

# Factor Graph Algorithms for Equalization

Robert J. Drost\* and Andrew C. Singer, *Senior Member, IEEE*

## Abstract

We use the factor graph framework to describe the statistical relationships that arise in the equalization of data transmitted over an intersymbol interference channel, and use it to develop several new algorithms for linear and decision feedback approaches. Specifically, we examine both unconstrained and constrained linear equalization and decision feedback equalization of a sequence of nonidentically distributed symbols that is transmitted over a linear, possibly time-varying, finite-length channel and then corrupted by additive white noise. Factor graphs are used to derive algorithms for each of these equalization tasks, including fast implementations. One important application of these algorithms is linear turbo equalization, which requires a linear equalizer that can process observations of nonidentically distributed transmitted symbols. We show how the output of these factor graph-based algorithms can be used in an efficient implementation of a linear turbo equalizer.

## Index Terms

Factor graphs, equalizers, intersymbol interference, turbo equalization, iterative methods

## I. INTRODUCTION

We examine the use of factor graphs to develop fast equalization algorithms for data with *a priori* information that have been transmitted over a known linear, possibly time-varying, finite-length channel and then corrupted by additive white noise. As in turbo equalization, we consider the equalization problem given both the received channel output as well as a set of priors over the transmitted symbols (soft information). The estimation of a transmitted sequence from the output of such a channel is a fundamental problem in communications [1], and there is much research into algorithms that not only perform well in

The authors are with the Coordinated Science Laboratory and the Department of Electrical and Computer Engineering at the University of Illinois at Urbana-Champaign, 1308 West Main Street, Urbana, IL 61801. Email: {drost,singer}@ifp.uiuc.edu

This work was supported in part by the Department of the Navy, Office of Naval Research, under grant N00014-01-1-0117 and by the National Science Foundation under grant CCR-0092598 (CAREER)

some sense, but that are also computationally efficient. While the maximum *a posteriori* (MAP) estimate can be computed and is optimal in the sense of minimizing the probability of bit error, the computational complexity of forming this estimate can be restrictive even when using an efficient implementation such as the BCJR algorithm [2]. Hence, approximations to the MAP estimate have been developed to reduce the computational burden [3]. Alternatively, criteria other than probability of bit error can be employed, such as minimum mean squared error (MMSE). Unfortunately, optimizing the MMSE between the transmitted sequence and the estimated sequence can still be computationally complex. However, by restricting the estimates to be linear functions of the received data, the solution simplifies significantly [1]. It is this problem of efficiently finding the linear minimum mean squared error (LMMSE) estimate of the transmitted sequence that we address, and to this end, we employ the factor graph framework.

Factor graphs are a relatively new modeling framework that have proven useful in a wide variety of applications [4]. By depicting the factorization of a global function of several variables as the product of several functions, each of a subset of the original set of variables, factor graphs can provide the foundation for a divide-and-conquer strategy to the problem at hand. The sum-product algorithm is one algorithm that exploits the structure depicted in a factor graph to yield low complexity algorithms. The sum-product algorithm operates on a factor graph to yield either exact or approximate marginal functions, depending on whether or not the graph is a tree [4]. This algorithm has successfully been applied to a host of problems spanning many areas including error correction decoding [5], Kalman filtering [6], image segmentation [7], phase unwrapping [8], and network tomography [9], and will be the basis for the algorithms presented here. Consequently, we provide a brief overview of factor graphs in Section II.

In Sections III-V, we develop three LMMSE optimal equalization algorithms. The first is an unconstrained equalizer, i.e., estimates of the transmitted data are formed that are linear functions of the entire received sequence. The second algorithm is a constrained complexity linear equalizer, in which only a particular subset of observations are considered when estimating a given symbol. Finally, we develop a decision feedback equalizer (DFE) which makes hard decisions based on constrained LMMSE estimates that were formed assuming prior decisions were correct.

In each case, we detail fast implementations of the algorithms and describe the number of computations they require. As for the performance of each algorithm, we note that because the factor graphs that we employ are trees, the sum-product algorithm is guaranteed to converge in a finite number of steps and will recover exactly the estimates of the well-known unconstrained and constrained LMMSE equalizers and the DFE. So, we simply refer the reader to [1] for details on the performance of the algorithms.

The theoretical significance of deriving these equalizers within the factor graph framework is twofold. First, the range of problems to which factor graphs can be applied is extended, further indicating the broad applicability of this framework. Second, the method by which the algorithms are derived provides an interpretation of LMMSE estimation as MAP estimation based on an approximate model. This not only provides additional insight into LMMSE equalization, but also demonstrates a relationship between the algorithms presented here that might not be apparent when viewed outside of the factor graph framework.

The algorithms are also of practical use given the fundamental importance of equalization in the area of communications. That these equalizers can be employed in situations where the communication channel is time-varying or the priors are nonidentical demonstrates their broad applicability. For example, in turbo equalization, it is necessary to consider nonidentical priors, even when the channel is time-invariant [10]. Consider a coded binary sequence that is transmitted over an intersymbol interference channel. In turbo equalization, an equalizer and a decoder exchange information in the form of log-likelihood ratios (LLRs) of each bit. The LLRs generated by the decoder are treated as *a priori* information by the equalizer, resulting, in general, in time-varying priors [10]. Hence, the fast algorithms presented here are ideally suited for this problem when a linear equalizer is desired. In Section VI, we show how the equalizer algorithms can be used in turbo equalization, addressing some issues that arise in efficiently converting the symbol estimates into LLRs.

It should be noted that a fast constrained equalizer and a fast DFE for turbo equalization are considered in [10] and [11]. However, while we do address turbo equalization, our primary goal is to demonstrate how the factor graph framework can be used to derive fast numerically stable algorithms for general equalization. The factor graph framework is not considered in [10] or [11], but instead fast algorithms

are developed based on an algebraic approach specifically in the context of turbo equalization. Also, the unconstrained case was not considered, and the DFE that we describe in Section V-A is more general than the DFE considered in [10] and [11] in that decisions can be made in any order and any chosen set of decisions can be incorporated into the estimate of a particular bit.

### A. Problem Setup

Throughout this paper, we consider the following discrete-time equivalent baseband model of the communication channel. The transmitted sequence  $x_n$ ,  $1 \leq n \leq L$ , where  $L$  is the arbitrary but known data sequence length, consists of independent bits drawn from a discrete alphabet  $\mathcal{B}$  according to *a priori* probability density functions  $f_n(x_n)$ . We describe the time-varying intersymbol interference of the channel by the sequence of channel response vectors  $\mathbf{h}_n$ ,  $1 \leq n \leq L + N - 1$ , where  $N$  is the arbitrary but known channel length,  $\mathbf{h}_n \in \mathbb{R}^n$  if  $n < N$ ,  $\mathbf{h}_n \in \mathbb{R}^N$  if  $N \leq n \leq L$ , and  $\mathbf{h}_n \in \mathbb{R}^{L+N-n}$  if  $n > L$ . Denote  $\mathbf{x}_n = [x_1 \ x_2 \ \dots \ x_n]^T$  if  $1 \leq n < N$ ,  $\mathbf{x}_n = [x_{n-N+1} \ x_{n-N+2} \ \dots \ x_n]^T$  if  $N \leq n \leq L$ , and  $\mathbf{x}_n = [x_{n-N+1} \ x_{n-N+2} \ \dots \ x_L]^T$  if  $n > L$ . Next, let  $w_n$ ,  $1 \leq n \leq L + N - 1$ , be a zero-mean additive white noise sequence of variance  $\sigma_n^2$ . The received sequence  $y_n$ ,  $1 \leq n \leq L + N - 1$ , is then given by  $y_n = \mathbf{h}_n^T \mathbf{x}_n + w_n$ . Finally, in the case of the constrained equalizer and the DFE, we denote the arbitrary but predetermined equalizer filter length by  $M$ .

Note that we set the form of the received sequence at the boundaries so as to be able to provide complete descriptions of the algorithms presented, but modifications to the algorithms can be made for other observation models. For example, though we consider observations extending past the end of the transmitted data sequence, which will reduce the mean squared error of our algorithms in the absence of model inaccuracy, the algorithms can be modified to only consider observations up to time  $L$ . Also, throughout this paper, we typically address the region of the data not affected by the boundaries of the transmitted sequence, denoting such an arbitrary time instant by  $n$ . This is especially the case when the modifications for the boundaries are straightforward. However, the implementations of the algorithms that we provide do appropriately take into account these boundaries.

## II. BACKGROUND

### A. Factor Graphs and the Sum-Product Algorithm

The following introduction to factor graphs and the sum-product algorithm is based largely on the material and notation of [4], to which we refer the reader for a more in-depth review.

Let  $X = \{x_1, x_2, \dots, x_L\}$  be a set of variables and let  $f(X)$  be a function of the variables in  $X$ . Suppose there exist functions  $g_k(X_k)$ ,  $1 \leq k \leq K$ , where  $X_k \subset X$ , such that

$$f(X) = \prod_{k=1}^K g_k(X_k). \quad (1)$$

A factor graph  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$  is an undirected bipartite graph that depicts the factorization of (1), where  $\mathcal{V} = X \cup \{g_1, g_2, \dots, g_N\}$  and  $\mathcal{E} = \{(g_k, x_l) : x_l \in X_k\}$ . (Note that the distinction between a vertex and the corresponding variable or function is usually ignored as the meaning should be clear from context.) Often in a graphical depiction, variable nodes, vertices corresponding to variables, are drawn as open circles while function nodes, vertices corresponding to functions, are drawn as filled circles.

The sum-product algorithm operates on a factor graph, producing either exact or approximate marginal functions depending on whether or not the graph is a tree. This algorithm is often described as a message passing scheme in which messages are sent along the edges of the factor graph. These messages are functions of the variable node incident on a particular edge. The message update rules for functions with discrete domains follow.

To be precise, given a variable or function, we denote the corresponding node by placing a dot over the variable or function so as to explicitly distinguish between the two. For example, the node corresponding to variable  $x$  will be denoted  $\dot{x}$ . Let  $m_{\dot{a} \rightarrow \dot{b}}$  denote the message passed from node  $\dot{a}$  to node  $\dot{b}$ , and let  $\mathcal{N}(\dot{n})$  denote the set of nodes adjacent to node  $\dot{n}$ . Then a message being sent along an edge from a variable node  $\dot{x}$  to a function node  $\dot{g}$  takes the form

$$m_{\dot{x} \rightarrow \dot{g}}(x) = \prod_{\dot{n} \in \mathcal{N}(\dot{x}) \setminus \{\dot{g}\}} m_{\dot{n} \rightarrow \dot{x}}(x), \quad (2)$$

and a message being sent along an edge from a function node  $\dot{g}$  to a variable node  $\dot{x}_k$  takes the form

$$m_{\dot{g} \rightarrow \dot{x}_k}(x_k) = \sum_{\sim \{x_k\}} \left( g(X) \prod_{\dot{x}_l \in \mathcal{N}(\dot{g}) \setminus \{\dot{x}_k\}} m_{\dot{x}_l \rightarrow \dot{g}}(x_l) \right), \quad (3)$$

where  $X$  is the set of arguments of the function  $g$ , and  $\sum_{\sim\{x_k\}}$  indicates the summation over all configurations of variables in  $X \setminus \{x_k\}$  as a function of  $x_k$  [4]. To handle factor graphs that model functions defined on a continuous domain, the above update equations must be modified to reflect that some sets of variables need to be integrated rather than summed.

After initializing all messages to the unity function, the sum-product algorithm proceeds by forming messages as above. (In practice, each message usually needs to be computed only to within a positive multiplicative constant, since the effect of the constant can often be easily compensated for in the output of the algorithm.) If the factor graph is a tree, as are all the factor graphs in this paper, the messages will cease to change after a finite number of iterations, signaling the termination of the algorithm. There is freedom in choosing the order in which messages are generated, but in the case of a tree, propagating messages starting from leaf nodes yields the most efficient schedule. The output marginal function for a particular variable is obtained as the product of incoming messages to that variable node.

In this paper, we will be interested in the case where all function nodes correspond to functions that are Gaussian, i.e., quadratic exponential functions, or degenerate forms of Gaussians, e.g., the Dirac delta function and constant functions. For conciseness, we introduce the function

$$\gamma(\mathbf{x}, \boldsymbol{\mu}, \mathbf{W}) = \exp \left\{ -\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^T \mathbf{W}(\mathbf{x} - \boldsymbol{\mu}) \right\}, \quad (4)$$

where  $\mathbf{x}$  and  $\boldsymbol{\mu}$  are vectors of some length  $m$  and  $\mathbf{W}$  is an  $m \times m$  matrix. Note that when  $\mathbf{W}$  is positive definite,  $\gamma$  is a scaled Gaussian distribution with mean vector  $\boldsymbol{\mu}$  and covariance matrix  $\mathbf{W}^{-1}$ . Also note that a constant function is a degenerate form of (4) in which  $\mathbf{W}$  is taken to be a zero matrix.

As discussed in [6], when a factor graph represents a Gaussian distribution or a degenerate form, all messages will also be (possibly degenerate) Gaussians. This is in part because the product of two functions of the form in (4) maintains the same form, as does the marginalization of such a function over some set of variables. In particular, we have

$$\gamma(\mathbf{x}, \boldsymbol{\mu}_1, \mathbf{W}_1)\gamma(\mathbf{x}, \boldsymbol{\mu}_2, \mathbf{W}_2) \propto \gamma(\mathbf{x}, (\mathbf{W}_1 + \mathbf{W}_2)^\dagger(\mathbf{W}_1\boldsymbol{\mu}_1 + \mathbf{W}_2\boldsymbol{\mu}_2), \mathbf{W}_1 + \mathbf{W}_2), \quad (5)$$

and

$$\int_{-\infty}^{\infty} \gamma \left( \left[ \begin{array}{c} \mathbf{x} \\ \mathbf{y} \end{array} \right], \left[ \begin{array}{c} \boldsymbol{\mu}_x \\ \boldsymbol{\mu}_y \end{array} \right], \left[ \begin{array}{c|c} \mathbf{W}_x & \mathbf{W}_{xy} \\ \hline \mathbf{W}_{yx} & \mathbf{W}_y \end{array} \right] \right) d\mathbf{x} \propto \gamma(\mathbf{y}, \boldsymbol{\mu}_y, \mathbf{W}_y - \mathbf{W}_{yx} \mathbf{W}_x^\dagger \mathbf{W}_{xy}), \quad (6)$$

where  $\mathbf{W}^\dagger$  is the pseudoinverse of  $\mathbf{W}$  [6]. (Throughout this paper, the term pseudoinverse refers to the Moore-Penrose pseudoinverse.) We also make use of the following:

$$\gamma(\mathbf{x}, \boldsymbol{\mu}_x, \mathbf{W}_x) \gamma(\mathbf{y}, \mathbf{h}^T \mathbf{x}, W_y) \propto \gamma(\mathbf{x}, \boldsymbol{\mu}, \mathbf{W}), \quad (7)$$

where  $\mathbf{W} = \mathbf{W}_x + \mathbf{h} W_y \mathbf{h}^T$ ,  $\boldsymbol{\mu} = \mathbf{W}^\dagger (\mathbf{W}_x \boldsymbol{\mu}_x + W_y y \mathbf{h})$ , and  $y$  is a constant parameter.

### B. Matrix Inversion

We make frequent use of two matrix inversion formulas. Let  $\mathbf{W}$  be an invertible matrix that can be written as  $\mathbf{W} = \mathbf{A} + \mathbf{BCD}$  with  $\mathbf{A}$  and  $\mathbf{C}$  invertible. Then the well-known matrix inversion lemma [12] gives  $\mathbf{W}^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1} \mathbf{B} (\mathbf{C}^{-1} + \mathbf{D} \mathbf{A}^{-1} \mathbf{B})^{-1} \mathbf{D} \mathbf{A}^{-1}$ . We also make use of the block matrix inversion formula. Let

$$\mathbf{W} = \left[ \begin{array}{c|c} \mathbf{A} & \mathbf{B} \\ \hline \mathbf{C} & \mathbf{D} \end{array} \right], \quad \text{and} \quad \mathbf{W}^{-1} = \left[ \begin{array}{c|c} \mathbf{E} & \mathbf{F} \\ \hline \mathbf{G} & \mathbf{H} \end{array} \right],$$

where we assume that  $\mathbf{W}$  is invertible. Then,  $\mathbf{E} = (\mathbf{A} - \mathbf{B} \mathbf{D}^{-1} \mathbf{C})^{-1}$ ,  $\mathbf{F} = -\mathbf{A}^{-1} \mathbf{B} (\mathbf{D} - \mathbf{C} \mathbf{A}^{-1} \mathbf{B})^{-1}$ ,  $\mathbf{G} = -(\mathbf{D} - \mathbf{C} \mathbf{A}^{-1} \mathbf{B})^{-1} \mathbf{C} \mathbf{A}^{-1}$ , and  $\mathbf{H} = (\mathbf{D} - \mathbf{C} \mathbf{A}^{-1} \mathbf{B})^{-1}$ , provided the required inverses exist [12].

## III. UNCONSTRAINED LINEAR EQUALIZATION

In this section, we present unconstrained linear equalization algorithms based on the factor graph framework. Given the observed sequence  $y = \{y_1, y_2, \dots, y_{L+N-1}\}$ , we wish to find the LMMSE estimate of the transmitted sequence  $x = \{x_1, x_2, \dots, x_L\}$  based on the observation model outlined in Section I-A. In the standard interpretation of an LMMSE equalizer, this amounts to determining the set of functions  $g_n : \mathbb{R}^{L+N-1} \rightarrow \mathbb{R}$  that minimize  $\mathbb{E}\{[x_n - g_n(y)]^2\}$  for each  $n$ , where, for complexity reduction,  $g_n$  is constrained to be an affine function of the data ( $\mathbb{E}\{\cdot\}$  denotes expectation.)

We take a different view of linear equalization as follows. The observation model implies the following factorization of the *a posteriori* distribution of the transmitted sequence  $x$  given the received sequence  $y$ :

$$f(x|y) = \frac{1}{Z} \left[ \prod_{n=1}^{L+N-1} f(y_n | \mathbf{x}_n) \right] \left[ \prod_{m=1}^L f_m(x_m) \right],$$

where  $Z$  is a normalization constant depending only on  $y$ ,  $f(y_n|\mathbf{x}_n)$  is the conditional distribution of  $y_n$  given  $\mathbf{x}_n$ , and  $f_m(x_m)$  is the *a priori* distribution of  $x_m$ . The factor graph that is induced by this factorization is a tree, so the sum-product algorithm could be used to obtain exact marginal distributions from which the MAP estimate of  $x$  could be obtained. However, the computational complexity of this direct application of the sum-product algorithm would be exponential in  $N$ .

To obtain reduced complexity algorithms, we consider instead factor graphs that model an approximate form of the *a posteriori* distribution. For example, we derive the unconstrained linear equalizer by considering the distribution to be of the form

$$\hat{f}(x|y) = \frac{1}{Z'} \left[ \prod_{n=1}^{L+N-1} \hat{f}(y_n|\mathbf{x}_n) \right] \left[ \prod_{m=1}^L \hat{f}_m(x_m) \right], \quad (8)$$

where  $Z'$  is a normalization constant depending only on  $y$ ,  $\hat{f}(y_n|\mathbf{x}_n)$  is a conditional Gaussian distribution with the same conditional mean  $\mathbf{h}_n^T \mathbf{x}_n$  and variance  $\sigma_n^2$  as the distribution  $f(y_n|\mathbf{x}_n)$ , and  $\hat{f}_m(x_m)$  is a Gaussian distribution with the same mean  $\bar{x}_n$  and variance  $v_n$  as the true *a priori* distribution  $f_m(x_m)$ . (Throughout this paper, given a probability distribution  $f$ ,  $\hat{f}$  refers to a Gaussian distribution with the same first- and second-order statistics.)

The resulting approximate model is then that of a Gaussian process, so the sum-product algorithm can be applied with significant reduction in computation, and the resulting approximate MAP estimate of  $x$  will be the LMMSE estimate (based on the approximate model.) However, LMMSE estimates depend only on first- and second-order statistics. So, since the first- and second-order statistics of the true and approximate systems are identical, the MAP estimate of the approximate system is the LMMSE estimate of the original system. Hence, by choosing a convenient approximate *a posteriori* distribution that allows for a reduction in computational complexity, we have indirectly arrived at the LMMSE estimate. This is the underlying methodology on which all of the algorithms in this paper are based.

As an aside, it should be noted that an alternative approach to developing equalization algorithms would be to construct a factor graph based on a state space representation of the problem. Such a state space approach is often employed to develop factor graph algorithms addressing a wide variety of signal processing problems [13], including Kalman filtering [6] and turbo equalization [14], because of the ease

with which the sum-product algorithm can be applied. In fact, the algorithms in this paper could be developed using such an approach. However, for the purposes of this paper, basing the factor graphs on factorizations such as that given in (8) offers several advantages. First, the identities of messages formed during an algorithm are more transparent. For example, in applying the sum-product algorithm to the portion of a factor graph modeling the observation equation of an appropriate state-space model, a message equal to  $\hat{f}(y_n|\mathbf{x}_n)$  is formed. This message appears explicitly as a factor in (8), and such transparency aids in developing fast algorithms as well as the DFE. Also, the elimination of auxiliary matrices associated with a state space model tends to illuminate relationships that are useful in developing the fast algorithms and the DFE. Finally, as will be discussed in Section III-A, a numerical issue arising in factor graph-based equalization can be easily remedied using our approach, but such a solution is not as natural when considering the state space approach.

#### A. Basic Algorithm

To compute the unconstrained LMMSE estimate of the transmitted signal  $x$  given the received signal  $y$ , we make use of the factor graph depicted in Figure 1 that models the factorization in (8) of the distribution  $\hat{f}(x|y)$ . Since each  $x_n$  appears in several vectors  $\mathbf{x}_n$ , it is necessary to ensure that all realizations of  $x_n$  are consistent, i.e., that there is zero probability that a particular  $x_n$  takes on different values in two vectors  $\mathbf{x}_k$  and  $\mathbf{x}_l$ ,  $k \neq l$ . (This is a slight abuse of notation. Strictly speaking, the variable nodes in a factor graph are all distinct variables. Hence, while variable nodes  $\mathbf{x}_k$  and  $\mathbf{x}_l$  might each contain an element corresponding to the random variable  $x_n$ , these elements should be named differently in each vector, say  $x_n^{(1)}$  and  $x_n^{(2)}$ . However, such notation would be cumbersome, and since these elements are to be made equal with probability 1, it is convenient to simply denote each element as  $x_n$ .)

To ensure this consistency, we introduce additional factors into the factor graph. Let  $\mathbf{n}$  and  $\mathbf{m}$  be vectors of some length  $K$  with elements that index some subset of the elements of vectors  $\mathbf{x}_k$  and  $\mathbf{x}_l$ , respectively. Then we define

$$\delta(\mathbf{x}_k, \mathbf{x}_l, \mathbf{n}, \mathbf{m}) = \prod_{i=1}^K \delta(\mathbf{x}_k(\mathbf{n}_i) - \mathbf{x}_l(\mathbf{m}_i)),$$

where  $\delta(x)$  is the Dirac delta function,  $\mathbf{n}_i$  and  $\mathbf{m}_i$  are the  $i$ th components of  $\mathbf{n}$  and  $\mathbf{m}$ , respectively, and for a vector  $\mathbf{x}$ ,  $\mathbf{x}(j)$  is the  $j$ th element of  $\mathbf{x}$ . Although omitted from (8) for conciseness, these factors are introduced into the factorization as necessary to ensure consistency and are depicted in Figure 1, where the first two arguments of each  $\delta$  function can be inferred from the neighboring variable nodes, and the last two arguments, i.e., the indexing sets, are determined by which elements of each variable node correspond to the same random variable.

The LMMSE estimate is obtained by applying the sum-product algorithm to the factor graph of Figure 1 to obtain the marginal distributions  $\hat{f}(x_n|y)$ . This will be a Gaussian distribution, so the MAP estimate of  $x_n$  based on the approximate model, and hence the LMMSE estimate based on the true model, is given by the mean parameter of  $\hat{f}(x_n|y)$ . The algorithm consists of a forward pass of messages (from left to right) and a backward pass of messages. These messages are then combined to form the messages passed to the variable nodes  $x_n$ . Note that the message passing rules for this basic algorithm are similar to those derived in [6] for a Kalman filter implemented on a factor graph. So, while we do provide the message passing rules with some justification, we refer the reader to [6] for a more detailed derivation.

In Figure 1, we have labeled several messages  $m_k^{(n)}$  that result from the application of the sum-product algorithm (the superscripts have been omitted for readability). As previously discussed, all messages will be (scaled) Gaussian densities or degenerate forms. That is, we have  $m_k^{(n)} = \gamma(\cdot, \boldsymbol{\mu}_k^{(n)}, \mathbf{W}_k^{(n)})$ , where the first argument of the  $\gamma$  function is given by the variable node to which or from which the message is being passed, except in the case of  $m_3^{(n)}$ , where the argument is  $y_n$ . So, for example,

$$m_2^{(n)} = \hat{f}_n(x_n) \propto \exp\left\{-\frac{1}{2v_n}(x_n - \bar{x}_n)^2\right\} = \gamma(\mathbf{x}_n, \bar{x}_n \mathbf{s}_N, v_n^{-1} \mathbf{s}_N \mathbf{s}_N^T), \quad (9)$$

where for  $1 \leq l \leq N$ ,  $\mathbf{s}_l = [\mathbf{0}_{1 \times l-1} \ 1 \ \mathbf{0}_{1 \times N-l}]^T$ . (Note that even though  $m_2^{(n)}$  only depends on  $x_n$ , we have written the first argument of the  $\gamma$  function in (9) as  $\mathbf{x}_n$  to explicitly show that  $m_2^{(n)}$  can be expressed as a function of  $\mathbf{x}_n$ , the variable to which this function is attached in the factor graph.) Hence,  $\boldsymbol{\mu}_2^{(n)} = \bar{x}_n \mathbf{s}_N$  and  $\mathbf{W}_2^{(n)} = v_n^{-1} \mathbf{s}_N \mathbf{s}_N^T$ . The parameters of messages  $m_3^{(n)}$  and  $m_4^{(n)}$  can be similarly obtained from the following:

$$m_3^{(n)} = \hat{f}(y_n|\mathbf{x}_n) \propto \exp\left\{-\frac{1}{2\sigma_n^2}(y_n - \mathbf{h}_n^T \mathbf{x}_n)^2\right\} = \gamma(y_n, \mathbf{h}_n^T \mathbf{x}_n, \sigma_n^{-2}),$$

and

$$m_4^{(n)} = \int_{-\infty}^{\infty} 1 \cdot \delta(\mathbf{x}_n, x'_n) dx'_n = \int_{-\infty}^{\infty} \delta(x_n - x'_n) dx'_n = 1 = \gamma(\mathbf{x}_n, \mathbf{0}_{N \times 1}, \mathbf{0}_{N \times N}),$$

where in the last equation, we have applied the general update rule given in (3).

The propagation of messages then involves computing the parameters of the messages leaving each node given the parameters of the messages that arrive at that node. Suppose we have  $\boldsymbol{\mu}_1^{(n)}$  and  $\mathbf{W}_1^{(n)}$ .

From (2) we have  $m_5^{(n)} = m_1^{(n)} m_2^{(n)} m_3^{(n)} m_4^{(n)}$ . Then the use of (5) and (7) yields

$$\mathbf{W}_5^{(n)} = \mathbf{W}_1^{(n)} + \mathbf{W}_2^{(n)} + \mathbf{W}_3^{(n)} + \mathbf{W}_4^{(n)} = \mathbf{W}_1^{(n)} + v_n^{-1} \mathbf{s}_N \mathbf{s}_N^T + \sigma_n^{-2} \mathbf{h}_n \mathbf{h}_n^T,$$

and

$$\begin{aligned} \boldsymbol{\mu}_5^{(n)} &= (\mathbf{W}_5^{(n)})^\dagger (\mathbf{W}_1^{(n)} \boldsymbol{\mu}_1^{(n)} + \mathbf{W}_2^{(n)} \boldsymbol{\mu}_2^{(n)} + \mathbf{W}_3^{(n)} \boldsymbol{\mu}_3^{(n)} + \mathbf{W}_4^{(n)} \boldsymbol{\mu}_4^{(n)}) \\ &= (\mathbf{W}_5^{(n)})^\dagger (\mathbf{W}_1^{(n)} \boldsymbol{\mu}_1^{(n)} + \frac{\bar{x}_n}{v_n} \mathbf{s}_N + \sigma_n^{-2} y_n \mathbf{h}_n). \end{aligned}$$

To compute the parameters of  $m_1^{(n+1)}$ , we partition  $\boldsymbol{\mu}_5^{(n)}$  as  $\boldsymbol{\mu}_5^{(n)} = [\boldsymbol{\mu}_5^{(n)}(1) \mid \boldsymbol{\mu}_5^{(n)}(2:N)^T]^T = [\boldsymbol{\mu}_{5A}^{(n)} \mid (\boldsymbol{\mu}_{5B}^{(n)})^T]^T$ , where for a vector  $\boldsymbol{\mu}$ ,  $\boldsymbol{\mu}(k:l)$  is the subvector formed from elements  $k$  through  $l$  of  $\boldsymbol{\mu}$ . We also partition  $\mathbf{W}_5^{(n)}$  according to

$$\mathbf{W}_5^{(n)} = \left[ \begin{array}{c|c} \mathbf{W}_5^{(n)}(1,1) & \mathbf{W}_5^{(n)}(1,2:N) \\ \hline \mathbf{W}_5^{(n)}(2:N,1) & \mathbf{W}_5^{(n)}(2:N,2:N) \end{array} \right] = \left[ \begin{array}{c|c} W_{5A}^{(n)} & \mathbf{W}_{5B}^{(n)} \\ \hline \mathbf{W}_{5C}^{(n)} & \mathbf{W}_{5D}^{(n)} \end{array} \right], \quad (10)$$

where for a matrix  $\mathbf{W}$ ,  $\mathbf{W}(k_1:k_2, l_1:l_2)$  is defined in an analogous manner as the vector case. Then using (3) and (6), we have  $\boldsymbol{\mu}_1^{(n+1)} = [(\boldsymbol{\mu}_{5B}^{(n)})^T \ 0]^T$  and

$$\mathbf{W}_1^{(n+1)} = \left[ \begin{array}{c|c} \mathbf{W}_{5D}^{(n)} - \mathbf{W}_{5C}^{(n)} (\mathbf{W}_{5A}^{(n)})^\dagger \mathbf{W}_{5B}^{(n)} & \mathbf{0}_{(N-1) \times 1} \\ \hline \mathbf{0}_{1 \times (N-1)} & 0 \end{array} \right]. \quad (11)$$

Repeated application of the above formulas at each stage of the factor graph yields the messages for the forward pass of the sum-product algorithm. (Modifications must be made at the boundaries of the data sequence.) The message updates for the backward pass are similar and will not be described in detail. All that remains is to compute messages  $m_8^{(n)}$  and  $m_9^{(n)}$ . From (2), the former is given by  $m_8^{(n)} = m_1^{(n)} m_2^{(n)} m_3^{(n)} m_6^{(n)} = m_5^{(n)} m_6^{(n)}$ , where we have used that  $m_5^{(n)} = m_1^{(n)} m_2^{(n)} m_3^{(n)} m_4^{(n)}$  and  $m_4^{(n)} = 1$ . So, using (5), we have  $\mathbf{W}_8^{(n)} = \mathbf{W}_5^{(n)} + \mathbf{W}_6^{(n)}$  and  $\boldsymbol{\mu}_8^{(n)} = (\mathbf{W}_8^{(n)})^\dagger (\mathbf{W}_5^{(n)} \boldsymbol{\mu}_5^{(n)} + \mathbf{W}_6^{(n)} \boldsymbol{\mu}_6^{(n)})$ .

Now, from (3), it can be seen that  $m_9^{(n)}$  is given by marginalizing  $m_8^{(n)}$  over all variables in  $\mathbf{x}_n$  other than  $x_n$ . The partitions  $\boldsymbol{\mu}_8^{(n)} = [\boldsymbol{\mu}_8^{(n)}(1:N-1) \mid \boldsymbol{\mu}_8^{(n)}(N)]^T = [(\boldsymbol{\mu}_{8A}^{(n)})^T \mid \mu_{8B}^{(n)}]^T$  and

$$\mathbf{W}_8 = \left[ \begin{array}{c|c} \mathbf{W}_8^{(n)}(1:N-1, 1:N-1) & \mathbf{W}_8^{(n)}(1:N-1, N) \\ \hline \mathbf{W}_8^{(n)}(N, 1:N-1) & \mathbf{W}_8^{(n)}(N, N) \end{array} \right] = \left[ \begin{array}{c|c} \mathbf{W}_{8A}^{(n)} & \mathbf{W}_{8B}^{(n)} \\ \hline \mathbf{W}_{8C}^{(n)} & \mathbf{W}_{8D}^{(n)} \end{array} \right]$$

allow us to apply (6) to compute  $\mu_9^{(n)} = \mu_{8B}^{(n)}$  and  $W_9^{(n)} = W_{8D}^{(n)} - \mathbf{W}_{8C}^{(n)}(\mathbf{W}_{8A}^{(n)})^\dagger \mathbf{W}_{8B}^{(n)}$ . Since  $m_9^{(n)} \propto \hat{f}(x_n|y)$ , the LMMSE estimate  $\hat{x}_n$  of  $x_n$  is given by  $\mu_9^{(n)}$ .

While the above algorithm is theoretically correct, straightforward implementation gives rise to an issue with respect to the conditioning of the matrices involved. Note that the matrices formed in the forward pass of messages are inverse covariance matrices. For example,  $(\mathbf{W}_5^{(n)})^{-1} = \text{COV}(\mathbf{x}_n, \mathbf{x}_n | \mathbf{y}_1^n) \triangleq \mathbf{E}(\mathbf{x}_n \mathbf{x}_n^T | \mathbf{y}_1^n) - \mathbf{E}(\mathbf{x}_n | \mathbf{y}_1^n) \mathbf{E}(\mathbf{x}_n^T | \mathbf{y}_1^n)$  (under the assumed approximate distribution), where  $\mathbf{y}_1^n = [y_1 \ y_2 \ \dots \ y_n]^T$ . Consequently, these matrices can generally be inverted without numerical issues. However, the matrices in the backward pass are not inverse covariance matrices and therefore give no assurance of being well-conditioned. Also, the pseudoinverses are inconvenient in implementation and in the development of fast updates. In the forward pass, the pseudoinverses are only necessary because some matrices have an entire row and column that are zero. In these cases, the invertible and noninvertible portions of the matrices can be treated separately. However, in the backward pass, because the matrices involved are not inverse covariance matrices, there is no guarantee that the noninvertible part of the matrix can be handled as conveniently.

Fortunately, both of the above issues can be handled with a modification to the model and resulting factor graph. Figure 2 depicts another factorization of the *a posteriori* probability density function in which the prior distributions have been factored as  $\hat{f}_n(x_n) = \sqrt{\hat{f}_n(x_n)} \sqrt{\hat{f}_n(x_n)}$ . One factor is attached to the variable  $\mathbf{x}_n$ , which is similar to the previous factor graph, whereas the other factor is attached to  $\mathbf{x}_{n+N-1}$ . Note that the global function represented by the original factor graph is unchanged by these modifications, so the sum-product algorithm will still produce the desired result. However, with this modification, the matrices computed in both the forward and backward passes of the algorithm correspond to inverse covariance matrices and are therefore generally well-conditioned. (Note that these

matrices do not correspond to the true covariance matrices since they assume priors that have altered variances. However, this does not affect the conclusion that the matrices are well-conditioned.) The resulting algorithm differs from the original algorithm only in the updates of messages  $m_5^{(n)}$  and  $m_7^{(n)}$ . For example, message  $m_5^{(n)}$  is now parameterized by  $\mathbf{W}_5^{(n)} = \mathbf{W}_1^{(n)} + \mathbf{W}_{2a}^{(n)} + \mathbf{W}_{2b}^{(n)} + \sigma_n^{-2} \mathbf{h}_n \mathbf{h}_n^T$  and  $\boldsymbol{\mu}_5^{(n)} = (\mathbf{W}_5^{(n)})^\dagger (\mathbf{W}_1^{(n)} \boldsymbol{\mu}_1^{(n)} + \mathbf{W}_{2a}^{(n)} \boldsymbol{\mu}_{2a}^{(n)} + \mathbf{W}_{2b}^{(n)} \boldsymbol{\mu}_{2b}^{(n)} + \sigma_n^{-2} y_n \mathbf{h}_n)$ , where  $\boldsymbol{\mu}_{2a}^{(n)} = \bar{x}_n \mathbf{s}_N$ ,  $\boldsymbol{\mu}_{2b}^{(n)} = \bar{x}_{n-N+1} \mathbf{s}_1$ ,  $\mathbf{W}_{2a}^{(n)} = (2v_n)^{-1} \mathbf{s}_N \mathbf{s}_N^T$ , and  $\mathbf{W}_{2b}^{(n)} = (2v_{n-N+1})^{-1} \mathbf{s}_1 \mathbf{s}_1^T$ .

### B. Fast Algorithm

The complexity of using the above algorithm is  $\mathcal{O}(N^3)$  per estimate. However, we may compute the pseudoinverses recursively using the matrix inversion lemma and the block matrix inversion formula in  $\mathcal{O}(N^2)$ , yielding an algorithm that is  $\mathcal{O}(N^2)$  per estimate.

First, suppose we have  $\mathbf{W}_1^{(n)}$  and  $(\mathbf{W}_1^{(n)})^\dagger$ . From (11),  $\mathbf{W}_1^{(n)}$  is of the form

$$\mathbf{W}_1^{(n)} = \left[ \begin{array}{c|c} \tilde{\mathbf{W}}_1^{(n)} & \mathbf{0}_{(N-1) \times 1} \\ \hline \mathbf{0}_{1 \times (N-1)} & 0 \end{array} \right],$$

for some matrix  $\tilde{\mathbf{W}}_1^{(n)}$ . Furthermore, as discussed previously, it can be shown that  $\mathbf{W}_1^{(n)}$  is an inverse (modified) covariance matrix and is therefore positive definite and invertible. So, we have

$$(\mathbf{W}_1^{(n)})^\dagger = \left[ \begin{array}{c|c} (\tilde{\mathbf{W}}_1^{(n)})^{-1} & \mathbf{0}_{(N-1) \times 1} \\ \hline \mathbf{0}_{1 \times (N-1)} & 0 \end{array} \right].$$

Then,

$$(\mathbf{W}_1^{(n)} + \mathbf{W}_{2a}^{(n)})^{-1} = \left[ \begin{array}{c|c} (\tilde{\mathbf{W}}_1^{(n)})^{-1} & \mathbf{0}_{(N-1) \times 1} \\ \hline \mathbf{0}_{1 \times (N-1)} & 2v_n \end{array} \right].$$

Since  $\mathbf{W}_{2b}^{(n)}$  and  $\mathbf{W}_3^{(n)}$  are both rank-one,  $(\mathbf{W}_5^{(n)})^{-1} = [(\mathbf{W}_1^{(n)} + \mathbf{W}_{2a}^{(n)}) + \mathbf{W}_{2b}^{(n)} + \sigma_n^{-2} \mathbf{h}_n \mathbf{h}_n^T]^{-1}$  can be computed with two applications of the matrix inversion lemma to  $(\mathbf{W}_1^{(n)} + \mathbf{W}_{2a}^{(n)})^{-1}$ . Finally,

$$(\mathbf{W}_1^{(n+1)})^\dagger = \left[ \begin{array}{c|c} (\mathbf{W}_5^{(n)})^{-1} (2:N, 2:N) & \mathbf{0}_{(N-1) \times 1} \\ \hline \mathbf{0}_{1 \times (N-1)} & 0 \end{array} \right].$$

The recursive computation of the pseudoinverses in the backward pass can be performed similarly.

Now, an  $\mathcal{O}(N^2)$  update exists for  $(\mathbf{W}_8^{(n)})^{-1}$  from  $(\mathbf{W}_8^{(n+1)})^{-1}$  that can be derived algebraically using the matrix inversion formulas of Section II-B. However, it is perhaps simpler to note that  $(\mathbf{W}_8^{(n+1)})^{-1} = \text{COV}(\mathbf{x}_{n+1}, \mathbf{x}_{n+1}|y)$  and  $(\mathbf{W}_8^{(n)})^{-1} = \text{COV}(\mathbf{x}_n, \mathbf{x}_n|y)$ . So, partitioning  $(\mathbf{W}_8^{(n)})^{-1}$  according to

$$(\mathbf{W}_8^{(n)})^{-1} = \left[ \begin{array}{c|c} (\mathbf{W}_8^{(n)})^{-1}(1, 1) & (\mathbf{W}_8^{(n)})^{-1}(1, 2 : N) \\ \hline (\mathbf{W}_8^{(n)})^{-1}(2 : N, 1) & (\mathbf{W}_8^{(n)})^{-1}(2 : N, 2 : N) \end{array} \right] = \left[ \begin{array}{c|c} V_{8A}^{(n)} & \mathbf{V}_{8B}^{(n)} \\ \hline \mathbf{V}_{8C}^{(n)} & \mathbf{V}_{8D}^{(n)} \end{array} \right],$$

gives  $\mathbf{V}_{8D}^{(n)} = (\mathbf{W}_8^{(n+1)})^{-1}(1 : N - 1, 1 : N - 1)$ . Next, from (10), we can write

$$\mathbf{W}_8^{(n)} = \left[ \begin{array}{c|c} W_{5A}^{(n)} & \mathbf{W}_{5B}^{(n)} \\ \hline \mathbf{W}_{5C}^{(n)} & \tilde{\mathbf{W}}_{8D}^{(n)} \end{array} \right],$$

for some matrix  $\tilde{\mathbf{W}}_{8D}^{(n)}$ . Then, the use of the matrix inversion formulas of Section II-B yields  $V_{8A}^{(n)} = [W_{5A}^{(n)} + \mathbf{W}_{5B}^{(n)} \mathbf{V}_{8D}^{(n)} \mathbf{W}_{5C}^{(n)}] / (W_{5A}^{(n)})^2$ ,  $\mathbf{V}_{8B}^{(n)} = \mathbf{W}_{5B}^{(n)} \mathbf{V}_{8D}^{(n)} / W_{5A}^{(n)}$ , and  $\mathbf{V}_{8C}^{(n)} = \mathbf{V}_{8D}^{(n)} \mathbf{W}_{5C}^{(n)} / W_{5A}^{(n)}$ .

Finally, to obtain  $W_9^{(n)}$ , we note that  $(\mathbf{W}_8^{(n)})^{-1}$  is the conditional covariance matrix of  $\mathbf{x}_n$  given  $y$ . So the conditional variance of  $x_n$  given  $y$  is  $(\mathbf{W}_8^{(n)})^{-1}(N, N)$ , and  $W_9^{(n)} = 1 / [(\mathbf{W}_8^{(n)})^{-1}(N, N)]$ .

Table I describes the fast algorithm, where we have used that it is sometimes more convenient to update  $\boldsymbol{\mu}'_k \triangleq \mathbf{W}_k \boldsymbol{\mu}_k$  instead of  $\boldsymbol{\mu}_k$ . Also, in all tables, the assignment of an equation to the inverse of a matrix indicates that the matrix inverse is to be computed in the manner specified by the equation and stored. This implementation, along with some additional optimization to take advantage of the symmetry of the computed matrices and the zero elements of the computed matrices and vectors, requires  $(12N^2 - 16N + 18)L - (55/6)N^3 + (33/2)N^2 - (61/3)N + 6$  additions and  $(12N^2 + 4N + 11)L - (55/6)N^3 + N^2 - (29/6)N + 2$  multiplications, where subtraction is counted as addition and division is counted as multiplication. For brevity, it is not possible to derive these formulas here. Note that the first term of each formula describes the computations that grow linearly with the data length, while the remaining terms describe any overhead that is independent of the data length.

#### IV. CONSTRAINED LINEAR EQUALIZATION

##### A. Basic Algorithm

Next, we consider the use of factor graphs to find the constrained LMMSE estimate of  $x_n$  given the  $M = M_1 + M_2 + 1$  element vector  $\mathbf{y}_n = [y_{n-M_1} \ y_{n-M_1+1} \ \dots \ y_{n+M_2}]^T$ , i.e., we solve for the estimates

$\hat{x}_n$  minimizing  $E\{(x_n - \hat{x}_n)^2\}$  subject to the constraint that  $\hat{x}_n$  can be expressed as an affine function of  $\mathbf{y}_n$ .

Although the use of such an equalizer instead of the unconstrained equalizer considered in section III would result in a degradation of mean squared error in the absence of model inaccuracy, there are several reasons why the constrained equalizer might be preferable for a particular application. For example, the constrained equalizer can operate sequentially, in contrast with the unconstrained equalizer, which must wait until the entire data block is received before beginning the backward passage of messages. Also, many analytic results on the asymptotic behavior of turbo equalization require that the equalizer output for a particular bit  $x_n$  be independent of the *a priori* information regarding bits far from  $x_n$  [15]. This is the case for the constrained equalizer, but clearly not so, in general, for the unconstrained equalizer. Finally, unlike the unconstrained case, the constrained equalizer allows for some degree of parallel processing. While we do not detail such a parallel system here, it would basically consist of subdividing the data block, with overlap, and processing the subblocks independently by the multiple processors.

To perform the constrained equalization, we consider the distribution

$$\begin{aligned} \hat{f}(\mathbf{x}_{n-M_1-N+1}^{n+M_2} | \mathbf{y}_n) &\propto \hat{f}(\mathbf{y}_n | \mathbf{x}_{n-M_1-N+1}^{n+M_2}) \hat{f}(\mathbf{x}_{n-M_1-N+1}^{n+M_2}) \\ &\propto \left[ \prod_{k=n-M_1}^{n+M_2} \hat{f}(y_k | \mathbf{x}_k) \right] \left[ \prod_{l=n-M_1-N+1}^{n+M_2} \hat{f}_l(x_l) \right], \end{aligned}$$

where  $\mathbf{x}_a^b \triangleq [x_a \ x_{a+1} \ \dots \ x_b]^T$ . The factor graph of Figure 3 represents this factorization, where, as in the unconstrained case, we have used the factorization  $\hat{f}_n(x_n) = \sqrt{\hat{f}_n(x_n)} \sqrt{\hat{f}_n(x_n)}$  to ensure well conditioning. (Again, omitted arguments can be inferred from the attached variable node(s).)

Note that the subscripts of the labeled messages in this figure do not generally correspond to the subscripts of the labeled messages in the previous factor graphs, nor has the meaning of the superscripts been unchanged. In particular, in the previous factor graphs, the superscript ( $n$ ) indicates that a message is being passed along an edge incident on  $x_n$  or  $\mathbf{x}_n$ , whereas in Figure 3, the superscript ( $n$ ) indicates that this is a message produced during the process of forming the estimate  $\hat{x}_n$ . This slightly modified notation will become useful when considering a fast constrained linear equalizer.

The application of the sum-product algorithm to this factor graph results in message updates that are

similar to the unconstrained linear equalizer and so will not be described in detail. The resulting message  $m_g^{(n)}$  is the desired marginal distribution, the mean of which is the desired LMMSE estimate of  $x_n$ .

It is interesting to note that an equivalent constrained equalizer can actually be implemented on the factor graph of Figure 2 for the full distribution  $\hat{f}(x|y)$  by applying a particular message passing update schedule. In particular, we initialize all messages to unity. We then pass messages from all of the  $\sqrt{\hat{f}_n(x_n)}$ . Next, we simultaneously send  $N - 1$  messages to the right from the  $\mathbf{x}_n$  to the  $\mathbf{x}_{n+1}$  for all  $n$ . (By this we mean that messages are first passed to the right from the  $\mathbf{x}_n$  to the  $\delta$  functions for all  $n$ , and then messages are passed to the right from the  $\delta$  functions to the  $\mathbf{x}_{n+1}$  for all  $n$ . The two-step process is then repeated for a total of  $N - 1$  times.) We then simultaneously pass  $N - 1$  messages to the left. The messages from all of the  $\hat{f}(y_n|\mathbf{x}_n)$  are then sent. Next, we simultaneously pass  $M_1$  messages to the right from the  $\mathbf{x}_n$  to the  $\mathbf{x}_{n+1}$  for all  $n$ , and then  $M_2$  messages to the left. Finally, we pass the messages up from the  $\mathbf{x}_n$  to the  $x_n$  for all  $n$ . After sending the messages from the  $\hat{f}(y_n|\mathbf{x}_n)$ , the messages formed in the two algorithms are equivalent, so the message passed to  $x_n$  is proportional to  $\hat{f}(x_n|\mathbf{y}_n)$ , as desired. Hence, constrained linear equalization can be viewed as the implementation of an unconstrained linear equalizer on a factor graph using a suboptimal message passing schedule.

### B. Fast Algorithm

While the last reformulation of the constrained equalizer is interesting, it amounts to a simple reordering of steps. Ideally, we would like to use the messages formed in estimating  $x_n$  to be able to more quickly estimate  $x_{n+1}$ . In particular, given  $m_g^{(n)}$ , we find a fast update for  $m_g^{(n+1)}$ . Unfortunately, because some variables of  $\mathbf{x}_{n-M_1-N+1}^{n+M_2}$  are integrated out in computing  $m_g^{(n)}$ , information is lacking from  $m_g^{(n)}$  that could be useful in developing such an update. So, we modify the previous factor graph so as to delay the marginalization steps of the sum-product algorithm. In particular, we replace the variable nodes  $\mathbf{x}_k$ ,  $k < n$ , with  $\mathbf{x}_{n-M_1-N+1}^k$ ,  $\mathbf{x}_n$  with  $\mathbf{x}_{n-M_1-N+1}^{n+M_2}$ , and  $\mathbf{x}_k$ ,  $k > n$ , with  $\mathbf{x}_k^{n+M_2}$ . The  $\delta$  functions must also be modified to ensure that newly added representatives of each variable are consistent. With the exception of the modified  $\delta$  functions, the factorization represented by this factor graph is unchanged, since no modifications were made to the remaining function nodes. So even though the updates of the

sum-product algorithm will be different, the final result will be the same.

Because the sum-product updates for the modified factor graph are similar to those for the previous case, we do not describe them in detail. Table II provides a description of the algorithm for the case of  $n = 1$ . (We show this specific case since it can be used to initialize the recursive algorithm we describe later.) Note that we compute  $\boldsymbol{\mu}'_7 \triangleq \mathbf{W}_7 \boldsymbol{\mu}_7$  instead of  $\boldsymbol{\mu}_7$  to avoid some unnecessary matrix-vector multiplications. (The quantities  $\mathbf{W}_7$  and  $\boldsymbol{\mu}_7$  parameterize the message passed to the left from the  $\mathbf{x}_k^{M_2+1}$ .)

Now, it is easily verified that for the modified factor graph,

$$m_8^{(n+1)} = m_8^{(n)} \frac{(m_3^{(n+1)})^2 m_4^{(n+1)}}{(m_1^{(n)})^2 m_2^{(n)}},$$

where we analogously label the messages in the modified factor graph as in Figure 3. Noting that  $1/\gamma(\mathbf{x}, \boldsymbol{\mu}, \mathbf{W}) = \gamma(\mathbf{x}, \boldsymbol{\mu}, -\mathbf{W})$ , we can compute  $m_8^{(n+1)}$  from  $m_8^{(n)}$  using (5) and (7). Upon dividing by  $(m_1^{(n)})^2 m_2^{(n)}$ , the message is no longer a function of  $x_{n-M_1}$ , so the elements of the parameters of the message corresponding to this variable can be eliminated, i.e., the first row and column of the inverse covariance matrix and the first element of the mean vector can be deleted.

The matrix updates involved in the above computation are all rank-one updates. Hence, the required pseudoinverses can be computed efficiently using the matrix inversion lemma. Furthermore, by updating  $\boldsymbol{\mu}'_8^{(n)} \triangleq \mathbf{W}_8^{(n)} \boldsymbol{\mu}_8^{(n)}$  instead of  $\boldsymbol{\mu}_8^{(n)}$ , it is not necessary to compute  $\mathbf{W}_8^{(n)}$ , only  $(\mathbf{W}_8^{(n)})^{-1}$ .

The above algorithm, including the optimizations just mentioned, is described in Table III, where  $\mathbf{h}_{n-M_1-1}^t = [\mathbf{0}_{1 \times k} \mid \mathbf{h}_{n-M_1-1}^T]^T$ ,  $\mathbf{h}_{n+M_2}^b = [\mathbf{h}_{n+M_2}^T \mid \mathbf{0}_{1 \times l}]^T$ , and  $k$  and  $l$  are chosen so that the calculations in which these vectors appear are well-defined. Also, for an  $i \times j$  matrix  $\mathbf{W}$ , we define  $\text{size}(\mathbf{W}, k)$  to be  $i$  if  $k = 1$  and  $j$  if  $k = 2$ . Finally, in line 1 of Table III, the quantities  $\mathbf{W}_8^{(1)}$ ,  $(\mathbf{W}_8^{(1)})^{-1}$ , and  $\boldsymbol{\mu}_8^{(1)}$  can be computed using the previous constrained equalization algorithm described in Table II. This implementation, along with some optimization to take advantage of the symmetry of the computed matrices and the zero elements of the computed matrices and vectors, requires  $(3N^2 + 4NM + M^2 - 2M)L + g_1(N, M_1, M_2, K_2)$  additions and  $(3N^2 + 4NM + M^2 + 3N + M + 1)L + g_2(N, M_1, M_2, K_2)$  multiplications, where  $g_1$  and  $g_2$  are multivariate polynomials of total degree 3, and  $K_2 = \min(N, M_2 + 1)$ .

Note that the first term of each equation describes the computations that grow linearly with the data length, while the functions  $g_1$  and  $g_2$  describe any overhead that is independent of the data length.

### C. Extracting Filter Coefficients

The last constrained equalizer we present allows for the extraction of the equalizer coefficients. This is most useful in the case of a shift invariant channel, i.e.,  $\mathbf{h}_n = \mathbf{h}$  for all  $n$ , with the same prior distribution on each symbol. Then the constrained LMMSE filter is the same for all  $n$  (except at the borders), so computing this filter allows for efficient computation of the symbol estimates.

We consider the factorization of the function  $\hat{f}(\mathbf{x}_{n-M_1-N+1}^{n+M_2} | \mathbf{y}_n)$  depicted in the factor graph of Figure 3 with the following modifications. We replace the variable nodes  $\mathbf{x}_k$ ,  $n - M_1 \leq k < n$ , with the vectors  $[\mathbf{x}_k^T | y_{n-M_1} \ y_{n-M_1+1} \ \dots \ y_k]^T$  and we replace the variable nodes  $\mathbf{x}_k$ ,  $k > n$ , with the vectors  $[\mathbf{x}_k | y_k \ y_{k+1} \ \dots \ y_{n+M_2}]^T$ . Finally, we replace the variable node  $\mathbf{x}_n$  with  $[\mathbf{x}_n^T | \mathbf{y}_n^T]^T$  and the variable node  $x_n$  with  $[x_n | \mathbf{y}_n^T]^T$ . Again, modifications must be made to the  $\delta$  functions to ensure the consistency of the variables in the factor graph, but this will not affect the final result of the sum-product algorithm.

The purpose of these modifications is to allow the observations to be treated as variables instead of constant parameters. When treating  $y_k$  as a variable, it is convenient to rewrite the messages leaving nodes  $\hat{f}(y_k | \mathbf{x}_k)$  as  $\gamma([\mathbf{x}_k | y_k]^T, \mathbf{0}_{(N+1) \times 1}, \sigma_k^{-2} [\mathbf{h}_k^T | -1]^T [\mathbf{h}_k^T | -1])$ . With this modification, the sum-product algorithm can be applied in a straightforward manner. The message passed to variable node  $[x_n | \mathbf{y}_n^T]^T$  is then proportional to  $\hat{f}(x_n | \mathbf{y}_n)$ . Let this message be given by  $\gamma([x_n | \mathbf{y}_n^T]^T, \boldsymbol{\mu}, \mathbf{W})$ , where  $\boldsymbol{\mu}$  and  $\mathbf{W}$  are partitioned according to  $\boldsymbol{\mu} = [\boldsymbol{\mu}(1) | \boldsymbol{\mu}(2 : M+1)]^T = [\mu_A | \boldsymbol{\mu}_B^T]^T$  and

$$\mathbf{W} = \left[ \begin{array}{c|c} \mathbf{W}(1,1) & \mathbf{W}(1,2 : M+1) \\ \hline \mathbf{W}(2 : M+1,1) & \mathbf{W}(2 : M+1,2 : M+1) \end{array} \right] = \left[ \begin{array}{c|c} W_A & \mathbf{W}_B \\ \hline \mathbf{W}_C & \mathbf{W}_D \end{array} \right].$$

Then, solving for the conditional mean of  $x_n$  from this distribution yields

$$\hat{x}_n = \mu_A - \frac{\mathbf{W}_B + \mathbf{W}_C^T}{2W_A} (\mathbf{y}_n - \boldsymbol{\mu}_B) = \mu_A - \frac{\mathbf{W}_B}{W_A} (\mathbf{y}_n - \boldsymbol{\mu}_B),$$

where we have applied the symmetry of  $\mathbf{W}$ . Hence, the coefficients of the LMMSE filter are given by  $-\mathbf{W}_B/W_A$ . When using recursive updates, the complexity of computing this filter is  $\mathcal{O}(M(M+N)^2)$ .

## V. DECISION FEEDBACK EQUALIZATION

The last factor graph technique we present is a decision feedback equalizer. In such an equalizer, symbol decisions are made in stages. At each stage, equalization is performed assuming that symbol decisions from previous stages are correct [1], [16]. The resulting real-valued estimates are then used in deciding an additional set of symbols. In general, any equalization criterion can be used, but here we employ a constrained LMMSE equalizer. Also, there is freedom in choosing the order in which symbol decisions are made and in which previous decisions are considered while making a current estimate (though usually one considers all relevant previous decisions). We present first a general algorithm that, if desired, can be specialized to provide a more efficient implementation of a particular DFE. We then illustrate this specialization to the case of a DFE in which decisions are made sequentially from the beginning of the data sequence to the end.

### A. General Algorithm

As previously stated, the DFE we describe will use a constrained LMMSE equalizer. As in Section IV, the length- $M$  equalizer will solve for the LMMSE estimate of  $x_n$  given  $\mathbf{y}_n$ . The key observation in deriving the DFE is that decisions can be incorporated into the LMMSE estimates by computing each estimate as before but after replacing the *a priori* distribution  $\hat{f}_k(x_k)$  with the distribution  $\check{f}_k(x_k) = \delta(x_k - \check{x}_k)$  for some choice of symbols  $\{x_k\}$ , each with decision  $x_k = \check{x}_k$ . This replacement results in some modifications to the message passing updates. Also, since it would be inefficient to perform the equalization afresh after each symbol decision is made, a fast update can be considered. Both of these concerns are addressed in the following.

Our implementation of the DFE will use a modification of the factor graph depicted in Figure 3. As in Section IV-B, we replace, when applicable, the variable nodes  $\mathbf{x}_k$ ,  $k < n$ , with  $\mathbf{x}_{n-M_1-N+1}^k$ ,  $\mathbf{x}_n$  with  $\mathbf{x}_{n-M_1-N+1}^{n+M_2}$ , and  $\mathbf{x}_k$ ,  $k > n$ , with  $\mathbf{x}_k^{n+M_2}$ , modifying the  $\delta$  functions appropriately. Again, while these modifications result in changes to the updates of the sum-product algorithm that are useful in deriving a fast algorithm, the final result of the sum-product algorithm is unchanged.

The algorithm is initialized by performing equalization over the entire set of data as described in Section

IV-B and in Table III. From this step, we store the messages  $m_8^{(n)}$  and  $m_9^{(n)}$  for all  $n$ . As mentioned previously,  $m_9^{(n)}$  contains the symbol estimate as the mean of the distribution. Now, we make decisions on some subset of symbols. This subset can be predetermined, randomly chosen, or based on properties of the estimates themselves. For example, it might be reasonable to make decisions first for the variables with the smallest conditional variance as given by  $W_9^{-1}$ . However this set is chosen, the decisions  $\{\check{x}_n\}$  are made based on the estimates produced by equalization.

Next, we must update the stored messages to reflect the newly formed decisions. For example, we consider updating message  $m_8^{(n)} = \gamma(\cdot, \boldsymbol{\mu}_8^{(n)}, \mathbf{W}_8^{(n)})$  given the new decision  $x_{n-M_1-N+1} = \check{x}_{n-M_1-N+1}$ . We first must divide out the *a priori* distribution, i.e.,  $\hat{f}_{n-M_1-N+1}(x_{n-M_1-N+1})$ , from the message. We then multiply in a new distribution reflecting the assumed absolute certainty in the decision, i.e.,  $\delta(x_{n-M_1-N+1} - \check{x}_{n-M_1-N+1})$  [17]. Finally, since the variable for which the decision was made no longer plays a role in the estimation problem (only the decision does), it can be integrated out of any message in which it appears. Because of the new *a priori* distribution that asserts absolute certainty in the decision, this is equivalent to evaluating the message at the decision.

In our example, the updated message is

$$\begin{aligned} \tilde{m}_8^{(n)} &= \left[ m_8^{(n)} / \hat{f}_{n-M_1-N+1}(x_{n-M_1-N+1}) \right]_{x_{n-M_1-N+1}=\check{x}_{n-M_1-N+1}} \\ &= \frac{1}{\hat{f}_{n-M_1-N+1}(\check{x}_{n-M_1-N+1})} \left[ m_8^{(n)} \right]_{x_{n-M_1-N+1}=\check{x}_{n-M_1-N+1}} \\ &\propto \left[ m_8^{(n)} \right]_{x_{n-M_1-N+1}=\check{x}_{n-M_1-N+1}}. \end{aligned}$$

Since we only need messages up to a scale factor, it follows that it is unnecessary to divide out the original *a priori* distribution. As such, we need only to evaluate the message at the decision to obtain the update. The partitions  $\boldsymbol{\mu}_8^{(n)} = [\boldsymbol{\mu}_8^{(n)}(1) \mid \boldsymbol{\mu}_8^{(n)}(2:P)]^T = [\boldsymbol{\mu}_{8A}^{(n)} \mid (\boldsymbol{\mu}_{8B}^{(n)})^T]^T$  and

$$\mathbf{W}_8^{(n)} = \left[ \begin{array}{c|c} \mathbf{W}_8^{(n)}(1,1) & \mathbf{W}_8^{(n)}(1,2:P) \\ \hline \mathbf{W}_8^{(n)}(2:P,1) & \mathbf{W}_8^{(n)}(2:P,2:P) \end{array} \right] = \left[ \begin{array}{c|c} W_{8A}^{(n)} & \mathbf{W}_{8B}^{(n)} \\ \hline \mathbf{W}_{8C}^{(n)} & \mathbf{W}_{8D}^{(n)} \end{array} \right],$$

where  $P = M + N - 1$ , allow us to write  $\tilde{m}_8^{(n)} = \gamma(\cdot, \tilde{\boldsymbol{\mu}}_8^{(n)}, \tilde{\mathbf{W}}_8^{(n)})$ , where  $\tilde{\mathbf{W}}_8^{(n)} = \mathbf{W}_{8D}^{(n)}$  and

$$\begin{aligned}\tilde{\boldsymbol{\mu}}_8^{(n)} &= \frac{1}{2}(\tilde{\mathbf{W}}_8^{(n)})^\dagger \left\{ \left[ \mathbf{W}_{8D}^{(n)} + (\mathbf{W}_{8D}^{(n)})^T \right] \boldsymbol{\mu}_{8B}^{(n)} + \left[ \mathbf{W}_{8C}^{(n)} + (\mathbf{W}_{8B}^{(n)})^T \right] \left( \tilde{x}_{n-M_1-N+1} - \mu_{8A}^{(n)} \right) \right\} \\ &= (\tilde{\mathbf{W}}_8^{(n)})^\dagger \left[ \mathbf{W}_{8D}^{(n)} \boldsymbol{\mu}_{8B}^{(n)} + \mathbf{W}_{8C}^{(n)} \left( \tilde{x}_{n-M_1-N+1} - \mu_{8A}^{(n)} \right) \right].\end{aligned}$$

(The last equality applies the symmetry of  $\mathbf{W}_8^{(n)}$ .) Finally, we note that  $(\tilde{\mathbf{W}}_8^{(n)})^\dagger$  can be computed from  $(\mathbf{W}_8^{(n)})^\dagger$  efficiently using block matrix inversion.

We must update each message that corresponds to an estimate in which we wish to incorporate the previous decision and which is actually affected by the decision. Such updates are similar to the update above for  $m_8^{(n)}$ . Then the algorithm can make a decision for another set of symbols, beginning the next iteration in the process. The overall complexity of the algorithm can be shown to be  $\mathcal{O}((M + N)^3)$  per step, assuming one decision is made per step and all messages that are affected by each decision are updated.

### B. Sequential DFE

The above algorithm can simplify significantly given a particular DFE. We next consider a sequential DFE, in which decisions  $\check{x}_n$  are made sequentially based on the LMMSE estimate of  $x_n$  given the  $M = M_1 + M_2 + 1$  observations  $[y_{n-M_1} \ y_{n-M_1+1} \ \dots \ y_{n+M_2}]^T$  under the assumption that the decisions  $\check{x}_k, k < n - M_1$ , are correct. (Note that we allow  $M_1 = 0$  in the development.) Because the order in which decisions are made is predetermined, it is not necessary to equalize the entire set of data prior to making a decision. Furthermore, since the decisions are made sequentially, it is possible to use the messages formed when estimating  $x_{n-1}$  to form the LMMSE estimate (with decision feedback) of symbol  $x_n$ .

Given the message  $m_8^{(n-1)}$  on which the estimate for  $x_{n-1}$  is based, we determine  $m_8^{(n)}$ , taking into account the appropriate past decisions. This update requires two steps. First, the decision for  $x_{n-M_1-1}$  must be incorporated, and the variable  $x_{n-M_1-1}$  integrated out, yielding the distribution  $\tilde{m}_8^{(n-1)} \propto \hat{f}(\mathbf{x}_{n-M_1}^{n+M_2-1} | \mathbf{y}_1^{n+M_2-1}, \check{\mathbf{x}}_1^{n-M_1-1})$ . This operation was treated in the previous subsection, and is equivalent to evaluating  $m_8^{(n-1)}$  at  $x_{n-M_1-1} = \check{x}_{n-M_1-1}$ .

Comparing the distributions described by  $\tilde{m}_8^{(n-1)}$  and  $m_8^{(n)}$ , we have

$$m_8^{(n)} = \tilde{m}_8^{(n-1)} \hat{f}(y_{n+M_2} | \mathbf{x}_{n+M_2}, \tilde{\mathbf{x}}_1^{n-M_1-1}) \hat{f}_{n+M_2}(x_{n+M_2}),$$

which is the second step of the update. Note that this equation is similar to one of the steps in the fast constrained equalization algorithm from Section IV-B. However, there is one difference. In the update here, we multiply in the distribution  $\hat{f}(y_{n+M_2} | \mathbf{x}_{n+M_2}, \tilde{\mathbf{x}}_1^{n-M_1-1})$  instead of  $\hat{f}(y_{n+M_2} | \mathbf{x}_{n+M_2})$ . If  $N \leq M$ , then the two distributions are equal and the difference vanishes. However, if  $N > M$ , then

$$\hat{f}(y_{n+M_2} | \mathbf{x}_{n+M_2}, \tilde{\mathbf{x}}_1^{n-M_1-1}) = \hat{f}(y_{n+M_2} - \mathbf{h}_{n+M_2}(1 : N - M)^T \tilde{\mathbf{x}}_{n-M_1-(N-M)}^{n-M_1-1} | \mathbf{x}_{n-M_1}^{n+M_2}). \quad (12)$$

Table IV gives a description of the sequential DFE algorithm. Several comments should be made regarding this implementation. First, the computation of  $(\mathbf{W}_8^{(M_1+1)})^{-1}$ ,  $\boldsymbol{\mu}_8^{(M_1+1)}$ , and  $\hat{x}_k$ ,  $1 \leq k \leq M_1+1$ , in line 1 can be performed by using the fast constrained linear equalization algorithm given in Table III by modifying line 4 of Table III to read “**for**  $n = 2$  to  $M_1 + 1$  **do**”. Second, in lines 16 and 17, we define

$$\tilde{\mathbf{h}}_{n+M_2}^b = \begin{cases} [\mathbf{0}_{1 \times (P-N)} | \mathbf{h}_{n+M_2}^T]^T, & \text{if } P > N \\ \mathbf{h}_{n+M_2}(N - P + 1 : N), & \text{otherwise,} \end{cases}$$

where  $P = \text{size}(\mathbf{W}_8^{(n)}, 1)$ . Third, through algebraic simplification, lines 6 and 18 are written only in terms of  $\mathbf{W}_8^{-1}$ , not  $\mathbf{W}_8$ . Consequently,  $\mathbf{W}_8$  is no longer needed in the algorithm, and so it is not computed. Finally, we implemented (12) in line 9 by subtracting the contribution of each decision from the affected observations as each decision is made, instead of waiting until a particular observation is needed.

This implementation, along with some optimization to take advantage of the symmetry of the computed matrices and the zero elements of the computed matrices and vectors, requires  $[(1/2)M^2 + MK + (1/2)K^2 + (1/2)M + (3/2)K + N]L + g_1(N, M_1, M_2, K, K_2)$  additions and  $[(1/2)M^2 + MK + (1/2)K^2 + (5/2)M + (5/2)K + N]L + g_2(N, M_1, M_2, K, K_2)$  multiplications, where  $g_1$  and  $g_2$  are multivariate polynomials of total degree 3,  $K = \min(N, M)$ , and  $K_2 = \min(N, M_2)$ . Again, we have separated those terms that grow linearly with the length of the data sequence and those terms, giving rising to the functions  $g_1$  and  $g_2$ , that are independent of the data length.

## VI. APPLICATION TO TURBO EQUALIZATION

An important application of the equalization algorithms presented in this paper is that of so-called turbo equalization. A turbo equalizer consists of a decoder and an equalizer that alternately pass estimates of the transmitted data between them in the form of soft information. The decoder forms estimates by taking into account the structure of the code and the soft information received from the equalizer, while the equalizer considers some portion of the observed sequence, the channel model, and the soft information received from the decoder. Both components treat the soft information as *a priori* information [10]. Here we focus on the equalizer portion of the system, and consider the case of a binary phase shift keying (BPSK) transmission, i.e.,  $\mathcal{B} = \{-1, +1\}$ . For details on common graphical implementations of the decoder subsystem, we refer the reader to [18], [5], and [4].

The soft information sent to the equalizer will be a set of distributions  $f_n(x_n)$ ,  $1 \leq n \leq L$ , that, in the BPSK case, are often parameterized by log-likelihood ratios  $L_D(x_n) = \log(f_n(1)/f_n(-1))$ ,  $1 \leq n \leq L$ . The *a priori* means and variances can be extracted from this distribution and used to compute the LMMSE estimate  $\hat{x}_n$  of each bit  $x_n$  using any of the algorithms in this paper. However, these estimates are insufficient for two reasons. First, the equalizer in a turbo equalization scheme must output a log-likelihood ratio for each  $x_n$  of the form  $L_E(x_n) = \Pr[\hat{x}_n|x_n = +1]/\Pr[\hat{x}_n|x_n = -1]$ , where  $\Pr[A]$  is the probability of event  $A$  [10]. After making the usual assumption that the conditional distribution of  $\hat{x}_n$  given  $x_n$  is Gaussian with mean  $\mu_{n,x} = E\{\hat{x}_n|x_n = x\}$  and variance  $\sigma_{n,x}^2 = \text{Var}\{\hat{x}_n|x_n = x\} \triangleq E\{\hat{x}_n^2|x_n = x\} - E\{\hat{x}_n|x_n = x\}^2$ , we have [10]

$$L_E(x_n) = \frac{2\hat{x}_n\mu_{n,+1}}{\sigma_{n,+1}}. \quad (13)$$

Hence, there are additional computations in finding  $L_E(x_n)$  other than those associated with finding  $\hat{x}_n$ , which could result in a greater computational complexity. (Note that (13) is given merely to justify this last statement; for details on this equation and its role in turbo equalization, we refer to [10].)

The second reason that the equalization algorithms presented heretofore are insufficient for turbo equalization is that for each particular  $n$ ,  $L_E(x_n)$  must not depend on  $L_D(x_n)$ . This requirement, an essential component of the turbo principle [19], [20], implies that  $\hat{x}_n$  must not depend on  $L_D(x_n)$  [10],

[21]. In [14], a joint decoder/linear equalizer graph is employed that, due to the nature of the sum-product algorithm, results in an algorithm that automatically satisfies this principle. However, because of alterations we have made to obtain fast and stable algorithms, such as employing the factorization  $\hat{f}_n(x_n) = \sqrt{\hat{f}_n(x_n)}\sqrt{\hat{f}_n(x_n)}$ , additional steps must be taken to remove the dependency of  $L_E(x_n)$  on  $L_D(x_n)$  for each  $n$ .

In [10], the turbo principle is satisfied by setting  $L_D(x_n) = 0$  when forming the estimate  $\hat{x}_n$ , resulting in the substitution of  $\bar{x}_n = 0$  and  $v_n = 1$ . Equation (13) then becomes [10]

$$L_E(x_n) = \frac{2\mathbf{s}^T \mathbf{R}_n^{-1}}{1 - v_n \mathbf{s}^T \mathbf{R}_n^{-1} \mathbf{s}} (\mathbf{y}_n - \mathbf{H}_n \bar{\mathbf{x}}_n + \mathbf{s} \bar{x}_n), \quad (14)$$

where  $\mathbf{H}_n$  is the  $N \times (N + M - 1)$  time-varying channel convolution matrix,  $\mathbf{R}_n = \mathbf{H}_n \mathbf{V}_n \mathbf{H}_n^T + \mathbf{\Sigma}_n$ ,  $\mathbf{V}_n = \text{diag}(v_{n-N-M_1+1}, v_{n-N-M_1+2}, \dots, v_{n+M_2})$ ,  $\mathbf{\Sigma}_n = \text{diag}(\sigma_{n-M_1}^2, \sigma_{n-M_1+1}^2, \dots, \sigma_{n+M_2}^2)$ ,  $\bar{\mathbf{x}}_n = [\bar{x}_{n-N-M_1+1} \ \bar{x}_{n-N-M_1+2} \ \dots \ \bar{x}_{n+M_2}]^T$ ,  $\mathbf{y}_n = [y_{n-M_1} \ y_{n-M_1+1} \ \dots \ y_{n+M_2}]^T$ ,  $\mathbf{s} = \mathbf{H}_n [\mathbf{0}_{1 \times (N+M_1-1)} \ 1 \ \mathbf{0}_{1 \times M_2}]^T$ , and  $\text{diag}(a_1, a_2, \dots, a_m)$  is a diagonal matrix with elements  $a_1, a_2, \dots, a_m$ . (For conciseness, we consider the constrained equalizer, but the sequel can be modified for any of the equalizers in this paper.) For the algorithms to be useful in turbo equalization, we must efficiently compute  $L_E(x_n)$  of (14).

Note that the LMMSE estimate of  $x_n$  given  $\mathbf{y}_n$  can be expressed as [10]

$$\hat{x}_n = \bar{x}_n + v_n \mathbf{s}^T \mathbf{R}_n^{-1} (\mathbf{y}_n - \mathbf{H}_n \bar{\mathbf{x}}_n). \quad (15)$$

Denote  $\tilde{x}_n(\mu, v)$  as the LMMSE estimate of  $x_n$  given  $\mathbf{y}_n$  after replacing  $\bar{x}_n$  and  $v_n$  with  $\mu$  and  $v$ , respectively. Then from (15), we have the following two results to be used in expressing  $L_E(x_n)$ :

$$\tilde{x}_n(0, v_n) = v_n \mathbf{s}^T \mathbf{R}_n^{-1} (\mathbf{y}_n - \mathbf{H}_n \bar{\mathbf{x}}_n + \bar{x}_n \mathbf{s}),$$

and

$$\tilde{x}_n(0, \infty) \triangleq \lim_{v \rightarrow \infty} \tilde{x}_n(0, v) = \frac{\mathbf{s}^T \mathbf{R}_n^{-1}}{\mathbf{s}^T \mathbf{R}_n^{-1} \mathbf{s}} (\mathbf{y}_n - \mathbf{H}_n \bar{\mathbf{x}}_n + \bar{x}_n \mathbf{s}).$$

It is then easily verified that

$$L_E(x_n) = \frac{2\tilde{x}_n(0, v_n)\tilde{x}_n(0, \infty)}{v_n[\tilde{x}_n(0, \infty) - \tilde{x}_n(0, v_n)]}.$$

All that remains is to show how the factor graph algorithms can be used to compute  $\tilde{x}(0, v_n)$  and  $\tilde{x}(0, \infty)$ .

Note that  $m_9^{(n)} \propto \hat{f}(x_n|\mathbf{y}_n) \propto \hat{f}(x_n, \mathbf{y}_n) = \hat{f}(\mathbf{y}_n|x_n)\hat{f}_n(x_n)$ , where  $\hat{f}(\mathbf{y}_n|x_n)$  does not depend on the *a priori* distribution of  $x_n$ . So the *a posteriori* distribution  $\tilde{f}(x_n|\mathbf{y}_n)$  that assumes that the *a priori* distribution of  $x_n$  is  $\tilde{f}_n(x_n) = \gamma(x_n, 0, v^{-1})$  is simply

$$\tilde{f}(x_n|\mathbf{y}_n) \propto \frac{m_9^{(n)} \tilde{f}_n(x_n)}{\hat{f}_n(x_n)} \propto \gamma(x_n, (W_9 + v^{-1} - v_n^{-1})^\dagger (W_9 \mu_9 - v_n^{-1} \bar{x}_n), W_9 + v^{-1} - v_n^{-1}).$$

Setting  $v = v_n$  gives  $\tilde{x}_n(0, v_n) = W_9^\dagger (W_9 \mu_9 - v_n^{-1} \bar{x}_n)$  and letting  $v \rightarrow \infty$  gives  $\tilde{x}_n(0, \infty) = (W_9 - v_n^{-1})^\dagger (W_9 \mu_9 - v_n^{-1} \bar{x}_n)$ . Using these estimates,  $L_E(x_n)$  can now be computed.

Though we do not discuss it explicitly, this method of computing  $L_E(x_n)$  can be applied in a straightforward manner to all of the algorithms presented here. We refer the reader to [10] for details on the performance of turbo equalization using the constrained LMMSE equalizer and the DFE.

## VII. CONCLUSIONS

We have provided several fast algorithms for equalization with priors based on the factor graph framework and described how these algorithms can be used to compute log-likelihood ratios suitable for turbo equalization. In addition to providing practical algorithms, deriving these equalizers within the factor graph framework has yielded an alternative view of linear equalization that provides insight into LMMSE equalization itself, as well as the relationship between the unconstrained LMMSE equalizer, the constrained LMMSE equalizer, and the DFE.

It is also appealing that all of these algorithms can be derived using factor graphs, suggesting the possibility of forming new algorithms within this framework. For example, such algorithms might be formed by considering factor graphs containing cycles or non-Gaussian factors, or by allowing soft decision feedback. Major obstacles with such algorithms that would need to be addressed include the analysis of the convergence properties when using factor graphs that are not trees, complexity reduction when using non-Gaussian factor graphs, and performance analysis.

## REFERENCES

- [1] J. G. Proakis, *Digital Communications*. New York: McGraw-Hill, Inc., 1995.

- [2] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Trans. Inform. Theory*, vol. 20, no. 2, pp. 284–287, March 1974.
- [3] G. Ungerboeck, "Nonlinear equalization of binary signals in Gaussian noise," *IEEE Trans. Commun.*, vol. 19, no. 6, pp. 1128–1137, December 1971.
- [4] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Trans. Inform. Theory*, vol. 47, no. 2, pp. 498–519, February 2001.
- [5] F. R. Kschischang and B. J. Frey, "Iterative decoding of compound codes by probability propagation in graphical models," *IEEE J. Select. Areas Commun.*, vol. 16, no. 2, pp. 219–230, February 1998.
- [6] H.-A. Loeliger, "Least squares and Kalman filtering," in *Codes, Graphs, and Systems*, R. Blahut and R. Koetter, Eds. Boston: Kluwer, 2002, pp. 113–135.
- [7] R. J. Drost and A. C. Singer, "Factor graph methods for three-dimensional shape reconstruction as applied to LIDAR imaging," *J. Opt. Soc. America A*, vol. 21, no. 10, pp. 1855–1868, October 2004.
- [8] B. J. Frey, R. Koetter, and N. Petrovic, "Codes on images and iterative phase unwrapping," in *Proc. IEEE Inform. Theory Workshop*, September 2001, pp. 9–11.
- [9] M. Coates and R. Nowak, "Network for networks: Internet analysis using graphical statistical models," in *Proc. IEEE Workshop on Neural Networks for Signal Processing*, vol. 2, December 2000, pp. 755–764.
- [10] M. Tüchler, R. Koetter, and A. C. Singer, "Turbo equalization: Principles and new results," *IEEE Trans. Commun.*, vol. 50, no. 5, pp. 754–766, May 2002.
- [11] M. Tüchler, "Iterative equalization using priors," Master's thesis, University of Illinois at Urbana-Champaign, 2000.
- [12] T. K. Moon and W. C. Stirling, *Mathematical Methods and Algorithms for Signal Processing*. Upper Saddle River, NJ: Prentice Hall, 2000.
- [13] S. Korl, "A factor graph approach to signal modeling, system identification and filtering," Ph.D. dissertation, Swiss Federal Institute of Technology, Zurich, 2005.
- [14] M. Tüchler, R. Koetter, and A. C. Singer, "Graphical models for coded data transmission over inter-symbol interference channels," *Eur. Trans. Telecomm.*, vol. 15, no. 4, pp. 307–321, July-August 2004.
- [15] A. Kavčić, X. Ma, and M. Mitzenmacher, "Binary intersymbol interference channels: Gallager codes, density evolution, and code performance bounds," *IEEE Trans. Inform. Theory*, vol. 49, no. 7, pp. 1636–1652, July 2003.
- [16] M. E. Austin, "Decision-feedback equalization for digital communication over dispersive channels," MIT Lincoln Laboratory, Tech. Rep. 437, August 1967.
- [17] H.-A. Loeliger, "Some remarks on factor graphs," in *Proc. 3rd Int. Symp. Turbo Codes*, September 2003, pp. 111–115.
- [18] N. Wiberg, H.-A. Loeliger, and R. Kötter, "Codes and iterative decoding on general graphs," *Eur. Trans. Telecomm.*, vol. 6, no. 5, pp. 513–525, September-October 1995.
- [19] C. Berrou and A. Glavieux, "Near optimum error correcting coding and decoding: Turbo-codes," *IEEE Trans. Commun.*, vol. 44, no. 10, pp. 1261–1271, October 1996.

- [20] J. Hagenauer, "The turbo principle: Tutorial introduction and state of the art," in *Proc. Int. Symp. Turbo Codes*, September 1997, pp. 1–11.
- [21] C. Douillard, A. M. Jézéquel, C. Berrou, A. Picart, P. Didier, and A. Glavieux, "Iterative correction of intersymbol interference: Turbo-equalization," *Eur. Trans. Telecomm.*, vol. 6, no. 5, pp. 507–511, September-October 1995.

### LIST OF FIGURES AND TABLES

Fig. 1. Factor graph for unconstrained LMMSE equalization.

Fig. 2. Modified factor graph for unconstrained LMMSE equalization.

Fig. 3. Factor graph for constrained LMMSE equalization.

Table I. Unconstrained LMMSE equalization algorithm.

Table II. Constrained LMMSE equalization algorithm for  $\hat{x}_1$ .

Table III. Constrained LMMSE equalization algorithm.

Table IV. Decision feedback equalization algorithm.

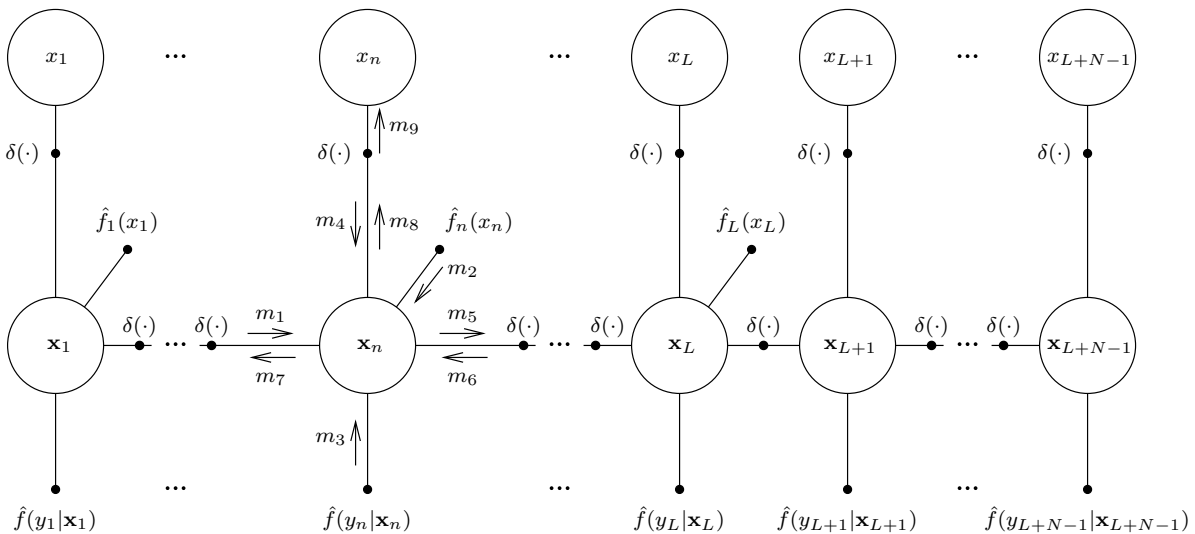


Fig. 1. Factor graph for unconstrained LMMSE equalization.

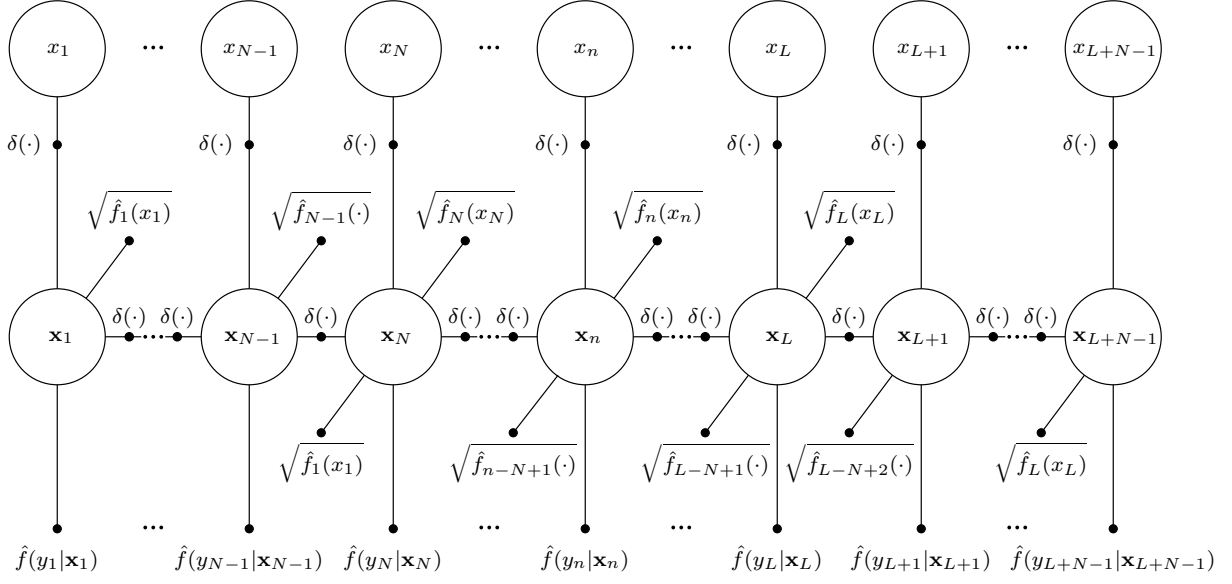


Fig. 2. Modified factor graph for unconstrained LMMSE equalization.

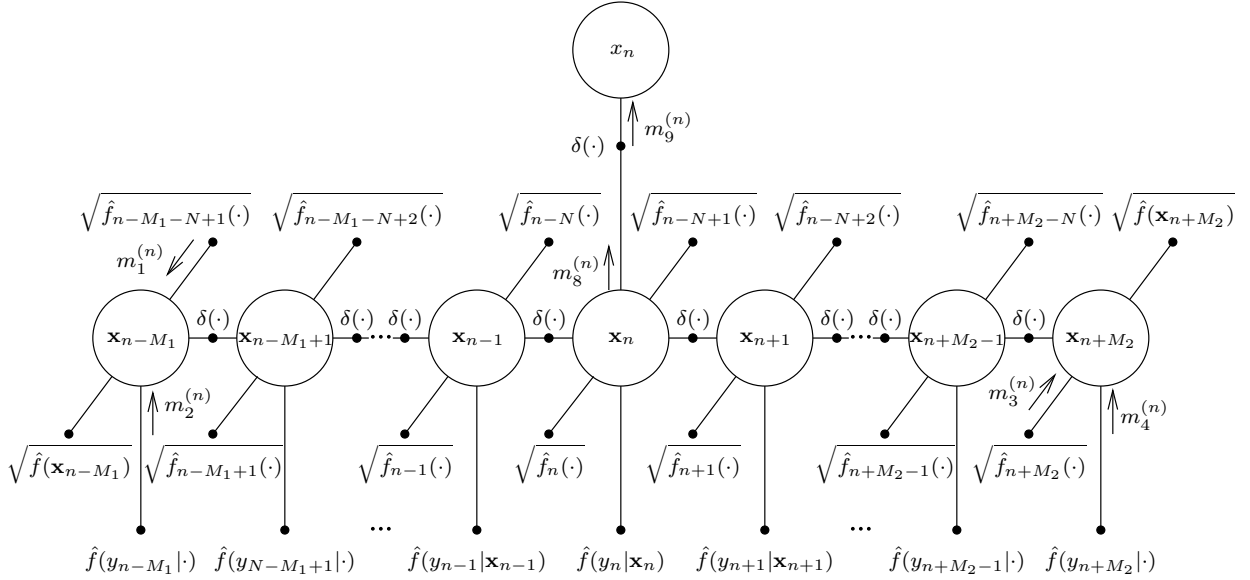


Fig. 3. Factor graph for constrained LMMSE equalization.

TABLE I

## UNCONSTRAINED LMMSE EQUALIZATION ALGORITHM

$\mathbf{W}_1 = \boldsymbol{\mu}'_1 = 0$   
**for**  $n = 1$  to  $N - 1$  **do**  
 $\mathbf{W}_1(n, n) \leftarrow 1/(2v_n)$   
 $\mathbf{W}_1 \leftarrow \mathbf{W}_1 + \sigma_n^{-2} \mathbf{h}_n \mathbf{h}_n^T$   
5:  $\mathbf{W}_1 \leftarrow \begin{bmatrix} \mathbf{W}_1 & \mathbf{0}_{n \times 1} \\ \mathbf{0}_{1 \times n} & 0 \end{bmatrix}$   
 $\boldsymbol{\mu}'_1(n) \leftarrow \bar{x}_n/(2v_n)$   
 $\boldsymbol{\mu}'_1 \leftarrow \boldsymbol{\mu}'_1 + \sigma_n^{-2} y_n \mathbf{h}_n$   
 $\boldsymbol{\mu}'_1 \leftarrow [\boldsymbol{\mu}'_1^T \ 0]^T$   
**end for**  
10: Compute  $\mathbf{W}_1^\dagger$  from  $\mathbf{W}_1$   
 $\boldsymbol{\mu}_1 \leftarrow \mathbf{W}_1^\dagger \boldsymbol{\mu}'_1$   
**for**  $n = N$  to  $L$  **do**  
 $\mathbf{W}_5^{(n)} \leftarrow \mathbf{W}_1$   
 $\mathbf{W}_5^{(n)}(N, N) \leftarrow 1/(2v_n)$   
15:  $\mathbf{W}_5^{(n)}(1, 1) \leftarrow \mathbf{W}_5^{(n)}(1, 1) + 1/(2v_{n-N+1})$   
 $\mathbf{W}_5^{(n)} \leftarrow \mathbf{W}_5^{(n)} + \sigma_n^{-2} \mathbf{h}_n \mathbf{h}_n^T$   
 $\mathbf{W}_5^{-1} \leftarrow \mathbf{W}_1^\dagger$   
 $\mathbf{W}_5^{-1}(N, N) \leftarrow 2v_n$   
 $\mathbf{W}_5^{-1} \leftarrow \mathbf{W}_5^{-1} - \mathbf{W}_5^{-1}(:, 1)$   
 $\quad \times [2v_{n-N+1} + \mathbf{W}_5^{-1}(1, 1)]^{-1} \mathbf{W}_5^{-1}(1, :)$   
20:  $\mathbf{W}_5^{-1} \leftarrow \mathbf{W}_5^{-1}$   
 $\quad - \mathbf{W}_5^{-1} \mathbf{h}_n (\sigma_n^2 + \mathbf{h}_n^T \mathbf{W}_5^{-1} \mathbf{h}_n)^{-1} \mathbf{h}_n^T \mathbf{W}_5^{-1}$   
 $\boldsymbol{\mu}_5 \leftarrow \mathbf{W}_1 \boldsymbol{\mu}_1$   
 $\boldsymbol{\mu}_5(N) \leftarrow \bar{x}_n/(2v_n)$   
 $\boldsymbol{\mu}_5(1) \leftarrow \boldsymbol{\mu}_5(1) + \bar{x}_{n-N+1}/(2v_{n-N+1})$   
 $\boldsymbol{\mu}_5^{(n)} \leftarrow \boldsymbol{\mu}_5 + \sigma_n^{-2} y_n \mathbf{h}_n$   
25:  $\boldsymbol{\mu}_5 \leftarrow \mathbf{W}_5^{-1} \boldsymbol{\mu}_5^{(n)}$   
Partition  $\mathbf{W}_5^{(n)} =$   
 $\begin{bmatrix} \mathbf{W}_{5A}^{(n)}(1 \times 1) & \mathbf{W}_{5B}^{(n)}(1 \times N-1) \\ \mathbf{W}_{5C}^{(n)}(N-1 \times 1) & \mathbf{W}_{5D}^{(n)}(N-1 \times N-1) \end{bmatrix}$   
 $\mathbf{W}_1 \leftarrow \begin{bmatrix} \mathbf{W}_{5D}^{(n)} - \mathbf{W}_{5C}^{(n)} (\mathbf{W}_{5A}^{(n)})^{-1} \mathbf{W}_{5B}^{(n)} & \mathbf{0}_{N-1 \times 1} \\ \mathbf{0}_{1 \times N-1} & 0 \end{bmatrix}$   
 $\mathbf{W}_1^\dagger \leftarrow \begin{bmatrix} \mathbf{W}_5^{-1}(2 : N, 2 : N) & \mathbf{0}_{N-1 \times 1} \\ \mathbf{0}_{1 \times N-1} & 0 \end{bmatrix}$   
 $\boldsymbol{\mu}_1 \leftarrow [\boldsymbol{\mu}_5(2 : N)^T \ 0]^T$   
30: **end for**  
 $\mathbf{W}_6 = \boldsymbol{\mu}'_6 = 0$   
**for**  $n = L + N - 1$  to  $L + 1$  by  $-1$  **do**  
 $\mathbf{W}_6(1, 1) \leftarrow 1/(2v_{n-N+1})$   
 $\mathbf{W}_6 \leftarrow \mathbf{W}_6 + \sigma_n^{-2} \mathbf{h}_n \mathbf{h}_n^T$   
35:  $\mathbf{W}_6 \leftarrow \begin{bmatrix} 0 & \mathbf{0}_{1 \times L+N-n} \\ \mathbf{0}_{L+N-n \times 1} & \mathbf{W}_6 \end{bmatrix}$   
 $\boldsymbol{\mu}'_6(1) \leftarrow \bar{x}_{n-N+1}/(2v_{n-N+1})$   
 $\boldsymbol{\mu}'_6 \leftarrow \boldsymbol{\mu}'_6 + \sigma_n^{-2} y_n \mathbf{h}_n$   
 $\boldsymbol{\mu}'_6 \leftarrow [0 \ \boldsymbol{\mu}'_6^T]^T$   
**end for**  
40: Compute  $\mathbf{W}_6^\dagger$  from  $\mathbf{W}_6$   
 $\boldsymbol{\mu}_6 \leftarrow \mathbf{W}_6^\dagger \boldsymbol{\mu}'_6$   
 $\mathbf{W}_8 \leftarrow \mathbf{W}_5 + \mathbf{W}_6$   
Compute  $\mathbf{W}_8^{-1}$  from  $\mathbf{W}_8$   
 $\boldsymbol{\mu}_8 \leftarrow \mathbf{W}_8^{-1}(\boldsymbol{\mu}_5^{(L)} + \boldsymbol{\mu}'_6)$   
45:  $(W_9^{(L)})^{-1} \leftarrow \mathbf{W}_8^{-1}(N, N)$   
 $\hat{x}_L \leftarrow \boldsymbol{\mu}_8(N)$   
**for**  $n = L - 1$  to  $N$  by  $-1$  **do**  
 $\mathbf{W}_7 \leftarrow \mathbf{W}_6$   
 $\mathbf{W}_7(1, 1) \leftarrow 1/(2v_{n-N+2})$   
50:  $\mathbf{W}_7(N, N) \leftarrow \mathbf{W}_7(N, N) + 1/(2v_{n+1})$   
 $\mathbf{W}_7 \leftarrow \mathbf{W}_7 + \sigma_{n+1}^{-2} \mathbf{h}_{n+1} \mathbf{h}_{n+1}^T$   
 $\mathbf{W}_7^{-1} \leftarrow \mathbf{W}_6^\dagger$   
 $\mathbf{W}_7^{-1}(1, 1) \leftarrow 2v_{n-N+2}$   
 $\mathbf{W}_7^{-1} \leftarrow \mathbf{W}_7^{-1} - \mathbf{W}_7^{-1}(:, N)$   
 $\quad \times [2v_{n+1} + \mathbf{W}_7^{-1}(N, N)]^{-1} \mathbf{W}_7^{-1}(N, :)$   
55:  $\mathbf{W}_7^{-1} \leftarrow \mathbf{W}_7^{-1} - \mathbf{W}_7^{-1} \mathbf{h}_{n+1}$   
 $\quad \times (\sigma_{n+1}^2 + \mathbf{h}_{n+1}^T \mathbf{W}_7^{-1} \mathbf{h}_{n+1})^{-1} \mathbf{h}_{n+1}^T \mathbf{W}_7^{-1}$   
 $\boldsymbol{\mu}_7 \leftarrow \mathbf{W}_6 \boldsymbol{\mu}_6$   
 $\boldsymbol{\mu}_7(1) \leftarrow \bar{x}_{n-N+2}/(2v_{n-N+2})$   
 $\boldsymbol{\mu}_7(N) \leftarrow \boldsymbol{\mu}_7(N) + \bar{x}_{n+1}/(2v_{n+1})$   
 $\boldsymbol{\mu}_7 \leftarrow \mathbf{W}_7^{-1}(\boldsymbol{\mu}_7 + \sigma_{n+1}^{-2} y_{n+1} \mathbf{h}_{n+1})$   
60:  $\mathbf{W}_{6D} \leftarrow \mathbf{W}_7(1 : N-1, 1 : N-1) - \mathbf{W}_7(1 : N-1, N)$   
 $\quad \times \mathbf{W}_7(N, N)^{-1} \mathbf{W}_7(N, 1 : N-1)$   
 $\mathbf{W}_6 \leftarrow \begin{bmatrix} 0 & \mathbf{0}_{1 \times N-1} \\ \mathbf{0}_{N-1 \times 1} & \mathbf{W}_{6D} \end{bmatrix}$   
 $\mathbf{W}_6^\dagger \leftarrow \begin{bmatrix} 0 & \mathbf{0}_{1 \times N-1} \\ \mathbf{0}_{N-1 \times 1} & \mathbf{W}_7^{-1}(1 : N-1, 1 : N-1) \end{bmatrix}$   
 $\boldsymbol{\mu}_6 \leftarrow [0 \ \boldsymbol{\mu}_7(1 : N-1)^T]^T$   
 $\mathbf{V}_{8D} \leftarrow \mathbf{W}_8^{-1}(1 : N-1, 1 : N-1)$   
65:  $\mathbf{V}_{8A} \leftarrow (\mathbf{W}_{5A}^{(n)} + \mathbf{W}_{5B}^{(n)} \mathbf{V}_{8D} \mathbf{W}_{5C}^{(n)}) / (\mathbf{W}_{5A}^{(n)})^2$   
 $\mathbf{V}_{8B} \leftarrow -\mathbf{W}_{5B}^{(n)} \mathbf{V}_{8D} / \mathbf{W}_{5A}^{(n)}$   
 $\mathbf{V}_{8C} \leftarrow -\mathbf{V}_{8D} \mathbf{W}_{5C}^{(n)} / \mathbf{W}_{5A}^{(n)}$   
 $\mathbf{W}_8^{-1} \leftarrow \begin{bmatrix} \mathbf{V}_{8A} & \mathbf{V}_{8B} \\ \mathbf{V}_{8C} & \mathbf{V}_{8D} \end{bmatrix}$   
 $\boldsymbol{\mu}_8 \leftarrow \mathbf{W}_8^{-1}(\boldsymbol{\mu}_5^{(n)} + \mathbf{W}_6 \boldsymbol{\mu}_6)$   
70:  $(W_9^{(n)})^{-1} \leftarrow \mathbf{W}_8^{-1}(N, N)$   
 $\hat{x}_n \leftarrow \boldsymbol{\mu}_8(N)$   
**end for**  
**for**  $n = N - 1$  to  $1$  by  $-1$  **do**  
 $(W_9^{(n)})^{-1} \leftarrow \mathbf{W}_8^{-1}(n, n)$   
75:  $\hat{x}_n \leftarrow \boldsymbol{\mu}_8(n)$   
**end for**

TABLE II

CONSTRAINED LMMSE EQUALIZATION ALGORITHM FOR  $\hat{x}_1$ .

```

if  $M_2 > N - 1$  then
   $\mathbf{W}_7 \leftarrow \text{diag}\{v_{M_2-N+2}^{-1}, v_{M_2-N+3}^{-1}, \dots, v_{M_2+1}^{-1}\}$ 
   $\boldsymbol{\mu}'_7 \leftarrow [\bar{x}_{M_2-N+2}/v_{M_2-N+2} \ \bar{x}_{M_2-N+3}/v_{M_2-N+3}$ 
     $\dots \bar{x}_{M_2+1}/v_{M_2+1}]^T$ 
else
5:   $\mathbf{W}_7 \leftarrow \text{diag}\{v_1^{-1}, v_2^{-1}, \dots, v_{M_2+1}^{-1}\}$ 
     $\boldsymbol{\mu}'_7 \leftarrow [\bar{x}_1/v_1 \ \bar{x}_2/v_2 \ \dots \ \bar{x}_{M_2+1}/v_{M_2+1}]^T$ 
end if
   $\mathbf{W}_7 \leftarrow \mathbf{W}_7 + \sigma_{M_2+1}^{-2} \mathbf{h}_{M_2+1} \mathbf{h}_{M_2+1}^T$ 
   $\boldsymbol{\mu}'_7 \leftarrow \boldsymbol{\mu}'_7 + \sigma_{M_2+1}^{-2} y_{M_2+1} \mathbf{h}_{M_2+1}$ 
10: for  $n = M_2$  to 1 by -1 do
    if  $n \geq N$  then
       $\mathbf{W}_7 \leftarrow \begin{bmatrix} v_{n-N+1}^{-1} & \mathbf{0}_{1 \times N+M_2-n} \\ \mathbf{0}_{N+M_2-n \times 1} & \mathbf{W}_7 \end{bmatrix}$ 
       $\mathbf{W}_7(1:N, 1:N) \leftarrow \mathbf{W}_7(1:N, 1:N) + \sigma_n^{-2} \mathbf{h}_n \mathbf{h}_n^T$ 
       $\boldsymbol{\mu}'_7 \leftarrow [\bar{x}_{n-N+1}/v_{n-N+1} \ \boldsymbol{\mu}'_7]^T$ 
15:   $\boldsymbol{\mu}'_7(1:N) \leftarrow \boldsymbol{\mu}'_7(1:N) + \sigma_n^{-2} y_n \mathbf{h}_n$ 
    else
       $\mathbf{W}_7(1:n, 1:n) \leftarrow \mathbf{W}_7(1:n, 1:n) + \sigma_n^{-2} \mathbf{h}_n \mathbf{h}_n^T$ 
       $\boldsymbol{\mu}'_7(1:n) \leftarrow \boldsymbol{\mu}'_7(1:n) + \sigma_n^{-2} y_n \mathbf{h}_n$ 
    end if
20: end for
   $\mathbf{W}_8 \leftarrow \mathbf{W}_7$ 
  Compute  $\mathbf{W}_8^{-1}$  from  $\mathbf{W}_8$ 
   $\boldsymbol{\mu}_8 \leftarrow \mathbf{W}_8^{-1} \boldsymbol{\mu}'_7$ 
   $(W_9^{(1)})^{-1} \leftarrow \mathbf{W}_8^{-1}(1, 1)$ 
25:  $\hat{x}_1 \leftarrow \boldsymbol{\mu}_8(1)$ 

```

TABLE III

CONSTRAINED LMMSE EQUALIZATION ALGORITHM.

Compute  $\mathbf{W}_8^{(1)}$ ,  $(\mathbf{W}_8^{(1)})^{-1}$  and  $\boldsymbol{\mu}_8^{(1)}$   
 $\mathbf{W}_8^{-1} \leftarrow (\mathbf{W}_8^{(1)})^{-1}$   
 $\boldsymbol{\mu}'_8 \leftarrow \mathbf{W}_8^{(1)} \boldsymbol{\mu}_8^{(1)}$   
**for**  $n = 2$  to  $L$  **do**  
 5: **if**  $n > M_1 + 1$  **then**  
      $\mathbf{W}_8^{-1} \leftarrow \mathbf{W}_8^{-1} - \mathbf{W}_8^{-1} \mathbf{h}_{n-M_1-1}^t (-\sigma_{n-M_1-1}^2$   
          $+ \mathbf{h}_{n-M_1-1}^{tT} \mathbf{W}_8^{-1} \mathbf{h}_{n-M_1-1}^t)^{-1} \mathbf{h}_{n-M_1-1}^{tT} \mathbf{W}_8^{-1}$   
      $\boldsymbol{\mu}'_8 \leftarrow \boldsymbol{\mu}'_8 - \sigma_{n-M_1-1}^{-2} y_{n-M_1-1} \mathbf{h}_{n-M_1-1}^t$   
   **end if**  
   **if**  $n > M_1 + N$  **then**  
 10:  $\mathbf{W}_8^{-1} \leftarrow \mathbf{W}_8^{-1}(2 : \text{end}, 2 : \text{end})$   
      $\boldsymbol{\mu}'_8 \leftarrow \boldsymbol{\mu}'_8(2 : \text{end})$   
   **end if**  
   **if**  $n < L - M_2 + 1$  **then**  
      $\mathbf{W}_8^{-1} \leftarrow \begin{bmatrix} \mathbf{W}_8^{-1} & \mathbf{0}_{\text{size}(\mathbf{w}_8^{-1},1) \times 1} \\ \mathbf{0}_{1 \times \text{size}(\mathbf{w}_8^{-1},2)} & v_{n+M_2} \end{bmatrix}$   
 15:  $\boldsymbol{\mu}'_8 \leftarrow [\boldsymbol{\mu}'_8^T \quad \bar{x}_{n+M_2}/v_{n+M_2}]^T$   
   **end if**  
   **if**  $n < L + N - M_2$  **then**  
      $\mathbf{W}_8^{-1} \leftarrow \mathbf{W}_8^{-1} - \mathbf{W}_8^{-1} \mathbf{h}_{n+M_2}^b (\sigma_{n+M_2}^2$   
          $+ \mathbf{h}_{n+M_2}^{bT} \mathbf{W}_8^{-1} \mathbf{h}_{n+M_2}^b)^{-1} \mathbf{h}_{n+M_2}^{bT} \mathbf{W}_8^{-1}$   
      $\boldsymbol{\mu}'_8 \leftarrow \boldsymbol{\mu}'_8 + \sigma_{n+M_2}^{-2} y_{n+M_2} \mathbf{h}_{n+M_2}^b$   
 20: **end if**  
      $\boldsymbol{\mu}_8 \leftarrow \mathbf{W}_8^{-1} \boldsymbol{\mu}'_8$   
     **if**  $n < M_1 + N + 1$  **then**  
        $(W_9^{(n)})^{-1} \leftarrow \mathbf{W}_8^{-1}(n, n)$   
        $\hat{x}_n \leftarrow \boldsymbol{\mu}_8(n)$   
 25: **else**  
      $(W_9^{(n)})^{-1} \leftarrow \mathbf{W}_8^{-1}(M_1 + N, M_1 + N)$   
      $\hat{x}_n \leftarrow \boldsymbol{\mu}_8(M_1 + N)$   
   **end if**  
**end for**

TABLE IV

DECISION FEEDBACK EQUALIZATION ALGORITHM.

Compute  $(\mathbf{W}_8^{(M_1+1)})^{-1}$ ,  $\boldsymbol{\mu}_8^{(M_1+1)}$ , and  $\hat{x}_k$ ,  $k \leq M_1 + 1$   
 Determine  $\check{x}_k$  from  $\hat{x}_k$ ,  $k \leq M_1 + 1$   
 $\mathbf{W}_8^{-1} \leftarrow (\mathbf{W}_8^{(M_1+1)})^{-1}$   
 $\boldsymbol{\mu}_8 \leftarrow \boldsymbol{\mu}_8^{(M_1+1)}$   
 5: **for**  $n = M_1 + 2$  to  $L$  **do**  
      $\boldsymbol{\mu}_8 \leftarrow \boldsymbol{\mu}_8(2 : \text{end}) + [\check{x}_{n-M_1-1} - \boldsymbol{\mu}_8(1)]$   
          $\times \mathbf{W}_8^{-1}(2 : \text{end}, 1) / \mathbf{W}_8^{-1}(1, 1)$   
      $\mathbf{W}_8^{-1} \leftarrow \mathbf{W}_8^{-1}(2 : \text{end}, 2 : \text{end})$   
          $- \mathbf{W}_8^{-1}(2 : \text{end}, 1) \mathbf{W}_8^{-1}(1, 2 : \text{end}) / \mathbf{W}_8^{-1}(1, 1)$   
     **for**  $m = n + M_2$  to  $n - M_1 + N - 2$  **do**  
        $y_m \leftarrow y_m - \check{x}_{n-M_1-1} \mathbf{h}_m(n - M_1 + N - 1 - m)$   
 10: **end for**  
     **if**  $n < L - M_2 + 1$  **then**  
        $\mathbf{W}_8^{-1} \leftarrow \begin{bmatrix} \mathbf{W}_8^{-1} & \mathbf{0}_{\text{size}(\mathbf{w}_8^{-1},1) \times 1} \\ \mathbf{0}_{1 \times \text{size}(\mathbf{w}_8^{-1},2)} & v_{n+M_2} \end{bmatrix}$   
        $\boldsymbol{\mu}_8 \leftarrow [\boldsymbol{\mu}_8^T \quad \bar{x}_{n+M_2}]^T$   
     **end if**  
 15: **if**  $n < L + N - M_2$  **then**  
      $\mathbf{W}_8^{-1} \leftarrow \mathbf{W}_8^{-1} - \mathbf{W}_8^{-1} \tilde{\mathbf{h}}_{n+M_2}^b (\sigma_{n+M_2}^2$   
          $+ \tilde{\mathbf{h}}_{n+M_2}^{bT} \mathbf{W}_8^{-1} \tilde{\mathbf{h}}_{n+M_2}^b)^{-1} \tilde{\mathbf{h}}_{n+M_2}^{bT} \mathbf{W}_8^{-1}$   
      $\boldsymbol{\mu}_8 \leftarrow \boldsymbol{\mu}_8 + \sigma_{n+M_2}^{-2} (y_{n+M_2} - \tilde{\mathbf{h}}_{n+M_2}^{bT} \boldsymbol{\mu}_8)$   
          $\times \mathbf{W}_8^{-1} \tilde{\mathbf{h}}_{n+M_2}^b$   
     **end if**  
      $(W_9^{(n)})^{-1} \leftarrow \mathbf{W}_8^{-1}(M_1 + 1, M_1 + 1)$   
 20:  $\hat{x}_n \leftarrow \boldsymbol{\mu}_8(M_1 + 1)$   
     Determine  $\check{x}_n$  from  $\hat{x}_n$   
**end for**