depicted in Figs. 1 and 2 can easily be implemented by micro-electronics. One chip can hold a digital realization of one order section of the lattice (a ' rung ' of a ' ladder '), with the inputs to the chip being $\epsilon(t)$ and $\zeta(t)$ at each order. The chips can be connected together to form a ladder, where the number of chips used equals the maximum order of the system. The inputs are entered at the bottom of the ladder, with each increase in system order corresponding to a rung on the ladder. The residuals are processed up the ladder, with the highest order residual being the ladder output. Intersample time can be efficiently used, with each new data point $x(t)$, $y(t)$ entering at the bottom of the ladder as the data arrives in real time.

The linear least square lattice can be embedded as part of other algorithms that solve a broader class of problems. Often a valid method of solving a non-linear problem is to iterate on a sequence of linearizations of the problem. One such iteration occurs in system parameter estimation (Åström and Mayne 1982). Also problems that are in some sense close to the normal equation (α-stationary (Levi-Ari and Kailath 1981)) can be solved using lattices.

If an application does not require recursive minimization of a linear least squares criterion, the lattice form algorithm should not be used unless a clever method of embedding exists as discussed above. For example, the Kalman filter is the most effective solution of the general state estimation problem, and the lattice algorithm can be applied only to a very small subclass of state estimation problems (see § 2.7).

1.4. *Literature concerning lattices*

Gauss (1809) originated least squares about 1794 and used it to compute the orbit of Ceres in 1801. One hundred years later, Yule and Walker (1927) derived the normal equation for the linear problem. In 1947 Levinson published an algorithm for recursive solution of the Yule–Walker equation that contains the basic elements of the lattice algorithm. Kalman filtering (Kalman 1960) was originated in 1960, at the same time that Widrow (1960) published his approximate solution to the recursive least squares problem. Widrow's approximate gradient algorithm was later adapted to lattices by Griffiths (1977) and Makhoul (1977). A lattice form recursive linear least squares algorithm was first mentioned (Morf *et al.*) in the literature in 1977, and an ever growing stream of papers has appeared since then. The survey papers by Lee *et al.* (1981) and by Friedlander (1982) summarize current literature and give pertinent references. The material in this tutorial is expanded upon in Wiberg *et al.* (1983).

1.5. *AR and MA representations*

Consider one data sequence $\{y(0), y(1), \ldots, y(t)\}$ that has an autoregressive (AR) representation

$$y(t)=c_{(1)}y(t-1)+c_{(2)}y(t-2)+\ldots+c_{(N)}y(t-N)+w(t) \tag{1.5.1}$$

where $w(t)$ is a white noise process. Suppose the coefficients $c_{(1)}, c_{(2)}, \ldots, c_{(N)}$ are unknown. Then a method of computing the estimates $\hat{c}_{(1)}(t), \hat{c}_{(2)}(t), \ldots, \hat{c}_{(N)}(t)$ is to use the single process lattice, as illustrated in Fig. 3. Figure 3 shows that when each estimate $\hat{c}_{(i)}(t)=c_{(i)}$ for $i=1, 2, \ldots, N$, then the numerator polynomial in the delay operator Z^{-1} cancels the denominator polynomial, and the residual produced by the lattice, $\epsilon_N(t)$, is equal to $w(t)$ and therefore $\epsilon_N(t)$ is white. For this reason, single data process lattices are sometimes called whitening filters. Also, notice that the physical process to be whitened, $y(t)$, is an AR process, and the lattice algorithm describes $\epsilon_N(t)$ as a moving average (MA) of $y(t)$, the lattice input.

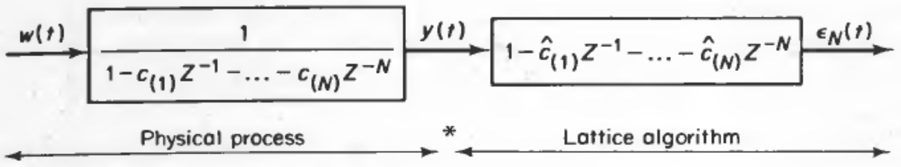
$w(t)$
$\frac{1}{1-c_{(1)}Z^{-1}-\ldots-c_{(N)}Z^{-N}}$
$y(t)$
$1-\hat{c}_{(1)}Z^{-1}-\ldots-\hat{c}_{(N)}Z^{-N}$
$\epsilon_N(t)$
Physical process
*
Lattice algorithm

Figure 3. AR-process $y(t)$ filtered by a single data sequence lattice.

Now consider the two data sequence, in which the given process $x(t)$ is to be expressed as a moving average of the other given process $y(t)$, i.e.

$$x(t)=c_{(0)}y(t)+c_{(1)}y(t-1)+\ldots+c_{(N)}y(t-N)+w(t) \qquad (1.5.2)$$

Then $x(t)$ is *MA* of $y(t)$, and $y(t)$ is an arbitrary process, i.e. $y(t)$ could be a general *ARMA* process. This is pictured in Fig. 4. Optimally predicting $\hat{x}_N(t)$ as *MA* in $y(t)$ will then minimize the linear least square problem of eqn. (1.1.1).

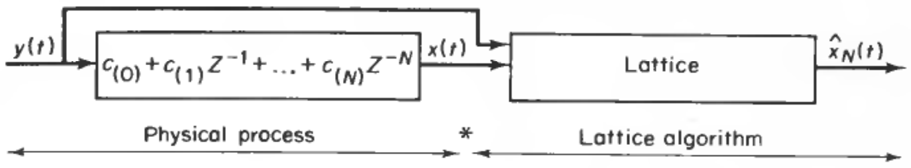

Figure 4. The $x(t)$ data sequence is a moving average of $y(t)$ in a two data sequence lattice.

2. Application of lattices

This section discusses briefly some industrial applications of lattice form linear least square algorithms. Speech generation, speech recognition (spectral estimation, signal identification), geophysical applications, adaptive antenna arrays, adaptive channel equalization (adaptive line enhancement), and recursive system parameter estimation are current uses. Lattices are not directly useful for the general case of state estimation and non-linear systems. Finally, noise cancelling is considered.

2.1. *Speech generation*

Computers, rather than recordings, can now be used to generate speech sounds. Talking toys, such as ' Speak-N-Spell ', and user messages, such as the ' voices ' of telephone operators, use lattice filter networks (not recursive least square, but in lattice form) to produce a remarkable likeness to human speech. Computers using lattices have replaced recordings because information can be stored much more efficiently with lattices. Sampling an analogue recording would require two times the 20 000 cycles of highest heard pitch to be stored for one second of speech. Lattices reduce this information storage requirement by a factor of more than one thousand, because only the value of the reflection coefficients need be stored for each speech sound.

The reason for this efficiency is that lattice filter networks model the process of human speech. Human speech is produced by modifying an acoustic white noise (corresponding to the ' s ' sound) or a single frequency sinusoid (pure tone) produced by the larynx. The tone or noise is modified by appropriate constrictions in the throat that cause the sound waves to reflect in the correct way to produce a certain speech sound. But, as explained in § 4, lattices model the reflection of waves. Therefore, a single set of reflection coefficient values plus the white noise and sinusoid amplitudes and frequency determine a human speech sound. A table of sounds can be assembled and looked up by a computer program that creates speech.

2.2. *Speech recognition (spectral estimation, signal identification)*

In contrast to the speech generation process, a speech recognition problem has unknown reflection coefficients that must be estimated from the data. Therefore, speech generation is only a lattice form filter, whereas speech recognition is a recursive linear least square algorithm in lattice form. Perhaps it is a lucky accident that the recursive linear least square solution can be put in lattice form.

Given speech data $\{y(0), y(1), \ldots, y(t)\}$, the speech recognition problem is to compute reflection coefficients recursively in system order to minimize the sum of squares $S(t, N)$ of eqn. (1.1.2) such that the Nth order forward residual $\epsilon_N(t)$ is the sum of white noise and a single pitch sinusoid. This is the inverse of the speech generation process just previously described, in which white noise plus a single pitch sinusoid is the input and speech is the output.

Once the reflection coefficients are computed, the sound whose reflection coefficients values are closest to the computed values is chosen. Thus, people can talk to a computer.

Speech is an *AR* process, and the coefficients of any *AR* signal can be estimated. Since the coefficients are estimated recursively in order, system order determination techniques can be used. In filtering terminology, the process is equivalent to *AR* spectral identification.

2.3. *Geophysical applications*

Sound waves reflecting from the boundaries of layered media of different sound transmissivities can be modelled by an *AR*-process, as discussed further in § 4. Therefore, a least squares fit of the reflection coefficients can be performed upon seismic data, obtained as pictured in Fig. 5.

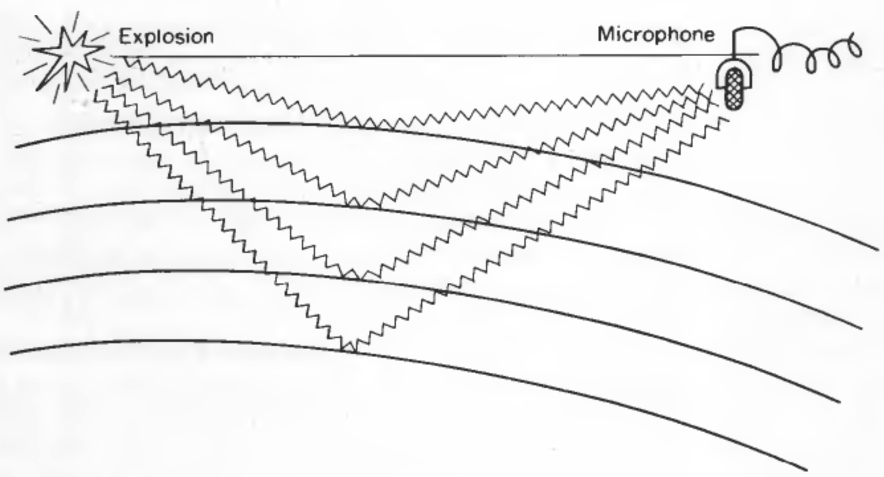

Figure 5. Sound waves reflecting off layered media.

Values of the reflection coefficient can be compared to known values of different media interfaces, for example, to that of water to oil. Obtaining a reflection coefficient value about equal to that of water–oil could mean discovery of oil if the next reflection coefficient had a value close to that of oil to rock.

2.4. *Adaptive antenna arrays*

An array of antennas leading to a radio receiver is pictured in Fig. 6, with a signal coming from one direction and a jamming noise coming from another direction.

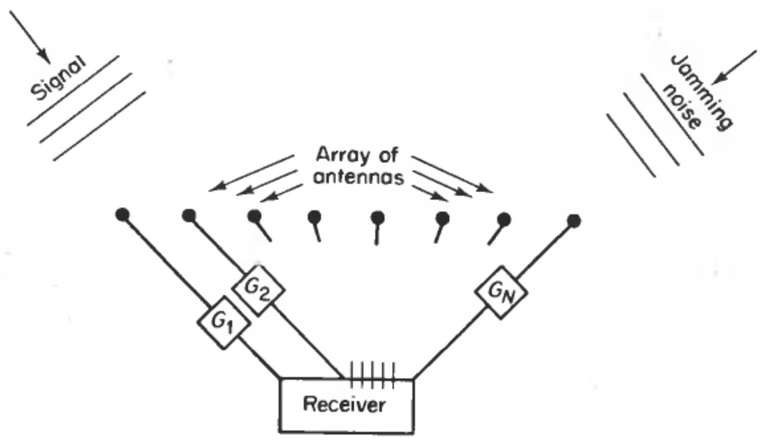

Figure 6. An adaptive antenna array.

The grains G_1, G_2, ..., G_N from each of the N antennas are adjusted to minimize $S(t, N)$, the least square sum defined by

$$S(t, N) = \sum_{\tau=0}^{t} (r(\tau) - G_1 s_1(\tau) - \dots - G_N s_N(\tau))^2 \tag{2.4.1}$$

where $r(t)$ = a reference signal and $s_1(t)$, ..., $s_N(t)$ are the signals from each antenna. The reference signal can be obtained from a number of sources, depending upon the exact application. The above equation (2.4.1) can be put into the least squares formulation of eqn. (1.1.4) by defining

$$y(t) = (s_1(t), s_2(t), \dots, s_N(t))^T \tag{2.4.2}$$

and

$$x(t) = (r(t), 0, \dots, 0)^T \tag{2.4.3}$$

Then the antenna gains G_1, G_2, ..., G_N can be approximated as the top row of the C_0 matrix.

The above example results in a least squares problem of order zero. For such a low order problem, lattice algorithms are not needed and a direct solution is available. However, applications in adaptive arrays often require much more complex computation, for which lattice algorithms are useful.

2.5. *Adaptive channel equalization*

Adaptive channel equalization, also known as adaptive line enhancement, is used in communications to minimize the distortion of a signal transmitted over a channel. An equalizer is installed in the channel as shown in Fig. 7. The output of the equalizer is $c_0 y(t) + c_1 y(t-1) + \dots + c_N y(t-N)$. A known 'training' signal $x(t)$ is transmitted as the input to the channel. The equalizer is supposed to make its output appear as close as possible to the training signal $x(t)$, i.e. the equalizer must choose $c_0, c_1, \ldots, c_N$ to minimize $S(t, N)$ where:

$$S(t, N) = \sum_{\tau=0}^{t} (x(\tau) - c_0 y(\tau) - c_1 y(\tau-1) - \ldots - c_N y(\tau-N))^2 \tag{2.5.1}$$

which is eqn. (1.1.1). The estimate for the true values of the constants $c_0, c_1, \ldots, c_N$ must converge quickly, so that the training signal can be as short in duration as possible. This permits the channel to have other messages as inputs for a longer duration before the training signal need be sent again to update the estimates of $c_0, c_1, \ldots, c_N$.

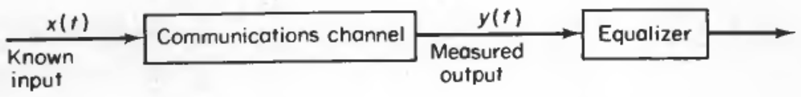

Figure 7. A communications channel with an equalizer.

Satorius and Pack (1980) simulated the response of the lattice form recursive linear least square algorithm used as a channel equalizer. They found the lattice to be much more quickly convergent than both a lattice form of the gradient algorithm and the tap delay form of the gradient algorithm, which were in previous use as channel equalizers.

2.6. *Recursive system parameter estimation*

A method developed by Åström and Mayne (1982) illustrates a possible use of lattice algorithms in system parameter estimation. In this case the system parameter estimation problem is non-linear. The lattice algorithms can be embedded in a larger algorithm, and the overall algorithm solves the non-linear problem as a sequence of linear approximations.

2.7. *Direct state estimation*

Dynamical processes are often described in state space form:

$$\left.\begin{aligned} \boldsymbol{x}(t+1) &= A\boldsymbol{x}(t) + B\boldsymbol{u}(t) + \boldsymbol{w}(t) \\ \boldsymbol{y}(t) &= C\boldsymbol{x}(t) + \boldsymbol{e}(t) \end{aligned}\right\} \tag{2.7.1}$$

where $\boldsymbol{x}(t)$ is an n-dimensional state vector, $\boldsymbol{u}(t)$ is an m-dimensional control vector, $\boldsymbol{w}(t)$ is an n-dimensional zero mean white noise (' process noise ') of variance Q, $\boldsymbol{y}(t)$ is an l-dimensional measured vector, $\boldsymbol{e}(t)$ is an l-dimensional zero mean white noise with variance R (' measurement vector '), and A, B and C are matrices of compatible dimension, possibly depending on time t. A controlled *ARMA* process can be put in the form (2.7.1) by choosing

$$A = \begin{bmatrix} -a_1 & 1 & 0 & \ldots & 0 \\ -a_2 & 0 & 1 & \ldots & 0 \\ \vdots & \vdots & \vdots & & \\ -a_n & 0 & 0 & \ldots & 0 \end{bmatrix}, \quad B = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}, \quad \boldsymbol{w}(t) = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix} w(t) \tag{2.7.2}$$

$C = (1 \ 0 \ \ldots \ 0)$ and $\boldsymbol{e}(t) = w(t)$

The controlled *ARMA* process can only be solved directly by lattice algorithms if $c_1, c_2, \ldots, c_{n-1}$ are all zero, because otherwise estimation becomes a non-linear problem. Therefore, general state estimation can NOT be solved directly by using lattice algorithms and a Kalman filter (Meditch 1969) with improved numerical properties (Bierman 1977) must be used.

2.8. *Estimation of non-linear system parameters*

Lattice form recursive linear least square algorithms can NOT be used directly for non-linear systems. For example, the autonomous non-linear state space representation

$$\left.\begin{aligned} \boldsymbol{x}(t+1) &= \boldsymbol{f}(\boldsymbol{x}(t)) \\ \boldsymbol{y}(t) &= \boldsymbol{g}(\boldsymbol{x}(t)) + \boldsymbol{w}(t) \end{aligned}\right\} \qquad (2.8.1)$$

where $\boldsymbol{f}(\boldsymbol{x})$ and/or $\boldsymbol{g}(\boldsymbol{x})$ are non-linear in $\boldsymbol{x}$, and $\boldsymbol{w}(t)$ is a white noise, does not admit direct solution by lattice algorithms.

2.9. *Noise cancellation*

Consider the simple example illustrated in Fig. 8. A signal is transmitted over one telephone wire, but not over another, parallel, telephone wire. Lightning strikes both wires, producing the same noise in each wire. At the receiving end, a simple noise canceller recovers the signal by subtracting the noise from the signal plus noise.

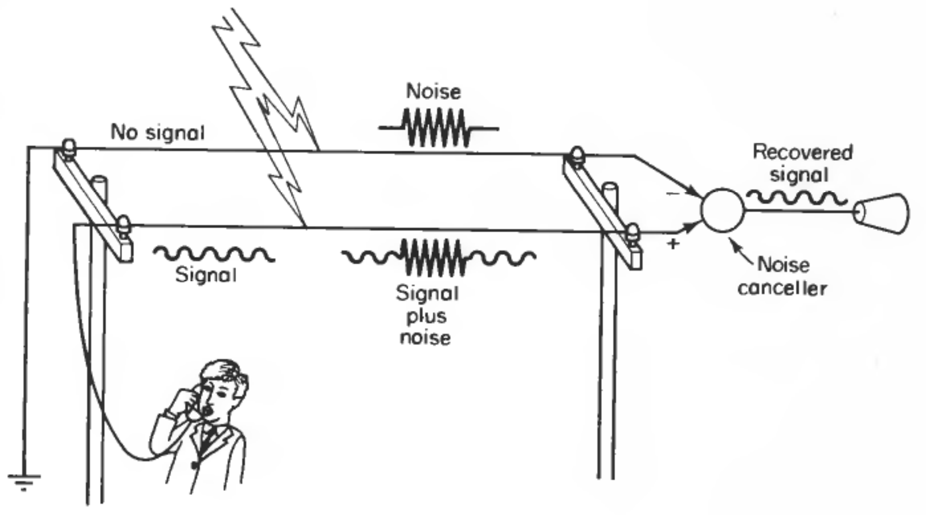

Figure 8. A simple example of noise cancelling.

The preceding example is perhaps the simplest case of noise cancelling. In more complicated cases, for example when a large distance separates the parallel telephone wires in Fig. 8, the noises on each wire are not exactly equal but are statistically related. In such a case the noise canceller becomes an estimator of the noise on the signal line given the noise on the line with no signal. This situation is illustrated in Fig. 9, which is the general configuration for noise cancelling.

Noise cancelling is defined in Fig. 9. There are two random processes $u(t)$ and $v(t)$ that are inputs to the noise canceller. The input $u(t)$ is known to contain the signal $s(t)$ plus other noise $r(t)$. The input $v(t)$ is known NOT to contain the signal $s(t)$, but $v(t)$ is statistically related to $r(t)$. Only $v(t)$ can be used to form an estimate $\hat{r}(t)$ of the noise $r(t)$ in $u(t)$, producing the signal estimate $\hat{s}(t)$ as

$$\hat{s}(t)=u(t)-\hat{r}(t)=u(t)-f(v(y), \ldots, v(0), t) \tag{2.9.1}$$

This equation defines noise cancelling.

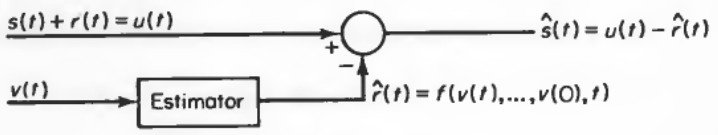
$s(t)+r(t)=u(t)$
$\hat{s}(t)=u(t)-\hat{r}(t)$
+
−
$v(t)$
Estimator
$\hat{r}(t)=f(v(t),\ldots,v(0),t)$

Figure 9. The noise cancelling configuration.

Noise cancelling is different from signal estimation in the following manner. In signal estimation, both random process $v(t)$ and $u(t)$ can be used to find $\hat{s}(t)$, i.e.

$$\hat{s}(t)=g(u(t), \ldots, u(0), v(t), \ldots, v(0), t) \tag{2.9.2}$$

whereas noise cancelling must take the form of eqn. (2.9.1).

In eqn. (2.9.1), the function f of $v(t)$ that forms the estimate $\hat{r}(t)$ can be selected optimally to minimize var $(\hat{s}(t)-s(t))$, the signal error variance. This can be done without knowing $s(t)$ by minimizing var $\hat{s}(t)$. Decompose the inputs $u(t)$ and $v(t)$ into two separate processes each as

$$u(t)=s(t)+r(t) \tag{2.9.3}$$

$$v(t)=n(t)+w(t) \tag{2.9.4}$$

Here $s(t)$, $n(t)$ and $w(t)$ are independent, and $r(t)$ is somehow correlated to $n(t)$. Then from (2.9.1)

$$\text{var}\, \hat{s}(t)=\text{var}\, [u(t)-\hat{r}(t)] \tag{2.9.5}$$

Using (2.9.3) gives

$$\text{var}\, \hat{s}(t)=\text{var}\, [s(t)+r(t)-\hat{r}(t)] \tag{2.9.6}$$

Since $r(t)$ and $v(t)$ are independent of $s(t)$, and $\hat{r}(t)$ is a function of $v(t)$ only

$$\text{var}\, \hat{s}(t)=\text{var}\, s(t)+\text{var}\, [r(t)-\hat{r}(t)] \tag{2.9.7}$$

Since $s(t)$ is a fixed signal, minimizing var $\hat{s}(t)$ minimizes var $[r(t)-\hat{r}(t)]$. But substituting (2.9.3) into (2.9.1) gives

$$\hat{s}(t)=s(t)+r(t)-\hat{r}(t) \tag{2.9.8}$$

Rearranging and taking variances gives

$$\text{var}\, [s(t)-\hat{s}(t)]=\text{var}\, [r(t)-\hat{r}(t)] \tag{2.9.9}$$

Therefore, minimizing var $\hat{s}(t)$ is equivalent to minimizing var $[s(t)-\hat{s}(t)]$ in the noise cancelling configuration.

If the function f that is the estimate of $\hat{r}(t)$ in (2.9.1) can be approximated as a linear combination of the past N values of $v(t)$, i.e.

$$\hat{r}(t)=c_0v(t)+c_1v(t-1)+\ldots+c_Nv(t-N) \tag{2.9.10}$$

and if a time average can be used to approximate the statistical average, then

$$\begin{aligned} t \operatorname{var} \hat{s}(t) &= \sum_{\tau=0}^{t} [u(\tau)-\hat{r}(\tau)]^2 \\ &= \sum_{\tau=0}^{t} [u(\tau)-c_0v(\tau)+c_1v(\tau-1)+\ldots+c_Nv(\tau-N)]^2 \end{aligned} \tag{2.9.11}$$

which is to be minimized by choice of $c_0, c_1, \ldots, c_N$.

This linear least squares sum must be minimized recursively in time, which lattice algorithms can do. Furthermore, noise cancelling has been found to be equivalent to optimal estimation in the case where signal variance is infinite (Wiberg *et al.* 1985).

3. Lattice recursion relationships

This section contains the equations (recursion relationships) for both the one and two data sequence lattice algorithms. The equations are presented in full generality: for vector-valued data sequences and with a forgetting factor. The proofs are to be found in the references, as indicated.

3.1. *Notation*

For clarity, the following notational conventions are used throughout this tutorial. Capital letters denote matrices, bold letters denote vectors, and lower case letters denote scalar quantities. System order is indicated by subscripts, and components of a vector are subscripts in parenthesis. For example, the forward residual $\boldsymbol{\epsilon}$ for a vector-valued data sequence in an Nth order lattice at time t is

$$\boldsymbol{\epsilon}^T(t)=(\epsilon_{N(1)}(t), \epsilon_{N(2)}(t), \ldots, \epsilon_{N(N)}(t)) \tag{3.1.1}$$

The cap T superscript denotes transpose. Then $\boldsymbol{\epsilon}(t)$ is an N-dimensional column vector.

Random processes are defined on a probability triple (sample space, Borel field, and probability measure). However, because only linear systems are considered, the sample variable will be suppressed. For example, $w(t)$ will denote either the value of a sample function of the random process w at time t or the random variable formed by the random process w at time t. The meaning should be clear from the context.

3.2. *The one data sequence lattice*

Figure 10 is a vector-valued version of Fig. 2. Figure 10 illustrates the basic recursion relation between the forward and backward residuals $\boldsymbol{\epsilon}_N(t)$ and $\boldsymbol{\zeta}_N(t)$ and their values for one order higher, $\boldsymbol{\epsilon}_{N+1}(t)$ and $\boldsymbol{\zeta}_{N+1}(t)$. Mathematically, Fig. 10 illustrates the order–update relationships

$$\boldsymbol{\epsilon}_{N+1}(t)=\boldsymbol{\epsilon}_N(t)-K_N^{\epsilon}(t)\boldsymbol{\zeta}_N(t-1) \tag{3.2.1}$$

$$\boldsymbol{\zeta}_{N+1}(t)=\boldsymbol{\zeta}_N(t-1)-K_N^{\zeta}(t)\boldsymbol{\epsilon}_N(t) \tag{3.2.2}$$

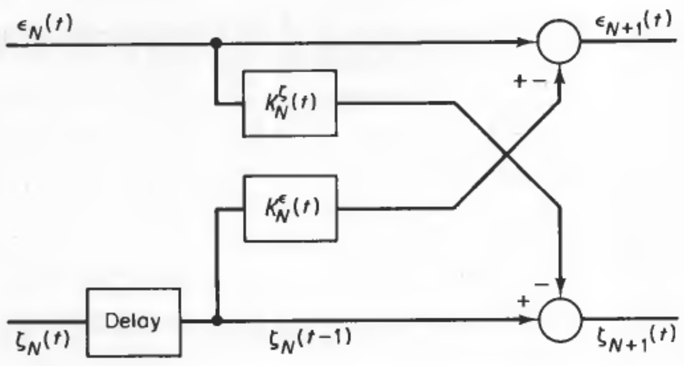

Figure 10. A segment of the vector-valued lattice.

If the one data sequence input to the lattice $\{y(0), y(1), \ldots, y(t)\}$ is a sequence of p-dimensional vectors, i.e.

$$y^T(t) = (y_{(1)}(t), y_{(2)}(t), \ldots, y_{(p)}(t)) \tag{3.2.3}$$

then the forward and backward residuals $\epsilon_N(t)$ and $\zeta_N(t)$ are also p-dimensional vectors.

The gains $K_N^{\epsilon}(t)$ and $K_N^{\zeta}(t)$ must then be $p \times p$ matrices. These gains are functions of quantities computed at the previous time instant. Specifically for the single data sequence lattice, denote the forward and backward residual sample variances $R_N^{\epsilon}(t)$ and $R_N^{\zeta}(t)$, and the sample partial covariance $\Delta_N(t)$. The reason for this terminology will be apparent from the mathematical derivation given in Lee *et al.* (1981) and Friedlander (1982). These quantities must be introduced here in order to compute the gains $K_N^{\epsilon}(t)$ and $K_N^{\zeta}(t)$.

$$K_N^{\epsilon}(t) = \Delta_{N+1}(t)[R_N^{\zeta}(t-1)]^{-1} \tag{3.2.4}$$

$$K_N^{\zeta}(t) = \Delta_{N+1}{}^T(t)[R_N^{\epsilon}(t)]^{-1} \tag{3.2.5}$$

Note that $\Delta_{N+1}(t)$, $R_N^{\epsilon}(t)$ and $R_N^{\zeta}(t)$ are all $p \times p$ matrices. Additionally, R_N^{ϵ} and R_N^{ζ} are symmetric, and initially are chosen equal to a given matrix R.

The final quantities that must be introduced are the scalars, $\theta_N(t)$, $\alpha_N(t)$ and λ. $\theta_N(t)$ and $\alpha_N(t)$ are the square of the sine of the angle between subspaces, as explained in Lee *et al.* (1981, p. 634). ($\theta_N(t)$ is used for Lee's $\cos^2 \theta_{1,n,t}$ and $\alpha_{n+1}(t)$ for $\cos^2 \theta_{0,n,t}$.) However, here both θ_N and α_N can be viewed simply as a correction factor to the time updates for Δ, R^{ϵ} and R^{ζ}. The scalar λ, where $0 < \lambda \leqslant 1$, is the forgetting factor. As was explained by eqn. (1.1.11), incorporation of the forgetting factor λ gives less weight to older data. The forgetting factor λ permits the tracking of possible changes in the *AR* coefficients.

The recurrence relations are presented below. For each new data point $y(t)$, cycle through the order recurrence relations up to the given maximum order of the lattice *NMAX*. The one data sequence lattice can be concisely described as follows:

Input parameters:

NMAX—maximum lattice order
TMAX—maximum time
$y(t)$—data point at time t
λ—exponential forgetting factor
R—initial residual variance

Variables:

$\epsilon_n(t)$, $\zeta_n(t)$—forward (backward) residuals
$R_n^{\epsilon}(t)$, $R_n^{\zeta}(t)$—forward (backward) residual sample variances
$\Delta_n(t)$—sample partial covariance
$\theta_n(t)$, $\alpha_n(t)$—subspace angles

The following order recursions are performed once for each time step ($t=0, \ldots, TMAX$).
At the zeroth order of the lattice:

$$\epsilon_0(t)=\zeta_0(t)=y(t) \tag{3.2.6}$$

$$R_0^{\epsilon}(t)=R_0^{\zeta}(t)=\lambda R_0^{\epsilon}(t-1)+y(t)y^T(t), \quad \text{with } R_0^{\epsilon}(-1)=R \tag{3.2.7}$$

$$\theta_0(t)=1 \tag{3.2.8}$$

For orders $n=1$ to min ($NMAX$, $TMAX$):

$$\Delta_{n+1}(t)=\lambda\Delta_{n+1}(t-1)+\epsilon_n(t)\zeta_n^T(t-1)/\theta_n(t) \tag{3.2.9}$$

with

$$\Delta_{n+1}(-1)=0 \quad \text{and} \quad \zeta_n^T(-1)=\mathbf{0} \tag{3.2.10}$$

$$\theta_{n+1}(t)=\theta_n(t)-\zeta_n^T(t-1)[R_n^{\zeta}(t-1)]^{-1}\zeta_n(t-1) \tag{3.2.11}$$

with

$$R_n^{\zeta}(-1)=R \tag{3.2.12}$$

$$\epsilon_{n+1}(t)=\epsilon_n(t)-\Delta_{n+1}(t)[R_n^{\zeta}(t-1)]^{-1}\zeta_n(t-1) \tag{3.2.13}$$

$$\zeta_{n+1}(t)=\zeta_n(t-1)-\Delta_{n+1}^T(t)[R_n^{\epsilon}(t)]^{-1}\epsilon_n(t) \tag{3.2.14}$$

$$R_{n+1}^{\epsilon}(t)=R_n^{\epsilon}(t)-\Delta_{n+1}(t)[R_n^{\zeta}(t-1)]^{-1}\Delta_{n+1}^T(t) \tag{3.2.15}$$

$$R_{n+1}^{\zeta}(t)=R_n^{\zeta}(t-1)-\Delta_{n+1}^T(t)[R^{\epsilon}(t)]^{-1}\Delta_{n+1}(t) \tag{3.2.16}$$

Note: If any of the quantities $\theta_n(t)$, $R_n^{\epsilon}(t)$ and $R_n^{\zeta}(t)$ does not have an inverse, or is somehow close to being not invertible, see Porat *et al.* (1982). In the scalar case, this non-invertibility reduces to replacing the inverse by zero.

In another form of the one data sequence lattice, the residual sample variances are recursed in time, not in order. This alternative form has advantages in the vector valued case where the algorithm of Gill *et al.* (1975), mentioned in Miller and Wrathall (1980, p. 138), can update $R_n^{\epsilon}(t)$ and $R_n^{\zeta}(t)$. The algorithm of Gill *et al.* (1975) should always be used in the vector valued case to update eqn. (3.2.7). The alternative lattice recursions are:

$$\alpha_n(t)=\begin{cases}1, & \text{if } n=0\\ \theta_{n-1}(t)-\epsilon_{n-1}^T(t)[R_{n+1}^{\epsilon}(t)]^{-1}\epsilon_{n-1}(t), & \text{if } n>0\end{cases} \tag{3.2.17}$$

$$R_n^{\epsilon}(t)=\lambda R_n^{\epsilon}(t-1)+\epsilon_n(t)\epsilon_n^T(t)/\theta_n(t) \tag{3.2.18}$$

$$R_n^{\zeta}(t)=\lambda R_n^{\zeta}(t-1)+\zeta_n(t)\zeta_n^T(t)/\alpha_n(t) \tag{3.2.19}$$

with initial time conditions for all n as

$$\zeta_n(n-1)=\mathbf{0}, \quad \Delta_n(n-1)=0, \quad R_n^{\epsilon}(-1)=0, \quad R_n^{\zeta}(n-1)=R \tag{3.2.20}$$

The values of $\zeta_n(t)$, $R_n^{\zeta}(t)$ and $\Delta_n(t)$ are not needed before $t=n-1$.

3.3. *The two data sequence lattice*

For the two data sequence lattice, the inputs are the vector-valued data sequences $\{x(0), x(1), \ldots, x(t)\}$ and $\{y(0), y(1), \ldots, y(t)\}$. Assume both $x(t)$ and $y(t)$ are of dimension p, and if they are not, the vector of the lesser dimension can be filled up with zeros to make the dimensions of x and y equal. The two data sequence lattice uses the same equations as the one data sequence lattice given in § 3.2, plus the following.

Input parameters:

$x(t)$—corresponding data point at time t

Variables:

$\epsilon_n{}^x(t)$—forward residual in x
$\Delta_n{}^x(t)$—sample partial x covariance

The following additional recursions are performed once for each time step ($t = 0, \ldots, TMAX$).

Initially:

$$\epsilon_{-1}{}^x(t) = x(t) \tag{3.3.1}$$

$$\Delta_n{}^x(t) = 0, \quad \text{for } n > t \tag{3.3.2}$$

For orders $n = 0$ to min ($NMAX$, $TMAX$):

$$\Delta_n{}^x(t) = \lambda \Delta_n{}^x(t-1) + \epsilon_{n-1}{}^x(t) \zeta_n{}^T(t) / \alpha_n(t) \tag{3.3.3}$$

$$\epsilon_n{}^x(t) = \epsilon_{n-1}{}^x(t) - \Delta_n{}^x(t) [R_n{}^\zeta(t)]^{-1} \zeta_n(t) \tag{3.3.4}$$

Of course, $\epsilon_n{}^x(t)$ is a p-vector and $\Delta_n{}^x(t)$ is a $p \times p$ matrix.

In the case of noise cancelling, the residual after the highest order desired $\epsilon_{NMAX}{}^x(t)$ · equals $\hat{s}(t)$, the estimate of the signal in the x process.

3.4. *Derivation of the lattice algorithm*

The proofs are presented in Friedlander (1982) and Lee *et al.* (1981). Also see Ljung and Soderstrom (1983) for a complementary development of lattices, including some proofs.

4. **Properties of the lattice structure**

This section discusses various properties of the lattice structure. First it is shown that the reflection of waves travelling through layered media results in the lattice structure. This gives physical meaning to the reflection coefficients, and a physical basis for the inherent stability of lattices. The relationship between AR and MA representations, and how to approximate $ARMA$ processes, is the last topic of this section.

4.1. *Reflection coefficients*

Ladder networks have a physical interpretation in the transmission and reflection of waves through a layered medium as presented in Claerbout (1976). Consider an upward directed sinusoidal wave of unity amplitude, reflecting off, and transmitting through, a boundary between layers of different media (Fig. 11). The normalized amplitude of the reflected wave, c, is called the reflection coefficient. The amplitude of the transmitted and reflected waves sum to one, which is the amplitude of the incident wave. However, because the waveshape is reversed upon reflection, c is of opposite sign, so that:

$$t+(-c)=1 \tag{4.1.1}$$

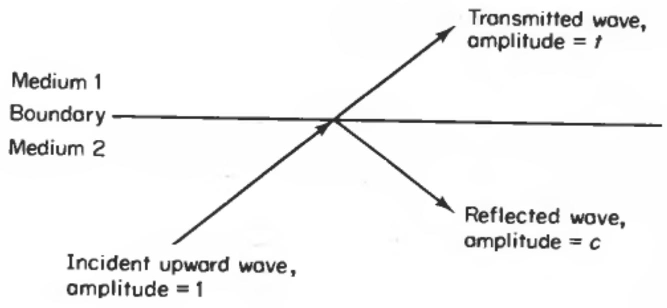

Figure 11. A wave reflecting off a layer boundary.

The same situation occurs for a downward incident wave, which is denoted with primed superscripts:

$$t'+(-c')=1 \tag{4.1.2}$$

The energy associated with any wave is the square of the wave amplitude times Y, a proportionality factor dependent on the medium. Equating the energy in the upward incident wave to the energy in the reflected and transmitted waves gives

$$Y_2 1^2 = Y_2 c^2 + Y_1 t^2 \tag{4.1.3}$$

Using (4.1.1) to substitute for t in the above equation and solving for c gives a value for the reflection coefficient c as

$$c=\frac{Y_2-Y_1}{Y_1+Y_2} \tag{4.1.4}$$

Repeating this argument for the downward incident wave, or more simply interchanging Y_1 and Y_2 in the above formula, gives

$$c'=-c \tag{4.1.5}$$

Now consider two waves incident on the same boundary (see Fig. 12). Call the upward incident wave amplitude U and the downward incident wave amplitude D'. Then the amplitudes of the resultant wave U' and D are

$$\left.\begin{aligned} U'&=c'D'+tU \\ D&=t'D'+cU \end{aligned}\right\} \tag{4.1.6}$$

Eliminating c', t' and t in terms of c from (4.1.1), (4.1.2) and (4.1.5), and then solving for U and D gives the relationship of the amplitude in one medium to that in the next.

$$\begin{pmatrix} U \\ D \end{pmatrix} = \frac{1}{1+c} \begin{pmatrix} 1 & c \\ c & 1 \end{pmatrix} \begin{pmatrix} U' \\ D' \end{pmatrix} \tag{4.1.7}$$

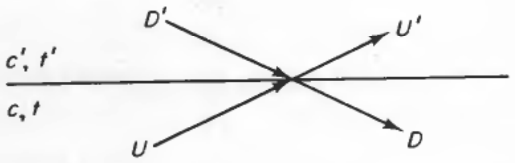

Figure 12. Two waves incident on the same boundary.

Now consider wave travel through a number of layers, with equal travel time in all layers. Let T be the two way travel time, and define

$$z = \exp(j\omega T) \tag{4.1.8}$$

Then multiplication by $\sqrt{z}$ is a delay of $T/2$. Within the kth layer (Fig. 13) the downward directed wave at the top of the layer is denoted D_k, and the upward wave is U_k. At the bottom of the kth layer these variables are primed, as shown. Because the downward directed wave is delayed by $T/2$ in travel through the medium, then

$$D'_k = D_k\sqrt{z} \tag{4.1.9}$$

and similarly for the upward wave

$$U_k = U'_k\sqrt{z} \tag{4.1.10}$$

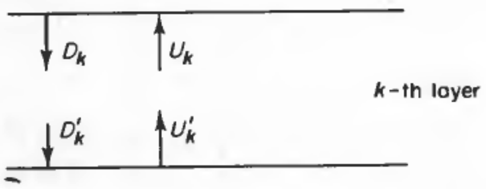

Figure 13. Reflection in the kth layer.

Combining (4.1.7), (4.1.9) and (4.1.10) gives the formula for the waves in one layered medium in terms of the waves in the next layered medium.

$$\begin{pmatrix} D_{k+1} \\ U_{k+1} \end{pmatrix} = \frac{\sqrt{z}}{(1+c_k)} \begin{pmatrix} 1 & c_k z^{-1} \\ c_k & z^{-1} \end{pmatrix} \begin{pmatrix} D_k \\ U_k \end{pmatrix} \tag{4.1.11}$$

Now compare this equation with the lattice order recursion depicted in Fig. 3. Calling the delay operator z^{-1}, from this figure it can be seen that

$$\begin{pmatrix} \epsilon_{N+1} \\ \zeta_{N+1} \end{pmatrix} = \begin{pmatrix} 1 & -k_N{}^{\epsilon} z^{-1} \\ -k_N{}^{\zeta} & z^{-1} \end{pmatrix} \begin{pmatrix} \epsilon_N \\ \zeta_N \end{pmatrix} \tag{4.1.12}$$

Therefore, if $k_N^\zeta = k_N^\epsilon$, this agrees with the previous equation except for the multiplicative scalar $\sqrt{z}/(1+c_k)$. Because no energy is gained in the reflection of a wave travelling through layered media, the process is passive and therefore stable. Of course the reflection coefficient $|c_k| \leqslant 1$, so that the stability condition for the lattice is $|k_N^\epsilon k_N^\zeta| < 1$. Due to the series connection of order recursions (4.1.12), the necessary and sufficient condition for input–output stability of the lattice recursions is that $|k_N^\epsilon(t) k_N^\zeta(t)| < 1$ for all t and N.

4.2. *System representation of the lattice*

Consider the lattice order recursion at a specific time instant t, with input $y(t)$ and output $\epsilon_n(t)$, the forward innovations of order n. The lattice for this situation is pictured in Fig. 14.

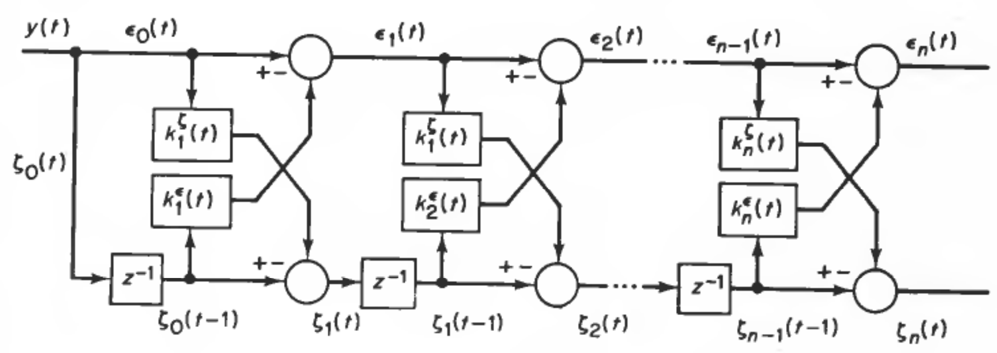

Figure 14. The lattice recursion in order.

The feed-forward nature of the lattice with n delays z^{-1} indicates that it is equivalent to an nth order moving average (MA) model, otherwise known as a tapped-delay line or finite impulse response model.

$$\epsilon_n(t) = c_0 y(t) + c_1 y(t-1) + \ldots + c_n y(t-n) \tag{4.2.1}$$

It is difficult to compute the values of the tapped delay coefficients $c_0, c_1, \ldots, c_n$ in terms of the reflection coefficients $k_1^\epsilon, k_1^\zeta, \ldots, k_n^\epsilon, k_n^\zeta$ for large n. Porat *et al.* (1982) present a method for updating the tapped delay coefficients in which the number of computational operations is proportional to n.

4.3. *AR, MA and ARMA representations*

The previous section described an MA representation (4.2.1) of the lattice, in which $y(t)$ is the input and $\epsilon_n(t)$ is the output. In AR process generation (§ 2.2) $\epsilon_n(t)$ is the input occurring as white noise and $y(t)$ is the output. The lattice recursions of § 3 occur only in the MA representation and are shown in Fig. 15.

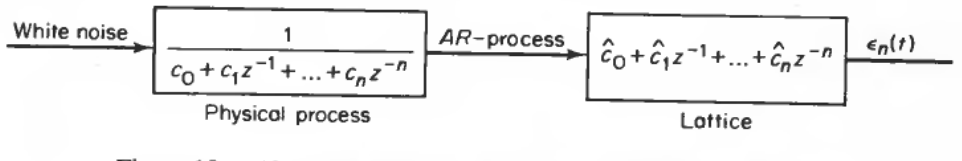
White noise
$\dfrac{1}{c_0 + c_1 z^{-1} + \ldots + c_n z^{-n}}$
Physical process
AR-process
$\hat{c}_0 + \hat{c}_1 z^{-1} + \ldots + \hat{c}_n z^{-n}$
Lattice
$\epsilon_n(t)$

Figure 15. *AR* processes are inputs to single data sequence lattices.

However, for the two data sequence lattices of eqn. (1.1.1), consider an x process that is MA in terms of y, the other process.

$$x(t)=c_0y(t)+c_1y(t-1)+\ldots+c_ny(t-n) \tag{4.3.1}$$

Neither x nor y need be white.

Fast $ARMA$ processes can be adequately approximated by MA processes. Consider the AR process x driven by white noise w as follows

$$x(t)-\alpha x(t-1)=w(t) \tag{4.3.2}$$

Then taking z transforms gives

$$x(z)=w(z)/(1-\alpha z^{-1}) \tag{4.3.3}$$

Expanding the fraction in z^{-1} gives

$$x(z)=(1+\alpha z^{-1}+\alpha^2z^{-2}+\ldots)w(z) \tag{4.3.4}$$

If $|\alpha|\ll 1$ this series can be truncated to obtain an adequate MA representation.

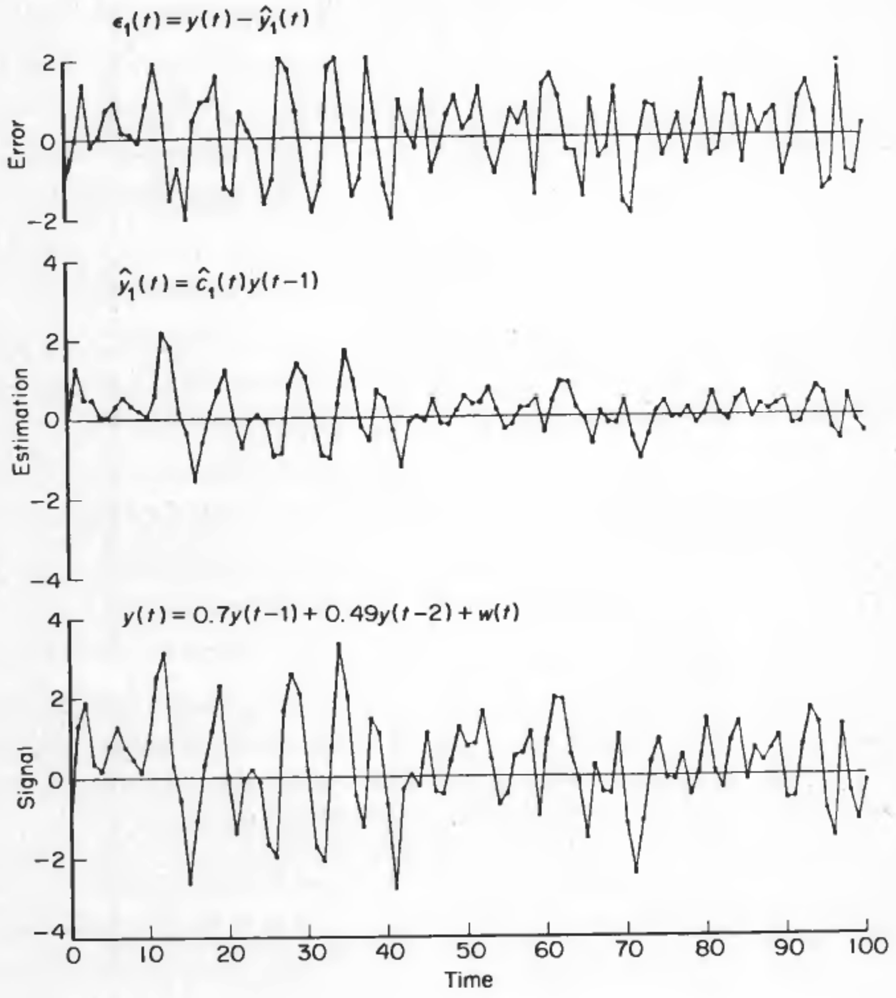

Figure 16. A first order lattice filtering a second order AR process.

5. Simulation results

This section gives the results of simulations of data sequences filtered by recursive linear least square algorithms in lattice form, as described in the previous sections.

5.1. *Simulation of one data sequence lattices*

A number of simulations were carried out involving a one data sequence lattice (whitening filter). Figure 16 shows a first order lattice attempting to whiten a second order AR process. The second order AR process that is the input to the lattice is shown at the bottom of the figure. The forward innovation $\epsilon_1(t)$ shown at the top is the output of the first order lattice. The first order estimate $\hat{y}_1(t)$, obtained by subtracting the top graph of $\epsilon_1(t)$ from the bottom graph of $y(t)$, is shown in the middle.

Figure 17 shows the same simulation as Fig. 16 except that a second order lattice was used in place of the first order lattice. The output of the second order lattice is the forward innovation $\epsilon_2(t)$ shown at the top of Fig. 17. The autocorrelation of this

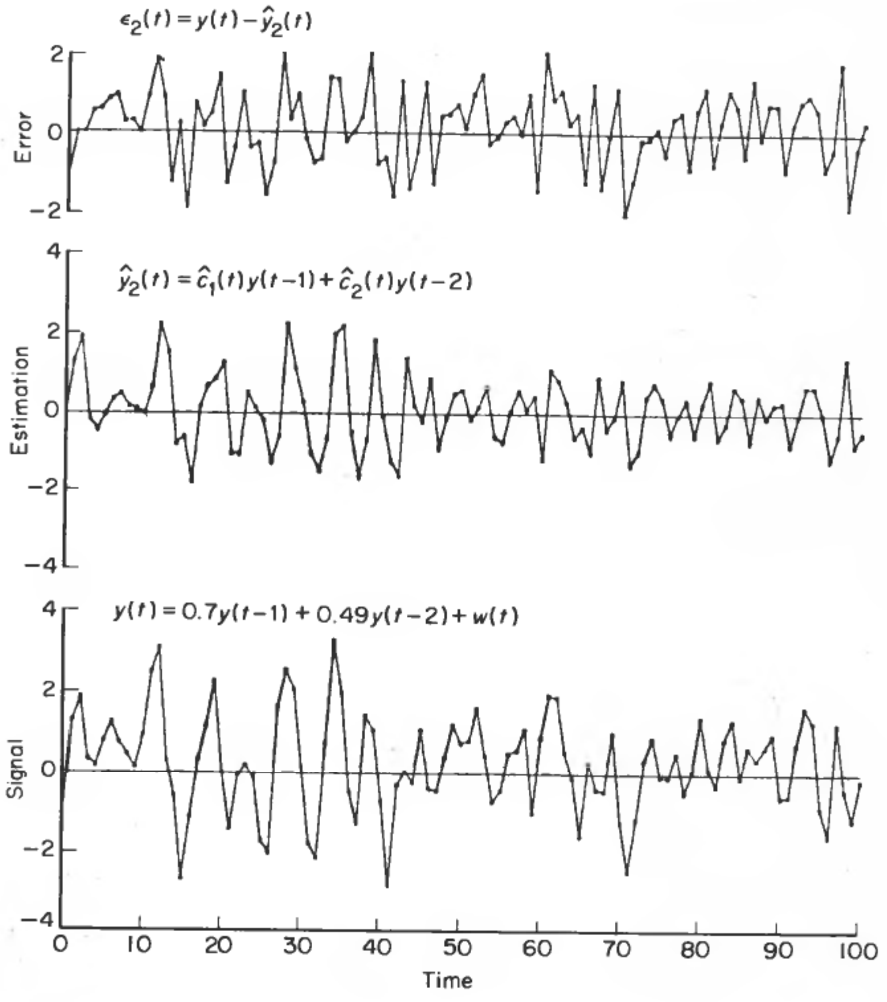

Figure 17. A second order lattice filtering a second order AR process.

$\epsilon_2(t)$ is compared with that of the first order forward residual $\epsilon_1(t)$ of Fig. 16 in Fig. 18. Notice that $\epsilon_2(t)$ appears white, whereas $\epsilon_1(t)$ is not white.

Since sine waves obey a second order *AR* relationship, they can be identified by a single process lattice of order greater than one. A sine wave of period 50 samples with a small additive white noise was generated as pictured at the bottom of Fig. 19. This generated data was the input to a second order single data sequence lattice, whose output was the forward innovation $\epsilon_2(t)$ pictured at the top of the figure. The estimated signal, $\hat{y}_2(t)$ shown in the middle of the figure, was obtained by subtracting $\epsilon_2(t)$ from $y(t)$.

A similar result was obtained for a square wave, as shown in Fig. 20. Since constants obey a first order *AR* relation, a first order lattice could have been used.

5.2. *Simulation of two data sequence lattices*

A number of simulations were carried out involving a two data sequence lattice (noise canceller). The bottom of Fig. 21 shows a slow sinusoidal signal plus a small noise as one data sequence $x(t)$, and the same small noise as the other data sequence $y(t)$. The sinusoidal signal's estimate $\hat{s}(t)$ by a second order lattice is shown at the top of the figure. Note that it takes slightly more than one sinusoisal period to

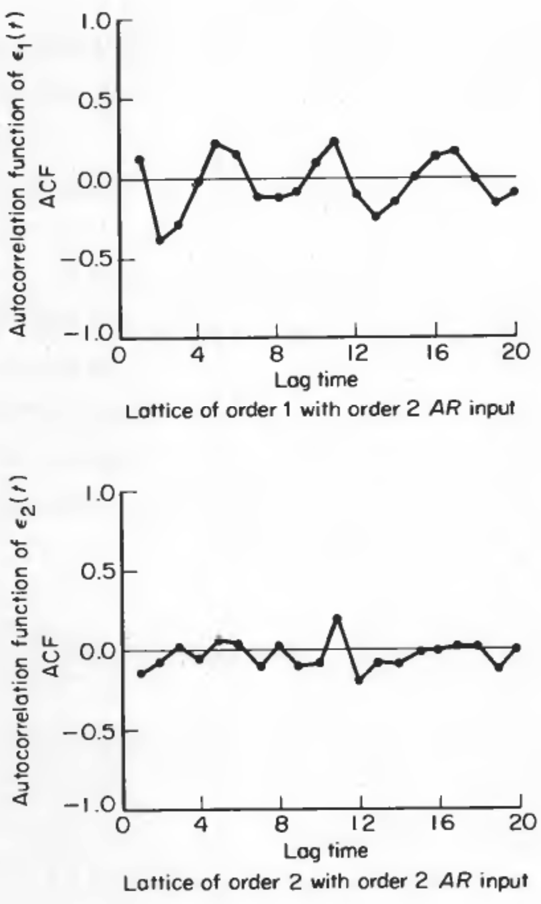

Figure 18. Comparison of the autocorrelation of the residuals of figures.

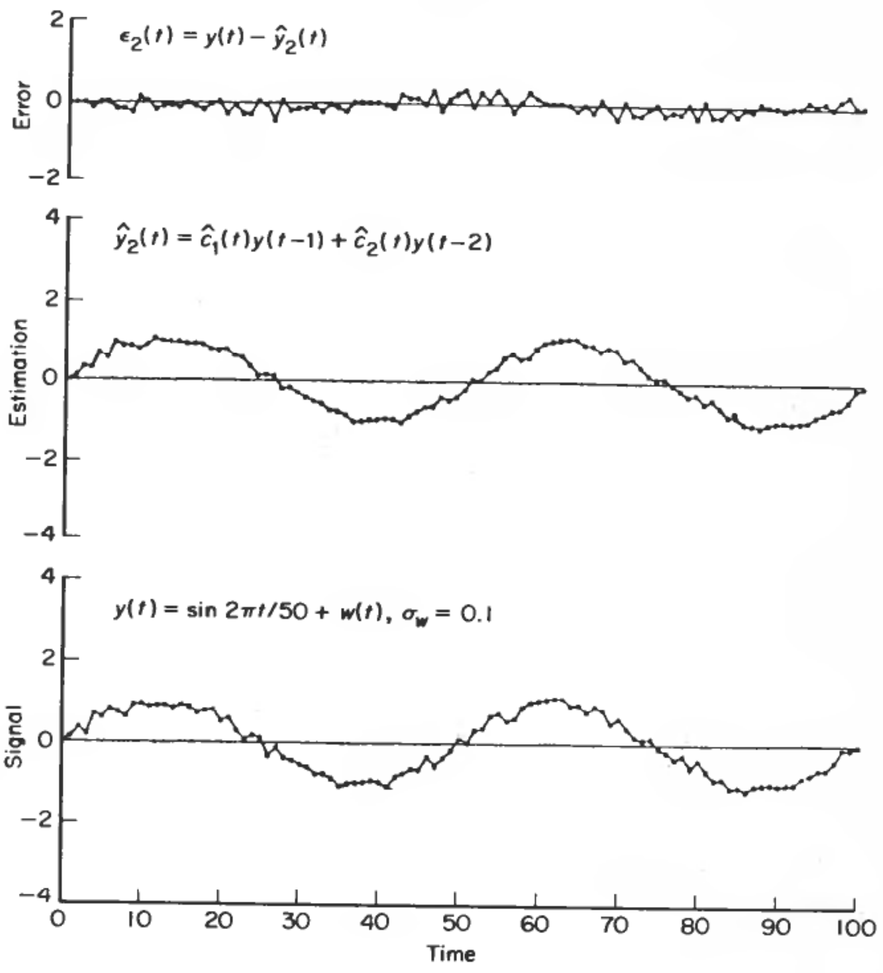

Figure 19. Sinusoidal input to a second order lattice.

recover the sine wave signal. In cases of low noise, the signal plus noise appears to be as accurate as the estimated signal over this first period of the sinusoid.

Figure 22 shows the same situation as Fig. 21, except in high noise. In this case the signal is almost completely masked by the noise, as can be seen at the bottom of Fig. 22. Since the noise is cancelled, the same signal estimate $\hat{s}(t)$ is produced as in Fig. 21.

Figure 23 shows a higher frequency sinusoid in the signal than Fig. 22. Again, it takes slightly more than one sinusoidal period to recover the sine wave signal. Because the frequency is higher, it appears that the noise canceller converges faster, but sinusoidal signal recovery appears to depend on frequency much more than absolute time.

Figures 24 to 31 simulate noise cancelling from a square wave signal. First order two data sequence lattices attempt to cancel the noise in Figs. 24 to 27. Figure 24 shows the cancellation in low noise, while Fig. 25 shows the same recovered signal in high noise. This is the same situation as in Figs. 21 and 22 except for a square wave rather than a sine wave. Figure 26 shows that the lattice is unable to recover

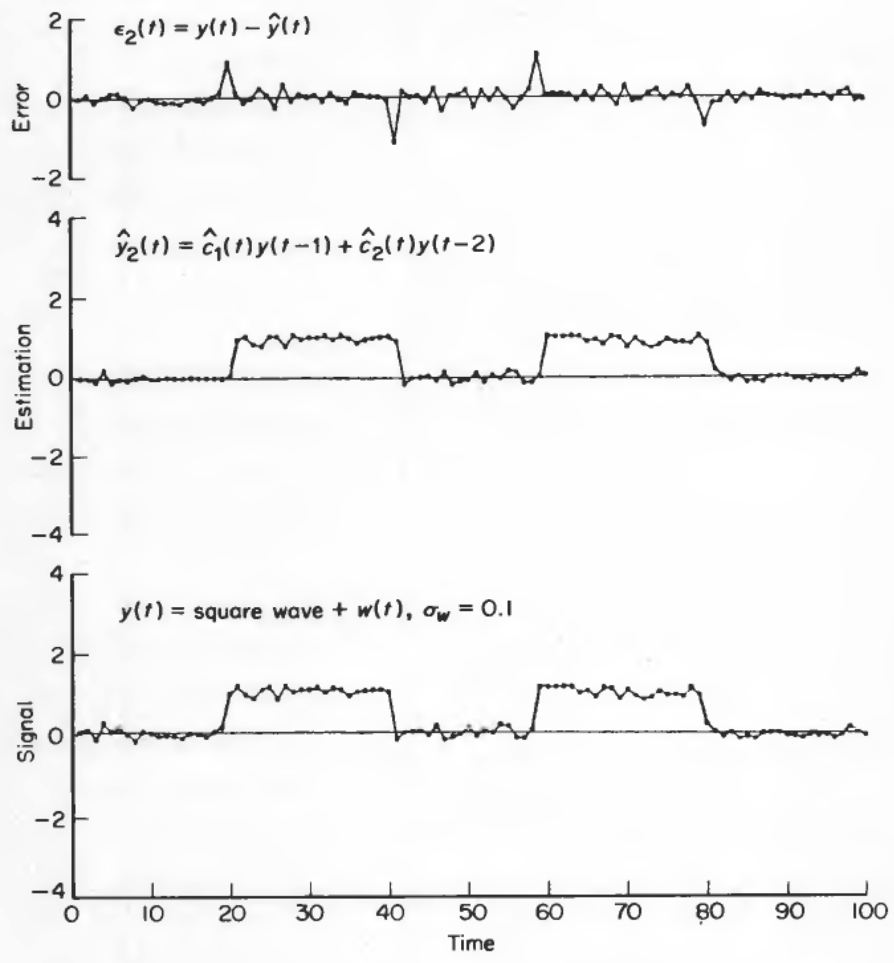

Figure 20. Square wave input to a second order lattice.

the signal when the noises in the two data sequence inputs are independent. Figures 24 to 26 give the same results as if one data sequence had merely been subtracted from the other. Figure 27 shows how a higher order lattice can be effective when the noise on the signal is a moving average of the other data sequence.

Figures 28, 29 30 and 31 show a sequence of increasing order lattices cancelling noise that is *ARMA* in the other data sequence. A poor estimate of the signal results from the second order lattice shown in Fig. 28, just as in the independent noise case of Fig. 26.

An increase in order to four gives some barely discernible steps in Fig. 29. These steps are rough but quite recognizable when the order is increased to six, as in Fig. 30. The tenth order lattice, depicted in Fig. 31, sharpens the square wave a little bit more than the sixth order lattice of Fig. 30.

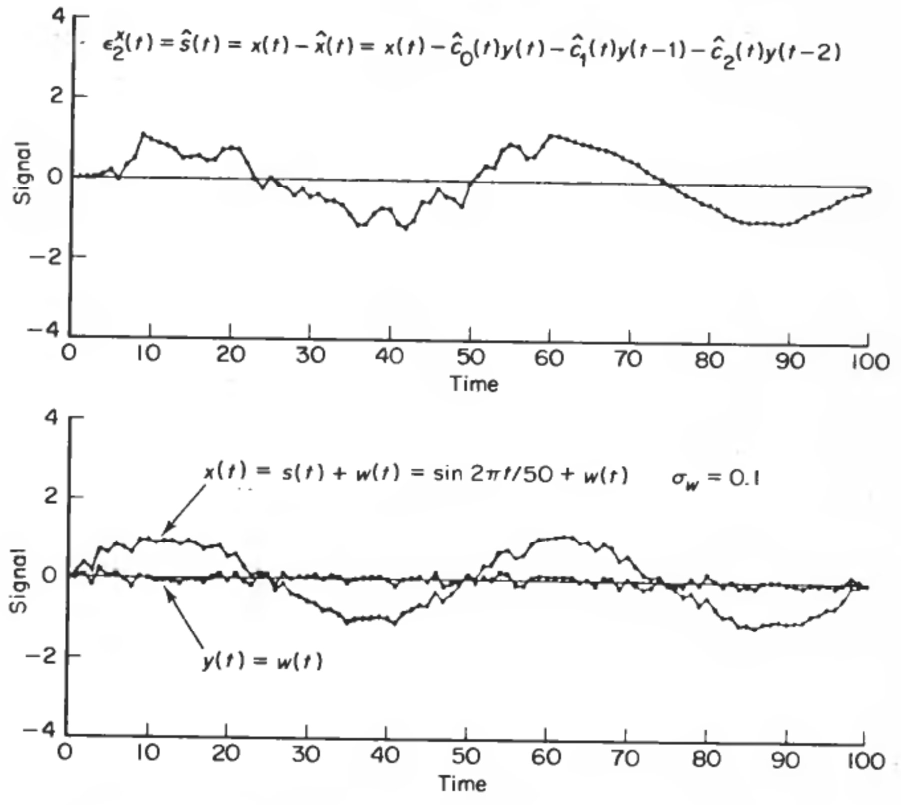

Figure 21. Noise cancellation for a slow sinusoidal signal in low noises.

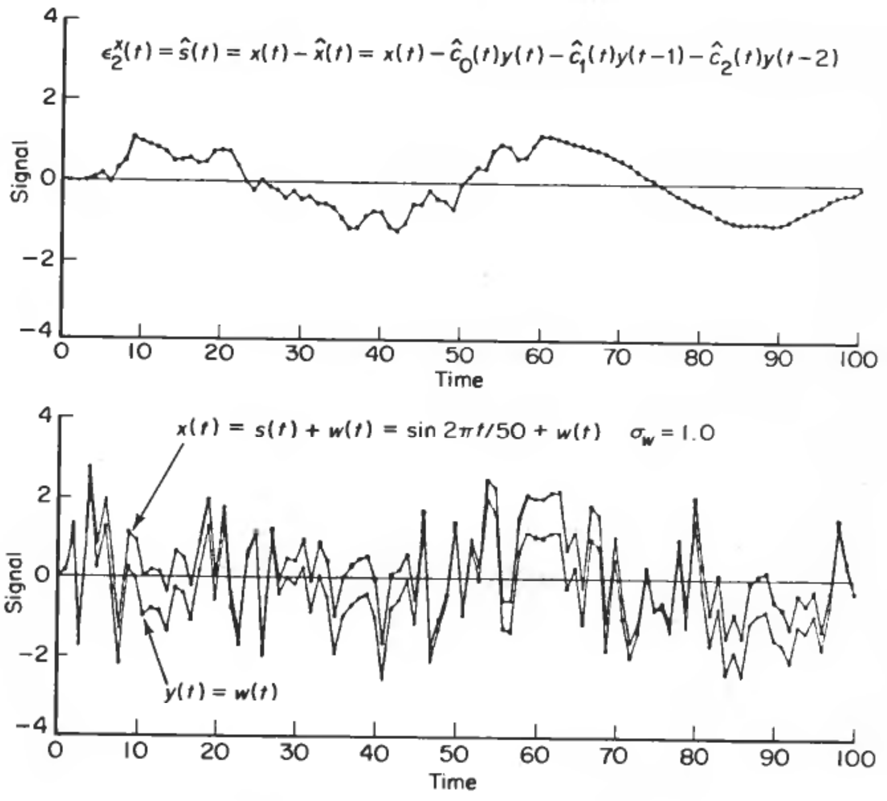

Figure 22. Noise cancellation for a slow sinusoidal signal in high noise.

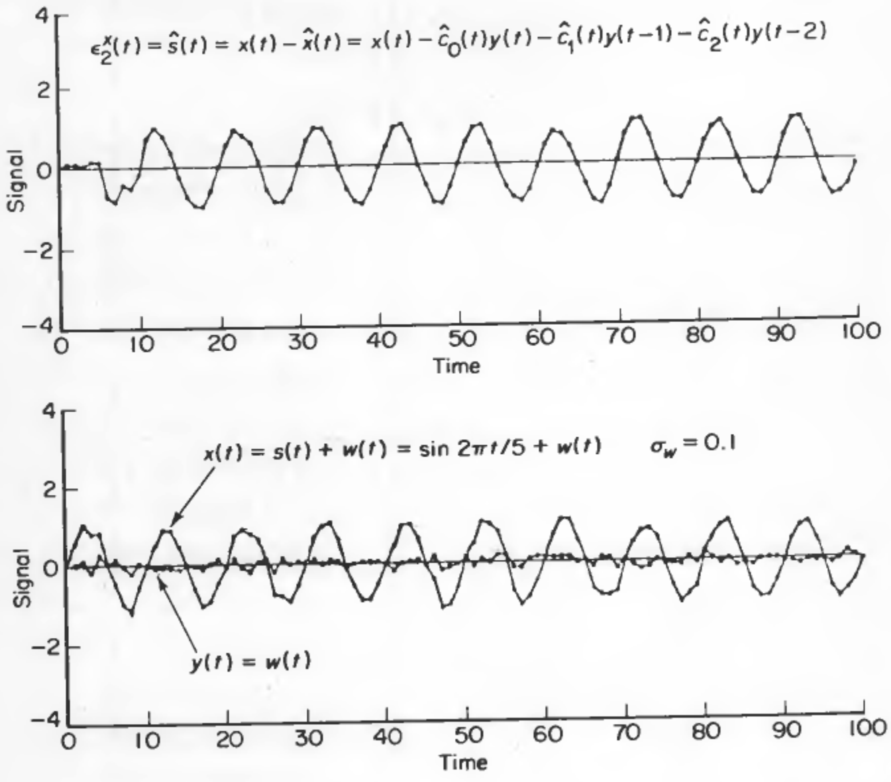

Figure 23. Noise cancellation for a fast sinusoidal signal in low noise.

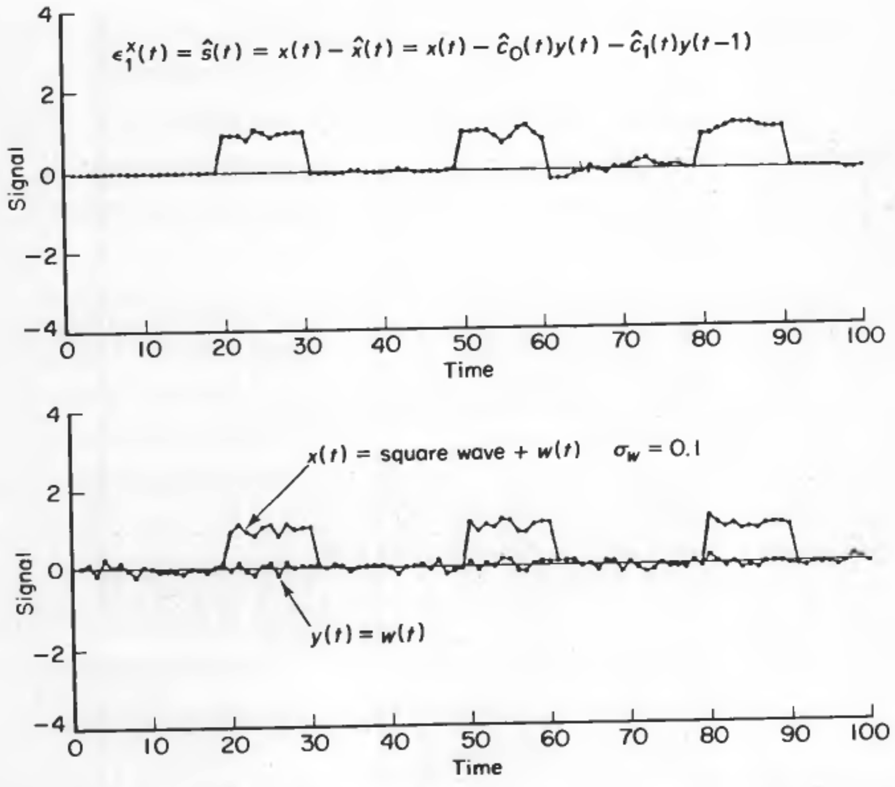

Figure 24. Noise cancellation for a square wave signal in low noise.

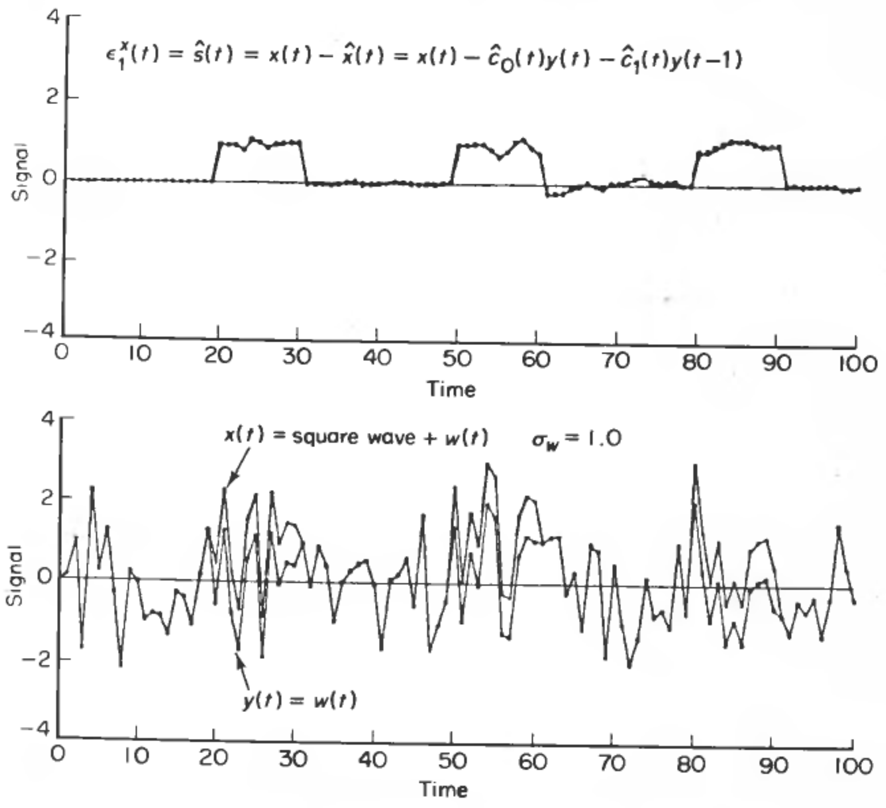

Figure 25. Noise cancellation for a square wave signal in high noises.

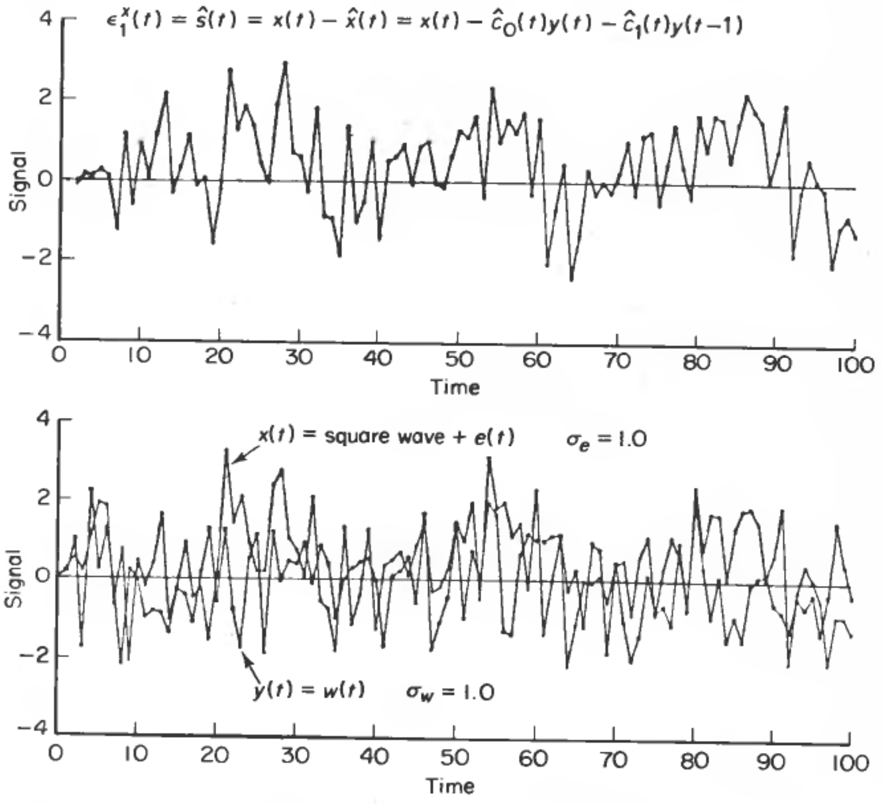

Figure 26. Noise cancellation for a square wave signal with uncorrelated high noise.

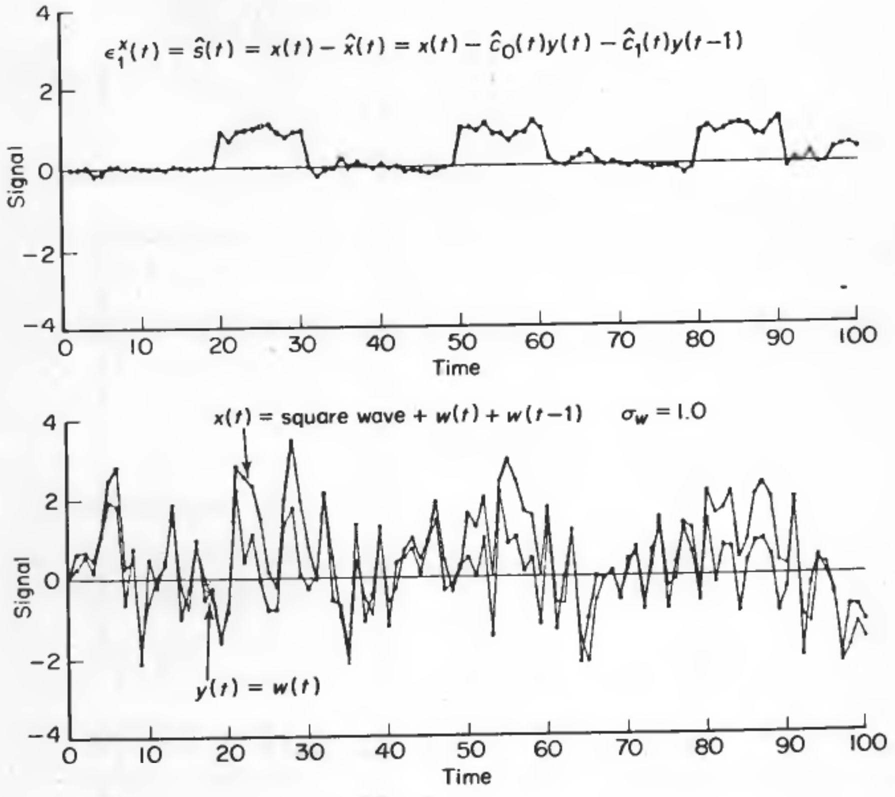

Figure 27. Noise cancellation for a square wave signal with *MA* correlated high noise.

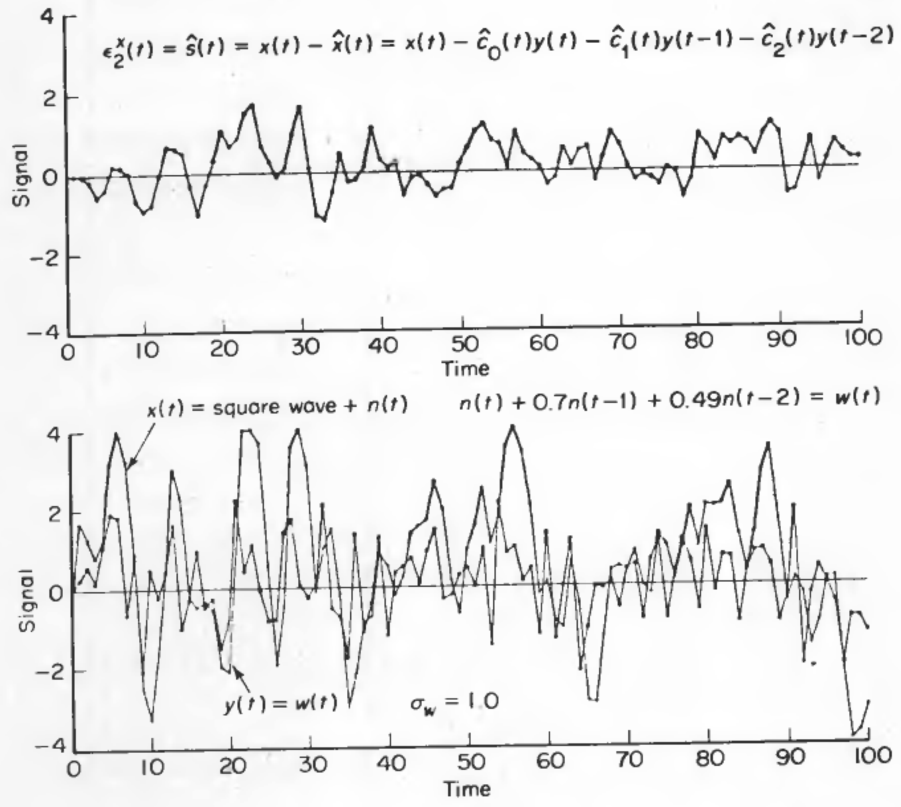

Figure 28. Noise cancellation of second order for a square wave signal with *ARMA* correlated high noise.

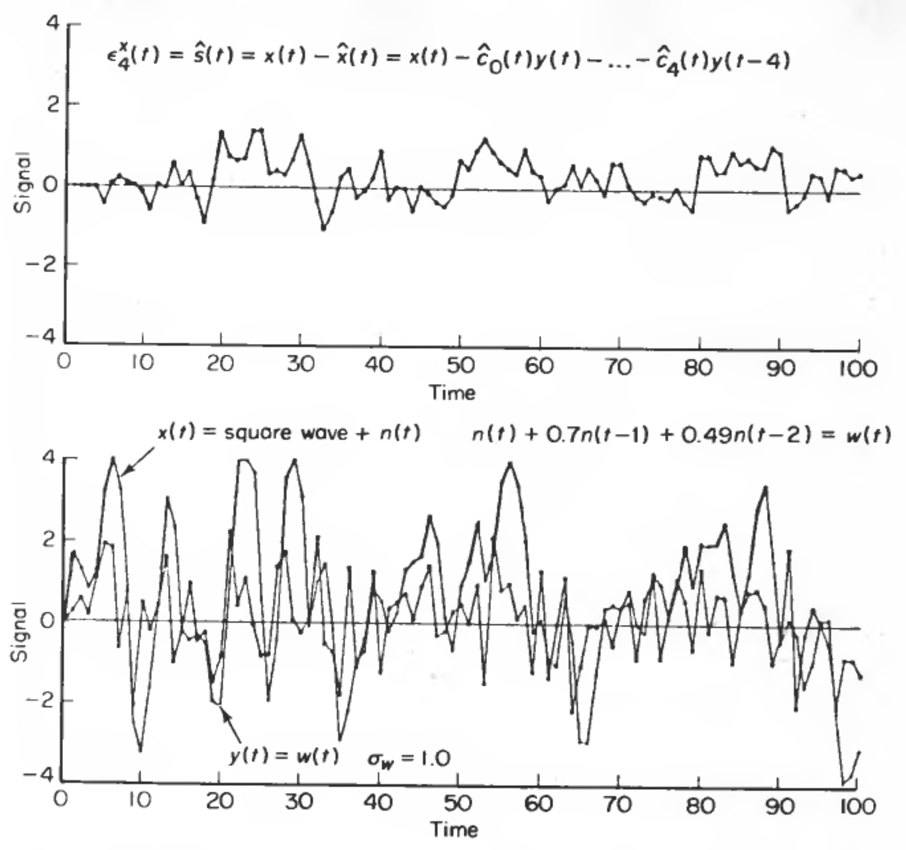

Figure 29. Noise cancellation of fourth order for a square wave signal with *ARMA* correlated high noise.

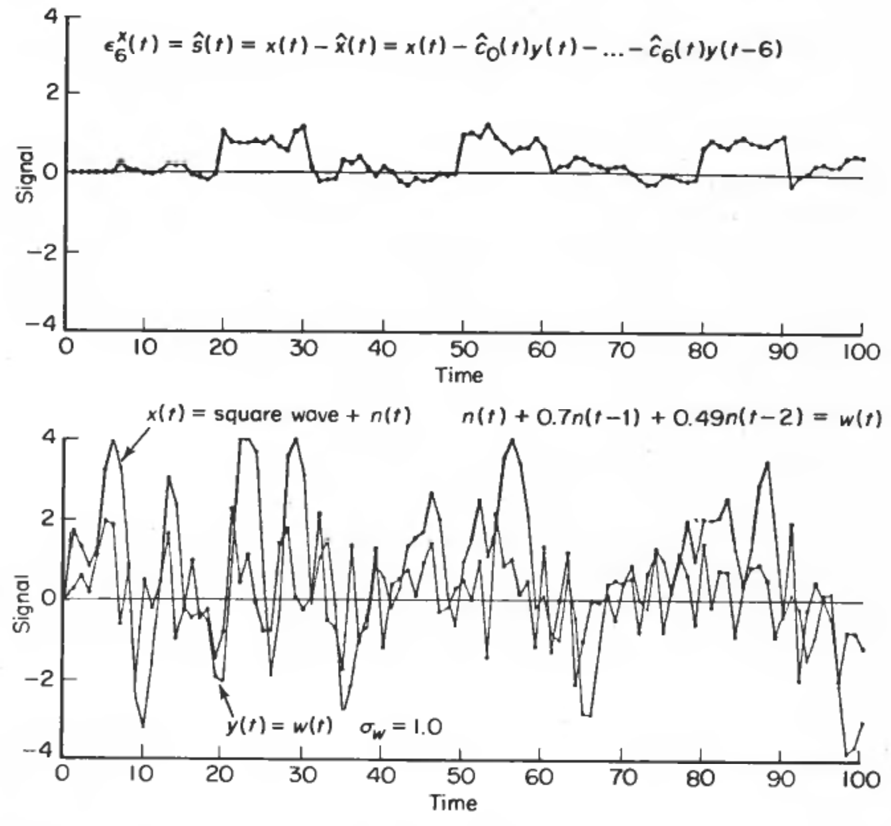

Figure 30. Noise cancellation of sixth order for a square wave signal with *ARMA* correlated high noise.

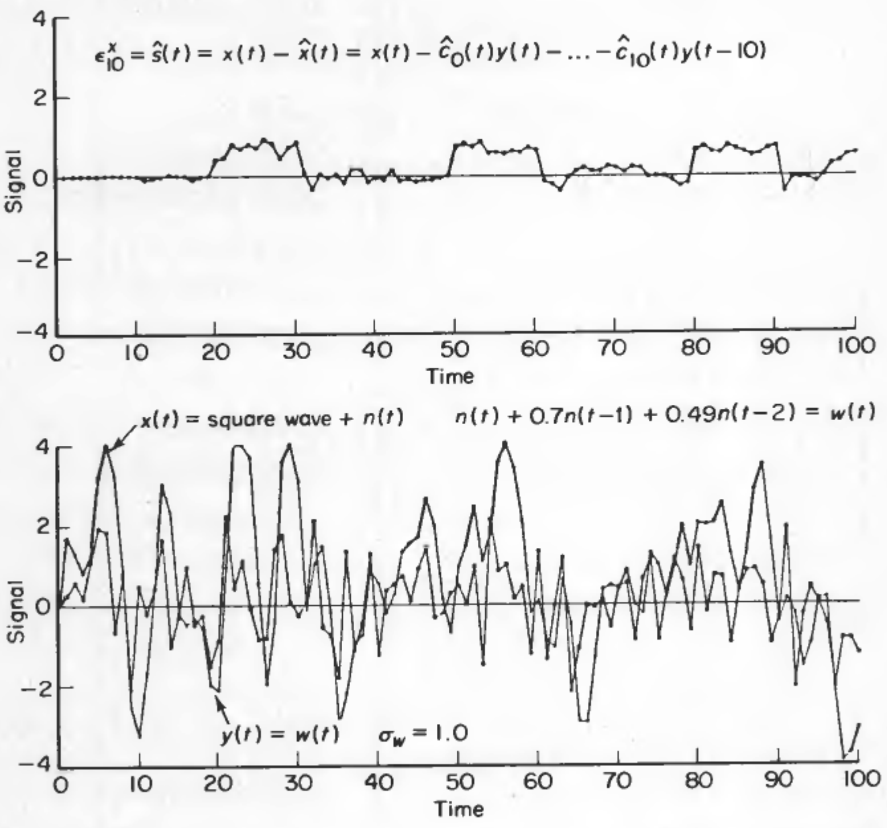

Figure 31. Noise cancellation of tenth order for a square wave signal with *ARMA* correlated high noise.

6. Conclusions

Linear least squares parameter estimation can be applied to a number of important industrial problems. Lattice algorithms permit the exact solution of these problems in cases where previously only approximate solutions could be used because of the size of the system. Very high order systems result from multiple-input, multiple-output problems, and lattices are especially useful in these cases. The difficulty with using approximate solutions has been that poor performance has resulted because of the approximation. The difficulty with the exact recursive least square algorithm used previously is that the computational burden and the resulting round-off error accumulation becomes too high for large systems, a shortcoming which is avoided by lattices. However, lattices can solve only the linear least-square problem, and are not suitable for general filtering applications.

Perhaps micro-computer chips implementing the lattice will be available in the near future. Until these are available, mini-computers can easily by programmed with the equations given in this tutorial to implement the lattice in practice.

References

Goodwin, G. C., and Payne, R. L. (1977). *Dynamic System Identification, Experimental Design and Data Analysis* (Academic Press, N.Y.).

LEV-ARI, H., and KAILATH, T. (1981). Schur and Levinson algorithms for nonstationary processes, *Proceedings of I.E.E.E. Conference on Decision and Control*, Dec. 1981, pp. 860–864.

ÅSTRÖM, K. J., and MAYNE, D. Q. (1982). A new algorithm for recursive estimation of controlled *ARMA* processes, *Proceedings of IFAC Conference on Identification and System Parameter Estimation*, Washington, D.C., June 1982, pp. 122–126.

BIERMAN, G. J. (1977). *Factorization Methods for Discrete Sequential Estimation* (Academic Press, N.Y.).

GAUSS, K. F. (1963). *Theoria Motus Corporum Coelestium*, 1809, translated as *Theory of The Motion of the Heavenly Bodies Moving About the Sun in Conic Sections* (Dover, N.Y.).

YULE, G. U. (1927). On a method of investigating periodicities in disturbed series, with special reference to Wolfer's sunspot numbers. *Phil. Trans.* A, **267**, 226.

LEVINSON, N. (1947). The Wiener RMS (root-mean-square) error criterion in filter design and prediction. *J. Math. Phys.*, **25**, 261–278.

KALMAN, R. E. (1960). A new approach to linear filtering and prediction problems. *J. Basic Engineering*, **82**, 342–345.

WIDROW, B., and HOFF, M., Jr. (1960). Adaptive switching circuits. *I.R.E. WESCON Conv. Record*, Pt. 4, pp. 96–104.

MORF, B., DICKINSON, B., KAILATH, T., and VIEIRA, A. (1977). Efficient solutions of covariance equations for linear prediction. *I.E.E.E. Trans. Acoustics, Speech, and Signal Processing*, **25**, 429–435.

LEE, D. T. L., MORF, M., and FRIEDLANDER, B. (1981). Recursive least squares ladder estimation algorithms. *I.E.E.E. Trans. Acoustics, Speech, and Signal Processing*, **29**, 627–641.

FRIEDLANDER, B. (1982). Lattice filters for adaptive processing. *Proc. I.E.E.E.*, **70**, 829–867.

FRIEDLANDER, B. (1982). Lattice methods for spectral estimation. *Proc. I.E.E.E.*, **70**, 990–1017.

SATORIUS, E. H., and PACK, J. D. (1980). A least squares adaptive lattice equalizer algorithm, *Naval Ocean Systems Center Technical Report* 575, Sept. 1980, San Diego, California 92152, U.S.A.

PORAT, B., FRIEDLANDER, B., and MORF, M. (1982). Square root covariance ladder algorithms. *I.E.E.E. Trans. autom. Control*, **27**, 813–829.

MEDITCH, J. S. (1969). *Stochastic Optimal Linear Estimation and Control* (McGraw-Hill, New York).

GILL, P. E., MURRAY, W., and SAUNDERS, M. A. (1975). Methods for computing and modifying the LDV factors of a matrix. *Mathematics of Computation*, **29**, 1051–1077.

MILLER, W., and WRATHALL, C. (1980). *Software for Roundoff Analysis of Matrix Algorithms* (Academic Press, New York).

LJUNG, L., and SODERSTROM, T. (1983). *Theory and Practice of Recursive Identification* (MIT Press, Cambridge, MA).

CLAERBOUT, J. F. (1976). *Fundamentals of Geophysical Data Processing* (McGraw-Hill, New York).

WIBERG, D. M., BASKIN, F., and LINDSAY, R. D. (1985). On the dynamics of recursive least squares and lattices, *7th IFAC Symposium on Identification and System Parameter Estimation*, York, United Kingdom, July 1985.

LJUNG, S., amd LJUNG, L. (1984). Error propagation properties of recursive least squares adaption algorithms, *9th IFAC Congress*, Colloquium on System Identification, Budapest, 1984. An extended version will appear in *Automatica*.

SAMSON, C., and REDDY, V. U. (1983). Fixed point error analysis of the normalized ladder algorithm. *I.E.E.E. Trans. Acoustics, Speech, and Signal Processing*, **31**, 1177–1191.

GRIFFITHS, L. J. (1977). A continuously adaptive filter implemented as a lattice structure, *Proc. I.E.E.E. Conf. Acoustics, Speech, Sig. Proc.*, Hartford, C.T., May 1977, pp. 683–686.

MAKHOUL, J. (1977). Stable and efficient lattice methods for linear prediction. *I.E.E.E. Trans. Acoustics, Speech, and Signal Processing*, **25**, 423–428.

WIBERG, D. M., BASKIN, F., and LINDSAY, R. D. (1983). Lattice form recursive linear least-square algorithms, especially for noise cancelling, *Aerospace Report No.* ATR–83(9975)–2, The Aerospace Corp., El Segundo, California 90009, U.S.A.