On the Use of Kernel Structure for Blind Equalization

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

Abstract—The mathematical theory of kernel (null space) structure of Hankel and Hankel-like matrices is applied to the problem of blind equalization of cochannel signals. This approach provides a new perspective on the blind equalization problem and gives insights into the identifiability conditions already presented in the literature. An algorithm is presented that tracks the exact null space of the symbol matrix even in the presence of noise. This work exploits shift structure in the oversampled channel output and the finite alphabet property of the signals. Previously, these two properties were used independently in a two-step (equalize then separate) process. A contribution of the new approach is that it allows simultaneous exploitation of both shift structure and the finite alphabet property of the signals.

Index Terms—Adaptive equalizers, communications channels, deconvolution, equalizers, Hankel matrices, intersymbol interference, multipath channels, Toeplitz matrices.

I. INTRODUCTION

M ANY structured matrices (e.g., Hankel, Toeplitz, Vandermonde) have kernels (null spaces) that are also structured. Kernel structure theory plays an implicit role in a surprisingly wide variety of problems in mathematics, signal processing, control, and coding. We are interested in applying this theory to the blind equalization problem.

The problem of blind equalization of a single source has received considerable attention in the literature [1]–[3]. Recently, several methods have been derived based on the cyclostationarity of oversampled channel outputs [4]–[10]. The oversampling forces special shift structure into the data model that can be exploited to remove intersymbol interference (ISI). Removing ISI in the case of multiple cochannel signals can still be accomplished by exploiting only the shift structure, but determines the signals only up to an instantaneous linear mixture. As shown in [11], the signals can be separated by exploiting the finite alphabet property. These ideas are combined and extended in [12]–[14], leading to methods for blind equalization of multiple cochannel signals. Yet, the two properties are used independently. First, the signals are equalized by exploiting the shift structure, and then, they are separated using the finite alphabet property. Unlike these methods, the new method proposed in this paper simultaneously exploits the shift structure and the finite alphabet property.

The format of the paper is as follows. In Section II, the data model is briefly developed. Section III suggests a new approach to the blind equalization problem. Section IV gives a tutorial summary on kernel structure theory for Hankel matrices that commonly arise in signal processing problems. Then, identifiability conditions found in the literature that seem unrelated on the surface are put into a unifying framework using the kernel structure theory. Before deriving our kernel structure-based blind equalizer in Section VI, we investigate formulas for updating and downdating the kernel parameters in Section V. These efficient formulas are a key element to the development of our new equalizer. Finally, we conclude the paper with some representative simulation results.

II. DATA MODEL

Suppose a linear channel is driven by $d$ digital communication signals. We assume that the channel has a finite impulse response spanning $L$ symbol periods. By means of spatial and/or temporal oversampling, $M$ samples are collected during each symbol period. Arranging all samples collected during the $n$th symbol period into an $M$ vector, the following baseband model is obtained:

$$
x[n] = \sum_{i=1}^{d} \sum_{j=0}^{L-1} h_i[j] s_i[n-j]$$

$$= \begin{bmatrix} H_{L-1} & \cdots & H_0 \end{bmatrix} \begin{bmatrix} s_{L-1} \\ \vdots \\ s_0 \end{bmatrix}$$

where

$$H_j = [h_1[j] \cdots h_d[j]]$$

is an $M \times d$ matrix representing the $j$th tap of the MIMO FIR communication channel, and

$$s_n = [s_1[n] \cdots s_d[n]]^T$$

is a $d \times 1$ symbol vector containing the symbols transmitted by each source during the $n$th symbol period. In this paper, we deal only with finite alphabet (FA) signals where the symbols $s_i[n]$ are chosen from a finite symbol constellation or alphabet $\Omega$. Implicitly, (1) assumes the cochannel signals arrive synchronously at the receiver. However, for asynchronous signals, the associated delays can be included in the channel matrix $H$ as in [12].

Arranging consecutive vector samples into the columns of a matrix gives

$$X_n^{(N-L+1)} = [x_{n-N+L} \cdots x_n] = HS_n^{(N-L+1)}$$

where the symbol matrix

$$S_n^{(N-L+1)} = \begin{bmatrix} s_{n-N+1} & \cdots & s_{n-L+1} \\ \vdots & \vdots & \vdots \\ s_{n-N+L} & \cdots & s_n \end{bmatrix}$$

Manuscript received April 15, 1999; revised July 29, 1999. The associate editor coordinating the review of this paper and approving it for publication was Dr. Vikram Krishnamurthy.

J. H. Gunther is with Merasoft, Inc., Provo, UT 84602 USA.
A. Lee Swindlehurst is with the Department of Electrical and Computer Engineering, Brigham Young University, Provo, UT 84620 USA.

Publisher Item Identifier S 1053-587X/00$10.00 © 2000 IEEE
is a block Hankel matrix with $d \times 1$ blocks, and $s_n$ and $H$ are defined in (1). The superscript on the data and symbol matrices indicates the number of columns, whereas the subscript gives the index of the last (bottom right) block element in the matrix.

For now, we neglect the influence of noise and assume that (2) is an exact mathematical description of the observations. In this paper, we assume that $H$ has full column rank, giving $X^{(N-L+1)}_n$ a low-rank structure. When this is true ($M > dL$ is necessary), then the row spans and the null spaces of the data and the symbol matrices coincide. It follows that

$$X^{(N-L+1)}_nG_n = 0 \iff S^{(N-L+1)}_nG_n = 0 \quad (3)$$

where the columns of $G_n$ form a basis for the null space of $X^{(N-L+1)}_n$ and $S^{(N-L+1)}_n$. This is the key equation for blind equalization. It expresses a relationship between the observed data $X^{(N-L+1)}_n$ and the unknown symbols $S^{(N-L+1)}_n$ through the null matrix $G_n$. Recall that the symbol matrix $S^{(N-L+1)}_n$ is block Hankel with elements in the symbol alphabet. Hence, only a finite number of legitimate symbol matrices is possible.

III. PROBLEM DEVELOPMENT

There are at least two approaches to blind equalization based on (3), depending on which side of the equation is used first.

Approach 1—(A1): One possibility is to start on the left-hand side and compute a null basis $G^X_n$ for the observed data $X^{(N-L+1)}_n$. According to the right-hand side of (3), when arranged in a block Hankel matrix of appropriate dimensions, the correct symbol sequences will be orthogonal to the columns of $G^X_n$. Therefore, the symbols could be estimated by testing every possible legitimate symbol matrix for orthogonality to $G^X_n$. This is a finite (though lengthy) search. The correct symbols will be orthogonal to $G^X_n$ at least when there is no noise and $H$ has full column rank. This exhaustive search can be replaced by a simpler suboptimal procedure [12]–[14]. First, the unique (non FA) vector sequence, which is most orthogonal to $G^X_n$ in the least-squares sense, could be determined. Then, symbol estimates could be computed by projecting the least-squares estimates onto the symbol alphabet.

Approach 2—(A2): A different approach to blind equalization using (3) might start on the right-hand side. Again, every possible legitimate symbol matrix is considered. For any candidate symbol matrix $S^{(N-L+1)}_n$, an exact symbol null basis $G^S_n$ could be computed. According to (3), $G^S_n$ is orthogonal to the data if and only if the corresponding symbols are correct. Therefore, the validity of the symbols in question could be tested by whether or not the observed data is orthogonal to $G^S_n$.

From a practical standpoint, both approaches have advantages and disadvantages. First, suppose noise is present. Then, the low-rank structure of the data matrix is lost. However, an approximate data null space $G^X_n$ can still be computed using a rank-revealing matrix decomposition such as the singular value decomposition (SVD). In this case, $X^{(N-L+1)}_nG^X_n \neq 0$, and $S^{(N-L+1)}_nG^S_n \neq 0$, even for the correct symbols. On the other hand, in approach A2, the symbol null space $G^S_n$ can be computed exactly for any symbol matrix, even when noise is present in the data. However, the noise does make $X^{(N-L+1)}_nG^X_n \neq 0$, even for the correct symbols. In the first approach, $G^X_n$ is only computed once, whereas in the second approach, $G^S_n$ must be computed for every candidate symbol sequence. Thus, A1 has a computational advantage (at least when using the SVD to compute the null basis), whereas A2 has the relative advantage of working with exact symbol null matrices.

Next, recall the form of the data and symbol matrices. $X^{(N-L+1)}_n$ is an unstructured matrix, whereas $S^{(N-L+1)}_n$ has special block Hankel structure. In the next section, we closely examine the structure of null spaces of Hankel and block Hankel matrices. Due to its special structure, $S^{(N-L+1)}_n$ also has a very highly structured null space. This has several consequences. First, it leads to a parameterization of the null space $G^S_n$ in terms of a very few variables. On the other hand, the data null space $G^X_n$ is grossly overparameterized. For example, an SVD of $X^{(N-L+1)}_n$ does not produce a structured null basis. Furthermore, the sparse parameterization of the symbol kernel can be computed efficiently and updated/downated efficiently, but this cannot be said about the SVD and many other matrix decompositions used for computing $G^X_n$. Combining this fact with the finite alphabet property of the signals leads to an algorithm to implement the second approach given above. That is, every feasible symbol matrix is considered. We avoid a complete enumeration by taking a recursive approach, which allows an efficient tree/trellis search of the whole list of feasible symbol matrices.

IV. KERNEL STRUCTURE OF HANKEL MATRICES

This section reviews some basic facts about kernel structure that will be needed to develop the blind equalizer. We focus on Hankel matrices, but many of these facts can be adapted to hold for other types of structured matrices. We begin with some notational conventions.

A. Notation and Definitions

Let $S(n, N) = \{s_{n-N+1}, \cdots, s_n\}$ be an $N$-point sequence terminating in $s_n$. The family of Hankel matrices

$$S^{(k)}_n = \begin{bmatrix} s_{n-N+1} & \cdots & s_{n-k+1} \\ \vdots & \ddots & \vdots \\ s_{n-k+1} & \cdots & s_n \end{bmatrix} \quad (4)$$

is associated with the sequence $S(n, N)$, where $k + l = N + 1$, $k = 1, \cdots, N$, and as before, the superscript indicates the number of columns. If the $s_k$ are scalars, then $S^{(k)}_n$ is an $(N + 1 - k) \times k$ Hankel matrix. If the $s_k$ are matrices themselves, then the $S^{(k)}_n$ are block Hankel matrices. Note that $S^{(1)}_n$ is a column, and $S^{(N)}_n$ is a row. Consider the following example.

Example 1: Suppose we have the following sequence: $S(n, T) = \{a, b, c, d, e, f, g\}$. Then, the family $\{S^{(k)}_n\}_{k=1}^T$ of Hankel matrices is

$$S^{(1)}_n = \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \end{bmatrix}, \quad S^{(2)}_n = \begin{bmatrix} a & b \\ c & d \\ e & f \end{bmatrix}, \quad S^{(3)}_n = \begin{bmatrix} a & b & c \\ b & c & d \\ c & d & e \end{bmatrix}$$
Many of the results that follow are more easily expressed in polynomial language rather than using matrix-vector notation. We will freely switch back and forth between these representations using the natural correspondence between spaces of length \( n \) vectors and the space of polynomials of degree less than \( n \).

Denote the kernel (null space) of \( S_n \) by \( N_n \),
\[
N_n = \left\{ u(\lambda) ; S_n u = 0 \right\}. 
\]

If the \( s_i \) are \( p \times q \) matrices, then \( u(\lambda) \) is a vector polynomial with coefficients that are \( q \) vectors.

### B. Kernel Structure Theorem

The first theorem characterizes the null space of block Hankel matrices \( S_n \) for \( k = 1, \ldots, N+1 \).

**Theorem 1:** Given a sequence \( S(n, N) = \{ s_{n-N+1}, \ldots, s_n \} \) of \( p \times q \) matrices, there is a uniquely defined \( (p+q) \)-tuple of integers \( \{n_1, \ldots, n_{p+q} \} \), \( 0 \leq n_1 \leq \cdots \leq n_{p+q} \leq N+1 \), and vector polynomials \( u_i(\lambda) \), \( i = 1, \ldots, p+q \), \( p+q = \delta \), where \( \delta \) is the row defect of \( S_n \) such that \( \deg(u_i(\lambda)) \leq n_i \) and the polynomials
\[
u_i(\lambda), \lambda u_i(\lambda), \ldots, \lambda^{n_i-1} u_i(\lambda)
\]
form a basis of \( N_n \), where \( i \) runs over all indices with \( n_i < k \).

**Proof:** See [15].

Following [15], we call \( \{ u_i(\lambda) \}_{i=1}^{p+q} \) the fundamental system (FS) and \( \{ n_i \}_{i=1}^{p+q} \) the characteristic degrees of \( S(n, N) \). The polynomials in (6) are referred to as a shift chain. The fundamental system is not unique but has the property that the \( u_i(\lambda) \) have no roots in common. In the following, we assume that the \( (n_i, u_i(\lambda)) \) pairs are ordered so that the \( n_i \) are nondecreasing. The parameter \( \delta \) in the theorem is the row defect of \( S_n(N) \). It is the number of linearly dependent rows in \( S_n \). In the blind equalization problem, the cochannel symbol sequences that are the rows of \( S_n(N) \) are typically assumed to be linearly independent. Therefore, in the sequel, we will only consider the case \( \delta = 0 \).

Additionally, in blind equalization, the sequence under consideration is a \( d \times 1 \) symbol vector sequence so that \( (p, q) = (d, 1) \). A consequence of \( q = 1 \) is that the polynomials in the theorem are simply scalar polynomials. Hence, we only consider \( q = 1 \) and scalar polynomials in the rest of the paper.

The following discussion is intended to explain Theorem 1 and elaborate on its consequences. Essentially, the theorem characterizes the null spaces of any block Hankel matrix constructed from a particular vector sequence. Suppose a linearly independent \( N \times 1 \) vector sequence \( S(n, N) \) is given. According to the theorem, there is a special set of polynomials or vectors that completely parameterizes the null spaces of \( S_n(k) \) for all \( k \). However, not all of the polynomials are needed to represent the null space of \( S_n(k) \) for all \( k \). For a particular \( k \), if \( n_k \) is greater than or equal to \( k \), then \( u_k \) is not needed in the representation of the null space of \( S_n(k) \). Since the \( n_1, n_2, \ldots, n_{p+q} \) are fixed by the sequence, the theorem indicates that as \( k \) increases, more of the \( u_i \) (and more shifts of them) are needed to represent the null space of \( S_n(k) \). Because the polynomials in the fundamental system are coprime, the polynomial shift chain is linearly independent.

The matrix-vector equivalent of the polynomial shift chain is
\[
\begin{bmatrix}
x_0 \\
\vdots \\
x_{n-1}
\end{bmatrix} \iff x(\lambda) = x_0 + x_1 \lambda + \cdots + x_{n-1} \lambda^{n-1}.
\]

When noise is present, (8) no longer holds. However, if we let \( G_n^S \) be an SVD derived null basis for \( S_n(N+1) \), then we still have
\[
\text{span} \left\{ G_n^S \right\} = \text{span} \left\{ U_n^S \right\}.
\]
by Theorem 1 on the rank of $S_n^{(k)}$ or, equivalently, the dimension of $N_n^{(k)}$. For this purpose, define

$$\delta_k = \text{card}\{i : n_i < k\}$$

where $\text{card}\{A\}$ is the cardinality of (number of elements in) the set $A$. Thus, $\delta_k$ is the number of vectors from the fundamental system that are needed to completely parameterize $N_n^{(k)}$. $\delta_k$ takes on the values $0, 1, \ldots, d+1$ in sequence as $k$ increases from $0$ to $N+1$. The number of columns in $U_n^{(k)}$ corresponding to $u_k$ is $k - n_i$. Therefore, the total number of columns in $U_n^{(k)}$ is

$$\dim N_n^{(k)} = \sum_{k=1}^{\delta_k} (k - n_i) = k\delta_k - \sum_{k=1}^{\delta_k} n_i, \quad (9)$$

For $n_i \leq k < n_{i+1}, \delta_k$ is a constant (equal to $\delta_i$), and therefore the sum $\sum_{k=1}^{\delta_k} n_i$ is also constant. Hence, when $k$ is between two characteristic degrees, then the dimension of the null space increases in steps of $\delta_i$. From this information, we can deduce the rank profile of $S_n^{(k)}$ as a function of $k$.

For simplicity, consider the single source ($d = 1$) case. In this case, $p = d = 1, q = 1$, and the theorem implies that $p + q = 2$ polynomials (vectors) are needed to characterize the null space $N_n^{(k)}$ of $S_n^{(k)}$ for all $k$. (This fact was overlooked in [17], where it was assumed that only one polynomial was needed to span $N_n^{(k)}$ for all $k$.) Now, given a sequence $S(n_i, N)$, there are unique characteristic degrees $(n_1, n_2)$, and there is a fundamental system $(u_1, u_2)$ that satisfies Theorem 1. In this case

$$\delta_k = \begin{cases} 0 & 1 \leq k \leq n_1 \\ n_1 < k \leq n_2 \\ n_2 < k \leq N, \end{cases} \quad (10)$$

The rank of $S_n^{(k)}$ is equal to the number of its columns less the dimension of $N_n^{(k)}$.

$$\text{rank}\{S_n^{(k)}\} = k - k\delta_k + \sum_{i=1}^{\delta_k} n_i = \sum_{i=1}^{\delta_k} n_i - k(\delta_k - 1).$$

Fig. 1 illustrates a typical graph of $\delta_k$ and rank$\{S_n^{(k)}\}$ versus $k$. In the lower graph, the solid line is rank$\{S_n^{(k)}\}$, and the dashed line is $k$, which is the number of columns in $S_n^{(k)}$. The difference between these two graphs is the dimension of $N_n^{(k)}$. As illustrated, the dimension of $N_n^{(k)}$ increases in increments of $\delta_n + 1$ for $n_1 < k \leq n_2$ and then in increments of $\delta_{n+1} + 1 = 2$ for $n_2 < k \leq N$. Next, we consider how the polynomials in the FS play a role in these regions.

Consider the three cases in (10). First, if $1 \leq k \leq n_1$, then $S_n^{(k)}$ is full rank, and therefore, the null space is empty. Now, when $k = n_1 + 1, S_n^{(n_1+1)}$ is rank deficient by 1. Let $u_k$ be the $(n_1 + 1) \times 1$ vector that spans the null space of $S_n^{(n_1+1)}$. For $n_1 < k \leq n_2$, the null space is spanned by the structured shift chain basis parameterized completely by $u_k$. This is because of the Hankel shift structure in $S_n^{(k)}$. When $k = n_2 + 1$, the $u_1$ shift chain spans all but a one-dimensional (1-D) subspace of $N_n^{(n_2+1)}$. Therefore, an $(n_2 + 1) \times 1$ vector $v_2$ can be found in the null space of $S_n^{(n_2+1)}$, which is linearly independent of the $u_1$ shift chain. These two cases are illustrated in Fig. 2. A key point that will be used later is that only when $k > n_2$ are both $u_1$ and $u_2$ needed for a complete parameterization of the null space.

C. Typical Values of the Characteristic Degrees

For any given sequence, Theorem 1 asserts that there is a unique set of characteristic degrees $(n_1, n_2)$. What are typical values for the characteristic degrees of a generic symbol sequence? Consider the $d = 1$ case. First, note that the relationship between $n_1, n_2$ and $N$ is

$$n_1 \leq n_2 \leq N \quad \text{and} \quad n_1 + n_2 = N + 1. \quad (11)$$

In addition, assume that $N$ is odd so that $(N + 1)/2$ is an integer. In light of (11), the largest $n_1$ can ever be is $(N + 1)/2$, regardless of what the sequence is. First, consider a sequence $S(n_i, N)$, which is the sum of $m$ distinct complex exponentials. For this sequence, the matrices $S_n^{(k)}$ have full column rank for $1 \leq k \leq m$, assuming $S_n^{(k)}$ is wide [i.e., $N$ is large enough so that $m \leq (N + 1)/2$]. Hence, for $k$ in this range, it will be the case that $n_1 = m$. However, when $m = (N + 1)/2 + 1, S_n^{(m)}$ has two more columns than rows. Therefore, it has a two-dimensional (2-D) null space. In this case, $n_1 = n_2 = (N + 1)/2$. (If $N$ was even, then $n_1 = [(N + 1)/2] = N/2$, and $n_2 = [(N + 1)/2] = (N/2) + 1$.) Therefore, it can be concluded that for any $m > (N + 1)/2, n_1$ and $n_2$ will be fixed at $(N + 1)/2$. Following [17], we will call $n_1$ the number of modes in the sequence. The modes are the $n_1$ roots of $u_1(\lambda)$. The conclusion of the above discussion is that if a sequence is composed of a sufficiently large number of complex exponentials, then $n_1 = n_2 = (N + 1)/2$. Next, let $S(n_i, N)$ be a finite alphabet symbol sequence. The question about typical values for the characteristic degrees now corresponds to how many complex exponentials are typically required to represent the sequence. It is often assumed that the symbols are independent and identically distributed (i.i.d.). Thinking in terms of the DFT of an i.i.d. sequence, we would expect that it would have a nonzero projection onto each of the $N$ elements of the complex exponential DFT.
basis. Therefore, most of the time, \( n_1 = n_2 = (N + 1)/2 \) in the blind equalization problem. For large \( N \), it is a rare event when \( n_1 < n_2 \). However, sequences such as constant sequences or sequences of the same symbol alternating in sign, and so on, have a nonzero probability of occurring (error correcting coding may prevent this). For these degenerate cases, \( n_1 < (N+1)/2 < n_2 \). These cases present a problem to the block methods [12]–[14], but the kernel structure based method developed in Section VI adapts naturally to the number of modes in the symbol sequence.

**D. The Inverse Problem**

Before relating these ideas to previous work on blind equalization, we need to consider the inverse of Theorem 1. In summary, Theorem 1 states that given a sequence, there is a set of coprime polynomials forming a fundamental system. We now ask, given a collection of coprime polynomials, is there a sequence that has as its fundamental system this particular collection of polynomials and is the corresponding sequence unique? First, consider a theorem by Heinig [18] that answers this question for the \( d = 1 \) case.

**Theorem 2:** Given two coprime polynomials \( (u_1(\lambda), u_2(\lambda)) \) with degrees \( (n_1, n_2) \), there is an \( N \)-point sequence \( S(n, N) \), where \( N = n_1 + n_1 - 1 \), for which \( (u_1(\lambda), u_2(\lambda)) \) is a fundamental system.

The proof is based on the following lemma.

**Lemma 1:** The polynomials \( (u_1(\lambda), u_2(\lambda)) \) form a fundamental system for \( S(n, N) \) if and only if
\[
S_n^{(N)} U_n^{(N)} = 0. 
\]

**Proof:** See [18].

In (12), \( S_n^{(N)} \) is a \( 1 \times N \) row vector. We know that \( U_n^{(N)} \) is \( N \times N - 1 \) and has linearly independent columns. Therefore, it has a 1-D left null space. Lemma 1 states that \( (u_1(\lambda), u_2(\lambda)) \) is a fundamental system for any sequence in the left null space of \( U_n^{(N)} \). All of these sequences are the same up to a scaling. Hence, given two coprime polynomials, there exists a sequence that is unique up to a multiplicative constant. This idea can be extended for \( d > 1 \). In this case, the multiplicative constant is a \( d \times d \) matrix. Hence, for multiple sources, there remains an ambiguity in the inverse problem. Later, when we apply kernel structure theory to the blind equalization problem, this ambiguity does not arise.

The essential outcome of Theorem 2 is the following fact.

**Fact 1:** Both \( u_1(\lambda) \) and \( u_2(\lambda) \) are needed to identify the sequence and suffice to identify it up to a multiplicative constant.

**E. Unification of Identifiability Conditions**

Together with earlier results on kernel structure theory, Fact 1 sheds light on the blind identifiability conditions in [12], [13], and on the degeneracy described in [13]. First, we examine the identifiability conditions on the symbol sequences. We assume the channel matrix has full column rank.

In the context of this paper, Liu’s conditions [13] for blind identifiability (\( d = 1 \)) is that the symbol sequence must have more than \( L \) modes. Recall that the number of modes is equal to \( n_1 \), which is the characteristic degree of \( u_1(\lambda) \), whose roots \( z_1, \ldots, z_{n_1} \) are the modes. Liu’s identifiability condition is that

\[ n_1 > L. \]

Recall that for \( d = 1 \), we have \( n_1 + n_2 = N + 1 \). Eliminating \( n_1 \) from these two equations, we have

\[ N - L + 1 > n_2. \]

This equation implies that both \( u_1 \) and \( u_2 \) are required to form \( U_n^{(N-L+1)} \), which is a null basis for \( S_n^{(N-L+1)} \). According to Fact 1, these are precisely the necessary and sufficient conditions for blind identifiability. In other words, Liu’s requirement that the input consist of more than \( L \) modes is equivalent to requiring the full fundamental system (both \( u_1 \) and \( u_2 \)) for a complete parameterization of the null space.

In the context of this paper, van der Veen’s [12] identifiability conditions are that \( M > d, N > dL \) (necessary), and \( S_n^{(N-L+1)} \) must have full row rank. Consider again the \( d = 1 \) case. Because of oversampling, we know that \( M > d = 1 \) is satisfied. In addition, we must assume that the number of columns \( N - L + 1 \) in \( S_n^{(N-L+1)} \) is greater than one (in order to have a...
null space), which implies that \( N > L \), which corresponds exactly with van der Veen’s necessary condition. Finally, consider the condition that the \( L \) rows of \( S_n^{(N-L+1)} \) must be linearly independent. This implies that the null space be exactly \( N - 2L + 1 \) dimensional. The null space dimension is given in (9). For the case when the signals are identifiable according to Fact 1, we must have \( \delta_{N-L+1} = 2 \). Inserting this into (9) gives

\[
\dim \mathcal{N}_{n_1}^{(N-L+1)} = (N - L + 1) \delta_{N-L+1} - \sum_{i=1}^{\delta_{N-L+1}} n_i = 2(N - L + 1) - (n_1 + n_2) = N - 2L + 1.
\]

This corresponds exactly with van der Veen’s sufficient condition. These results can be generalized to the case \( d > 1 \) as well.

Next, consider Liu’s degenerate case where the input contains too few modes to be identifiable. When there are no more than \( L \) modes (i.e., \( n_2 \leq L \)) in the symbol sequence, the inequality in (13) reverses to

\[
N - L + 1 \leq n_2.
\]

This is the condition where only \( u_1 \) appears in \( U_n^{(N-L+1)} \). Therefore, according to Fact 1, the symbol sequence is not identifiable. In this case, as described by Liu, the symbol sequence can be written as

\[
s_k = \sum_{i=1}^{n_1} a_i z_k^{-i}
\]

for \( k = 1, \ldots, N \). Liu suggests treating this as a frequency estimation problem to determine the modes \( z_k \) and then exploiting the FA property to determine the amplitudes. Note that the fundamental system provides the modes as the roots of \( u_1(\lambda) \). Additionally, if \( u_2(\lambda) \) is known, then the sequence is completely specified (up to a multiplicative constant). In the method we propose later, we track both \( u_1 \) and \( u_2 \). Therefore, even if this degenerate condition occurs, we will still be able to recover the sequence. The fundamental system will adapt naturally to the number of modes in the symbol sequence.

F. Information Available in \( u_2 \)

As discussed above, it is not possible to determine the whole symbol sequence knowing only \( u_1 \). We might ask what knowing \( u_2 \) does tell us. Assume that \( N - L + 1 \leq n_2 \) and that the last element of \( u_1 \) is nonzero. Then, we have

\[
S_n^{(N)} U(u_2) = [s_{n-N+1}, \ldots, s_n] U(u_1) = 0
\]

where the \( N \times N - n_1 \) matrix \( U(u_1) \) is given by

\[
U(u_1) = \begin{bmatrix}
\mathbf{u}_1 & \mathbf{u}_2 \\
\mathbf{u}_4 & \mathbf{u}_5
\end{bmatrix}
\]

Now, partition \( U(u_2) \) according to

\[
U = \begin{bmatrix} C \\ D \end{bmatrix}
\]

where \( C \) is the first \( n_1 \) rows, and \( D \) is the last \( N - n_2 \) rows of \( U(u_2) \). Then, if the first \( n_2 \) symbols are known, the last \( N - n_2 \) symbols can be determined by

\[
\begin{bmatrix}
[s_{n-p+1}, \ldots, s_n] \\
[s_{n-N+1}, \ldots, s_{n-p}]
\end{bmatrix} = -[s_{n-N+1}, \ldots, s_{n-p}] CD^{-1}
\]

where \( p = N + n_1 \). Note that very efficient algorithms exist for inverting \( D \) since it is an upper triangular Toeplitz matrix. Hence, knowing \( u_2 \) and a few symbols can be helpful. This could be used in a batch algorithm. \( u_1 \) can be determined as the common factor of the two least dominant right singular vectors of \( X_n^{[(N+1)/2]} \) for \( N \) odd or from the least dominant right singular vectors of \( X_n^{[(N+1)/2]} \) for \( N \) even.

Furthermore, note that while knowing \( u_1 \) at a time \( n \) may not be enough to identify the symbol sequence \( S(n, N) \), but knowing \( u_1 \) at times \( n, n + 1, \ldots, n + Q - 1 \) may be sufficient to identify \( S(n + Q - 1, N + Q - 1) \). Denote \( u_1 \) and \( n_1 \) at time \( n \) by \( u_1(n) \) and \( n_1(n) \). Then, choose \( Q \) such that

\[
QN - \sum_{i=0}^{Q-1} n_1(n + i) \geq N + Q.
\]

Then

\[
[U_1(n), \ldots, U_1(n+Q-1)] = 0
\]

is sufficient to identify the symbol sequence \( S(n + Q - 1, N + Q - 1) \).

Let \( u_1, u_2 \) be a fundamental system for \( S(n, N) \). In [19], it is shown that if \( p(\lambda) \) satisfies the polynomial congruence

\[
p(\lambda) u_2(\lambda) = 1 \mod u_2(\lambda)
\]

then the first \( N \) Markov parameters of the rational function \( r(\lambda) = p(\lambda) / u_1(\lambda) \) form the sequence \( S(n, N) \), where the degree of \( p(\lambda) \) is less than \( n_2 \). Hence, \( r(\lambda) \) may be rewritten as

\[
r(\lambda) = \sum_{i=0}^{n_1-1} \frac{p_i}{\lambda^i} u_1(\lambda)
\]

where \( p_i \) are the coefficients of \( p(\lambda) \). Denote the first \( N \) Markov parameters of the rational functions \( r_i(\lambda) = u_i(\lambda) / u_1(\lambda) \) by the sequences \( S_i(n, N) = [s_{n-N+1}^i, \ldots, s_n^i] \) for \( i = 0, \ldots, n_1 - 1 \). The sequences \( S_i(n, N) \) form a basis for an \( n_1 \)-dimensional space that contains \( S(n, N) = \sum_{i=0}^{n_1-1} p_i S_i(n, N) \). Hence, \( u_1 \) alone specifies the unknown symbol sequence to within an \( n_1 \)-dimensional subspace. Other known properties of the signal could then be used to resolve the remaining uncertainty.

In summary, although \( u_1 \) does not completely specify the unknown symbol sequence, it does provide a great deal of information. It can be used to recover the last \( N - n_2 \) symbols if the first \( n_2 \) symbols are known. If \( u_1 \) is known for \( Q \) consecutive symbol periods, then the set of \( u_1 \)’s can be used to uniquely recover the extended sequence \( S(n + Q - 1, N + Q - 1) \). Finally, we found that \( u_1 \) specifies the symbol sequence to within an \( n_1 \)-dimensional subspace.
V. KERNEL STRUCTURE UP/DOWNDATING

The fundamental system is easily modified as data are added to or removed from \( S(n, N) \). The process of adding a symbol \( s_{n+1} \) to the end of \( S(n, N) \) is referred to as extension and produces the sequence \( S(n+1, N+1) \). Removing a symbol \( s_{N+1} \) from the beginning of \( S(n, N) \) is referred to as reduction and produces the sequence \( S(n, N-1) \). The recursive kernel structure-based blind equalizer developed in the next section will depend on the ability to efficiently compute a FS for extended and reduced sequences given a FS for the original sequence.

Heinig has given algorithms for computing the FS for \( S(n+1, N+1) \) or \( S(n-1, N-1) \), given the FS for \( S(n, N) \) for both the single source (\( d = 1 \)) [20] and the multisource (\( d > 1 \)) [21] cases. These formulas can easily be manipulated to give fundamental systems for \( S(n, N-1) \), which is important to the blind equalization problem where we want to track the kernel of the symbols in a sliding \( N \)-point window.

A. Heinig’s Algorithm for the \( d = 1 \) Case

This section develops the basic modification formulas for the \( d = 1 \) case following [20]. Suppose a sequence \( S(n, N) \) is given along with its fundamental system \( (u^n_0, u^n_2) \) and characteristic degrees \( (n^n_1, n^n_2) \). Form a new sequence \( S(n+1, N+1) \) that is an extension of \( S(n, N) \) by one new symbol \( s_{n+1} \). We seek update formulas for \( (n^{n+1}_1, n^{n+1}_2) \) and \( (n^{n+1}_1, n^{n+1}_2) \).

First, note that

\[
n^{n+1}_1 + n^{n+1}_2 = N + 1 \quad \text{and} \quad n^{n+1}_1 + n^{n+1}_2 = N + 2.
\]

Define the \( 1 \times k \) vector \( \mathbf{b}^{(k)}(n_k) = [s_{n-k+1}, \ldots, s_n] \), and note that

\[
S^{(k)}(n+1) = \begin{bmatrix} S^{(k)}(n) \\ \mathbf{b}^{(k)}(n+1) \end{bmatrix}
\]

and

\[
S^{(k+1)}(n+1) = \begin{bmatrix} S^{(k)}(n) \\ (\mathbf{b}^{(k)}(n+1))^T \end{bmatrix}.
\]

This implies that

\[
\Lambda^{(k)}(n+1) = \Lambda^{(k)}(n) \cap \ker \mathbf{b}^{(k+1)}(n+1) \subseteq \Lambda^{(k)}(n) \subseteq \Lambda^{(k+1)}(n+1).
\]

The left (right) inclusion gives a lower (upper) bound for \( n^{n+1}_1 \) as follows:

\[
n^{n+1}_1 \leq n^{n+1}_1 \leq n^{n+1}_2.
\]

Together with (14), this implies that either \( n^{n+1}_1 = n^n_2 \), \( n^{n+1}_1 = n^n_2 + 1 \), or \( n^{n+1}_1 = n^n_2 + 2 \). The FS update formulas can be derived by first checking to see if \( (u^n_0, u^n_2) \) is already an FS for \( S(n+1, N+1) \) without any modifications. This can be determined by examining

\[
S^{(n+1)}(n+1) u^{(n+1)}_1 = \begin{bmatrix} S^{(n+1)}(n) \\ \mathbf{b}^{(n+1)}(n+1) \end{bmatrix} u^{n}_1 = \begin{bmatrix} 0 \\ \mathbf{b}^{(n+1)}(n+1) u^{n}_1 \end{bmatrix} = \begin{bmatrix} 0 \\ f^n_1 \end{bmatrix}
\]

and

\[
S^{(n+1)}(n+1) u^{(n+1)}_2 = \begin{bmatrix} S^{(n+1)}(n) \\ \mathbf{b}^{(n+1)}(n+1) \end{bmatrix} u^{n}_2 = \begin{bmatrix} 0 \\ \mathbf{b}^{(n+1)}(n+1) u^{n}_2 \end{bmatrix} = \begin{bmatrix} 0 \\ f^n_2 \end{bmatrix}.
\]

If \( f^n_1 = 0 \), then we find that \( n^{n+1}_1 = n^n_1 \), and \( u^{(n+1)}_1 \) is in the FS of \( S(n+1, N+1) \). In addition, the degree of \( u^{(n+1)}_2 \) must increase by (14) so that \( n^{n+1}_2 = n^n_2 + 1 \). Hence, \( u^{(n+1)}_2 \) must satisfy

\[
0 = S^{(n+1)}(n+1) u^{(n+1)}_2 = S^{(n+1)}(n+1) \mathbf{b}^{(n+1)}(n+1) u^{n+1}_2
\]

\[
= \left[ S^{(n+1)}(n+1) \left( (\mathbf{b}^{(n+1)+1})^T \right) \right] u^{n+1}_2.
\]

Hence, the simplest modification is

\[
u^{n+1}_2 = \begin{bmatrix} u^n_0 \\ 0 \end{bmatrix} \quad \leftrightarrow \quad u^{n+1}_2(\lambda) = u^n_2(\lambda).
\]

On the other hand, if \( f^n_1 \neq 0 \) but \( f^n_2 = 0 \), then the same type of modification occurs as in the \( f^n_1 = 0 \) case, except that \( (n^n_1, u^n_1) \) and \( (n^n_2, u^n_2) \) trade places. The final case to consider is where \( f^n_1 \neq 0 \) and \( f^n_2 \neq 0 \); then, the following update is sensible:

\[
\begin{align*}
(n^{n+1}_1, n^{n+1}_2) &= (n^n_1, n^n_2) + (1, 0) \\
u^{n+1}_1 &= \begin{bmatrix} u^n_1 \\ 0 \end{bmatrix} \quad \leftrightarrow \quad u^{n+1}_1(\lambda) = u^n_1(\lambda) \\
u^{n+1}_2 &= u^n_2 - \frac{f^n_2}{f^n_1} \begin{bmatrix} 0 \\ u^n_0 \end{bmatrix} \quad \leftrightarrow \quad u^{n+1}_2(\lambda) = u^n_2(\lambda) - p(\lambda)
\end{align*}
\]

where \( p(\lambda) = (f^n_2/f^n_1)\lambda^{n^n_2-n^n_1} u^n_0(\lambda) \). It is easy to see that this works. The \( u^n_1 \) update is obvious. The \( u^n_2 \) update deserves more attention.

\[
S^{(n+1)}(n+1) u^{n+1}_2 = S^{(n+1)}(n+1) u^{n+1}_2
\]

\[
= \left[ S^{(n+1)}(n) \left( (\mathbf{b}^{(n+1)+1})^T \right) \right] u^n_2 - \frac{f^n_2}{f^n_1} \begin{bmatrix} 0 \\ \mathbf{b}^{(n+1)+1} u^n_2 \end{bmatrix}
\]

\[
= \begin{bmatrix} 0 \\ f^n_1 \end{bmatrix}.
\]

In any of these cases, at the end of the update, always reorder the FS to ensure that the characteristic degrees are in nondecreasing order. Downdating formulas for reduction can be derived in a similar way and are not discussed further. To initialize Heinig’s algorithm, set \( (n^n_1, n^n_2) = (0, 1) \) and \( (u^n_1(\lambda), u^n_2(\lambda)) = (1, \lambda) \). This is the fundamental system for \( S(1, 1) = s_1 \). Now, apply Heinig’s extension and reduction formulas to build up the fundamental system for \( S(n, N) \).

These modification formulas were originally proposed as algorithms for fast inversion of Hankel and block Hankel matrices and can be applied to Toeplitz matrices as well. The formulas derived above are of the Levinson-type since they require an inner product to compute \( f^n_1 \). There are also Schur-type formulas that work with a “residual system” and are inner product free. Table I gives Heinig’s Levinson-type extension and reduction algorithms for the single source case. The important thing to note is the simplicity of the update. By itself, Heinig’s algorithm is numerically unstable. In practice, the FS must be scaled periodically. In addition, we replace the condition \( f^n_1 = 0 \) with \( |f^n_1| < \epsilon \) for some small \( \epsilon \).
TABLE I
Heinig’s Algorithm for the Case \(d = 1\)

Heinig’s Algorithm (Levinson-Type, \(d = 1\))

**Extension by \(s_{n+1}\)**

**Input:** a FS of \(S(n, N)\) and \(s_{n+1}\).

**Output:** a FS of \(S(n+1, N+1)\).

Let \(f_1 = [s_{n-1}, \cdots, s_{n+1}] u_1\),

\[
f_2 = [s_{n-2}, \cdots, s_{n+1}] u_2.
\]

\[
\begin{bmatrix}
    u_1(\lambda) & u_2(\lambda) \\
    n_1 & n_2
\end{bmatrix}
\leftarrow
\begin{bmatrix}
    u_1(\lambda) & u_2(\lambda) \\
    1 & c(\lambda) \\
    0 & 1
\end{bmatrix}
\begin{bmatrix}
    n_1 & n_2 + \eta
\end{bmatrix}
\]

If \(f_1 = 0\) then \(c(\lambda) = 0\) \(\eta = [0, 1]\)

Else if \(f_2 = 0\) then \(c(\lambda) = 0\) \(\eta = [1, 0]\)

Else \(c(\lambda) = -\frac{1}{e_1} \lambda^{n_2-n_1}\) \(\eta = [1, 0]\)

**Reduction by \(s_{n-N+1}\)**

**Input:** a FS of \(S(n, N)\).

**Output:** a FS of \(S(n, N-1) = \{s_{n-1}, \cdots, s_n\}\).

Let \(e_1 = u_{i,0}, e_2 = u_{i,0}\) and let \(u_i(\lambda) = \lambda^{n_i} u_i(\lambda^{-1})\) be the reciprocal polynomial of \(u_i(\lambda)\).

\[
\begin{bmatrix}
    u_1(\lambda) & u_2(\lambda) \\
    n_1 & n_2
\end{bmatrix}
\leftarrow
\begin{bmatrix}
    u_1(\lambda) & u_2(\lambda) \\
    1 & c(\lambda) \\
    0 & 1
\end{bmatrix}
\begin{bmatrix}
    n_1 & n_2 + \eta
\end{bmatrix}
\]

If \(e_1 = 0\) then \(c(\lambda) = 0\) \(\eta = [-1,0]\)

Else if \(e_2 = 0\) then \(c(\lambda) = 0\) \(\eta = [0,-1]\)

Else \(c(\lambda) = -\frac{1}{e_1} \lambda^{n_2-n_1}\) \(\eta = [0,-1]\)

**B. Heinig’s Algorithm for the \(d > 1\) Case**

For completeness, we present the generalization of Heinig’s algorithm to the multiple source case \((d > 1)\) and assume that the rows of \(S_n^{(N)}\) are linearly independent. The following is taken from [21]. We start with some definitions. Let \(u_i(\lambda)\) be the \(i\)th polynomial \((i = 1, \cdots, d+1)\) in the FS of the \(d \times 1\) vector sequence \(S(n, N)\), and let \(u_{i,j}\) be the coefficient of \(\lambda^j, j = 0, \cdots, n_i\) in \(u_i(\lambda)\). Define the fundamental matrix \(U(\lambda)\) to be the matrix with the \(u_i(\lambda)\) as its columns

\[
U(\lambda) = [u_1(\lambda) \cdots u_{d+1}(\lambda)].
\]

For purposes of reducing the vector sequence, the matrix \(E\) defined by

\[
E = [u_{1,0} \cdots u_{d+1,0}]
\]

will be needed. For extending the vector sequence, define \(F\) to be the matrix

\[
F = [f_1 \cdots f_{d+1}] \quad \text{where} \quad f_i = b_{n+i}^{(n+1)} u_i.
\]

As in the single-source case, the \(f_i\) indicate whether or not \(u_i\) appears unmodified in the fundamental system of the extended sequence \(S(n+1, N+1)\). In the recursions, it will often be the case that some of the columns of \(F\) and \(E\) will be zero. If the nonzero columns of a matrix form a basis of its column space, then the matrix is said to be column regular. Denote the set of indices of the nonzero columns of a column regular matrix by \(J\), which is called the set of proper indices. If \(F\) (or \(E\)) is not column regular, using Gaussian elimination, an upper triangular matrix \(T\) can be constructed so that \(FT\) (or \(ET\)) is column regular. The following theorem shows how to modify the fundamental matrix as data are added and removed from the vector sequence.

**Theorem 3:** Let \(U(\lambda)\) be a fundamental matrix of the \(d \times 1\) vector sequence \(S(n, N)\) with \(S_n^{(N)}\) full row rank.

**Reduction by \(s_{n-N+1}\):** Construct \(T\) such that \(ET\) is column regular with proper indices \(J\). Then, \(U \Lambda^{-1} T A\) is a fundamental matrix of \(S(n, N-1)\), where \(\Lambda = \text{diag}(\lambda^{n_1}, \cdots, \lambda^{n_{d+1}})\). Let \(n_i = n_i - 1\) for \(i \in J\).

**Extension by \(s_{n+1}\):** Construct \(T\) such that \(FT\) is column regular with proper indices \(J\). Then, \(UA^{-1} T A\) is a fundamental matrix of \(S(n+1, N+1)\), where \(\Lambda = \text{diag}(\lambda^{n_1}, \cdots, \lambda^{n_{d+1}})\). Let \(n_i = n_i + 1\) for \(i \in J\).

**Proof:** See [21]. Note the recursions in Theorem 3 reduce to the recursions given above for the \(d = 1\) case. As before, at the end of an extension or reduction, the FS may need to be reordered to preserve the nondecreasing order of the characteristic degrees.

**VI. Recursive Blind Equalization Using Structured Kernels**

Now, we combine the results from the previous two sections to develop an efficient recursive equalizer. For simplicity, we will present the single source case. To begin, we assume that \(N\) is chosen large enough so that \(S_n^{(N-L+1)}\) is wide. We also assume that \(H\) has full column rank. One of the advantages of the scheme we propose is that it adapts naturally to the number of modes in the signals.

By Corollary 1, for \(d = 1\), the columns of

\[
U_n^{(N-L+1)} = \begin{bmatrix}
    u_{1,0} & u_{2,0} \\
    \vdots & \vdots \\
    u_{1,n_1} & u_{2,n_2}
\end{bmatrix}
\]

form a basis for \(\Lambda_n^{(N-L+1)}\). When no noise is present, it follows from (3) that

\[
X_n^{(N-L+1)} U_n^{(N-L+1)} = 0 \quad \Rightarrow \quad S_n^{(N-L+1)} U_n^{(N-L+1)} = 0.
\]

Next, we show how to exploit the finite alphabet property of the signal. For simplicity, assume that the symbol alphabet is \(\{+1, -1\}\). Given an FS for \(S(n, N)\), we compute, using Heinig’s extension algorithm, an FS for a \(+1\) extension \(S(n+1, N+1)\).
1, \(N+1, +1\) and a \(-1\) extension \(S(n+1, N+1, -1)\). This leads to the shift chains \(U_{n+1}^{(N-L+2)}(1)\) and \(U_{n+1}^{(N-L+1)}(-1)\), respectively. One of these extensions is correct (i.e., corresponds to the sequence actually transmitted). Define \(\tau(s_{n+1})\) by
\[
\tau(s_{n+1}) = \left| \frac{X_{n+1}^{(N-L+2)}U_{n+1}^{(N-L+2)}(s_{n+1})}{X_{n+1}^{(N-L+1)}U_{n+1}^{(N-L+1)}(s_{n+1})} \right|^2 \tag{15}
\]
and suppose that the \(+1\) extension is actually the correct extension (i.e., \(s_{n+1} = +1\)). Then, with no noise, we will have
\[
\tau(+1) = 0 \quad \text{and} \quad \tau(-1) > 0.
\]
Therefore, the decision is easy, at least when there is no noise, since the subspace relation (3) holds exactly. In practice, we compute \(\tau(s_{n+1})\) and decide \(s_{n+1} = +1\) if \(\tau(+1) < \tau(-1)\) and \(s_{n+1} = -1\), otherwise. Additionally, we should point out that Heinig’s algorithm gives an FS that is not scaled appropriately for comparison of \(\tau(+1)\) with \(\tau(-1)\). For this reason, we replace \(U_{n+1}^{(N-L+2)}(s_{n+1})\) with the \(Q\) factor from its QR factorization, which can be computed very efficiently by exploiting displacement structure [22]. Before moving on to \(n+2\), we execute a reduction step via Heinig’s reduction algorithm to remove \(s_{n-N+1}\). Hence, we track the FS for the most recent \(N\) samples of the symbol sequence.

The shift structure of the data has been fully exploited by using the shift chain basis of the null space. Furthermore, the FA property of the signal has been exploited by considering only feasible extensions. For arbitrary \(K\) element alphabets \(\Omega\), a total of \(K\) extensions must be computed (\(Ked\) for \(d\) signals). However, because the fundamental system is a parsimonious parameterization of the null space and can be extended and reduced very easily using Heinig’s algorithm, this results in a very efficient algorithm. The largest expense in this algorithm is the matrix product in the computation of \(\tau(s_{n+1})\).

We place no lower limit on the number of modes in the signals. Because we track the fundamental system of the exact symbol matrix, our algorithm can accommodate any number of signal modes from the minimum of 1 up to the maximum \([N + 1]/2\]. This makes the algorithm useful as a blind start up procedure that could be used to equalize at the very beginning of transmission without a channel estimate or training signals. Consider the \(d = 1\) case. We can assume without loss of generality that the very first symbol transmitted is \(s_0 = +1\) and \(s_i = 0\) for \(i < 0\). Initialize Heinig’s algorithm with \([n_1, n_2] = [1, L]\) and \([u_1(\lambda), u_2(\lambda)] = [1, \lambda^L]\). Now, apply Heinig’s extension algorithm to the data as it arrives in \(X_{i}^{(N+1)}\) for \(i = 1, 2, \ldots\) until the desired window size is reached, and then, interleave reductions with the extensions. At each step, we compare \(\tau(+1)\) and \(\tau(-1)\) to make a decision.

Thus far, we have assumed that the number of sources \(d\) and the channel length \(L\) were known. The algorithm, however, is not very sensitive to an incorrectly chosen \(L\). If \(L\) is underestimated, then fewer shifts are used, which does not significantly affect the performance. Overestimating \(L\), however, does impact the performance since more shifts are assumed in the data than are actually present. Therefore, in practice, care should be taken to ensure that \(L\) is not overestimated.

A fundamental limitation of this algorithm occurs due to noise amplification. Computing \(\tau(s_{n+1})\) amplifies the noise in the space spanned by the columns of \(U_{n+1}^{(N-L+2)}(s_{n+1})\). Denote this space by \(N(s_{n+1})\). Even if \(s_{n+1} = +1\) is correct, the noise power in \(N(+1)\) might be greater than that in \(N(-1)\), leading to \(\tau(+1) > \tau(-1)\) and an incorrect decision. As with any equalizer with decision feedback, a propagating error condition can occur. We suggest several remedies for this situation.

**Remedy 1:** If the noise is white, we would expect its projection onto \(N(+1)\) and \(N(-1)\) to be similar in “size” so that we would expect a large difference between \(\tau(+1)\) and \(\tau(-1)\). If the difference is small relative to some threshold \(T\)
\[
|\tau(+1) - \tau(-1)| < T
\]
then we cannot with certainty make a decision about \(s_{n+1}\). We could then delay our decision until the next symbol period by extending the fundamental systems and computing \(\tau(+1, +1)\), \(\tau(+1, -1)\), \(\tau(-1, +1)\), \(\tau(-1, -1)\). Then, a decision about the pair \([s_{n+1}, s_{n+2}]\) could be made by choosing the smallest \(\tau\).

**Remedy 2:** We could continue delaying our decision about \(s_{n+1}\) as in Remedy 1 until \(n-M\) by computing the FS for all \(2^M\) possible \([\pm 1, \ldots, \pm 1]\). Then, a decision about \(s_{n+1}\) could be made based on choosing the smallest \(\tau\) or comparing the sets \(\tau(+1, \pm 1, \ldots, \pm 1)\) with \(\tau(-1, \pm 1, \ldots, \pm 1)\) in which the sequences in each set are constant on \(s_{n+1}\). Either alternative could be implemented efficiently in a recursive manner using the Viterbi algorithm with \(\tau(\cdot)\) replacing the branch metric.

**Remedy 3:** If the transmitted symbol stream is actually encoded, then the code structure could be exploited as in [23]. Suppose the source signals are convolutionally encoded. Then, when making decisions at each node in the trellis at the receiver, the fundamental systems are extended according to the sequences labeling each branch. Survivor branches are chosen to be those that have the lowest cost (15). Exploiting code structure makes the algorithm more efficient since the kernel structure updates are easy, and the expensive cost function evaluations are done less often.

### VII. Simulations

Previously, we suggested that the proposed algorithm could be used to equalize at the beginning of transmission without a channel estimate and without a training sequence. This blind “start-up” feature distinguishes our algorithm from other recursive algorithms. Batch processing methods, which can also be used for blind startup, require at least 20–30 samples and can be very computationally demanding. In contrast, our efficient recursive method begins equalizing from the very start.

We simulated reception of a single source over a length \(L = 4\) channel. We assume that \(MP = 6\). The fundamental system was initialized as \([n_1, n_2] = [1, L]\) and \([u_1(\lambda), u_2(\lambda)] = [1, \lambda^L]\). Possible signal kernels were tracked as suggested in Remedy 2 using the Viterbi algorithm and a \(2^L\) state trellis. The results are displayed in Fig. 3. For comparison, we also give simulation results obtained using a block method for blind symbol estimation proposed by van der Veen et al. [12] and Liu et al. [13]. The bit error rate (BER) is plotted on a per sample basis. Observe that in most instances, the recursive kernel structure equalizer (with Viterbi algorithm) has a lower BER than the batch equalizer. The batch equalizer curves...
symbol periods, and an array of sensors errors above 15 dB.

The kernel structure equalizer exploiting the code did not make any code structure. This matches the results in [23]. Note that the structure equalizer performed much better when exploiting the at several SNR’s. The results are plotted in Fig. 4. The kernel was used to observe the signal. The window length was length

ally encoded using the rate 1/2 binary convolutional encoder exploiting the code structure. A single source was convolution-

samples. The BER’s were computed by averaging over 10^5 trials these symbols in the batch equalizer.

sequence. This is due to the higher variance in the estimates of (dotted) also show the error rate increases in the “ends” of the sequence. This is due to the higher variance in the estimates of these symbols in the batch equalizer.

A second simulation was performed that compares the per-

Fig. 3. Kernel structure recursive blind equalizer in “start-up” mode.

Fig. 4. Kernel structure recursive blind equalizer with and without exploiting code structure.

VIII. CONCLUSIONS

This paper presents a novel approach to the blind equaliza-

REFERENCES


[2] D. N. Godard, “Self-recovering equalization and carrier tracking in two-


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*. 