Recursive Blind Symbol Estimation of Convolutionally Coded Cochannel Signals

Jacob H. Gunther and A. Lee Swindlehurst, *Member, IEEE*

***Abstract*—A recursive blind equalizer is presented that directly estimates the transmitted symbols of multiple cochannel signals in the presence of ISI. The algorithm exploits shift structure present in the data model and the finite alphabet property of the signals. The proposed method possesses a separation property that allows the symbol sequences for each user to be estimated independently of the others. Problematic issues surrounding unknown and mismatched channel lengths for the cochannel users can be handled effectively in the recursive equalizer. Additionally, if the cochannel signals are encoded prior to transmission, we show how the code structure can be incorporated into the recursive equalizer to improve its performance.**

***Index Terms*—Adaptive equalizers, communication channels, convolutional codes, decision feedback equalizers, decoding, equalizers, multiuser channels, recursive estimation, sequence estimation.**

I. Introduction

TRANSMISSION of digitally modulated signals over bandwidth-limited channels has been an imoportant topic for several decades. In this situation, the receiver must compensate for channel induced intersymbol interference (ISI) in order to make reliable symbol decisions. The process of removing ISI is called equalization, and any device or algorithm for dealing with ISI is referred to as an equalizer.

Conventionally, many solutions to the equalization problem have focused on estimating the channel impulse response or its inverse rather than directly estimating the transmitted symbol sequence. Because the channel is unknown *a priori*, adjustable equalizer parameters are initially set, assuming a known training signal is transmitted. For time-varying channels, continuous updating of the adaptive equalizer parameters can be performed by periodically transmitting a known signal or using decision directed equalization.

To more efficiently use the channel and automatically restart in case of a broken data link, the use of self-recovering or blind adaptive equalization has been investigated [1]–[3]. These methods estimate the channel without a training sequence by forcing the equalizer output to possess a certain property such as constant modulus or finite alphabet. Nonadaptive methods for blind equalization have also been developed based on second and higher order statistics [4]–[10]. These methods process the data in blocks of consecutive samples to estimate the channel response. With the channel response in hand, the symbol sequence can be estimated using optimum maximum likelihood sequence estimation or some other technique.

An alternative to channel-identification-based blind equalization is blind sequence estimation [11], [12]. Here, the symbol sequence is estimated directly without the need for an explicit channel estimate. Combined with work on blind separation of cochannel signals transmitted over memoryless channels [13], blind sequence estimation can remove ISI and separate several cochannel finite alphabet signals. Channel-identification-based equalizers, on the other hand, are usually limited to the single-user case.

The principle on which *block* blind symbol estimation (BSE) [11], [12] is based can also be used to develop a *recursive* blind symbol estimator (RBSE) [14]. Here, estimates of the transmitted symbols are produced sequentially as new data samples are collected. Prior symbol decisions are fed back to help make decisions on new symbols, a process that can simplify the implementation of RBSE relative to BSE. In this paper, we develop a recursive blind symbol estimator and discuss some of its properties. This new recursive blind equalizer is unrelated to the classical decision feedback equalizer [15], where decisions are based on a linear combination of the received signal and previous symbol decisions. We also note that Tong [16] developed a recursive equalizer for a single user based on second-order statistics. Our method is new in the sense that it applies to multiple users, and no statistical assumptions on the users are necessary. We also show how to exploit coded signals in an efficient manner by incorporating code structure into the recursive blind equalizer. This leads to a method for simultaneously equalizing and decoding several cochannel signals. We restrict our attention to convolutional codes, but our results will apply in general to trellis-based codes, which are decoded iteratively, e.g., such as trellis-coded modulation and turbo codes. Linear block codes can also be decoded using trellis techniques [17]; therefore, our results apply to them as well.

The remainder of the paper is organized as follows. After reviewing the data model in Section II, the recursive blind equalizer is developed in Section III. In Section IV, we show how convolutional code structure can be incorporated into the equalizer. Finally, Section V presents some representative simulation results.

II. Data Model

Fig. 1 illustrates a scenario where d signals arrive at an M-element antenna array through independent FIR channels whose temporal support is assumed to span at most L symbol

Manuscript received January 27, 1998; revised October 1, 1999. The associate editor coordinating the review of this paper and approving it for publication was Prof. James A. Bucklew.

J. H. Gunther is with Merasoft, Inc., Provo, UT 84602 USA (e-mail: jakeg@merasoft.com).

A. L. Swindlehurst is with the Department of Electrical and Computer Engineering, Brigham Young University, Provo, UT 84620 USA.

Publisher Item Identifier S 1053-587X(00)02368-0.

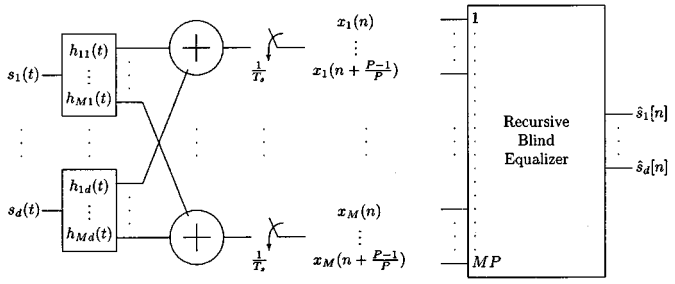


Fig. 1. System diagram.

periods. The channel impulse response $h_{ij}(t)$ accounts for both the transmitter pulse shaping filter and the physical propagation channel between the jth user and the ith sensor. We consider generic QAM communication signals described by

$$s^i(t) = \sum s_n^i \delta(t - nT) \qquad i = 1, \cdots, d$$

where

$\delta(t)$	Dirac delta function;
$T = PT_s$	symbol period;
$1/T_s$	sample rate;
$P \geq 1$	integer oversampling factor.

The transmitted symbols s_n^i are assumed to be drawn from a J-element alphabet Ω. Initially, we ignore the presence of any coding on the sequences, but in Section IV, coded sequences will be considered.

Although there are only M physical channels, the action of oversampling creates MP "virtual" channels. Collecting all samples of the array taken during the nth symbol period into an MP vector, we obtain

$$\boldsymbol{x}_n = \sum_{\ell=0}^{L-1} \boldsymbol{H}_\ell \boldsymbol{s}_{n-\ell} = \underbrace{[\boldsymbol{H}_0 \quad \cdots \quad \boldsymbol{H}_{L-1}]}_{\boldsymbol{H}} \begin{bmatrix} \boldsymbol{s}_n \\ \vdots \\ \boldsymbol{s}_{n-L+1} \end{bmatrix}. \quad (1)$$

Here, $\boldsymbol{H}_\ell$ is the ℓth tap of a MIMO FIR system with d inputs and MP outputs, and the $d \times 1$ symbol vector $\boldsymbol{s}_n$ contains one symbol for each user. Denote the elements of a matrix (vector) by $\boldsymbol{H}(\cdot, \cdot)$ $(\boldsymbol{x}(\cdot))$. Then

$$\begin{aligned} \boldsymbol{H}_n(i, j) &= h_{qj}\left(n + \frac{k}{P}\right) & i &= (q-1)P + k + 1 \\ \boldsymbol{x}_n(i) &= x_q\left(n + \frac{k}{P}\right) & q &= 1, \cdots, M \\ \boldsymbol{s}_n(\ell) &= s_n^\ell & k &= 0, \cdots, P-1 \\ & & \ell &= 1, \cdots, d. \end{aligned}$$

Later, we relate the subspace spanned by sequences of vector samples $\boldsymbol{x}_n$ and the corresponding symbol vectors $\boldsymbol{s}_n$. Therefore, for convenience, we collect N consecutive data vectors into the columns of $\boldsymbol{X}_n = [\boldsymbol{x}_{n-N+1}, \cdots, \boldsymbol{x}_n]$ and note that

$$\boldsymbol{X}_n = \boldsymbol{H}\boldsymbol{S}_n, \quad \boldsymbol{S}_n = \begin{bmatrix} \boldsymbol{s}_{n-N+1} & \cdots & \boldsymbol{s}_n \\ \vdots & & \vdots \\ \boldsymbol{s}_{n-N-L+2} & \cdots & \boldsymbol{s}_{n-L+1} \end{bmatrix}$$

where $\boldsymbol{S}_n$ is a block Toeplitz matrix whose elements are drawn from Ω.

In this paper, we require $\boldsymbol{H}$ to have full column rank. To alleviate the strict requirements this imposes on M and P in relation to d and L, rearrange the data into a block Hankel matrix with m block rows

$$\mathcal{X}_n^m = \mathcal{H}^{(m)} \mathcal{S}_n^{(m)} \quad (2)$$

where

$$\mathcal{X}_n^{(m)} = \begin{bmatrix} \boldsymbol{x}_{n-N+1} & \cdots & \boldsymbol{x}_{n-m+1} \\ \vdots & & \vdots \\ \boldsymbol{x}_{n-N+m} & \cdots & \boldsymbol{x}_n \end{bmatrix}$$

$$\mathcal{H}^{(m)} = \begin{bmatrix} & & \boxed{\boldsymbol{H}} \\ & \boxed{\boldsymbol{H}} & \\ \iddots & & \iddots \\ \boxed{\boldsymbol{H}} & & \end{bmatrix}$$

$$\mathcal{S}_n^{(m)} = \begin{bmatrix} \boldsymbol{s}_{n-N+m} & \cdots & \boldsymbol{s}_n \\ \vdots & & \vdots \\ \boldsymbol{s}_{n-N-L+2} & \cdots & \boldsymbol{s}_{n-m-L+2} \end{bmatrix}$$

$\mathcal{X}_n^{(m)}$: $mMP \times (N - m + 1)$ block-Hankel
$\mathcal{H}^{(m)}$: $mMP \times d(L + m - 1)$ block-Hankel
$\mathcal{S}_n^{(m)}$: $d(L + m - 1) \times (N - m + 1)$ block-Toeplitz.

The parameter m can be chosen by the user to satisfy certain identifiability conditions on the user sequences [11], [12]. In the following, we assume that m is chosen so that $\mathcal{H}^{(m)}$ has full column rank.

The proposed model (2) is based on several implicit assumptions. Perhaps the most restrictive of these is that the number of users d and the channel length L must be known. Although d might reasonably be known in a controlled communication system, the problem of channel length determination is difficult and becomes even more complicated when multiple cochannel users are present since the channel lengths for each user may be different. We will return to this issue later and indicate how our method may be applied when the channels have unequal but known lengths. Unless specified, assume that the channel for each user has length L.

Another less restrictive assumption implied by (2) is that the cochannel users are synchronized with respect to their symbol periods. As described in [12], this assumption may be relaxed to allow asynchronous signals by including the associated delays in the channel matrix.

III. Recursive Blind Symbol Estimation

Since $H^{(m)}$ in (2) has full column rank, the row spaces and null spaces of $\mathcal{X}_n^{(m)}$ and $\mathcal{S}_n^{(m)}$ coincide when no noise is present. If the symbol sequences are linearly independent, then $\text{rank}(\mathcal{X}_n^{(m)}) = \text{rank}(\mathcal{S}_n^{(m)}) = d(L + m - 1)$. Hence

$$\dim \text{null}\,(\mathcal{X}_n^{(m)}) = c_G = N - m + 1 - d(L + m - 1).$$

If we let the c_G columns of $\boldsymbol{G}_n$ be a basis for the null space of $\mathcal{X}^{(m)}$, then

$$\mathcal{X}_n^{(m)} \boldsymbol{G}_n = 0 \Leftrightarrow \mathcal{S}_n^{(m)} \boldsymbol{G}_n = 0 \quad (3)$$

holds. This equation is fundamental to the block methods [11], [12] for blind symbol estimation and plays a central role in the recursive method that follows. First, $\boldsymbol{G}_n$ is determined from the observed data using the left-hand side of (3). Then, the symbol sequences are estimated by exploiting the orthogonality to $\boldsymbol{G}_n$ illustrated by the right-hand side of (3).

In a recursive approach, $\boldsymbol{s}_n$ is estimated assuming the previous (hard or soft) symbol decisions $\hat{\boldsymbol{s}}_{n-1}, \hat{\boldsymbol{s}}_{n-2}, \cdots$ have been made. This information is available at the receiver from previous equalizer outputs or from a known training sequence. With previous symbols available, the only unknown on the right-hand side of (3) is $\boldsymbol{s}_n$, which appears on the first block row of $\mathcal{S}_n^{(m)}$. Let $\boldsymbol{B}_n$ be the first block row of $\mathcal{S}_n^{(m)}$:

$$\boldsymbol{B}_n = [\hat{\boldsymbol{s}}_{n-N+m}, \cdots, \hat{\boldsymbol{s}}_{n-1}, \boldsymbol{s}_n]$$

where previous symbols $\boldsymbol{s}_{n-i}$ have been replaced by training data or known symbol decisions $\hat{\boldsymbol{s}}_{n-i}$, $i = 1, \cdots, N-m$. By (3), all rows of $\mathcal{S}_n^{(m)}$ are in the left null space of $\boldsymbol{G}_n$; therefore,

$$\boldsymbol{B}_n \boldsymbol{G}_n = 0 \tag{4}$$

and, hence, a windowed segment of the cochannel symbol sequences is orthogonal to the column space of $\boldsymbol{G}_n$. Only the Toeplitz structure of the symbol matrix was used in deriving (4). A simple recursive estimator for $\boldsymbol{s}_n$ based on (4) would be

$$\hat{\boldsymbol{s}}_n = \arg\min_{\boldsymbol{s}_n \in \Omega^d} \left\| \underbrace{[\hat{\boldsymbol{s}}_{n-N+m}, \cdots, \hat{\boldsymbol{s}}_{n-1}, \boldsymbol{s}_n]}_{\boldsymbol{B}_n} \boldsymbol{G}_n \right\|_F^2$$

where the finite alphabet property has been imposed as a constraint for the minimization. This method can lead to errors since (4) is an underdetermined system of equations. The left null space of $\boldsymbol{G}_n$ is $d(L+m-1)$ dimensional, whereas $\boldsymbol{B}_n$ has only d linearly independent rows. Therefore, $\boldsymbol{s}_n$ may not be uniquely determined by (4).

Instead of (4), we can find a similar equation that uniquely determines $\boldsymbol{B}_n$ based on the sequence of matrices $\boldsymbol{G}_n, \cdots, \boldsymbol{G}_{n+Q-1}$, where $Q = L + m - 1$. We begin by making an observation about the shift structure present in the data model. Due to the memory of the channel, a particular symbol will influence the array output for L symbol periods after it is transmitted [see (1)]. For example, $\boldsymbol{s}_n$ influences outputs $\boldsymbol{x}_n, \boldsymbol{x}_{n+1}, \cdots, \boldsymbol{x}_{n+L-1}$. This manifests itself in the block-Toeplitz structure of $\mathcal{S}_n^{(m)}$ as

$$\begin{aligned} \boldsymbol{B}_n &= [\boldsymbol{s}_{n-N+m}, \cdots, \boldsymbol{s}_n] \\ &= \text{First block row of } \mathcal{S}_n^{(m)} \\ &= \text{Second block row of } \mathcal{S}_{n+1}^{(m)} \\ &\vdots \\ &= \text{Last block row of } \mathcal{S}_{n+Q-1}^{(m)}. \end{aligned} \tag{5}$$

Combining (3) and (5), we have the following Q equations involving $\boldsymbol{s}_n$:

$$\begin{aligned} \mathcal{S}_n^{(m)} \boldsymbol{G}_n = 0 &\Rightarrow \boldsymbol{B}_n \boldsymbol{G}_n = 0 \\ \mathcal{S}_{n+1}^{(m)} \boldsymbol{G}_{n+1} = 0 &\Rightarrow \boldsymbol{B}_n \boldsymbol{G}_{n+1} = 0 \\ &\vdots \\ \mathcal{S}_{n+Q-1}^{(m)} \boldsymbol{G}_{n+Q-1} = 0 &\Rightarrow \boldsymbol{B}_n \boldsymbol{G}_{n+Q-1} = 0. \end{aligned}$$

These equations indicate that the rows of $\boldsymbol{B}_n$ (which are the symbol sequences) are orthogonal to the column space of $\boldsymbol{G}_{n+i}$ for $i = 0, \cdots, Q-1$. In other words, the rows of $\boldsymbol{B}_n$ are orthogonal to the union of these spaces

$$\text{row}(\boldsymbol{B}_n) \perp \bigcup_{i=0}^{Q-1} \text{col}(\boldsymbol{G}_{n+i}). \tag{6}$$

Taking complements in (6), we have the interpretation that the rows of $\boldsymbol{B}_n$ lie in the intersection of the null spaces of $\boldsymbol{G}_{n+i}^*$ for $i = 0, \cdots, Q-1$:

$$\text{row}(\boldsymbol{B}_n) \subset \bigcap_{i=0}^{Q-1} \text{null}(\boldsymbol{G}_{n+i}^*). \tag{7}$$

Both of these expressions are summarized in matrix notation as

$$\boldsymbol{B}_n \mathcal{G}_n = 0 \tag{8}$$

where $\mathcal{G}_n = [\boldsymbol{G}_n, \cdots, \boldsymbol{G}_{n+Q-1}]$. Although (4) does not uniquely specify $\boldsymbol{B}_n$, (8) does. This relationship was independently derived in [18] and also exploited in the development of an adaptive blind equalizer.

The following theorem indicates that (in the absence of noise) the left null space of $\mathcal{G}_n$ is a d-dimensional space. Hence, there are only d linearly independent sequences from Ω that are orthogonal to the columns of $\mathcal{G}_n$.

Theorem 1: The left null space of $\mathcal{G}_n$ is exactly d dimensional.

Proof: See Appendix A.

This discussion suggests that the symbol sequences may be determined as the d sequences from Ω that are orthogonal to the columns of $\mathcal{G}_n$. When noise is present, orthogonality between $\boldsymbol{B}_n$ and $\mathcal{G}_n$ is lost. However, estimates for $\boldsymbol{s}_n$ can still be obtained by solving

$$\hat{\boldsymbol{s}}_n = \arg\min_{\boldsymbol{s}_n \in \Omega^d} \|[\hat{\boldsymbol{s}}_{n-N+m}, \cdots, \hat{\boldsymbol{s}}_{n-1}, \boldsymbol{s}_n] \mathcal{G}_n\|_F^2. \tag{9}$$

This minimization is the basis of our recursive symbol estimation algorithm, which is summarized in Table I.

A procedure for initializing the algorithm is not specified in Table I. To be completely "blind" in the sense that both the channel and the symbol sequences are unknown, a block algorithm such as BSE [11], [12] must be used to obtain initial estimates for $\hat{\boldsymbol{s}}_1, \cdots, \hat{\boldsymbol{s}}_{N-m}$. Then, we can proceed as above to estimate $\hat{\boldsymbol{s}}_{N-m+1}, \hat{\boldsymbol{s}}_{N-m+2}, \cdots$. Alternatively, the availability of a few training symbols could circumvent resort to the batch algorithms. Note that there is a $Q = L+m-1$ symbol delay between receiving the data $\boldsymbol{x}_n$ and making a decision on $\boldsymbol{s}_n$. This delay is required for forming $\mathcal{G}_n = [\boldsymbol{G}_n, \cdots, \boldsymbol{G}_{n+Q-1}]$.

Steps 1 and 3 in the algorithm (see Table I) call for shifts. This simply amounts to throwing away old data and adding the new. For example, in the shift

$$\mathcal{X}_n^{(m)} \leftarrow \mathcal{X}_{n-1}^{(m)}, \boldsymbol{x}_n$$

the first column of $\mathcal{X}_{n-1}^{(m)}$ is discarded, columns 2 through $N-m+1$ are shifted left, and a new column that preserves the Hankel structure is constructed using the new data $\boldsymbol{x}_n$. The shift

$$\mathcal{G}_n \leftarrow \mathcal{G}_{n-1}, \boldsymbol{G}_{n+Q-1}$$

is done in an obvious way.

TABLE I
RECURSIVE SYMBOL ESTIMATION ALGORITHM

Recursive Blind Symbol Estimation

Inputs:
- New data: $\mathbf{x}_{n+Q-1}$

Stored from previous iteration:
- Data matrix: $\mathcal{X}_{n+Q-2}^{(m)}$
- Null matrix: $\mathcal{G}_{n-1}$
- Previous decisions: $\hat{\mathbf{s}}_{n-N+m}, \cdots, \hat{\mathbf{s}}_{n-1}$

Outputs:
- Symbol estimate: $\hat{\mathbf{s}}_n$

Steps:
1. Shift: $\mathcal{X}_{n+Q-1}^{(m)} \leftarrow \mathcal{X}_{n+Q-2}^{(m)}, \mathbf{x}_{n+Q-1}$
2. Compute $\mathbf{G}_{n+Q-1}$ such that $\mathcal{X}_{n+Q-1}^{(m)}\mathbf{G}_{n+Q-1} = 0$
3. Shift: $\mathcal{G}_n \leftarrow \mathcal{G}_{n-1}, \mathbf{G}_{n+Q-1}$
4. Estimate $\hat{\mathbf{s}}_n$ using: $\hat{\mathbf{s}}_n = \arg\min_{\mathbf{s}_n \in \Omega^d} \left\| [\hat{\mathbf{s}}_{n-N+m}, \cdots, \hat{\mathbf{s}}_{n-1}, \mathbf{s}_n]\mathcal{G}_n \right\|_F^2$

When additive noise is present, $\mathcal{X}_n^{(m)}$ does not have a null space. In practice, the SVD can be used to approximate $\boldsymbol{G}_n$, but computing the full SVD is expensive. Fortunately, for the problem at hand, only the c_G least dominant right singular vectors are needed. In the above recursive approach, the matrices $\mathcal{X}_{n+1}^{(m)}$ and $\mathcal{X}_n^{(m)}$ will be nearly the same since only one data vector $\boldsymbol{x}_{n-N+1}$ has been discarded from $\mathcal{X}_n^{(m)}$ and one new vector $\boldsymbol{x}_{n+1}$ added. Hence, rather than recomputing the entire SVD at each new symbol period, it is desirable to perform an update/downdate operation. Several algorithms are available for this purpose but are still relatively expensive [19]–[22].

Two-sided (or complete) orthogonal decompositions have been used as substitutes for the SVD in least-squares problems [23]–[25]. Recently, rank revealing two-sided orthogonal URV and ULV decompositions have been introduced by Stewart [26], [27]. For our purposes, the main advantage of the URV over the SVD is that it allows for an efficient update and downdate. The up/downdating procedure requires $O((N-m+1)(mMP))$ time or $O(mMP)$ on a linear array of processors. Procedures for updating and downdating the URV/ULV have been presented in [26]–[30]. The quality of the subspaces obtained from the URV/ULV decompositions is described in [31], and it is shown that the ULV yields a more accurate estimate of the null space than the URV. To date, none of the algorithms for computing two-sided orthogonal decompositions (including the SVD) can exploit Hankel or Toeplitz structure like that found in $\mathcal{X}_n^{(m)}$.

A. Benefits of Recursive Blind Symbol Estimation

Among the benefits of our recursive symbol estimator is the fact that the users separate automatically so that the symbols for each user may be estimated independently of one another. To see this, recall that the d rows of $\boldsymbol{B}_n = [\hat{\boldsymbol{s}}_{n-N+m}, \cdots, \hat{\boldsymbol{s}}_{n-1}, \boldsymbol{s}_n]$ correspond to symbol sequences for each of the d users, and note that (9) represents a d-dimensional search that requires J^d evaluations to find the global minimum. However, using the property of the Frobenius norm

$$\left\| \begin{bmatrix} \boldsymbol{c}_1 \\ \vdots \\ \boldsymbol{c}_d \end{bmatrix} \boldsymbol{G} \right\|_F^2 = \sum_{i=1}^{d} \|\boldsymbol{c}_i \boldsymbol{G}\|_2^2$$

where the $\boldsymbol{c}_i$ are row vectors and $\boldsymbol{G}$ is a matrix, (9) is equivalent to d scalar minimizations

$$\hat{s}_n^i = \arg\min_{s_n^i \in \Omega} \left\| [\underbrace{\hat{s}_{n-N+m}^i, \cdots, \hat{s}_{n-1}^i}_{\triangleq \hat{\boldsymbol{b}}_n^i}, s_n^i]\mathcal{G}_n \right\|_2^2 \tag{10}$$

where $i = 1, 2, \cdots, d$, and $\hat{\boldsymbol{b}}_n^i$ is the ith row of $\boldsymbol{B}_n$ without the last element. In the remainder of the paper, we refer to the equivalence between (9) and (10) as the *source separation property* and call (10) the RBSE cost function. The d-dimensional minimization in (9), requiring J^d evaluations, reduces to d scalar minimizations in (10), each requiring J evaluations. Thus, the computational load has been reduced from $O(J^d)$ to $O(dJ)$. To exploit the source separation property in the RBSE algorithm in Table I, simply replace the d-dimensional enumeration in *Step 4* with d 1-D searches in (10). Next, we show that the enumeration in (10) is unnecessary in some cases.

An alternative *ad hoc* approach to minimizing (10) would be to relax the finite alphabet constraint and find a (closed-form) least-squares solution, followed by a projection to the nearest points in the symbol constellation. The ILSP approach [13] is an example of such a method. Partition $\mathcal{G}_n$ as

$$\mathcal{G}_n = \begin{bmatrix} \boldsymbol{A}_n \\ \boldsymbol{z}_n \end{bmatrix} \begin{matrix} \leftarrow \text{first N-m rows} \\ \leftarrow \text{ last row.} \end{matrix}$$

Then, the least-squares estimate of s_n^i is

$$\tilde{s}_n^i = -[\hat{s}_{n-N+m}^i, \cdots, \hat{s}_{n-1}^i]\boldsymbol{A}_n \boldsymbol{z}_n^\dagger = -\hat{\boldsymbol{b}}_n^i \boldsymbol{A}_n \boldsymbol{z}_n^\dagger \tag{11}$$

where $\boldsymbol{z}^\dagger = \boldsymbol{z}^*/\|\boldsymbol{z}\|_2^2$, and $\hat{s}_n^i$ is a hard decision based on $\tilde{s}_n^i$. This least-squares approach followed by a projection onto the alphabet Ω turns out to be equivalent to the 1-D search implied by (10) for any alphabet. For a proof of this, see Appendix B. Hence, optimal [in the sense of (10)] decisions about s_n^i can be made by simple decision rules based on $\tilde{s}_n^i$. For example, for the BPSK case, we have

$$\hat{s}_n^i = \begin{cases} +1, & \text{if } \mathrm{Re}\{\tilde{s}_n^i\} < 0 \\ -1, & \text{if } \mathrm{Re}\{\tilde{s}_n^i\} \geq 0. \end{cases}$$

For 16-QAM, compute $\tilde{s}_n^i$ as above, and then, decide

$$\mathrm{Re}\{\hat{s}_n^i\} = \begin{cases} +3, & \text{if } \mathrm{Re}\{\tilde{s}_n^i\} < -2 \\ +1, & \text{if } -2 \leq \mathrm{Re}\{\tilde{s}_n^i\} < 0 \\ -1, & \text{if } 0 \leq \mathrm{Re}\{\tilde{s}_n^i\} < 2 \\ -3, & \text{if } 2 \leq \mathrm{Re}\{\tilde{s}_n^i\}. \end{cases}$$

The decisions for $\mathrm{Im}\{\hat{s}_n^i\}$ are the same as above with $-\mathrm{Im}\{\cdot\}$ replacing $\mathrm{Re}\{\cdot\}$ on the right. The decision rules given above take $\hat{s}_n^i$ to be the element of the alphabet closest to the least-squares estimate $\tilde{s}_n^i$. Decision rules such as those above can provide a significant computational savings over enumeration, especially for large alphabets. A block diagram representation of the algorithm is given in Fig. 2.

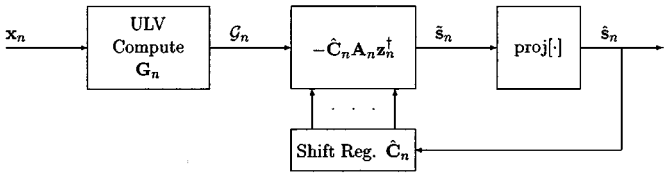


Fig. 2. Block diagram of the recursive blind symbol estimation algorithm.

B. Unknown or Mismatched Channel Lengths

We have assumed that all the channels have lengths equal to L symbol periods. However, in practice, it is difficult to determine the channel length L precisely. Furthermore, channels corresponding to different users may have different lengths. Both van der Veen [12] and Liu [32] give methods to compensate for unequal but known channel lengths in the block approach to blind symbol estimation. Next, we consider how to apply the recursive algorithm when channel lengths are unequal.

Recall that the recursive algorithm is based on the fact that the symbol sequence

$$[s_{n-N+m}^i, \cdots, s_{n-1}^i, s_n^i] \tag{12}$$

is orthogonal to the columns of

$$\mathcal{G}_n = [\boldsymbol{G}_n, \cdots, \boldsymbol{G}_{n+L+m-2}]$$

for each user ($i = 1, \cdots, d$). Consider the case where the channel for the ith user has length $L_i < L$. In this case, the sequence (12) is only orthogonal to a subset of the columns of $\mathcal{G}_n$ given by

$$\mathcal{G}_n^{(L_i)} = [\boldsymbol{G}_n, \cdots, \boldsymbol{G}_{n+L_i+m-2}].$$

Hence, to use channel length information about individual users in the RBSE, we simply replace $\mathcal{G}_n$ by $\mathcal{G}_n^{(L_i)}$ in (10). Furthermore, note that the sequence (12) is orthogonal to $\mathcal{G}_n^{(\hat{L}_i)}$ for any $\hat{L}_i \leq L_i$. Therefore, in applying recursive symbol estimation, we require that the channel length estimate for the ith user not be overestimated and be large enough so that $\mathcal{G}_n^{(\hat{L}_i)}$ has more columns than rows, i.e., so that $\mathcal{G}_n^{(\hat{L}_i)}$ is wide.

Communication channel impulse responses may have multiple peaks and long tails. For this reason, channel length is an ill-defined quantity. In practice, it is useful to choose L large enough to capture most of the channel response for the user with the longest channel, i.e., $L \geq \max_{i=1,\ldots,d} L_i$. In doing so, the channel matrix $\mathcal{H}^{(m)}$ may become ill conditioned. Notice the effect of overestimating L on the dimension of the null space of $\mathcal{X}_n^{(m)}$

$$\dim \text{null}(\mathcal{X}_n^{(m)}) = N - m + 1 - d(L + m - 1).$$

Overestimating L simply leads to fewer columns in $\boldsymbol{G}_n$ and does not present a problem in applying the recursive algorithm as long as $\mathcal{G}_n^{(\hat{L}_i)}$ is wide. This discussion leads to the following two observations in applying the recursive algorithm:

(O1) It is better to overestimate rather than underestimate L in computing the null matrix $\boldsymbol{G}_n$.

(O2) It is better to underestimate, rather than overestimate, L_i for computing the estimate s_n^i.

The use of these rules is illustrated in the following example. A channel with length $L = 4$ was driven by a BPSK source. The outputs of an $M = 4$ element array were oversampled by a factor of $P = 2$. The data matrices were constructed with $N = 30$ consecutive symbols, and no stacking ($m = 1$) was employed. A random complex matrix $MP \times L$ matrix was used as the channel matrix, and 10^5 data points were collected with a signal to noise ratio (SNR) of 2 dB. The same data set was processed by the RBSE algorithm several times, each time using different channel length information. The channel length L_{alg} was assumed in determining the number of right singular vectors of $\mathcal{X}_n^{(m)}$ used to form $\boldsymbol{G}_n$, that is, $\boldsymbol{G}_n$ had $N - m + 1 - d(L_{alg} + m - 1)$ columns. The channel length used for computing the symbol estimates is denoted by L_1, and $\mathcal{G}_n^{(L_1)}$ was used in the RBSE cost function instead of $\mathcal{G}_n$. Note that L_1 cannot be greater than L_{alg}. The purpose of this example is to evaluate the performance of RBSE for various choices of (L_{alg}, L_1). To illustrate the validity of (O1), the data was processed with $L_{alg} = 2, 3, 4, 5$ in RBSE. The four cases represented here are

1) channel length underestimation $L_{alg} = 2$;
2) channel length underestimation $L_{alg} = 3$;
3) correct estimation $L_{alg} = 4$;
4) overestimation $L_{alg} = 5$.

For each value of L_{alg}, L_1 took on all integer values between 2 and L_{alg}. For $L_1 < L = 4$, the channel is underestimated, $L_1 = 4$ is correctly estimated, and $L > 4$ is overestimated. A plot of the bit error rate (BER) versus L_1 appears in Fig. 3. First note that the BER is large when (O1) is violated ($L_{alg} = 2, 3$ curves). Additionally, it appears that worse violations of (O1) (lower values of L_{alg}) lead to higher BER. The $L_{alg} = 4, 5$ curves show that when (O1) is satisfied, then good BER performance is achieved, provided (O2) is not violated. The performance is almost the same when L_{alg} is overestimated as when it is correctly estimated, provided L_1 is not overestimated. This evidence suggests the validity of (O1) and (O2).

IV. Recursive Blind Estimation of Coded Sequences

In this section, we study the equalization problem in situations where the cochannel signals are encoded prior to transmission. Coding increases the effective "distance" between two symbol sequences, and exploiting this information in the recursive blind equalizer should improve its performance. There are several coding schemes that could be considered. In the discussion below, we will restrict attention to convolutional codes.

A. Convolutional Coding

For simplicity, we consider a rate one half binary convolutional code with ν memory elements. Here, interdependencies between the transmitted bits are introduced at the transmitter by the pair of mappings

$$[s_{2n}, s_{2n+1}] = \mathcal{F}(\sigma_n, \boldsymbol{\sigma}_n)$$
$$\boldsymbol{\sigma}_{n+1} = \mathcal{E}(\sigma_n, \boldsymbol{\sigma}_n)$$

where

- s_{2n}, s_{2n+1} are the encoded bits to be transmitted over the channel during the nth symbol period;
- σ_n is the convolutional encoder input at time n;
- $\boldsymbol{\sigma}_n = (\sigma_{n-1}, \cdots, \sigma_{n-\nu})$ represents the present state of the encoder at time n;

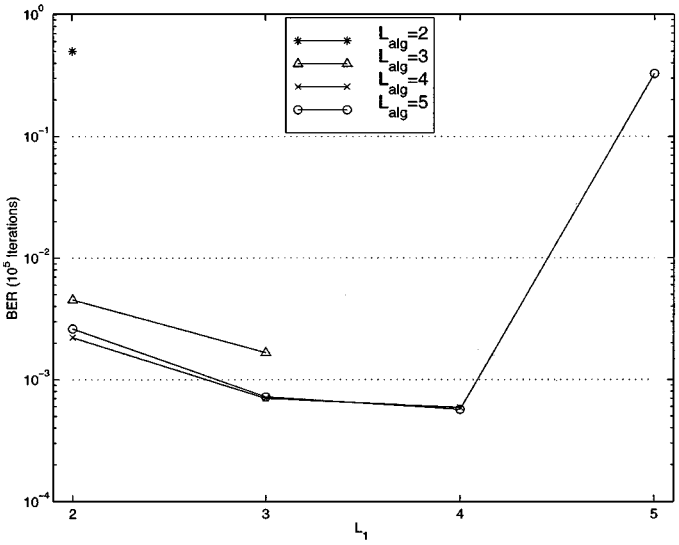


Fig. 3. BER performance of RBSE for various $(L_{alg},\ L_1)$.

- $\sigma_{n+1} = (\sigma_n, \cdots, \boldsymbol{\sigma}_{n-\nu+1})$ represents the next state of the encoder.

The convolutional encoder is a multirate system since the outputs are produced at twice the rate of the input. We have referenced the time indexing with respect to the information source that generates the σ_n bits.

The map $\mathcal{F}(\cdot, \cdot)$ encodes input information bits into output channel bits. For a linear convolutional code [e.g., Fig. 4(a)], $\mathcal{F}(\cdot, \cdot)$ is linear, and the map $\mathcal{E}(\cdot, \cdot)$ from present state to next state given the input is simply a shift. There are a total of 2^ν states. Because of $\mathcal{E}(\cdot, \cdot)$, not every transition between two states is possible, and hence, not every sequence of symbols is possible. Effectively, $\mathcal{E}$ increases the distance between feasible symbol sequences.

The map $\mathcal{E}(\cdot, \cdot)$ defining the state transitions can be expanded graphically into a trellis, e.g., Fig. 4(b). As time advances, the transmitter state traces out a single trajectory through the trellis, and the goal of the receiver is to determine this trajectory. To do so, the receiver considers all possible paths through the trellis. Each trellis stage consists of the set of all possible transmitter states at time n and the set of possible next states at time $n+1$.

B. The Viterbi Algorithm

The Viterbi algorithm (VA) is a procedure for choosing the surviving path with lowest cost among all the paths through the trellis that merge in a particular state at time $n+1$. Corresponding to the jth state $\boldsymbol{\sigma}_n^j$, $j = 1, \cdots, 2^\nu$ at time n is the pair $(\Gamma_n^j, \{s_i^j\})$, where

- Γ_n^j is the minimum accumulated cost associated with all paths leading to state $\boldsymbol{\sigma}_n^j$ at time n;
- $\{s_i^j\}$, $i = 2n - N + m, \cdots, 2n-1$ is the sequence of channel symbols leading to the state $\boldsymbol{\sigma}_n^j$ at time n, which minimizes Γ_n^j.

Corresponding to the branch connecting state $\boldsymbol{\sigma}_n^j$ to state $\boldsymbol{\sigma}_{n+1}^k$ is the triple

$$(\sigma_n^{jk},\ (s_{2n}^{jk},\ s_{2n+1}^{jk}),\ \delta_n^{jk}) \tag{13}$$

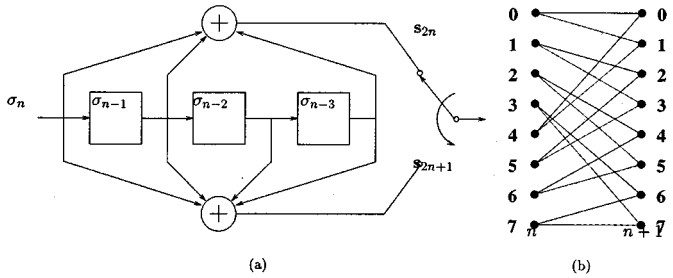


Fig. 4. (a) Rate one half linear binary convolutional encoder with $\nu = 3$ memory elements. (b) Single stage of the trellis associated with the encoder.

where

- σ_n^{jk} is the encoder input at time n that drives the transition from $\boldsymbol{\sigma}_n^j$ to $\boldsymbol{\sigma}_{n+1}^k$;
- $(s_{2n}^{jk},\ s_{2n+1}^{jk})$ is the encoder output corresponding to σ_n^{jk};
- δ_n^{jk} is the cost associated with outputs $(s_{2n}^{jk},\ s_{2n+1}^{jk})$.

Below, we show how to compute δ_n^{jk}.

At time $n+1$, a particular source arrived in state $\boldsymbol{\sigma}_{n+1}^k$ from state $\boldsymbol{\sigma}_n^j$ if the encoder input was σ_n^{jk} and the pair

$$(s_{2n}^{jk},\ s_{2n+1}^{jk}) \tag{14}$$

was transmitted. At the receiver, we can test the validity of this hypothesis using the pair (14) together with the symbol sequence that survived at state $\boldsymbol{\sigma}_n^j$ at time n. If a channel estimate was known, then the most likely pair (14) would minimize

$$\left\| \boldsymbol{x}_n - \sum_{\ell=0}^{L-1} \boldsymbol{H}_\ell \boldsymbol{s}_{n-\ell} \right\|_F^2 \tag{15}$$

which is the branch metric in the standard approach to maximum likelihood sequence estimation (MLSE) [33]. As illustrated in (15), both previous symbol decisions and a channel estimate are required in order to evaluate the MLSE branch metric. This leads to a large trellis that grows exponentially with the number of users d and the channel length L.

C. Modified VA for RBSE

The focus of this paper is on direct blind symbol estimation without the need for explicit channel estimates. Hence, the MLSE branch metric (15) is not applicable. In order to use the VA, we need a branch metric that does not require a channel estimate. In this case, the RBSE cost function (10) is a natural choice since it does not require a channel estimate, and it can be applied one user at a time. This leads to d small trellises whose complexity is fixed by the code and independent of the channel length. Consider the following channel-independent metric:

$$\delta_n^{jk} = \|[\hat{s}_{2n-N+m+1}^j, \cdots, \hat{s}_{2n-1}^j, s_{2n}^{jk}, s_{2n+1}^{jk}]\mathcal{G}_{2n+1}\|_2^2. \tag{16}$$

This metric measures the orthogonality between a hypothesized sequence and the columns of $\mathcal{G}_{2n+1}$. According to (8), in the absence of noise, the correct sequence will be orthogonal to $\mathcal{G}_{2n+1}$ and will thus minimize the branch metric. If the branch metric in (16) is the smallest among all branches that merge at state $\boldsymbol{\sigma}_{n+1}^k$, then we decide that the pair (14) was actually transmitted and append it to the sequence that survived at state $\boldsymbol{\sigma}_n^j$. Note that because of the trellis, it is not necessary to test every possible

TABLE II
RECURSIVE SYMBOL ESTIMATION ALGORITHM FOR CONVOLUTIONALLY ENCODED SIGNALS

Recursive Blind Equalization For Convolutionally Encoded Signals

Let: $Q = L + m - 1$
Inputs:
- New data: $\mathbf{x}_{2n+Q}, \mathbf{x}_{2n+Q+1}$

Stored from previous iteration:
- Data matrix: $\mathcal{X}^{(m)}_{2n+Q-1}$
- Null matrix: $\mathcal{G}_{2n-1}$
- Previous decisions for each user and each state: $\hat{s}^j_{2n-N+m}, \cdots, \hat{s}^j_{2n-1}$ for $j = 1, \cdots, 2^\nu$.

Outputs:
- Symbol estimates for each user: $\hat{s}_{2n-N+m}, \hat{s}_{2n-N+m+1}$.

Steps:
1. For $i = 0, 1$ do
   (a) Shift: $\mathcal{X}^{(m)}_{2n+Q+i} \leftarrow \mathcal{X}^{(m)}_{2n+Q-1+i}, \mathbf{x}_{2n+Q+i}$
   (b) Compute $\mathbf{G}_{2n+Q+i}$ such that $\mathcal{X}^{(m)}_{2n+Q+i}\mathbf{G}_{2n+Q+i} = 0$.
   (c) Shift: $\mathcal{G}_{2n+i} \leftarrow \mathcal{G}_{2n-1+i}, \mathbf{G}_{2n+Q+i}$
2. Compute branch metrics according to (16) for each user:
$$\delta^{jk}_n \text{ for } j = 1, \cdots, 2^\nu, \quad k = k_1, k_2$$
3. Choose survivors at each next state for each user:
$$\Gamma^k_{n+1} = \min_{j=j_1,j_2} \left\{ \Gamma^j_n + \delta^{jk}_n \right\} \text{ for } k = 1, \cdots, 2^\nu$$
$$j_k = \arg\min_{j=j_1,j_2} \left\{ \Gamma^j_n + \delta^{jk}_n \right\} \text{ for } k = 1, \cdots, 2^\nu$$
4. Update survivor sequences at each state for each user:
$$\left(\hat{s}^k_{2n}, \hat{s}^k_{2n-1}\right) = \left(s^{j_k k}_{2n}, s^{j_k k}_{2n+1}\right) \text{ for } k = 1, \cdots, 2^\nu$$
5. Output decisions for each user:
$$\eta = \arg\min_{k=1,\cdots,2^\nu} \Gamma^k_{n+1}$$
$$(\hat{s}_{2n-N+m}, \hat{s}_{2n-N+m+1}) = \left(s^{j_\eta \eta}_{2n-N+m}, s^{j_\eta \eta}_{2n-N+m+1}\right)$$

set of transmitted symbols. Only the feasible ones that represent state transitions as determined by $\mathcal{E}(\cdot, \cdot)$ need be considered.

In light of the above discussion, we propose replacing the standard metrics δ^{jk}_n in the VA with

$$\delta^{jk}_n = \left\| \left[\hat{s}^j_{2n-N+m+1}, \cdots, \hat{s}^j_{2n-1}, s^{jk}_{2n}, s^{jk}_{2n+1} \right] \mathcal{G}_{2n+1} \right\|^2_2$$
$$\Gamma^k_{n+1} = \min_j \left\{ \Gamma^j_n + \delta^{jk}_n \right\}.$$

Moving from time n to $n+1$ is thus a deterministic procedure using the symbols labeling each branch. We summarize the algorithm in Table II. This algorithm is novel in the sense that it combines decoding and equalization of multiple convolutionally encoded cochannel signals all into one operation.

The parameters k_1, k_2 in Step 2 are the indices of the states that follow state $\boldsymbol{\sigma}^j_n$. Similarly, the parameters j_1, j_2 in Step 3 are the indices of the states that precede state $\boldsymbol{\sigma}^k_{n+1}$. The j_k in Step 4 is the index of the previous state that survives at state k. Finally, η in Step 5 is the index of the state at time $n+1$, which has the lowest cost Γ^k_{n+1}. Symbol decisions for each user are made using the sequence terminating in state $\boldsymbol{\sigma}^\eta_{n+1}$.

Once $\mathcal{G}_{2n}$ and $\mathcal{G}_{2n+1}$ are computed (Step 1), all the computations associated with each user are independent of one another (Steps 2–5). Therefore, they may be carried out on a bank of d parallel processors. Thus, the fact that the users separate provides a major computational advantage to our algorithm over approaches where a single large trellis holds information for all the users together. The number of states in the MLSE trellis is $J^{d(L-1)}$, whereas the total number of states in the new approach is dJ^ν.

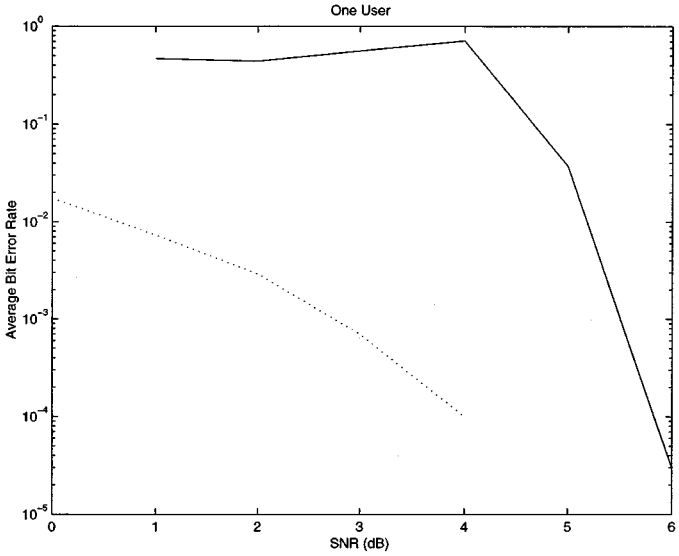


Fig. 5. Average BER for the single-user case (solid line). The average BER for the MMSE equalizer is also shown (dotted line).

Although we have only considered a rate one half binary convolutional code, nonbinary convolutional codes of any rate can be accommodated by the above framework. Trellis-coded modulation as well as other schemes can also be used with our recursive equalizer. The main difference is that the mappings $\mathcal{F}(\cdot, \cdot)$ and $\mathcal{E}(\cdot, \cdot)$ are different (see [34]), but the basic steps of the algorithm remain the same.

V. SIMULATIONS

In the first example, we simulate samples from an $M = 4$ element uniform linear array by passing a random BPSK sequence through a three-ray multipath channel. The symbols were pulse shaped using a raised cosine pulse with a roll-off factor of 0.1 and were oversampled by a factor of $P = 2$. The angles of arrival were -72.9547, 10.3021, and -12.6954° with respect to the array broadside, the time delays were 0, 0.1592, and 1.1302 symbol periods, and the complex gain factors were $0.0807 - 0.0624i$, $0.5913 + 0.1680i$, $-0.1692 + 1.2020i$. All these values were chosen at random. A GSM style frame was used that consisted of 26 known training symbols followed by 58 unknown symbols. The training symbols were used to initialize the algorithm described in Section III. In these simulations, we set the parameters $m = 9$ and $L = 5$. The bit error rates achieved by the algorithm versus SNR are shown in Fig. 5. These BER's were calculated by averaging over 100 000 symbols. Fig. 6 shows results (solid lines) for a two-user scenario. For comparison, the figures also show (dotted lines) the BER for a minimum mean square error (MMSE) equalizer. The MMSE equalizer implemented here is a one-tap spatio-temporal filter and is estimated using a 10^5 sample training sequence. In the single-user case, note the steepness of the slope of the BER curve. The BER decreases from 50% errors at 4 dB to 3×10^{-5} at 6 dB. This sudden change in BER is also observed in the two-user case. In the two-user case, note that the BER for the second user decreases below the BER for the same user in the MMSE equalizer.

To evaluate the relative performance of RBSE with and without use of the convolutional code structure, the following

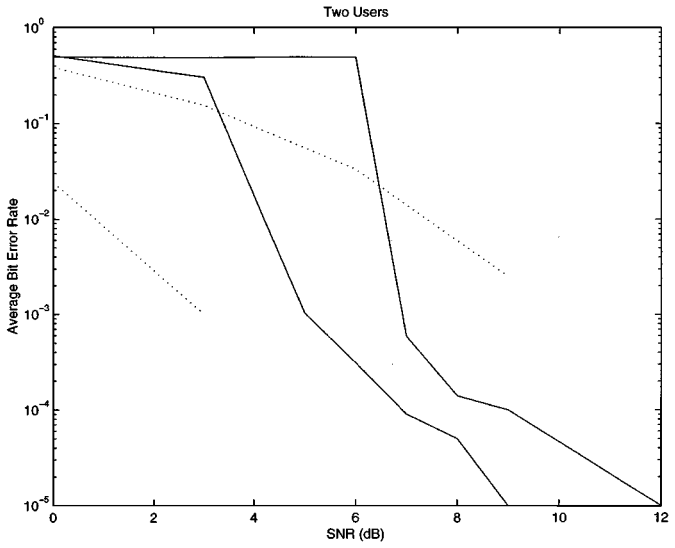


Fig. 6. Average BER for the two-user case (solid lines). The average BER for the MMSE equalizer is also shown (dotted lines).

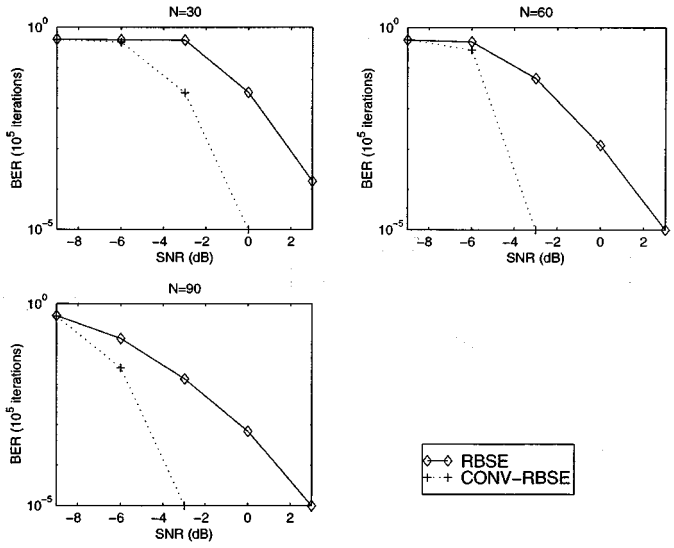


Fig. 7. BER performance of the RBSE algorithm versus SNR with (dotted) and without (solid) exploiting convolutional code structure for various values of N.

simulations were performed. A random complex $MP \times dL$ matrix was used to represent the channel. We assumed that $M = 4$ array elements were oversampled by $P = 2$ and that the channel spanned $L = 5$ symbol periods. The symbols were drawn at random from a BPSK alphabet. In the first simulation, the algorithms' dependence on the data window length N is evaluated. A single source was used with $N = 30$, 60, and 90. To test RBSE without coding, a random ± 1 sequence was passed through the channel. To test RBSE with the convolutional code (which is denoted CONV-RBSE in the figures), the convolutional encoder shown in Fig. 4 was used to encode the information bits, and the binary encoder outputs were converted to channel symbols.

The same channel matrix was used in all simulations and was normalized to have unit Frobenius norm. The signal-to-noise ratio is then defined by $\text{SNR} = 10\log_{10}(1/MP\sigma^2)$, where σ^2 is the noise variance. Fig. 7 shows plots of bit error rates versus SNR for each value of N.

As expected, the performance of RBSE with or without coding improves as the window length increases. As N increases, the null matrix $\mathcal{G}_n$ becomes wider, and more noise averaging takes place in computing the symbol estimates $\tilde{s}_n^i$. However, increasing N beyond a certain limit increases the computational load with diminishing performance gains. For example, the difference between bit error rates for $N = 60$ and $N = 90$ is not as dramatic as the difference for $N = 30$ and $N = 60$. In addition, note the significant performance improvement that results when coding is exploited.

A second simulation was conducted to evaluate the algorithms' performance for one and two users. For the two-user case, the submatrix of the channel matrix corresponding to each user was normalized to have unit Frobenius norm so that each user had the specified SNR. For the one- and two-user simulations, we fixed the number of columns of $\boldsymbol{G}_n$ to be 20 by changing N as necessary. For $d = 1$, this led to $N = 29$, and for $d = 2$, $N = 36$. Fig. 8 shows BER plots for each case.

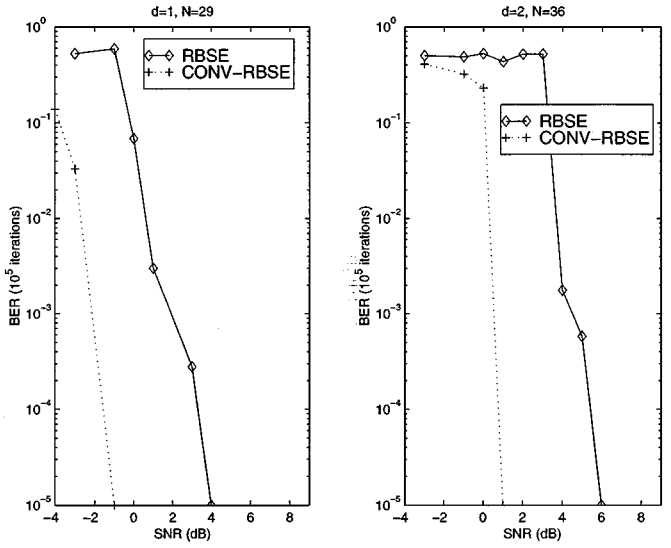


Fig. 8. BER performance of the RBSE algorithm versus SNR with (dotted) and without (solid) exploiting convolutional code structure for $d = 1$ and $d = 2$ user scenarios.

In the simulations conducted, CONV-RBSE achieves the same BER as RBSE with 4 to 6 dB less SNR at all useful bit error rates. From another viewpoint, for a fixed SNR, CONV-RBSE has bit error rates that are several orders of magnitude lower that RBSE. These results suggest that exploiting code structure in the equalizer can give a significant performance improvement. Note also the presence of a sharp performance threshold for CONV-RBSE as the SNR decreases. For the single-user case, CONV-RBSE decreases from 10^{-1} BER to 10^{-5} BER with only a 3-dB change in SNR. For two users, the effect is even more pronounced. This sharp drop in performance at low SNR is probably due to the correct path not being chosen at any state in the VA.

VI. Conclusions

In this paper, a recursive blind equalizer was derived that estimates cochannel symbol sequences directly without an explicit channel estimate. The principle benefit of this approach is that the users separate and can be estimated independently of one another. Efficient methods for up and downdating rank revealing matrix decompositions lead to an efficient implementation. The source separation property allows channel-length information to be used separately for each user. Hence, problems associated with unknown and mismatched channel lengths are overcome. Because of the recursive nature of the algorithm and the source separation property, the recursive blind equalizer is easily modified to exploit convolutional code structure in the source sequences. Simulations demonstrated a significant performance improvement when the code structure was incorporated into the equalizer.

Appendix A
Proof of Theorem 2

Lemma 1: Let $\mathcal{X}_{n+i}^{(m)} = \mathcal{H}^{(m)}\mathcal{S}_{n+i}^{(m)}$, where $\boldsymbol{H}^{(m)}$ and $\mathcal{S}_{n+i}^{(m)}$ have full column and row rank, respectively. Let $\mathcal{G}_n^{(i)} = [\boldsymbol{G}_n, \cdots, \boldsymbol{G}_{n+i-1}]$, where the $c_G = N - m + 1 - d(L + m - 1)$ columns of $\boldsymbol{G}_{n+i}$ span the null space of $\mathcal{X}_{n+i}^{(m)}$. Then

$$\text{rank}(\mathcal{G}_n^{(i)}) = c_G + (i-1)\,d$$

for $i = 1, \cdots, Q$.

Proof: $\boldsymbol{G}_{n+i}$ is an $(N-m+1)\times c_G$ matrix, where $c_G = N - m + 1 - d(L+m-1)$ with full column rank. The $i = 1$ case is trivial. Consider the $i = 2$ case. We want to show $\text{rank}([\boldsymbol{G}_n\ \boldsymbol{G}_{n+1}]) = c_G + d$. We know that

$$\begin{aligned}
&\text{rank}([\boldsymbol{G}_n\ \boldsymbol{G}_{n+1}]) \\
&\quad = \dim\{\text{span}(\boldsymbol{G}_n)\bigcup\text{span}(\boldsymbol{G}_{n+1})\} \\
&\quad = \dim\{\text{span}(\boldsymbol{G}_n)\} + \dim\{\text{span}(\boldsymbol{G}_{n+1})\} \\
&\qquad - \dim\{\text{span}(\boldsymbol{G}_n)\bigcap\text{span}(\boldsymbol{G}_{n+1})\} \\
&\quad = 2c_G - \dim\{\text{span}(\boldsymbol{G}_n)\bigcap\text{span}(\boldsymbol{G}_{n+1})\}. \qquad (17)
\end{aligned}$$

Let $N_1 = N + 1$ and $m_1 = m + 1$, and form a new matrix $\mathcal{X}_n^{(m_1)}$ with the same block Hankel structure as $\mathcal{X}_n^{(m)}$, but stack m_1 times instead of m. $\mathcal{X}_n^{(m_1)}$ has dimensions $(m_1 MP)\times(N_1 - m_1 + 1) = ((m+1)MP)\times(N-m+1)$, i.e., $\mathcal{X}_n^{(m_1)}$ has the same number of columns as $\mathcal{X}_n^{(m)}$ but has one more block row. Note that $\mathcal{X}_n^{(m_1)}$ contains $\mathcal{X}_n^{(m)}$ and $\mathcal{X}_{n+1}^{(m)}$ as submatrices in its first and last mMP rows, respectively. Stacking in this way, we have $\mathcal{X}_n^{(m_1)} = \mathcal{H}^{(m_1)}\mathcal{S}^{(m_1)}$, where $\mathcal{H}^{(m_1)}$ still has full column rank. Then, $\mathcal{X}_n^{(m_1)}$ has rank $d(L+m_1-1) = d(L+m-1)+d$, and $\text{null}(\mathcal{X}_n^{(m_1)}) = \text{null}(\mathcal{S}_n^{(m_1)})$.

Let the $N-m+1-d(L+m-1)-d = c_G - d$ columns of $\boldsymbol{G}_n^{(m_1)}$ be a basis for the null space of $\mathcal{X}_n^{(m_1)}$. Then, $\boldsymbol{G}_n^{(m_1)}$ is a basis for the intersection of $\text{null}(\mathcal{X}_n^{(m)})$ and $\text{null}(\mathcal{X}_{n+1}^{(m)})$. Hence

$$\begin{aligned}
&\dim\{\text{span}(\boldsymbol{G}_n)\bigcap\text{span}(\boldsymbol{G}_{n+1})\} \\
&\quad = \dim\{\text{null}(\mathcal{X}_n^{(m)})\bigcap\text{null}(\mathcal{X}_{n+1}^{(m)})\} \\
&\quad = \text{rank}(\boldsymbol{G}_n^{(m_1)}) = c_G - d.
\end{aligned}$$

Using this result in (17) leads to $\text{rank}([\boldsymbol{G}_n\ \boldsymbol{G}_{n+1}]) = c_G + d$.

In general, for $i = 3, \cdots, Q$, let $N_i = N + i - 1$ and $m_i = m+i-1$ and form $\mathcal{X}_n^{(m_i)}$, which has rank $d(L+m_i-1) = d(L+m-1)+d(i-1)$ and contains $\mathcal{X}_n^{(m)}, \cdots, \mathcal{X}_{n+i}^{(m)}$ as submatrices. Let the $N-m+1-d(L+m_i+1) = c_G - d(i-1)$ columns of $\boldsymbol{G}_n^{(m_i)}$ span the null space of $\mathcal{X}_n^{(m_i)}$. Then

$$\begin{aligned}
&\dim\left\{\bigcap_{k=1}^{i}\text{span}(\boldsymbol{G}_{n+k-1})\right\} \\
&\quad = \dim\left\{\bigcap_{k=1}^{i}\text{null}(\mathcal{X}_{n+k-1}^{(m)})\right\} \\
&\quad = \text{rank}(\boldsymbol{G}_n^{(m_i)}) = c_G - d(i-1).
\end{aligned}$$

The desired result follows by induction. ∎

Theorem 2: The left null space of $\mathcal{G}_n$ is exactly d-dimensional.

Proof: The dimension of the left null space of $\mathcal{G}_n$ is

$$\begin{aligned}
&(N-m+1) - \text{rank}(\mathcal{G}_n) \\
&\quad = (N-m+1) - (c_G + d(Q-1)) \ \text{ by Lemma 1} \\
&\quad = d(L+m-1) - d(L+m-2) = d
\end{aligned}$$

∎

Appendix B
Enumeration Is Equivalent to Least-Squares and Projection

The RBSE cost function has the form

$$\begin{aligned}
J_0 &= \left\| [\boldsymbol{b}\quad s]\begin{bmatrix}\boldsymbol{A}\\ \boldsymbol{z}\end{bmatrix}\right\|_2^2 \\
&= |s|^2\|\boldsymbol{z}\|_2^2 - 2\|\boldsymbol{z}\|_2^2\,\text{Re}\{s^*\tilde{s}\} + \boldsymbol{b}\boldsymbol{A}\boldsymbol{A}^*\boldsymbol{b}^* \qquad (18)
\end{aligned}$$

where $\tilde{s}$ is the least squares estimate of s given in (11) and accounts for the sign change of the middle term on the right-hand side of (18). The minimizing argument of J_0 in (18) does not change if we divide J_0 by the positive constant $\|\boldsymbol{z}\|_2^2$. This gives a new cost function

$$J_1 = |s|^2 - 2\,\text{Re}\{s^*\tilde{s}\} + c$$

where $c = \boldsymbol{b}\boldsymbol{A}\boldsymbol{A}^*\boldsymbol{b}^*/\|\boldsymbol{z}\|_2^2$. Now, expand the cost function J_1 in terms of the real and imaginary parts of s and $\tilde{s}$

$$J_1 = s_r^2 + s_i^2 - 2(\tilde{s}_r s_r + \tilde{s}_i s_i) + c.$$

Finally, after completing the square, we obtain

$$J_1 = (s_r - \tilde{s}_r)^2 + (s_i - \tilde{s}_i)^2 + c - (\tilde{s}_r^2 + \tilde{s}_i^2). \qquad (19)$$

Thus, we see that minimizing J_1 (which is equivalent to minimizing the RBSE cost function) is achieved by choosing the element of the symbol constellation that is closest in Euclidean distance to the least squares estimate $\tilde{s}$. This proof is general and holds for arbitrary symbol constellations.

References

[1] Y. Sato, "A method of self-recovering equalization for multi-level amplitude modulation," *IEEE Trans. Commun.*, vol. COMM-23, pp. 679–682, June 1975.

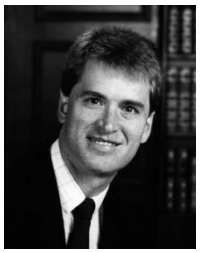
[2] D. N. Godard, "Self-recovering equalization and carrier tracking in two-dimensional data communication systems," *IEEE Trans. Commun.*, vol. COMM-28, pp. 1867–1875, Nov. 1980.

[3] A. Benveniste and M. Goursat, "Blind equalizers," *IEEE Trans. Commun.*, vol. COMM-32, pp. 871–883, Aug. 1984.

[4] J. K. Tugnait, "Identification of linear stochastic systems via second and fourth-order cumulant matching," *IEEE Trans. Inform. Theory*, vol. IT-33, pp. 393–407, May 1987.
[5] W. Gardner, "A new method of channel identification," *IEEE Trans. Commun.*, vol. 39, pp. 813–817, June 1991.
[6] L. Tong, G. Xu, and T. Kailath, "Blind identification and equalization based on second-order statistics: A time domain approach," *IEEE Trans. Inform. Theory*, vol. 40, pp. 340–349, Mar. 1994.
[7] J. K. Tugnait, "On blind equalization of multipath channels using fractional sampling and second-order cyclostationary statistics," in *Proc. Global Commun. Conf.*, Houston, TX, Dec. 1993.
[8] D. Slock, "Blind fractionally spaced equalization, perfect reconstruction filter banks, and multichannel linear prediction," in *Proc. Int. Conf. Acoust., Speech Signal Process.*, Adelaide, Australia, 1994, pp. IV-585–IV-588.
[9] E. Moulines, P. Duhamel, J. Cardoso, and S. Mayrargue, "Subspace methods for the blind identification of multichannel FIR filters," *IEEE Trans. Signal Processing*, vol. 43, pp. 516–525, Feb. 1995.
[10] Z. Ding, "Matrix outer-product decomposition method for blind multiple channel equalization," *IEEE Trans. Signal Processing*, vol. 45, pp. 3053–3061, Dec. 1997.
[11] H. Liu and G. Xu, "Closed-form blind symbol estimation in digital communications," *IEEE Trans. Signal Processing*, vol. 43, pp. 2714–2723, Nov. 1995.
[12] A. van der Veen, S. Talwar, and A. Paulraj, "A subspace approach to blind space-time signal processing for wireless communication systems," *IEEE Trans. Signal Processing*, vol. 45, pp. 173–190, Jan. 1997.
[13] S. Talwar, M. Viberg, and A. Paulraj, "Blind separation of synchronous co-channel digital signals using an antenna array—Part I: Algorithms," *IEEE Trans. Signal Processing*, vol. 44, June 1996.
[14] J. Gunther and A. Swindlehurst, "Blind sequential symbol estimation of co-channel finite alphabet signals," in *Proc. 30th IEEE Asilomar Conf.*, vol. 2, Nov. 1996, pp. 823–827.
[15] C. A. Belfiore and J. J. H. Park, "Decision-feedback equalization," *Proc. IEEE*, vol. 67, pp. 1143–1156, Aug. 1979.
[16] L. Tong, "Blind sequence estimation," *IEEE Trans. Commun.*, vol. 43, pp. 2986–2994, Dec. 1995.
[17] C. Schlegel, *Trellis Coding*. Piscataway, NJ: IEEE, 1997.
[18] P. Vandaele and M. Moonen, "An SVD+Viterbi algorithm for multiuser adaptive blind equalization of mobile radio channels," *IEEE Signal Processing Lett.*, vol. 5, pp. 260–264, Oct. 1998.
[19] M. Moonen, P. V. Dooren, and J. Vandewalle, "A singular value decomposition updating algorithm for subspace tracking," *SIAM J. Matrix Anal. Appl.*, vol. 13, pp. 1015–1038, Oct. 1992.
[20] J. R. Bunch and C. P. Nielsen, "Updating the singular value decomposition," *Numer. Math.*, vol. 31, pp. 111–129, 1978.
[21] H. Park and S. V. Huffel, "Two-way bidiagonalization scheme for downdating the singular-value decomposition," *Linear Algebra Appl.*, vol. 222, pp. 23–39, 1995.
[22] P. A. Businger, "Updating a singular value decomposition," *BIT*, vol. 10, pp. 376–397, 1970.
[23] G. H. Golub and C. F. Van Loan, *Matrix Computations*. Baltimore, MD: John Hopkins Univ. Press, 1989.
[24] C. L. Lawson and R. J. Hanson, *Solving Least-Squares Problems*. Englewood Cliffs, NJ: Prentice-Hall, 1974.
[25] A. Bjorck, *Numerical Methods for Least Squares Problems*. Philadelphia, PA: SIAM, 1996.
[26] G. W. Stewart, "An updating algorithm for subspace tracking," *IEEE Trans. Signal Processing*, vol. 40, pp. 1535–1541, June 1992.
[27] ——, "Updating a rank-revealing ULV decomposition," *SIAM J. Matrix Anal. Appl.*, vol. 14, no. 2, pp. 494–499, 1993.
[28] H. Park and L. Elden, "Downdating the rank-revealing URV decomposition," *SIAM J. Matrix Anal. Appl.*, vol. 16, no. 1, pp. 135–155, 1995.
[29] J. L. Barlow, P. A. Yoon, and H. Zha, "An algorithm and stability theory for downdating the ULV decomposition," *BIT*, vol. 36, pp. 14–40, Jan. 1996.
[30] J. L. Barlow and P. A. Yoon, "Solving recursive TLS problems using the rank-revealing ULV decomposition," *Proc. Int. Workshop TLS Errors Variables*, 1997.
[31] R. D. Fierro and J. R. Bunch, "Bounding the subspaces from rank revealing two-sided orthogonal decompositions," *SIAM J. Matrix Anal. Appl.*, vol. 16, no. 3, pp. 743–759, 1995.
[32] H. Liu and G. Xu, "Multiuser blind channel estimation and spatial channel pre-equalization," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, vol. 3, 1995, pp. 1756–1759.
[33] J. G. D. Forney, "Maximum-likelihood sequence estimation of digital sequences in the presenece of ISI," *IEEE Trans. Inform. Theory*, vol. IT-18, pp. 363–378, May 1972.
[34] E. Biglieri, D. Divsalar, P. McLane, and M. Simon, *Introduction to Trellis-Coded Modulation with Applications*. New York: Macmillan, 1991.

Jacob H. Gunther received the B.S., M.S., and Ph.D. degrees in electrical engineering from Brigham Young University (BYU), Provo, UT, in 1994, 1994, and 1998, respectively.

From 1992 to 1994, he was a Research Assistant with the Microwave Earth Remote Sensing Laboratory, BYU. From 1994 to 1995, he was with Lockheed-Martin, Manassas, VA, where he worked on target detection algorithms, sonar systems, and satellite digital communication systems. He is currently with Merasoft, Inc., Provo, where he is involved in design of noise and echo cancellation algorithms, speech recognition, and concatenative speech synthesis. His research interests include adaptive and nonlinear filtering, array signal processing for wireless communications, and speech processing.



A. Lee Swindlehurst (M'90) received the B.S. (summa cum laude) and M.S. degrees in electrical engineering from Brigham Young University (BYU), Provo, UT, in 1985 and 1986, respectively, and the Ph.D. degree in electrical engineering from Stanford University, Stanford, CA, in 1991.

From 1983 to 1984, he was with Eyring Research Institute, Provo, as a Scientific Programmer, writing software for a Minuteman missile simulation system. From 1984 to 1986, he was a Research Assistant with the Department of Electrical Engineering, BYU. He was awarded an Office of Naval Research Graduate Fellowship for 1985 to 1988, and during that time, he was affiliated with the Information Systems Laboratory at Stanford University. From 1986 to 1990, he was also with ESL, Inc., Sunnyvale, CA, where he was involved in the design of algorithms and architectures for a variety of radar and sonar signal processing systems. He joined the Faculty of the Department of Electrical and Computer Engineering, BYU, in 1990, where he is an Associate Professor. From 1996 to 1997, he held a joint appointment as a Visiting Scholar with both Uppsala University, Uppsala, Sweden, and at the Royal Institute of Technology, Stockholm, Sweden. His research interests include sensor array signal processing for radar and wireless communications, detection and estimation theory, and system identification.

Dr. Swindlehurst is a past Associate Editor for the IEEE TRANSACTIONS ON SIGNAL PROCESSING.