An Extending Window MMSE Turbo Equalization Algorithm

Lihai Liu, *Student Member, IEEE,* and Li Ping, *Member, IEEE*

***Abstract*—We present a modified turbo minimum mean-squared error (MMSE) equalization algorithm using an extending window approach. We show that the new method can achieve nearly the same performance and considerable cost reduction compared with a recently proposed sliding window MMSE equalization technique.**

***Index Terms*—Inter-symbol interference, MMSE equalizer, turbo equalization.**

I. Introduction

TURBO equalization has been studied for digital communication systems with inter-symbol interference (ISI) channels [1]–[7]. A turbo equalizer consists of two basic processors: a soft-in-soft-out (SISO) channel equalizer and an *a posteriori* probability (APP) decoder. The two processors operate in an iterative manner. The optimal realization of the SISO channel equalizer is the maximum *a posteriori* probability (MAP) algorithm [1]–[3] that has a relatively high complexity. A low-cost alternative based on the minimum mean-squared error (MMSE) principle has been proposed in [4], further investigated in [5]–[7] and extended to other applications involving interference cancellation [8]. In particular, the work in [6], [7] represents a good trade-off between performance and complexity.

In this letter, we propose modifying the MMSE equalization developed in [6], [7] by replacing the sliding window with an extending window and we will describe a low-cost recursive technique to compute the soft-output estimates. We will show that considerable cost reduction can be achieved without compromising performance. For ease of discussion, we will adopt the system models in [6], [7] and mostly follow the notations there. Our emphasis is to derive a low-cost implementation technique based on the background work developed in [6].

II. Turbo Linear MMSE Equalization

Consider an equivalent time-invariant ISI channel with M tap-coefficients. The signal model can be represented by [6], [7] (1a), as shown at the bottom of the next page, or in a more compact form as

$$\boldsymbol{z}_n = \boldsymbol{H}\boldsymbol{x}_n + \boldsymbol{w}_n \tag{1b}$$

where $\boldsymbol{z}_n$ is the the channel observation, $\boldsymbol{x}_n$ the transmitted signal, $\boldsymbol{H}$ the channel convolution matrix and $\boldsymbol{w}_n$ an appropriate-sized vector of additive white Gaussian noise (AWGN) with variance $\sigma^2 = N_0/2$ per dimension. In the following, we will first consider BPSK signaling and real tap-coefficients. Later in Section IV we will generalize the principle to complex signaling and complex tap-coefficients. We will call N_2+N_1+1 (the length of $\boldsymbol{z}_n$) "window size".

A turbo equalizer consists of a SISO equalizer and an APP decoder, operating in an iterative manner. We consider the MMSE linear equalizer (LE) developed in [6], [7] for the SISO equalization task. Denote by $\{L(x_n)\}$ the log-likelihood ratios (LLRs) of $\{x_n\}$ fed back by the APP decoder; these are initialized to zero at the start of the iteration. The MMSE LE calculates the extrinsic output $L_E(x_n)$ for $x_n \in \{+1,-1\}$ based on $\{L(x_n)\}$ and the channel observation

$$\boldsymbol{z}_n \equiv [\, z_{n-N_2} \quad z_{n-N_2+1} \cdots z_{n+N_1} \,]^T \tag{2}$$

where N_1 and N_2 are, respectively, referred to as the noncausal and causal lengths of the filter. The mean and variance of x_n are estimated as

$$\bar{x}_n \equiv \mathrm{E}(x_n) = \tanh\left(\frac{L(x_n)}{2}\right), \quad v_n \equiv \mathrm{Var}(x_n) = 1 - \bar{x}_n^2.$$

Let $\mathbf{0}_{1\times i}$ denote a length-i zero row vector and $\boldsymbol{I}$ denote an identity matrix of an appropriate size. The output of the MMSE LE developed in [6] and [7] is given by

$$L_E(x_n) = \frac{2\boldsymbol{c}_n^T(\boldsymbol{z}_n - \boldsymbol{H}\bar{\boldsymbol{x}}_n + \bar{x}_n\boldsymbol{s}_n)}{1 - \boldsymbol{s}_n^T\boldsymbol{c}_n} \tag{3}$$

where

$$\begin{aligned}
\bar{\boldsymbol{x}}_n &\equiv [\, \bar{x}_{n-N_2-M+1} \quad \bar{x}_{n-N_2-M+2} \cdots \bar{x}_{n+N_1} \,]^T \\
\boldsymbol{V}_n &\equiv \mathrm{Diag}\,(\, v_{n-N_2-M+1} \quad v_{n-N_2-M+2} \cdots v_{n+N_1} \,) \\
\boldsymbol{s}_n &\equiv \boldsymbol{H}\,[\, \mathbf{0}_{1\times(N_2+M-1)} \quad 1 \quad \mathbf{0}_{1\times N_1} \,]^T \\
\boldsymbol{c}_n &\equiv (\sigma^2\boldsymbol{I} + \boldsymbol{H}\boldsymbol{V}_n\boldsymbol{H}^T + (1-v_n)\boldsymbol{s}_n\boldsymbol{s}_n^T)^{-1}\boldsymbol{s}_n.
\end{aligned}$$

A recursive procedure is developed in [7] to evaluate (3) based on the sliding observation window defined in (2) with a fixed size of $N_1 + N_2 + 1$. In the next section, we will consider an alternative method using a progressively extending observation window. We will show that the new method leads to significant complexity reduction without compromising performance.

Manuscript received November 7, 2003; revised February 26, 2004. This work was supported by a grant from the Research Grant Council of the Hong Kong Special Administrative Region, China, under Project CityU 101/01. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Visa Koivunen.

The authors are with the Department of Electronic Engineering, City University of Hong Kong, Hong Kong (e-mail: eeliping@cityu.edu.hk).

Digital Object Identifier 10.1109/LSP.2004.835461

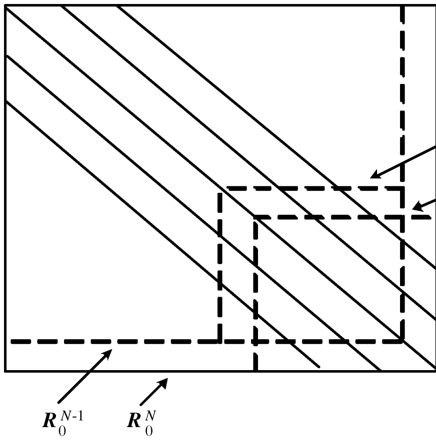

Fig. 1. Relationship between $\boldsymbol{R}_{n-1}$, $\boldsymbol{R}_n$, $\boldsymbol{R}_0^{N-1}$, and $\boldsymbol{R}_0^N$.

III. The Extending Window MMSE Equalization Algorithm

A. The Extending Window Equalizer

We now take a strategy different from that of [7]. Without loss of generality, assume that transmission starts from time 0. Instead of using a window with fixed size $N_2 + N_1 + 1$ as in [7], we will adopt a window from 0 until N with variable size $N + 1$, where $N = n + N_1$. In this case, the signal model in (1) can be rewritten in the equivalent form

$$\begin{bmatrix} z_0 \\ z_1 \\ \vdots \\ z_N \end{bmatrix} = \begin{bmatrix} h_0 & & & & & \\ \vdots & h_0 & & & \mathbf{0} & \\ h_{M-2} & & \ddots & & & \\ h_{M-1} & h_{M-2} & & \ddots & & \\ & h_{M-1} & \ddots & & \ddots & \\ \mathbf{0} & & \ddots & \ddots & & \ddots \\ & & & h_{M-1} & h_{M-2} & \cdots & h_0 \end{bmatrix} \times \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_N \end{bmatrix} + \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_N \end{bmatrix} \tag{4a}$$

or more compactly

$$\boldsymbol{z}_0^N = \boldsymbol{H}_0^N \boldsymbol{x}_0^N + \boldsymbol{w}_0^N \tag{4b}$$

where we have assumed that $x_{-M+1}, x_{-M+2}, \ldots, x_{-2}, x_{-1}$ are zero.

Now return to (3). Using the matrix inversion lemma [9], we rewrite it in an equivalent form as

$$L_E(x_n) = 2 \cdot \frac{\boldsymbol{s}_n^T \boldsymbol{R}_n^{-1}(\boldsymbol{z}_n - \boldsymbol{H}\bar{\boldsymbol{x}}_n) + \bar{x}_n \boldsymbol{s}_n^T \boldsymbol{R}_n^{-1} \boldsymbol{s}_n}{1 - v_n \boldsymbol{s}_n^T \boldsymbol{R}_n^{-1} \boldsymbol{s}_n} \tag{5a}$$

where $\boldsymbol{R}_n \equiv \sigma^2 \boldsymbol{I} + \boldsymbol{H}\boldsymbol{V}_n \boldsymbol{H}^T$. Equation (5a) can be used to derive the sliding window algorithm [7]. Here we take an alternative approach. Replacing the variables in (5a) by their respective counterparts in (4), we have

$$L_E(x_n) = 2 \cdot \frac{(\boldsymbol{s}_n^N)^T (\boldsymbol{R}_0^N)^{-1} (\boldsymbol{z}_0^N - \boldsymbol{H}_0^N \bar{\boldsymbol{x}}_0^N) + \bar{x}_n (\boldsymbol{s}_n^N)^T (\boldsymbol{R}_0^N)^{-1} \boldsymbol{s}_n^N}{1 - v_n (\boldsymbol{s}_n^N)^T (\boldsymbol{R}_0^N)^{-1} \boldsymbol{s}_n^N} \tag{5b}$$

where $\boldsymbol{R}_0^N \equiv \sigma^2 \boldsymbol{I} + \boldsymbol{H}_0^N \boldsymbol{V}_0^N (\boldsymbol{H}_0^N)^T$ and $\boldsymbol{s}_n^N \equiv \boldsymbol{H}_0^N [\mathbf{0}_{1\times n} \;\; 1 \;\; \mathbf{0}_{1\times N_1}]^T$. The extending window method based on (4b) and (5b) has both performance and complexity advantages over the sliding window method. The performance advantage results from the fact that (5b) is based on a larger observation set than (5a) (since $\boldsymbol{z}_n$ in (1b) is a sub-block of $\boldsymbol{z}_0^N$ in (4b)). However, the performance difference is marginal provided that N_1 and N_2 are sufficiently large (see below). Empirically, we observed gradual performance degradation when $N_1 < 2M$ for both methods or when $N_2 < M$ for the sliding window method. Provided that $N_1 \geq 2M$ and $N_2 \geq M$, the performance of both methods remains roughly the same for different choices of N_1 and N_2. (This behavior is similar to the truncated Viterbi algorithm employing a limited size of the decoding window [10]). Therefore, in the following, we will concentrate on the complexity advantage of the extending window method.

B. The Relationship Between the Sliding Window and Extending Window Methods

The relationship between $\boldsymbol{R}_{n-1}$, $\boldsymbol{R}_n$, $\boldsymbol{R}_0^{N-1}$ and $\boldsymbol{R}_0^N$ is illustrated in Fig. 1. Here we summarize the similarity and difference between the sliding and extending window methods.

- The cost for the sliding window method is dominated by computing $(\boldsymbol{R}_n)^{-1}$ in (5a). The recursive technique developed in [7] obtains $(\boldsymbol{R}_n)^{-1}$ based on $(\boldsymbol{R}_{n-1})^{-1}$, which takes advantage of the fact that $\boldsymbol{R}_{n-1}$ and $\boldsymbol{R}_n$ partially overlap (see Fig. 1).
- The cost for the extending window method is dominated by computing $(\boldsymbol{R}_0^N)^{-1}$ in (5b). Since $\boldsymbol{R}_0^{N-1}$ is a sub-block in $\boldsymbol{R}_0^N$, given the Cholesky factorization of $\boldsymbol{R}_0^{N-1}$, little extra effort is needed to find the Cholesky factorization of

$$\begin{bmatrix} z_{n-N_2} \\ z_{n-N_2+1} \\ \vdots \\ z_{n+N_1} \end{bmatrix} = \begin{bmatrix} h_{M-1} & h_{M-2} & \cdots & h_0 & & & & \\ & h_{M-1} & h_{M-2} & \cdots & h_0 & & \mathbf{0} & \\ & & \ddots & \ddots & & \ddots & & \\ & \mathbf{0} & & \ddots & \ddots & & \ddots & \\ & & & & h_{M-1} & h_{M-2} & \cdots & h_0 \end{bmatrix} \times \begin{bmatrix} x_{n-N_2-M+1} \\ x_{n-N_2-M+2} \\ \vdots \\ x_{n+N_1} \end{bmatrix} + \begin{bmatrix} w_{n-N_2} \\ w_{n-N_2+1} \\ \vdots \\ w_{n+N_1} \end{bmatrix} \tag{1a}$$

$\boldsymbol{R}_0^N$. This fact leads to a more efficient solution described below.

C. The Cholesky Factorization Technique for the Evaluation of (5b)

It is easy to verify that $\boldsymbol{R}_0^N$ is a symmetric, positive-definite band matrix with bandwidth $2M-1$. We can decompose $\boldsymbol{R}_0^N$ using the band Cholesky factorization [9] such that

$$\boldsymbol{R}_0^N = \boldsymbol{L}_0^N (\boldsymbol{L}_0^N)^T \tag{6}$$

where $\boldsymbol{L}_0^N$ is a lower triangular band matrix with bandwidth M. Then (5b) can be rewritten as

$$L_E(x_n) = 2 \cdot \frac{\boldsymbol{g}_n^T \boldsymbol{f}_n + \bar{x}_n \boldsymbol{g}_n^T \boldsymbol{g}_n}{1 - v_n \boldsymbol{g}_n^T \boldsymbol{g}_n} \tag{7a}$$

where

$$\boldsymbol{g}_n = (\boldsymbol{L}_0^N)^{-1} \boldsymbol{s}_n^N \text{ and } \boldsymbol{f}_n = (\boldsymbol{L}_0^N)^{-1}(\boldsymbol{z}_0^N - \boldsymbol{H}_0^N \bar{\boldsymbol{x}}_0^N). \tag{7b}$$

Suppose that $\boldsymbol{L}_0^{N-1}$ is available and consider the evaluation of $\boldsymbol{L}_0^N$ in (6). It can be shown that $\boldsymbol{R}_0^N$ has the form (see Fig. 1)

$$\boldsymbol{R}_0^N = \begin{bmatrix} \boldsymbol{R}_0^{N-1} & \boldsymbol{r} \\ \boldsymbol{r}^T & r \end{bmatrix} \tag{8a}$$

where r is a scalar and $\boldsymbol{r}$ is a column vector, and that

$$\boldsymbol{L}_0^N = \begin{bmatrix} \boldsymbol{L}_0^{N-1} & \boldsymbol{0} \\ \boldsymbol{l}^T & l \end{bmatrix} \tag{8b}$$

where l is a scalar and $\boldsymbol{l}$ is a column vector whose first $N-M+2$ entries are zero. From (8a) and (8b), given $\boldsymbol{L}_0^{N-1}$, only the last row in $\boldsymbol{L}_0^N$, i.e., $\boldsymbol{l}^T$ and l, needs to be calculated during the recursion, costing $M(M+1)/2-1$ multiplications, $M(M-1)/2$ additions and a square-root function [9], where M is the bandwidth of $\boldsymbol{L}_0^N$.

The evaluation of $\boldsymbol{g}_n$ and $\boldsymbol{f}_n$ is equivalent to solving the following equations:

$$\boldsymbol{L}_0^N \boldsymbol{g}_n = \boldsymbol{s}_n^N \tag{9a}$$
$$\boldsymbol{L}_0^N \boldsymbol{f}_n = \tilde{\boldsymbol{z}}_0^N \tag{9b}$$

where $\tilde{\boldsymbol{z}}_0^N = \boldsymbol{z}_0^N - \boldsymbol{H}_0^N \bar{\boldsymbol{x}}_0^N$. For (9a), since the first n entries in $\boldsymbol{s}_n^N$ are zero, only $M(N_1+1) - M(M-1)/2$ multiplications and $(M-1)(N_1+1) - M(M-1)/2$ additions are needed to compute each $\boldsymbol{g}_n$. For (9b), we have $\tilde{\boldsymbol{z}}_0^N = [(\tilde{\boldsymbol{z}}_0^{N-1})^T \tilde{z}]^T$ and $\boldsymbol{f}_n = [\boldsymbol{f}_{n-1}^T f]^T$, where $\tilde{z}$ and f are scalars. Given $\boldsymbol{f}_{n-1}$, only the last entry of $\boldsymbol{f}_n$, i.e., f, needs to be computed, costing M multiplications and $M-1$ additions. Some additional costs are also required relating to the calculation of the mean and variance values of $\{x_n\}$, $\tilde{\boldsymbol{z}}_0^N$ and the inner products in (7a).

For memory usage, recall that the recursion of $\boldsymbol{L}_0^N$ from $\boldsymbol{L}_0^{N-1}$ requires only the last row of $\boldsymbol{R}_0^N$ (M nonzero elements) and the $(M-1)$-by-$(M-1)$ subblock of $\boldsymbol{L}_0^{N-1}$ at the lower-right corner. Similarly, the recursions of $\boldsymbol{g}_n$ and $\boldsymbol{f}_n$ involve only the (N_1+1)-by-(N_1+1) sub-block of $\boldsymbol{L}_0^N$ at the lower-right corner and the last entry of $\tilde{\boldsymbol{z}}_0^N$. The calculation of (7a) involves only the last (N_1+1) entries of $\boldsymbol{g}_n$ and $\boldsymbol{f}_n$. Thus there is no need to store all the entries of $\boldsymbol{R}_0^N$, $\boldsymbol{L}_0^N$, $\boldsymbol{g}_n$, $\boldsymbol{f}_n$ and $\tilde{\boldsymbol{z}}_0^N$. Overall, the memory usage of the extending window technique is $O(N_1^2)$, which is similar to that of the exact sliding window technique [6], [7].

IV. Generalization to Complex Variables With QPSK Signaling

For complex signaling and channel models, we can adopt the following technique. Suppose that (4a) is a complex matrix equation with complex variables $\{z_n\}$, $\{h_n\}$, $\{x_n\}$ and $\{w_n\}$. Define

$$\vec{x}_n \equiv \begin{bmatrix} \mathrm{Re}(x_n) \\ \mathrm{Im}(x_n) \end{bmatrix}, \qquad \vec{z}_n \equiv \begin{bmatrix} \mathrm{Re}(z_n) \\ \mathrm{Im}(z_n) \end{bmatrix},$$
$$\vec{h}_m \equiv \begin{bmatrix} \mathrm{Re}(h_m) & -\mathrm{Im}(h_m) \\ \mathrm{Im}(h_m) & \mathrm{Re}(h_m) \end{bmatrix}, \qquad \vec{w}_n \equiv \begin{bmatrix} \mathrm{Re}(w_n) \\ \mathrm{Im}(w_n) \end{bmatrix}$$

where $\mathrm{Re}(\cdot)$ and $\mathrm{Im}(\cdot)$ denote the real and imaginary parts, respectively. Using these subblocks, (4a) can be equivalently rewritten in an augmented real matrix form

$$\begin{bmatrix} \vec{z}_1 \\ \vec{z}_2 \\ \vdots \\ \\ \\ \end{bmatrix} = \begin{bmatrix} \vec{h}_0 & & & & & & \\ \vec{h}_1 & \vec{h}_0 & & \boldsymbol{0} & & & \\ \vdots & & \ddots & & & & \\ \vec{h}_{M-1} & \vec{h}_{M-2} & \cdots & \vec{h}_0 & & & \\ & \vec{h}_{M-1} & \vec{h}_{M-2} & \cdots & \vec{h}_0 & & \\ \boldsymbol{0} & & \ddots & & & \ddots & \\ & & & \vec{h}_{M-1} & \vec{h}_{M-2} & \cdots & \vec{h}_0 \end{bmatrix} \times \begin{bmatrix} \vec{x}_1 \\ \vec{x}_2 \\ \vdots \\ \\ \end{bmatrix} + \begin{bmatrix} \vec{w}_1 \\ \vec{w}_2 \\ \vdots \\ \\ \end{bmatrix}. \tag{10}$$

Clearly, the discussion in Section III for real (4a) can be directly applied to (10).

V. Numerical Results

Consider a system employing a rate-1/2 systematic convolutional code with generators (1, 5/7). The information block length is 32 768 bits. A randomly generated interleaver is used before transmitting the coded bits in BPSK format. We employ tap-coefficients [0.227, 0.466, 0.668, 0.466, 0.227], $M=5$ and $N_1=9$; that is, the same as those in [6]. It is assumed that the receiver has exact knowledge of the channel fading coefficients.

TABLE I
COST COMPARISON OF DIFFERENT MMSE LEs. (UNIT: OPERATION NUMBERS PER SYMBOL PER ITERATION.) THE NUMBERS WITHIN THE BRACKETS ARE FOR THE SPECIAL CASE WITH $M = 5, N = 15$ AND $N_1 = 9$

Approach	Additions		Multiplications		Square root	
Exact sliding window [6]	$16N^2+4M^2+10M-4N-4$	(3686)	$8N^2+2M^2-10N+2M+4$	(1714)	0	(0)
Approximate sliding window [6]	$4N+8M$	(100)	$4N+4M-4$	(76)	0	(0)
Extending window	$4N_1M+10M+4N_1+8$	(274)	$4N_1M+8M+2N_1-1$	(237)	1	(1)

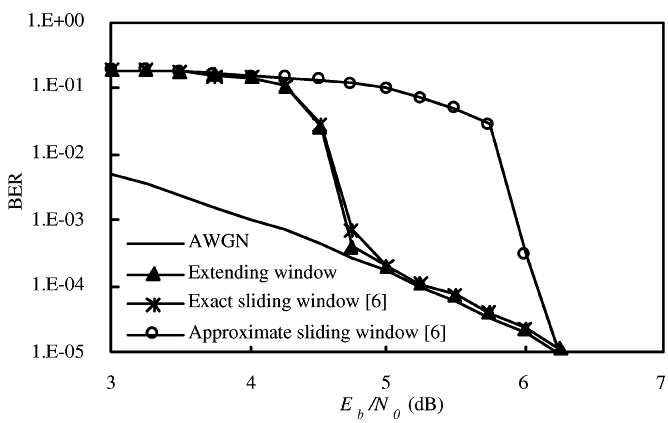

Fig. 2. Performance of linear MMSE turbo equalizers; 14 iterations are used.

Fig. 2 compares the extending window method with the exact sliding window method ($N_2 = 5$) [6] and the approximate sliding window method (I) ($N_2 = 5$) [6]. The performance in an AWGN channel without any ISI is included as a reference. It can be seen that the performance of the extending window method is very close to the exact sliding window method, and much better than the approximate sliding window approach.

Table I compares the number of multiplications and additions per symbol per iteration of these three equalizers, assuming complex channel coefficients. The cost for the extending window method takes into account the generalization from real tap-coefficients to complex ones as discussed in Section IV. Similar to [6], we ignore the cost of evaluating $\bar{x}_n$ and v_n based on $L(x_n)$. The numbers of multiplications and additions for the specific case used in Fig. 2 with $M = 5$, $N = 15$ and $N_1 = 9$ are also included.[1] The modified method requires less additions and multiplications than the exact sliding window method. (The cost of an extra square root operation per symbol per iteration for the former is negligible compared with additions and multiplications.) Comparing Fig. 2 and Table I, we can see that the proposed method provides a good compromise between performance and complexity.

[1]The methods in [6] are for a fixed window size N and their costs are related to N. For the extending window method proposed in this letter, the window size N grows during the recursion and the cost is determined by the noncausal length N_1 and the tap number M.

VI. CONCLUSION

We have proposed a modified MMSE turbo equalization algorithm. Significant cost reduction can be achieved without compromising performance as confirmed by simulation results.

REFERENCES

[1] C. Douillard, M. Jezequel, C. Berrou, A. Picart, P. Didier, and A. Glavieux, "Iterative correction of intersymbol interference: Turbo-equalization," *Eur. Trans. Telecommun.*, vol. 6, pp. 507–511, Sept./Oct. 1995.
[2] G. Bauch and V. Franz, "A comparison of soft-in/soft-out algorithms for 'Turbo detection'," in *Proc. Int. Conf. Telecomm.*, June 1998, pp. 259–263.
[3] P. Magniez, P. Duhamel, A. Roumy, and I. Fijalkow, "Turbo-equalization applied to trellis-coded-modulation," in *Proc. IEEE VTC99-Fall*, Amsterdam, The Netherlands, Sept. 1999, pp. 2556–2560.
[4] A. Glavieux, C. Laot, and J. Labat, "Turbo equalization over a frequency selective channel," in *Proc. Int. Symp.Turbo Codes*, Brest, France, Sept. 1997, pp. 96–102.
[5] D. Reynolds and X. Wang, "Low complexity turbo-equalization for diversity channels," *Signal Processing*, vol. 81, pp. 989–995, May 2001.
[6] M. Tüchler, R. Koetter, and A. Singer, "Turbo equalization: Principles and new results," *IEEE Trans. Commun.*, vol. 50, pp. 754–767, May 2002.
[7] M. Tüchler, A. Singer, and R. Koetter, "Minimum mean squared error equalization using *a priori* information," *IEEE Trans. Signal Processing*, vol. 50, pp. 673–683, Mar. 2002.
[8] X. Wang and H. V. Poor, "Iterative (turbo) soft interference cancellation and decoding for coded CDMA," *IEEE Trans. Commun.*, vol. 47, pp. 1046–1061, July 1999.
[9] B. N. Datta, *Numerical Linear Algebra*, California: Brooks/Cole Publishing Company, 1994.
[10] S. Haykin, *Communication Systems*, 4th ed. New York: Wiley, 2001, p. 663.