THE BELL SYSTEM TECHNICAL JOURNAL
Vol. 60, No. 8, October 1981
Printed in U.S.A.

Least-Squares Algorithms for Adaptive Equalizers

By M. S. MUELLER

(Manuscript received March 17, 1981)

Least-squares algorithms are the fastest converging algorithms for adaptive signal processors, such as adaptive equalizers. The Kalman, fast Kalman, and adaptive lattice algorithms using a least-squares cost function are investigated and extended to complex, fractionally spaced equalizers. It is shown that, for a typical telephone channel, these algorithms converge roughly three times as fast as the conventional stochastic-gradient technique. We analyze and compute the computational complexities and demonstrate that the fast Kalman algorithm is the most efficient in terms of overall performance.

I. INTRODUCTION

Adaptive channel equalization is a widespread technique used in most high-speed digital data modems. Generally, a transversal filter with adjustable coefficients is used as the equalizer. It can be adjusted adaptively to compensate for the undesired intersymbol interference introduced by the channel.

A large number of equalizer adjustment algorithms are conceivable, depending on the cost function. The currently prevailing technique is the so-called stochastic gradient algorithm. In the past years, three new rapidly converging algorithms were published, namely, the Kalman,[1] fast Kalman,[2] and adaptive lattice[3-7] algorithms. Here, we consider algorithms which minimize the sum-of-error-squares cost function. Because these least-squares algorithms make better use of all the past available information than the stochastic gradient algorithms, their start-up is faster.[8]

Originally the Kalman, fast Kalman, and adaptive lattice algorithms for equalizer update procedures were published for real-valued signals. In this paper, we present extensions of these algorithms to complex-valued signals which facilitate the analysis of quadrature-amplitude-

modulated (QAM) data transmission formats. We also extend the algorithms to include fractionally spaced equalizers.[9] An important characteristic of each algorithm is its computational complexity which we analyze for the least-squares, as well as for the stochastic gradient algorithms. Simulation results of the equalizer start-up using least-squares adjustment algorithms are presented for fractionally and symbol-spaced equalizers. Quadrature-amplitude-modulated and real-life voice-grade transmission channels are used for this study.

II. THE LEAST-SQUARES ALGORITHMS

In this section, we describe extensions of the Kalman, fast Kalman, and adaptive lattice adjustment algorithms for the coefficient adjustment of complex, fractionally spaced equalizers. It is assumed that equalizer output values are computed once for each symbol interval T, where T denotes the time interval between successive data values in the transmitter. The fractionally spaced equalizer is assumed to operate on T/p-spaced complex samples of the received signal.

Let $\xi(n)$ denote the complex p—dimensional vector of the new signal samples entering the fractionally spaced equalizer at time nT. Denote the M dimensional complex signal vector at time nT containing all signal samples over the past N time instances ($M = Np$) by

$$x(n)^* = [\xi(n)^*, \xi(n-1)^*, \cdots \xi(n-N+1)^*].\dagger \tag{1}$$

Then the output of the fractionally spaced equalizer is written as

$$y(n) = c(n-1)^* x(n), \tag{2}$$

where $c(n-1)$ is the M dimensional coefficient vector which was last updated at the previous time instant $n-1$. The desired data value at this instant is $d(n)$. Therefore, an output error

$$e(n) = d(n) - y(n) \tag{3}$$

results.

The objective of the least-squares algorithms is to determine the coefficient vector $c(n)$ which minimizes the weighted sum of all squared errors as if it were used over all the past received signal vectors, i.e., $c(n)$ minimizes

$$\sum_{k=0}^{n} \lambda^{n-k} |d(k) - c(n)^* x(k)|^2. \tag{4}$$

Setting the derivative of eq. (4) with respect to $c(n)$ to zero yields the discrete-time, Wiener-Hopf equation

† The * in eq. (1) denotes conjugate complex scalars and conjugate complex transposed vectors (matrices).

$$A(n)c(n) = v(n), \tag{5}$$

where

$$A(n) = \sum_{k=0}^{n} \lambda^{n-k} x(n)x(n)^* + \lambda^n \delta I_{MM} = \lambda A(n-1) + x(n)x(n)^*, \tag{6}$$

and

$$v(n) = \sum_{k=0}^{n} \lambda^{n-k} d(n)^* x(n) = \lambda v(n-1) + x(n)d(n)^*. \tag{7}$$

A small positive definite matrix δI_{MM} is included to ensure positive definiteness of $A(n)$ for all n. For $\lambda = 1$, $\delta = 0$ and large n, $1/n$ A(n) is an estimate of the channel correlation matrix, $1/n$ v(n) is an estimate of the cross correlation vector between the desired and the received signal. For $\lambda = 1$, all past information is weighted equally in calculating an updated coefficient vector; for $\lambda < 1$ the past is attenuated geometrically. Consequently, the present has a larger influence on the update than the past. This is a desired feature if time-varying channels are involved.

Since eqs. (6) and (7) can be written recursively, the updated coefficient vector can be calculated recursively as follows, cf. Appendix A:

$$c(n) = c(n-1) + g(n)e(n)^*, \tag{8}$$

where $g(n)$ is the Kalman gain defined as

$$g(n) = A(n)^{-1}x(n). \tag{9}$$

The Kalman, the fast Kalman, and the adaptive lattice algorithms all minimize the same cost function.[4] The difference is in the manner and the complexity with which it is achieved.

The remaining part of this section contains a brief discussion of the Kalman and the fast Kalman algorithms. The adaptive lattice algorithm is discussed in more detail; its derivation is given in Appendix A. Emphasis is placed on the signal transformation which is performed by the lattice structure. This signal transformation permits the evaluation of equalizers of increasing order in a computationally efficient way. The three algorithms are given in Appendix B in a form suitable for numerical evaluation.

2.1 The Kalman algorithm

The Kalman algorithm makes use of the recursive definition of $A(n)$ in eq. (6) and iteratively computes and stores its inverse $A(n)^{-1}$. The equalizer coefficient vector is then updated according to eqs. (8) and (9) at each iteration.

While the Kalman algorithm assures rapid equalizer start-up, it has the disadvantage of requiring matrix operations. Therefore, the number of calculations is proportional to M^2 and grows very fast with increasing M.

2.2 The fast Kalman algorithm

Ljung et al.[10] succeeded in formulating an equivalent algorithm with reduced complexity, where the number of operations is proportional to M. This algorithm was applied to the adaptive equalizer in Ref. 2. The algorithm exploits the fact that only p new signal samples enter the signal vector $x(n)$, p samples are discarded, and the remaining are just shifted. This is accomplished by means of $p \times M$ dimensional forward and backward predictors for the new and discarded values. Recurrence equations for these predictors and, finally, for the Kalman gain vector can be derived based on this. At most, $p \times M$ matrices have to be iterated—the chief reason for the reduced complexity.

2.3 The least-squares adaptive lattice algorithm

Recently the adaptive lattice algorithm for a least-squares cost function, originally published by Morf et al.,[11] was extended to equalizer update applications by Satorius and Pack[3] and Shichor.[4] Its application to the decision feedback equalizer is reported by Shensa.[6] Here, a further extension to the complex fractionally spaced equalizer is presented. A short form of this was published by Lim and Mueller.[7]

In the adaptive lattice structure, the equalizer coefficients operate on a transformed signal vector

$$\tilde{x}(n) = L(n-1)x(n), \tag{10}$$

where the transformation matrix is a lower triangular matrix formed by the backward prediction coefficients $c_m^b(n)$ of order m, $m = 1 \cdots N-1$, i.e.,

$$L(n) = \begin{bmatrix} I & 0 \cdots 0 \\ -c_1^b(n)^* & I\,0 \cdots \\ -c_2^b(n)^* & I\,0\,\cdot\cdot \\ \cdot & \cdots \\ \cdot & I\,0 \\ -c_{N-1}^b(n)^* & I \end{bmatrix}. \tag{11}$$

The backward predictor $c_m^b(n)$ of order m is a $mp \times p$ dimensional matrix satisfying

$$A(n)\begin{bmatrix} -c_m^b(n) \\ I \\ 0 \\ \cdot \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ \cdot \\ 0 \\ \epsilon^b(m, n) \\ \times \\ \cdot \\ \times \end{bmatrix}. \tag{12}$$

In eq. 12, $\epsilon^b(m, n)$ is the $p \times p$ dimensional backward prediction error residual of order m. For a detailed discussion of the backward predictor and the backward prediction error residual refer to Appendix A. In eqs. (11) and (12), I denotes a $p \times p$ dimensional identity matrix. The crosses in eq. (12) denote some unspecified elements which are of no further interest at this time. It follows from eqs. (11) and (12) that $A(n)L(n)^*$ is a lower block triangular matrix. Then it follows that $L(n)A(n)L(n)^*$ is lower block triangular, because it is the product of two lower block triangular matrices. On the other hand, $L(n)A(n)L(n)^*$ is Hermitian because $A(n)$ is Hermitian according to its definition in eq. (6). Therefore, $L(n)A(n)L(n)^*$ has to be a block diagonal matrix. Its diagonal element at the mth position is $\epsilon^b(m-1, n)$, i.e.,

$$L(n)A(n)L(n)^* = \text{diag}\,[\epsilon^b(m-1, n)]. \tag{13}$$

Hence, $L(n)$ diagonalizes $A(n)$.

We assume that all $\epsilon^b(m-1, n)$ are invertible and so we invert eq. (13). After premultiplying the result by $L(n)^*$ and postmultiplying it with $L(n)$ we obtain

$$A(n)^{-1} = L(n)^* \text{diag}\,[\epsilon^b(m-1, n)^{-1}]L(n). \tag{14}$$

Since it is desired that the adaptive lattice equalizer perform identically as the other least-squares equalizers, it follows that all the equalizer's output signals have to be equal, i.e.,

$$y(n) = c(n-1)^* x(n) = \tilde{c}(n-1)^* \tilde{x}(n). \tag{15}$$

From eqs. (10) and (15), it follows that the transformed coefficient vector $\tilde{c}(n)$ has to satisfy

$$\tilde{c}(n) = L(n)^{-*} c(n). \tag{16}$$

The matrix $L(n)^{-*}$, denotes the conjugate transposed inverse of $L(n)$. Upon substituting eqs. (5) and (14) into eq. (16), we obtain

$$\tilde{c}(n) = \text{diag}\,[\epsilon^b(m-1, n)^{-1}]L(n)v(n). \tag{17}$$

This suggests that the transformed coefficient vector is easily obtainable from the transformed correlation vector. The equalizer output of order N can be written as

$$y(n) = \sum_{m=1}^{N} z(m, n-1)^* \epsilon^b(m-1, n-1)^{-1} e^b(m-1, n), \qquad (18)$$

where we defined

$$\tilde{x}(n) = \begin{bmatrix} e^b(0, n) \\ e^b(1, n) \\ \cdot \\ \cdot \\ \cdot \\ e^b(N-1, n) \end{bmatrix}, \text{ and } L(n)v(n) = \begin{bmatrix} z(1, n) \\ z(2, n) \\ \cdot \\ \cdot \\ \cdot \\ z(N, n) \end{bmatrix}. \qquad (19)$$

Note that the elements of $\tilde{x}(n)$ are the backward prediction errors of order 0 to $N-1$. Therefore, the transformed equalizer operates on the backward prediction errors of order 0 to $N-1$.

Equations (10), (11), and (17) to (19) define the adaptive lattice equalizer as a transform of the ordinary transversal equalizer. An interesting property of this transform is due to the fact that $L(n)$ is a lower triangular matrix. This makes it possible to increase the dimension of the transformed signal vector and of the equalizer in a rather simple way, i.e., only one new p-vector is added to the vectors of order $m-1$ to form the vectors of order m. The already existing elements are unchanged. Accordingly, the equalizer output can be computed order-recursively in a very efficient way.

The time update algorithm of the lattice structure makes consequent use of the above-described order recursions. In addition to the backward predictor, the forward predictor and its error residual are iterated. Only the prediction errors and the prediction error residuals of order zero are updated in time. Then, using these elements as an anchor, the prediction errors and prediction error residuals of higher order are obtained recursively. It turns out that the predictions themselves are not needed. Finally, a time update of the elements $z(m, n)$ of the transformed vector $v(n)$ is obtained.

This scheme also makes use of all previously received data. Theoretically, its performance should be identical to the Kalman and the fast Kalman algorithms. Since there are no matrices involved, storage requirements and numbers of multiplications increase linearly with the equalizer length, though faster than in the fast Kalman algorithm.

A detailed derivation of the least-squares adaptive lattice equalizer algorithm is given in Appendix A. There, we adopt a notation which allowed us to describe the equalizer, the backward and the forward predictors as special cases of a general least-squares problem. In Appendix B, we list the adaptive lattice algorithm together with the two other least-squares algorithms in a form suitable for sequential execution.

III. COMPLEXITY

The number of multiplications (divisions are counted as multiplications) per iteration and the required precision are the dominant factors determining the complexity of real-time algorithms.

The effect of limited precision was investigated by T. L. Lim and the author in earlier work on that subject. There are no major differences between the three algorithms. With floating-point arithmetic, the requirements for the mantissa are from 11 to 12 bits for symbol-spaced equalizers and from 13 to 15 bits for $T/2$-spaced equalizers.

Table I gives the number of multiplications for the three least-squares algorithms and for the stochastic-gradient algorithm which is obtained when in eq. (8); $g(n)$, is replaced by a scalar. Results for both symbol- and $T/2$-spaced equalizers are given. The numbers for exponential weighting are included for the three least-squares algorithms.

The gradient algorithm requires the smallest number of multiplications, followed by the fast Kalman, the adaptive-lattice, and the Kalman algorithms. The gradient algorithm, of course, requires twice as many multiplications for the $T/2$ equalizer than for the symbol-spaced equalizer. For the Kalman algorithms, this factor is about four and for the adaptive-lattice, it is about five.

The fast Kalman algorithm has the lowest complexity of all least-squares algorithms; it requires about five times as many multiplications as the gradient algorithms for symbol-spaced equalizers and ten times as many for $T/2$-spaced equalizers. The adaptive-lattice algorithm requires more multiplications than the fast Kalman algorithm, especially for $T/2$-spaced equalizers. This is mainly because of the large number of matrix operations which is reflected in the large coefficient of p^3. However, it was pointed out in Refs. 3 and 4 that it offers a

Table I—Number of complex multiplications for equalizer spanning N symbol intervals with p samples per interval

		# Multiplications	$N = 31$ $p = 1$	$N = 31$ $p = 2$
Gradient		$2Np$	63	127
Kalman	$\lambda \neq 1$	$2N^2p^2 + 5Np$	2015	7998
	$\lambda = 1$	$Np(Np + 1)/2$ less than for $\lambda \neq 1$	1519	6045
Fast Kalman	$\lambda \neq 1$	$N(p^3 + 6p) + \frac{5}{3}p^3 + 2p^2 + \frac{4}{3}p$	316	1202
	$\lambda = 1$	$p(p + 1)/2$ less than for $\lambda \neq 1$	315	1199
Adaptive lattice	$\lambda \neq 1$	$N\left(\frac{13}{3}p^3 + 7p^2 + \frac{11}{3}p\right) - 4p^3 - 5p^2 - 2p$	454	2046
	$\lambda = 1$	$\left(N - \frac{1}{2}\right)p(p + 1)$ less than for $\lambda \neq 1$	393	1863

unique feature: the number of equalizer taps can be increased according to the actual need for the particular channel involved. Since for real-time applications the computational power for the longest required equalizer has to be provided, this cannot be regarded as an advantage and does not justify the considerably higher complexity for modem applications.

The Kalman algorithm requires the largest number of multiplications. Since it offers no additional features when compared with the fast Kalman algorithm, the latter is preferred for equalizer implementations.

IV. SIMULATED COMMUNICATION SYSTEM

Figure 1 shows the simulated system, where the transmitter is assumed to have a raised-cosine shaped transfer function with 12 percent excess bandwidth. Quadrature amplitude modulation with a symbol rate of 2400 baud and 2 bits per symbol is used. The carrier frequency is placed at 1700 Hz. The data symbols in the in-phase branch are taken from a binary pseudo random noise sequence (PRNS). The same sequence, shifted and reversed in time, is used in the quadrature branch.

Various channel transfer functions were considered. Figure 2 shows the transfer function of a channel which barely meets the requirements for basic conditioning of private lines. The eigenvalue spread of the autocorrelation matrix for symbol-spaced samples equals 9.8. The equivalent baseband impulse response of the combined transmitter and channel is used to generate the input data for the equalizer. Gaussian noise of specified power is added.

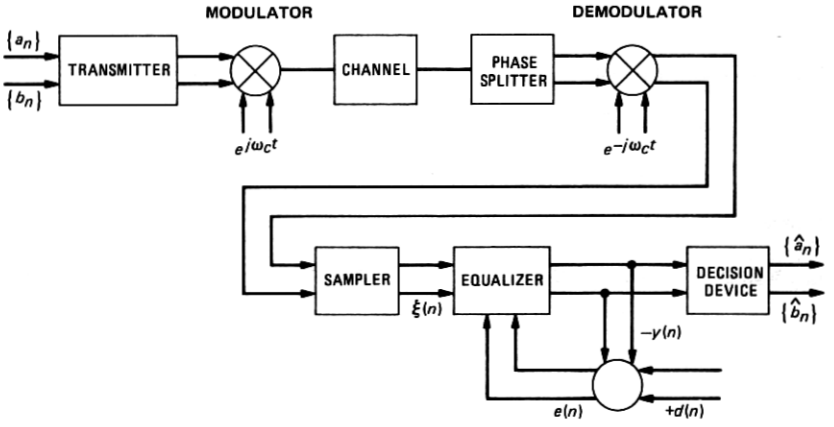

Fig. 1—Simulated transmission system.

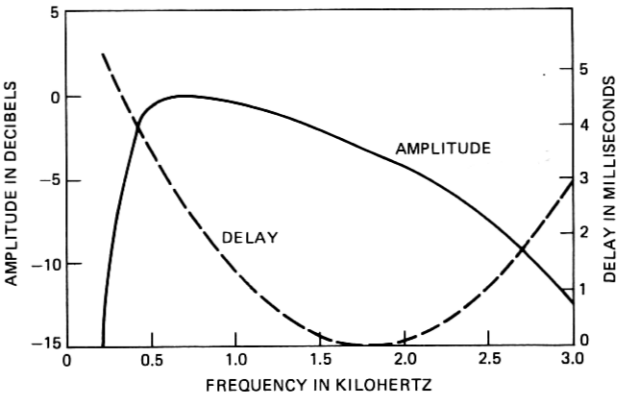

Fig. 2—Channel transfer function.

V. INITIAL CONVERGENCE

The initial convergence of the squared error at the output of the equalizer was determined for the Kalman, fast Kalman, and adaptive lattice equalizer structures. The behavior of the gradient algorithm, as well as of a fixed transversal filter with optimal coefficients were simulated for comparison purposes. Single precision floating-point arithmetic is used throughout, i.e., the mantissa is represented by 24 bits. The s/n is 25 dB and all equalizer coefficients are initially set to zero. A PRNS with a period of 127 symbols is used for the data symbols, and ten simulation runs with different starting points with respect to the PRNS are averaged. The resulting curve is smoothed with an exponential weighting factor of 0.9 to obtain the results shown in Fig. 3.

Figure 3 shows the results for the channel depicted in Fig. 2. The sampling phase is chosen to be 25 percent of a symbol interval away from the optimal sampling phase. Figure 3a corresponds to a 31-tap, symbol-spaced equalizer and Figure 3b to a $T/2$-spaced equalizer also spanning 31-symbol intervals. The behavior of the Kalman and fast Kalman algorithms has been observed to be identical [the difference in the output mean squared error (mse) is always smaller than 0.01 dB]. Therefore, the Kalman algorithm is not included on the plots.

The optimal fixed equalizer attains an output mse of 23.1. dB normalized to the signal level. For the symbol-spaced equalizer, about 125 iterations are required to converge to a normalized mse of 20 dB. For the $T/2$-spaced equalizer, all least-squares algorithms converge in about 150 iterations. The gradient algorithm requires about 400 iterations to converge to the same level. Very similar results were obtained for a channel with amplitude distortion as shown in Fig. 2 but with no

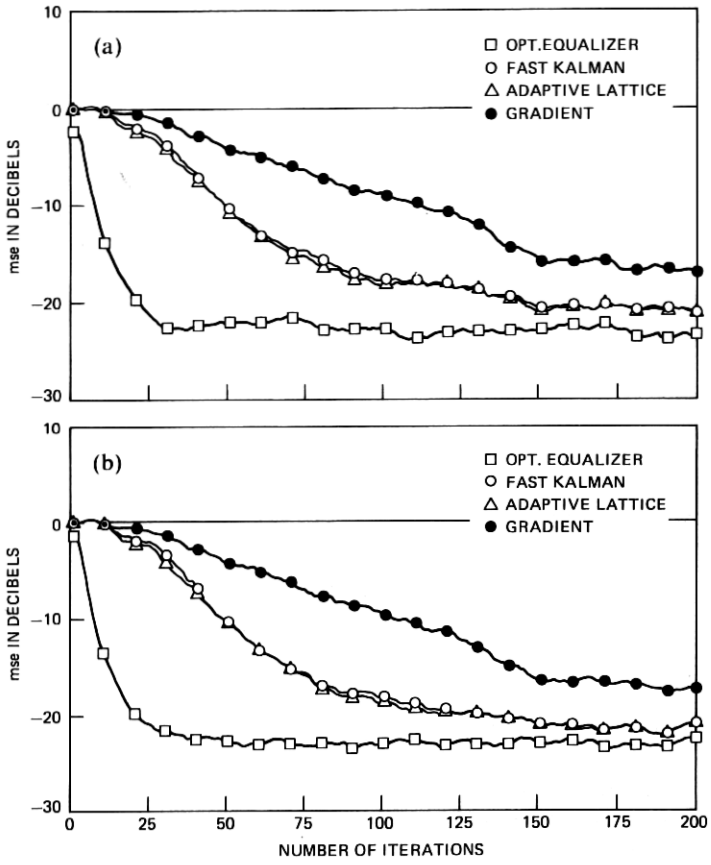

Fig. 3—(a) Convergence of symbol-spaced equalizer. $N = 31$, s/n = 25 dB. (b) Convergence of $T/2$-spaced equalizer. $N = 31$, $p = 2$, s/n = 25 dB.

phase distortion. If an ideal channel (no amplitude and no phase distortion) is used, the convergence time is reduced by about 35 percent.

These results indicate that, for realistic telephone channels, the least-squares algorithms behave very similarly and converge about three times faster than the stochastic gradient algorithm. Furthermore, it is found that the least-squares algorithms can be implemented successfully for both the symbol-spaced and $T/2$-spaced complex equalizers. Notice that the adaptive lattice algorithm requires a high number of matrix inversions per iteration if $p > 1$, which is often susceptible to numerical instabilities. However, our simulations did not uncover any stability problems.

The inclusion of an exponential weighting factor in the sum-of-squares cost function was proposed in Refs. 2 and 3 to allow for the

tracking of time varying parameters or channels. When this was included in our simulations, and single precision floating-point arithmetic was used, an unstable behavior of the fast Kalman algorithm resulted. Double precision arithmetic (i.e., 56-bits for the mantissa) was found to eliminate the instability. The Kalman and the adaptive-lattice algorithms did not show this instability.

VI. CONCLUSION

The Kalman, fast Kalman, and adaptive-lattice algorithms are the fastest known methods for the training of equalizers. In particular, it was found that they require only about a third as many iterations as the gradient algorithm to converge to within 3 dB of the optimal mse. For a $T/2$-spaced equalizer and a worst-case channel, the equalizer start-up requires about 150 iterations.

The fast Kalman algorithm possesses the lowest complexity of these schemes. It requires about ten times as many multiplications per iteration for a $T/2$-spaced equalizer as the stochastic-gradient algorithm.

The adaptive-lattice algorithm requires more multiplications per update but has the advantage of being able to increase the equalizer length adaptively when needed. This is an advantage for off-line or batch processing but not for real-time applications.

The Kalman algorithm possesses the highest complexity and offers no advantage over the two other schemes. Therefore, it is not recommended for implementation.

VII. ACKNOWLEDGMENT

I would like to thank T. L. Lim for his contributions on this subject.

APPENDIX A

Derivation of the Least-Squares Algorithms

Let $\xi(n)$ be a p-dimensional complex vector denoting the new elements in the pm dimensional signal vector $x_m(n)$

$$x_m(n) = [\xi(n)^*, \cdots \xi(n-m+1)^*]^*. \tag{20}$$

Let $c_m^s(n)$ be a complex pm dimensional coefficient vector, which denotes the equalizer coefficients if $s = e$, and let $c_m^s(n)$ be a complex $pm \times m$ dimensional matrix which stands for the forward predictor if $s = f$ and backward predictor if $s = b$.

Then the output of the equalizer, the forward and the backward predictors can generally be expressed as

$$y^s(m, n) = c_m^s(n-1)^* x_m(n). \tag{21}$$

The double argument (m, n) denotes order m and time n. The desired signal $d^s(m, n)$ is defined as

$$d^s(m, n) = \begin{cases} a(n - D) & \text{for } s = e \\ \xi(n + 1) & \text{for } s = f \\ \xi(n - m) & \text{for } s = b \end{cases}. \tag{22}$$

The transmission delay between transmitter and receiver is denoted by D. The equalization and prediction errors $e^s(m, n)$ are defined as

$$e^s(m, n) = d^s(m, n) - y^s(m, n). \tag{23}$$

For $s = e$, $y^s(m, n)$, $d^s(m, n)$ and $e^s(m, n)$ are complex scalars, for $s \neq e$ they are p–dimensional, complex vectors.

The equalizer and prediction coefficients are determined such that they minimize the trace of the following least-squares cost function

$$\sum_{k=0}^{n} \lambda^{n-k}[d^s(m, k) - c_m^s(n)^* x_m(k)][d^s(m, k) - c_m^s(n)^* x_m(k)]^*. \tag{24}$$

Lambda is a geometric weighting factor. Differentiating the above cost function with respect to $c_m^s(n)$ and equating the resulting expression to zero yields the discrete-time, Wiener-Hopf equation for the coefficients

$$A_m(n) c_m^s(n) = v_m^s(n), \tag{25}$$

where

$$A_m(n) = \sum_{k=0}^{n} \lambda^{n-k} x_m(k) x_m(k)^* = \lambda A_m(n - 1) + x_m(n) x_m(n)^* \tag{26}$$

$$v_m^s(n) = \sum_{k=0}^{n} \lambda^{n-k} x_m(k) d^s(m, k)^* = \lambda v_m^s(n - 1) + x_m(n) d^s(m, n)^*. \tag{27}$$

In eq. 27, $A_m(n)$ is an $mp \times mp$ dimensional, Hermitian matrix, and $v_m^s(n)$ is a $mp \times p$ dimensional complex matrix. They are equivalent to the autocorrelation matrix and the cross-correlation vectors which occur in the familiar mean-square approach.

The optimal value of the cost function is obtained when the solution resulting from eq. (25) is substituted into eq. (24)

$$\epsilon^s(m, n) = E^s(m, n) - v_m^s(n)^* c_m^s(n), \tag{28}$$

where

$$E^s(m, n) = \sum_{k=0}^{n} \lambda^{n-k} d^s(m, k) d^s(m, k)^* = \lambda E^s(m, n - 1) + d^s(m, n) d^s(m, n)^*. \tag{29}$$

For $s = e$, $\epsilon^s(m, n)$ is a scalar. For $s \neq e$, it is a $p \times p$ Hermitian matrix.

A.1 Time update recursions

We observe that eqs. (26) and (27) contain a recursive definition of $A_m(n)$ and $v_m^s(n)$. This allows us to obtain a recursion in time for the optimal coefficients. Upon combining eqs. (25) and (27) we have

$$A_m(n)c_m^s(n) = \lambda A_m(n-1)c_m^s(n-1) + x_m(n)d^s(m,n)^*. \quad (30)$$

Add and subtract $x_m(n)x_m(n)^*c_m^s(n-1)$ to the right-hand side of eq. (28), then use eq. (23), the recursive form of eq. (26), and premultiply both sides with $A_m(n)^{-1}$ to obtain the desired recursion

$$c_m^s(n) = c_m^s(n-1) + A_m(n)^{-1}x_m(n)e^s(m,n)^*. \quad (31)$$

To obtain time update recursions for various auxiliary variables, we consider $c_m^s(n)^*v_m^t(n)$, where s and $t \in \{e, b, f\}$. From the time update recursion for the coefficients eq. (31), we conclude that

$$c_m^s(n)^*v_m^t(n) = [c_m^s(n-1)^* + e^s(m,n)x_m(n)^*A_m(n)^{-1}]v_m^t(n). \quad (32)$$

We multiply out and use eq. (27) for the first term. In the second term, we apply eq. (25) and obtain

$$\begin{aligned} c_m^s(n)^*v_m^t(n) &= \lambda c_m^s(n-1)^*v_m^t(n-1) \\ &\quad + c_m^s(n-1)^*x_m(n)d^t(m,n)^* + e^s(m,n)x_m(n)^*c_m^t(n). \end{aligned} \quad (33)$$

Now add and subtract $d^s(m,n)d^t(m,n)^*$ and use eqs. (21) and (24) to obtain

$$\begin{aligned} c_m^s(n)^*v_m^t(n) &= \lambda c_m^s(n-1)^*v_m^t(n-1) \\ &\quad + d^s(m,n)d^t(m,n)e^s(m,n)[x_m(n)^*c_m^t(n) - d^t(m,n)]. \end{aligned} \quad (34)$$

From eq. (31) it follows that the error after updating the coefficients

$$e^t(m,n)^* \equiv -x_m(n)^*c_m^t(n) + d^t(m,n) = (1-\gamma(m,n))e^t(m,n)^*, \quad (35)$$

where we defined the real scalar

$$\gamma(m,n) = x_m(n)^*A_m(n)^{-1}x_m(n). \quad (36)$$

Finally, we obtain from eqs. (34) and (35)

$$\begin{aligned} c_m^s(n)^*v_m^t(n) &= \lambda c_m^s(n-1)^*v_m^t(n-1) + d^s(m,n)d^t(m,n) \\ &\quad - e^s(m,n)e^t(m,n)^*[1-\gamma(m,n)]. \end{aligned} \quad (37)$$

A time recursion for $\epsilon^s(m,n)$ can be obtained from eq. (28) by using eqs. (28), (29), and (37)

$$\epsilon^s(m,n) = \lambda\epsilon^s(m,n-1) + [1-\gamma(m,n)]e^s(m,n)e^s(m,n)^*. \quad (38)$$

The mth component of the transformed correlation vector is defined as, cf. eqs. (11) and (19)

$$z(m, n) = [-c^b_{m-1}(n)^*, I]v^e_m(n)$$
$$= -c^b_{m-1}(n)^* v^e_{m-1}(n) + \sum_{k=0}^{n} \lambda^{n-k}\xi(k+1-m)\,d^e(m, k)^*.$$

We note that $d^e(m, k) = d^e(m-1, k)$, and within apply eq. (37) and obtain the time recursion

$$z(m, n) = \lambda z(m, n-1)$$
$$+ [1 - \gamma(m-1, n)]e^b(m-1, n)\,e^e(m-1, n)^*. \quad (39)$$

For future use, we define the $p \times p$ matrix

$$k(m-1, n) = \sum_{k=1}^{n+1} \lambda^{n+1-k}\xi(k-m)\,\xi(k)^* - v^b_{m-1}(n)^* c^f_{m-1}(n).$$

We apply eq. (37) and observe that $d^f(m-1, n) = \xi(n+1)$ and that $d^b(m-1, n) = \xi(n-m+1)$. Thus, we obtain

$$k(m-1, n) = \lambda k(m-1, n-1)$$
$$+ [1 - \gamma(m-1, n)]e^b(m-1, n)\,e^f(m-1, n)^*. \quad (40)$$

A.2 Order update recursions

Observe that from eq. (20) it follows

$$x_{m+1}(k) = \begin{bmatrix} \xi(k) \\ \cdot \\ x_m(k-1) \\ \cdot \end{bmatrix} = \begin{bmatrix} \cdot \\ x_m(k) \\ \cdot \\ \xi(k-m) \end{bmatrix}. \quad (41a, b)$$

This relation is the order-time update equation for the signal vector. It allows the derivation of order-time update equations for the various coefficient vectors, for the error values and for the error residuals. Update equations for various auxiliary variables necessary for the algorithm are also derived.

From eq. (41) and the definitions eqs. (26) and (27) and under the condition that $x_m(0) = 0$ it follows

$$A_{m+1}(n+1) = \begin{bmatrix} E^f(m, n) & v^f_m(n)^* \\ v^f_m(n) & A_m(n) \end{bmatrix}$$
$$= \begin{bmatrix} A_m(n+1) & v^b_m(n+1) \\ v^b_m(n+1)^* & E^b(m, n+1) \end{bmatrix}. \quad (42a, b)$$

Upon combining eq. (26) and eq. (42a, b) we obtain the augmented Wiener-Hopf equations

$A_{m+1}(n+1)$

$$\left[\begin{array}{c|c} I & \\ \hline & -c_m^b(n+1) \\ -c_m^f(n) & \\ \hline & I \end{array}\right] = \left[\begin{array}{c|c} \epsilon^f(m,n) & 0 \\ 0 & \cdot \\ \cdot & \cdot \\ \cdot & \cdot \\ \cdot & \cdot \\ 0 & \epsilon^b(m,n+1) \end{array}\right], \qquad (43a, b)$$

where we used partitioned matrices to represent the two systems of equations for the forward and the backward predictors, having the same matrix of coefficients.

Similar equations can be derived for predictors of reduced order

$A_{m+1}(n+1)$

$$\left[\begin{array}{c|c} I & 0 \\ \hline & \\ -c_{m-1}^f(n) & -c_{m-1}^b(n) \\ \hline & \\ 0 & I \end{array}\right] = \left[\begin{array}{c|c} \epsilon^f(m-1,n) & k^b(m-1,n) \\ 0 & 0 \\ \cdot & \cdot \\ \cdot & \cdot \\ 0 & 0 \\ k^f(m-1,n) & \epsilon^b(m-1,n) \end{array}\right]. \qquad (44a, b)$$

Equation (44a) can easily be verified by applying the expansion of (42b) with reduced order and time indices to $A_m(n)$ in eq. (42a). The same method with the order reversed verifies eq. (44b).

The auxiliary variables $k^f(m-1,n)$ and $k^b(m-1,n)$ are defined as

$$k^f(m-1,n) = \sum_{k=0}^{n+1} \lambda^{n+1-k} \xi(k-m)\xi(k)^* - v_{m-1}^b(n)^* c_{m-1}^f(n) \qquad (45a)$$

$$k^b(m-1,n) = \sum_{k=0}^{n+1} \lambda^{n+1-k} \xi(k)\xi(k-m)^* - v_{m-1}^f(n)^* c_{m-1}^b(n). \qquad (45b)$$

From eq. (45a, b) and with the definition of the predictor coefficients from eq. (25), it is easily verified that

$$k^f(m-1,n) = k^b(m-1,n)^* \equiv k(m-1,n). \qquad (46)$$

The order update equations for the predictor coefficients and for the error residuals are now obtained through the combination of eqs. (43) and (44). We consider that

$$(43a) = (44a) - (44b)\epsilon^b(m-1,n)^{-1}k(m-1,n)$$

and

$$(43b) = (44b) - (44a)\epsilon^f(m-1,n)^{-1}k(m-1,n)^*.$$

This, with n reduced by one, yields an update equation for the prediction error residuals

$$\epsilon^f(m, n-1) = \epsilon^f(m-1, n-1) - k(m-1, n-1)^* \epsilon^b(m-1, n-1)^{-1} k(m-1, n-1) \quad (47a)$$

$$\epsilon^b(m, n) = \epsilon^b(m-1, n-1) - k(m-1, n-1) \epsilon^f(m-1, n-1)^{-1} k(m-1, n-1)^*. \quad (47b)$$

We assume now that $A_{m+1}(n+1)$ is nonsingular and premultiply eqs. (43) and (44) by its inverse. The same combinations of the new equations yield update equations for the coefficients.

$$c_m^f(n) = \left[\frac{c_{m-1}^f(n)}{0}\right] - \left[\frac{c_{m-1}^b(n)}{-I}\right] \cdot \epsilon^b(m-1, n)^{-1} k(m-1, n) \quad (48a)$$

$$c_m^b(n+1) = \left[\frac{0}{c_{m-1}^b(n)}\right] - \left[\frac{-I}{c_{m-1}^f(n)}\right] \cdot \epsilon^f(m-1, n)^{-1} k(m-1, n)^*. \quad (48b)$$

We premultiply eq. (48a) with $x_m(n+1)$ and eq. (48b) with $x_m(n+2)^*$. This yields eqs. (41), (21), and (23)

$$x_m(n+1)^* c_m^f(n) = x_{m-1}(n+1)^* c_{m-1}^f(n) - [x_{m-1}(n+1)^* \cdot c_{m-1}^b(n) - \xi(n-m+2)^*] \epsilon^b(m-1, n)^{-1} k(m-1, n) \quad (49a)$$

$$x_m(n+2)^* c_m^b(n+1) = x_{m-1}(n+1)^* c_{m-1}^b(n) - [x_{m-1}(n+1)^* \cdot c_{m-1}^f(n) - \xi(n+2)^*] \epsilon^f(m-1, n)^{-1} k(m-1, n)^*. \quad (49b)$$

With eq. (23) we identify the terms in the bracket of eq. (49a) as $e^b(m-1, n+1)^*$ and in the bracket of eq. (49b) as $e^f(m-1, n+1)^*$. We transpose eq. (49a), decrease n by 2, and use eq. (23) to obtain the update equations for $e^f(m, n-1)$. The update equation for $e^b(m, n)$ is obtained similarly.

$$e^f(m, n-1) = e^f(m-1, n-1) - k(m-1, n-2)^* \epsilon^b(m-1, n-2)^{-1} e^b(m-1, n-1) \quad (50a)$$

$$e^b(m, n) = e^b(m-1, n-1) - k(m, n-2) \epsilon^f(m, n-2)^{-1} e^f(m, n-1). \quad (50b)$$

The Kalman gain $g_{m+1}(n)$ is defined by

$$A_{m+1}(n) g_{m+1}(n) = x_{m+1}(n). \quad (51)$$

From eq. (40) we deduce

$$A_{m+1}(n)\left[\begin{array}{c|c} 0 & g_m(n) \\ \hline g_m(n-1) & 0 \end{array}\right]$$

$$= \left[\begin{array}{c|c} v_m^f(n-1)^* g_m(n-1) & x_m(n) \\ \hline x_m(n-1) & v_m^b(n)^* g_m(n) \end{array}\right]. \quad \text{(52a, b)}$$

We note from eq. (41) that eqs. (51), (52), and (43) are related as follows

$$(51) = (52a) + (43a)\epsilon^f(m, n-1)^{-1}[\xi(n) - v_m^f(n-1)^* g_m(n-1)]$$

$$(51) = (52b) + (43b)\epsilon^b(m, n)^{-1}[\xi(n-m) - v_m^b(n-1)^* g_m(n-1)].$$

We identify the terms in the brackets as the forward and backward prediction errors after updating the coefficients eq. (35). Performing the above-defined linear combinations of eqs. (43), (51), and (52) and premultiplying with $A_{m+1}(n)^{-1}$, yields order update equations for the Kalman gain.

$$g_{m+1}(n) = \left[\begin{array}{c} 0 \\ \hline g_m(n-1) \end{array}\right] + \left[\begin{array}{c} I \\ \hline -c_m^f(n-1) \end{array}\right] \cdot \epsilon^f(m, n-1)^{-1} \tilde{e}^f(m, n-1) \quad \text{(53a)}$$

$$g_{m+1}(n) = \left[\begin{array}{c} g_m(n) \\ \hline 0 \end{array}\right] + \left[\begin{array}{c} -c_m^b(n) \\ \hline I \end{array}\right] \epsilon^b(m, n)^{-1} \tilde{e}^b(m, n). \quad \text{(53b)}$$

We now proceed to obtain order update equations for $\gamma(m, n)$ as defined by eq. (36). Note from eqs. (36) and (51) that

$$\gamma(m, n) = x_m(n)^* g_m(n). \quad (54)$$

Upon using eqs. (41a), (53a), (41b), and (53b) respectively, we obtain

$$\gamma(m, n) = \gamma(m-1, n-1) + \tilde{e}^f(m-1, n-1)^* \cdot \epsilon^f(m-1, n-1)^{-1} \tilde{e}^f(m-1, n-1) \quad \text{(55a)}$$

$$\gamma(m, n) = \gamma(m-1, n) + \tilde{e}^b(m-1, n)^* \cdot \epsilon^b(m-1, n)^{-1} \tilde{e}^b(m-1, n). \quad \text{(55b)}$$

APPENDIX B

Least-Squares Equalizer Update Algorithms

Here the least-squares equalizer update algorithms are listed and ordered such that they can be evaluated in the given sequence. Emphasis is put on a simplified notation compared to Appendix A. Generally, capitals denote matrices and lower case letters denote scalars and vectors. Table II gives the correspondence of variables and their dimensions, where $M = Np$.

Table II—Correspondence of variables

Variable	Appendix A	Appendix B	Dimension
Signal vector	$x_m(n)$	$x(n)$	M
Correlation matrix	$A_m(n)$	$A(n)$	$M \times M$
Equalizer coefficients	$c_m^e(n)$	$c(n)$	M
Equalizer error	$e^e(m, n)$	$e(m, n)$	1
Forward prediction			
Coefficients	$c_m^f(n-1)$	$F(n)$	$M \times p$
Error	$e^f(m, n-1)$	$f(m, n)$	p
Error residual	$\epsilon^f(m, n-1)$	$E^f(m, n)$	$p \times p$
Backware prediction			
Coefficients	$c_m^b(n)$	$B(n)$	$M \times p$
Error	$e^b(m, n)$	$b(m, n)$	p
Error residual	$e^b(m, n)$	$E^b(m, n)$	$p \times p$
PARCOR coefficient	$k(m-1, n-1)$	$K(m, n)$	$p \times p$
Kalman gain	$g_m(n)$	$g(n)$	M

B.1 The Kalman algorithm

The Kalman algorithm makes use of the recursive definition of $A(n)$ in eq. (26) and the matrix inversion lemma, i.e.,

$$A(n)^{-1} = \frac{1}{\lambda}\left[A(n-1)^{-1} - \frac{A(n-1)^{-1}x(n)x(n)^*A(n-1)^{-1}}{\lambda + x(n)^*A(n-1)^{-1}x(n)}\right] \tag{56}$$

and defines

$$P(n) = A(n)^{-1}. \tag{57}$$

Upon using eqs. (2), (3), (8), (56), and (57) we obtain the Kalman algorithm for equalizer updating

$$t(n) = P(n-1)x(n), \tag{58}$$

$$g(n) = t(n)/(\lambda + x(n)^*t(n)), \tag{59}$$

$$P(n) = [P(n-1) - g(n)t(n)^*]\frac{1}{\lambda}, \tag{60}$$

$$y(n) = c(n-1)^*x(n), \tag{61}$$

$$e(n) = d(n) - y(n), \tag{62}$$

$$c(n) = c(n-1) + g(n)e(n)^*. \tag{63}$$

To initialize, set all variables to zero except $P(0)$ which is set to $P(0) = 1/\delta\, I$. Note that because of the Hermitian nature of $A(n)$ and consequently of $P(n) = A(n)^{-1}$, eq. (16) needs only be evaluated for the upper (or lower) triangle including the diagonal.

B.2 The fast Kalman algorithm

To obtain the fast Kalman algorithm, apply eqs. (21) to (23), (31), (35), and (38) for the forward predictor of fixed order m. This yields eqs. (64) to (67)

$$f(n) = \xi(n) - F(n-1)^* x(n-1) \tag{64}$$

$$F(n) = F(n-1) + g(n-1) f(n)^* \tag{65}$$

$$f(n)' = f(n)[1 - g(n-1)^* x(n-1)] \tag{66}$$

$$E(n) = \lambda E(n-1) + f(n)' f(n)^*. \tag{67}$$

Then use eq. (53a) to calculate the extended Kalman gain $\bar{g}(n)$ and partition as indicated

$$\bar{g}(n) = \begin{bmatrix} E(n)^{-1} f(n)' \\ \hline g(n-1) - F(n) E(n)^{-1} f(n)' \end{bmatrix} = \begin{bmatrix} g(n)' \\ \hline \mu(n) \end{bmatrix}. \tag{68) (69}$$

Now the backward prediction error $b(n)$ is calculated from eq. (23)

$$b(n) = \xi(n-N) - B(n-1)^* x(n). \tag{70}$$

Equations (31) and (53b) can now be used to update the backward predictor and finally to determine the updated Kalman gain

$$B(n) = [B(n-1) + g(n)' b(n)^*][I_{pp} - \mu(n) b(n)^*]^{-1} \tag{71}$$

$$g(n) = g(n)' + B(n)\mu(n). \tag{72}$$

Equations (21) to (23) and eq. (31) applied to the equalizer conclude the algorithm.

$$y(n) = c(n-1)^* x(n) \tag{73}$$

$$e(n) = d(n) - y(n) \tag{74}$$

$$c(n) = c(n-1) + g(n) e(n)^*. \tag{75}$$

To initialize, set

$$F(0) = B(0) = 0_{Mp}$$

$$x(0) = g(0) = c(0) = 0_M$$

and

$$E(0) = \delta I_{pp}.$$

Notice for the numerical evaluation that E is Hermitian and that the matrix inversions in eqs. (68) and (71) can be avoided if a p–dimensional system of linear equations is solved for multiple right-hand sides.

B.3 The lattice algorithm

For each time instant, the algorithm is initialized for order zero

$$y(0, n) = \gamma(0, n) = 0 \tag{76}$$

$$e(0, n) = d(n) \tag{77}$$

$$f(0, n) = b(0, n) = \xi(n) \tag{78}$$

$$E^f(0, n) = E^b(0, n) = \lambda E^f(0, n-1) + \xi(n)\xi(n)^*. \tag{79}$$

From eq. (40), the following time update equation follows

$$K(m, n) = \lambda K(m, n-1) + t(m, n-1) f(m-1, n)^*. \tag{80}$$

Then we obtain from eq. (50) order update equations for the prediction errors

$$f(m, n) = f(m-1, n) - G(m, n-1) b(m-1, n-1) \tag{81}$$

$$b(m, n) = b(m-1, n-1) - H(m, n-1) f(m-1, n), \tag{82}$$

where auxiliary $p \times p$ matrices are determined as

$$G(m, n) = K(m, n)^* E^b(m-1, n-1)^{-1} \tag{83}$$

$$H(m, n) = K(m, n) E^f(m-1, n)^{-1}. \tag{84}$$

Equation (48), together with eqs. (83) and (84), permits the update of the prediction error residuals

$$E^f(m, n) = E^f(m-1, n) - G(m, n) K(m, n) \tag{85}$$

$$E^b(m, n) = E^b(m-1, n-1) - H(m, n) K(m, n)^*. \tag{86}$$

The equalizer output and output error follow

$$y(m, n) = y(m-1, n) + z(m, n-1)^* \cdot E^b(m-1, n-1)^{-1} b(m-1, n) \tag{87}$$

$$e(m, n) = d(n) - y(m, n). \tag{88}$$

From eqs. (35) and (55), we have

$$t(m, n) = [1 - \gamma(m-1, n)] b(m-1, n) \tag{89}$$

$$\gamma(m, n) = \gamma(m-1, n) + t(m, n)^* E^b(m-1, n)^{-1} t(m, n). \tag{90}$$

Equation (39) finally allows to update the coefficients

$$z(m, n) = \lambda z(m, n-1) + t(m, n) e(m-1, n). \tag{91}$$

Equations (87), (88), and (91) are evaluated for $m \epsilon [1, N]$. The other equations are evaluated for $m \epsilon [1, N-1]$. To initialize, set all variables to zero except $E^f(0, 0) = E^b(0, 0) = \delta I_{pp}$.

For the numerical evaluation, it should be noted that $E^f(m, n)$ and $E^b(m, n)$ are Hermitian, thus, only the real diagonal and the upper or the lower triangle need be computed. Note also, that $G(m, n)$ and $H(m, n)$ are computed best as the solution of a p–dimensional system of linear equations with p right-hand sides.

REFERENCES

1. D. Godard, "Channel Equalization Using a Kalman Filter for Fast Data Transmission," IBM J. Res. Develop. (May 1974), pp. 267–73.
2. D. D. Falconer and L. Ljung, "Application of Fast Kalman Estimation to Adaptive Equalization," IEEE Trans. on Commun., *COM-26,* No. 10 (October 1978), pp. 1439–46.
3. E. H. Satorius and F. D. Pack, unpublished work.
4. E. Shichor, private communication.
5. D. D. Falconer, V. B. Lawrence, and S. K. Tewksbury, "Processor-Hardware Considerations for Adaptive Digital Filter Algorithms," Conf. Records of the ICC 1980, June 1980, Seattle, Washington, Paper No. 57.5.
6. M. J. Shensa, "A Least Squares Lattice Decision Feedback Equalizer," Conf. Records of the ICC 1980, June 1980, Seattle, Washington, Paper No. 57.6.
7. T. L. Lim and M. S. Mueller, "Rapid Equalizer Start-Up Using Least-Squares Algorithms," Conf. Records of the ICC 1980, June 1980, Seattle, Washington, Paper No. 57.7.
8. M. S. Mueller, private communication.
9. G. Ungerboeck, "Fractional Tap-Spacing Equalizers and Consequences for Clock Recovery for Data Modems," IEEE Trans. on Commun., *COM-24,* No. 8 (August 1976), pp. 856–64.
10. L. Ljung, M. Morf, and D. D. Falconer, "Fast Calculation of Gain Matrices for Recursive Estimation Schemes," Int. J. on Control (January 1978), pp. 1–19.
11. M. Morf, A. Vieira, and D. T. Lee, "Ladder Forms for Identification and Speech Processing," Proc. 1977 IEEE Conf. on Decision and Control, New Orleans, Louisiana, December 1977, pp. 1074–8.