1-1-2005

Turbo equalization based on factor graphs

Qinghua Guo
University of Hong Kong, qguo@uow.edu.au

Li Ping
City University of Hong Kong

Hans-Andrea Loeliger
ETH Zürich

Turbo equalization based on factor graphs

Abstract

This paper presents a factor graph approach to turbo equalization. Unlike the existing linear MMSE turbo equalization methods, which operate with truncated windows (sliding or extending window), the proposed is a full-window approach with low complexity. This approach supports a high-speed parallel implementation technique, which makes it an attractive option in practice.

Keywords

equalization, factor, turbo, graphs

Disciplines

Engineering | Science and Technology Studies

Publication Details

Q. Guo, L. Ping & H. Loeliger, "Turbo equalization based on factor graphs," in IEEE International Symposium on Information Theory - Proceedings, 2005, pp. 2021-2025.

Turbo Equalization Based on Factor Graphs

Qinghua Guo Li Ping
Department of Electronic Engineering
City University of Hong Kong, Hong Kong
Email: eeliping@cityu.edu.hk

Hans-Andrea Loeliger
The Signal Processing Lab (ISI)
ETH Zürich,Switzerland
Email: loeliger@isi.ee.ethz.ch

***Abstract*—This paper presents a factor graph approach to turbo equalization. Unlike the existing linear MMSE turbo equalization methods, which operate with truncated windows (sliding or extending window), the proposed is a full-window approach with low complexity. This approach supports a high-speed parallel implementation technique, which makes it an attractive option in practice.**

I. INTRODUCTION

Recently, turbo equalization [1]-[8] has been extensively studied to alleviate inter-symbol interference (ISI) over multipath channels. A turbo equalizer consists of two basic processors: an elementary signal estimator (ESE) (or sometimes called a soft-in-soft-out (SISO) channel equalizer) and an *a posteriori* probability (APP) channel decoder (DEC). The two processors operate in an iterative manner.

The optimal realization of the ESE is the maximum *a posteriori* probability (MAP) algorithm [1]-[3] that has very high complexity. A low-cost alternative based on the linear minimum mean-square error (LMMSE) principle has been proposed in [4], and further investigated in [5]-[8], which provide a good tradeoff between performance and complexity. Ideally, the LMMSE estimates should be based on the entire observation vector (denoted by $\boldsymbol{r}$ below, see (1)). However, due to cost concerns, the existing methods usually work on a window principle, i.e., using a truncated version of $\boldsymbol{r}$. The extending window algorithm derived in [8] is a low-cost alternative using a Cholesky decomposition technique. It can also be shown that the so-called joint Gaussian (JG) algorithm derived in [9] is equivalent to the LMMSE one, as can be verified by carefully comparing eqn. (5) in [8] and eqn. (19) in [9]. Note that a certain performance loss may be incurred by using these window methods. The loss can be reduced by increasing the window size at the price of increased complexity.

In this paper, we propose a factor graph [10][11] method. The new technique is a full-window method, meaning that it produces the LMMSE estimates based on the entire $\boldsymbol{r}$. The new method allows a high-speed parallel implementation technique based on the so-called flooding schedule, which makes it an attractive option for practical applications.

II. BASIC PRINCIPLE OF TURBO EQUALIZATION

Consider a coded linear system characterized by the following matrix equation

$$\boldsymbol{r} = \boldsymbol{H}\boldsymbol{x} + \boldsymbol{n}, \tag{1}$$

where $\boldsymbol{r}$ is a length-N observation vector, $\boldsymbol{H}$ a $N\times J$ system transfer matrix (usually representing the multiplicative effect of the channel), $\boldsymbol{x}$ a length-J sequence formed by the outputs of a forward error control (FEC) code and $\boldsymbol{n}$ an additive noise vector with zero mean and covariance $\sigma^2\boldsymbol{I}$. We assume that $\boldsymbol{x}$ is interleaved by an interleaver π before transmission. The system under consideration is "*binary*", in the sense that the elements of $\boldsymbol{x}$ are binary, i.e., $x_j \in \{+1,-1\}$, j=1, 2, …, J. Although we only discuss real $\boldsymbol{H}$ here, our discussion can be easily extended to the systems with QPSK signaling and complex channel coefficients, since a complex matrix equation can always be represented by an equivalent real one by equating the real and imaginary part separately [8].

A receiver will detect the transmitted codeword $\boldsymbol{x}$ based on $\boldsymbol{r}$. A sub-optimal "turbo" receiver structure is illustrated in Fig.1. It consists of an ESE and a DEC interconnected by a deinterleaver π^{-1} and an interleaver π. The following is a brief outline of the turbo detection principle. The extrinsic LLR generated by ESE is defined as follows

$$L_{ESE}(x_j) \equiv \ln \frac{\Pr\left(\boldsymbol{r} \mid x_j = +1\right)}{\Pr\left(\boldsymbol{r} \mid x_j = -1\right)} \quad j = 1, 2, \ldots, J \tag{2}$$

and is computed based on the channel observation and some *a priori* information about x_i, i = 1, 2, …, j-1, j+1, …, J. Notice that the *a priori* information about x_j itself is excluded here and hence the term "extrinsic" [12].

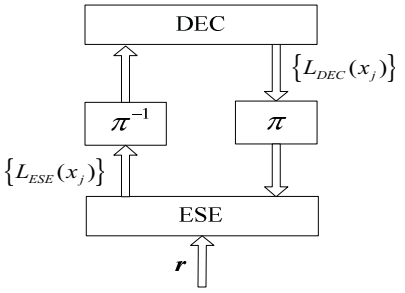

Fig. 1. The overall turbo receiver.

Similarly, we define $L_{DEC}(x_j)$ as the extrinsic LLR generated by the DEC [12]. The LLRs defined above are generated following the iterative process outlined below.

- The ESE computes $\{L_{ESE}(x_j)\}$ ignoring the FEC coding constraint (i.e., as if $\{x_j\}$ are un-coded bits). For the initial step, we assume that no *a priori* information is available at the ESE, so we set $\overline{x}_j = 0$ and $v_j = 1$, respectively, for all j.
- Using $\{L_{ESE}(x_j)\}$ as inputs, the DEC computes $\{L_{DEC}(x_j)\}$ based on the FEC coding constraint.
- After the DEC operations, $\{L_{DEC}(x_j)\}$ are fed back to ESE for the second iteration. In this case, the *a priori* mean and variance of x_j are updated by the feedback from the DEC [8] as

$$E\left(x_j\right) = \frac{\exp\left(L_{DEC}\left(x_j\right)\right) - 1}{\exp\left(L_{DEC}\left(x_j\right)\right) + 1}, \ \forall j$$

$$Var(x_j) = 1 - \left(E(x_j)\right)^2, \ \forall j. \tag{3}$$

  The ESE then re-compute $\{L_{ESE}(x_j)\}$ using these *a priori* means and variances.
- The above process continues iteratively. The DEC makes hard decisions on transmitted bits during the final iteration.

In the above, the DEC function involves a standard *a posteriori* probability (APP) decoding. In what follows, we will concentrate on the ESE function.

A. The Joint Gaussian (JG) Approach to ESE

Let us now focus on the estimation of one particular bit, say x_j, and treat all the other bits as interference. The basic signal model (1) can be rewritten as

$$\boldsymbol{r} = \boldsymbol{h}_j x_j + \boldsymbol{\xi}_j , \tag{4}$$

where $\boldsymbol{h}_j$ is the jth column of $\boldsymbol{H}$ and $\boldsymbol{\xi}_j = \sum_{i \neq j} \boldsymbol{h}_i x_i + \boldsymbol{n}$. The basic assumptions of the JG approach are as follows.

a) The entries of $\boldsymbol{\xi}_j$ are jointly Gaussian (according to the central limit theorem);
b) $\{x_j\}$ are mutually independent (i.e., the coding constraint is ignored); and
c) x_j is independent with $\boldsymbol{\xi}_j$.

Based on the above assumptions, we can compute the extrinsic LLR of x_j based on $\boldsymbol{r}$ as

$$\begin{aligned} L_{ESE}^{JG}\left(x_j\right) &= \ln \frac{P\left(\boldsymbol{r} \mid x_j = +1\right)}{P\left(\boldsymbol{r} \mid x_j = -1\right)} \\ &= \ln \frac{\frac{1}{\left(2\pi\right)^{N/2}\left|\boldsymbol{R}_j\right|^{1/2}} \exp\left[-\frac{1}{2}\left(\boldsymbol{r}-\boldsymbol{h}_j-\overline{\boldsymbol{\xi}}_j\right)^T \boldsymbol{R}_j^{-1}\left(\boldsymbol{r}-\boldsymbol{h}_j-\overline{\boldsymbol{\xi}}_j\right)\right]}{\frac{1}{\left(2\pi\right)^{N/2}\left|\boldsymbol{R}_j\right|^{1/2}} \exp\left[-\frac{1}{2}\left(\boldsymbol{r}+\boldsymbol{h}_j-\overline{\boldsymbol{\xi}}_j\right)^T \boldsymbol{R}_j^{-1}\left(\boldsymbol{r}+\boldsymbol{h}_j-\overline{\boldsymbol{\xi}}_j\right)\right]} \\ &= 2\boldsymbol{h}_j^T \boldsymbol{R}_j^{-1}\left(\boldsymbol{r}-\overline{\boldsymbol{\xi}}_j\right) \\ &= 2\boldsymbol{h}_j^T \boldsymbol{R}_j^{-1}\left(\boldsymbol{r}-\boldsymbol{H}\overline{\boldsymbol{x}}+\boldsymbol{h}_j\overline{x}_j\right). \end{aligned} \tag{5}$$

In the above equation, '$\overline{(\cdot)}$' is the expectation operator and

$$\boldsymbol{R}_j \equiv Cov(\boldsymbol{\xi}_j, \boldsymbol{\xi}_j) = \boldsymbol{H}\boldsymbol{V}_j\boldsymbol{H}^{\mathrm{T}} + \sigma^2\boldsymbol{I} \tag{6}$$

where $\boldsymbol{V}_j$ = diag $[v_1, \ldots v_{j-1}, 0, v_{j+1}, \ldots, v_J]$.

Eqn. (5) involves a matrix $\boldsymbol{R}_j$ of size $N \times N$ and it is generally different for different j. The computational cost involved in (5) can thus be substantial and an efficient implementation algorithm is an important issue. In the following, we will consider a factor graph technique. The complexity of the message passing algorithm in solving the factor graph is determined by the sparseness of the graph. Therefore the proposed approach is efficient when $\boldsymbol{R}_j$ is sparse.

As a comment, the key to the JG approach is to approximate interference and noise (with respect to a particular transmitted bit) by a joint Gaussian vector. This is done directly using the channel model in (1). A similar technique is also used in the LMMSE approach [7], except that the Gaussian approximation is applied after LMMSE filtering of the channel output. It can be shown that the two approaches are actually equivalent. However, it appears that the JG approach above is more concise and straightforward.

B. MMSE Estimation of a Gaussian System

In (1), the elements of $\boldsymbol{x}$ are binary. We now consider a system that is also characterized by (1): $\boldsymbol{r} = \boldsymbol{H}\boldsymbol{x} + \boldsymbol{n}$, but the distributions of the elements of $\boldsymbol{x}$ are Gaussian. We call this new system "the companion Gaussian system" of the original binary system. For the companion Gaussian system, it is well known that the MMSE estimation for $\boldsymbol{x}$ based on $\boldsymbol{r}$ is the *a posteriori* mean $\overline{\boldsymbol{x}}^{post} \equiv E(\boldsymbol{x}|\boldsymbol{r})$ [13]. It can be computed by

$$\overline{\boldsymbol{x}}^{post} = \overline{\boldsymbol{x}} + Cov(\boldsymbol{x}, \boldsymbol{r})\boldsymbol{R}^{-1}\left(\boldsymbol{r} - \boldsymbol{H}\overline{\boldsymbol{x}}\right) \tag{7}$$

where $\overline{\boldsymbol{x}}$ is the *a priori* mean of $\boldsymbol{x}$,

$$\boldsymbol{R} = Cov(\boldsymbol{r}, \boldsymbol{r}) = \boldsymbol{H}\boldsymbol{V}\boldsymbol{H}^{\mathrm{T}} + \sigma^2\boldsymbol{I}, \tag{8}$$

$\boldsymbol{V}$ = diag $[v_1, v_2, \ldots, v_J]$, and $\boldsymbol{I}$ is a unit matrix with the same size as $\boldsymbol{R}$. Comparing (6) and (8), we can see that

$$\boldsymbol{R} = \boldsymbol{R}_j + v_j \boldsymbol{h}_j \boldsymbol{h}_j^T . \tag{9}$$

C. The Connection between JG and MMSE Estimation

Now, let us again focus a particular bit x_j whose *a priori* mean and variance are $\overline{x}_j$ and v_j respectively. Its *a posteriori* mean $\overline{x}_j^{post}$ is the jth element of $\overline{\boldsymbol{x}}^{post}$, and its *a posteriori* variance is

$$v_j^{post} = E\left(\left(x_j - \overline{x}_j^{post}\right)^2\right). \tag{10}$$

The following theorem establishes a connection between the extrinsic LLR about x_j in the binary system and $(\overline{x}_j^{post}, v_j^{post})$ about x_j in the companion Gaussian system.

Theorem: The extrinsic LLR about x_j in the binary system is related to $(\overline{x}_j^{post}, v_j^{post})$ about x_j in the companion Gaussian system by

$$L_{ESE}^{JG}\left(x_j\right) = 2\left(\frac{\overline{x}_j^{post}}{v_j^{post}} - \frac{\overline{x}_j}{v_j}\right). \tag{11}$$

Proof: See Appendix. □

The theorem suggests a new approach to computing $L_{ESE}^{JG}\left(x_j\right)$, i.e., we can find $(\overline{x}_j^{post}, v_j^{post})$ about x_j in the companion Gaussian system and then obtain $L_{ESE}^{JG}\left(x_j\right)$ using (11). The advantage of this approach is that the companion Gaussian system can be solved using the factor graph technique studied in [11], as detailed below.

III. The Factor Graph Representation of an ISI Channel

A. The Factor Graph Representation of an ISI Channel

We consider the special case when (1) represents an equivalent discrete time-invariant ISI channel with $L=M+1$ tap-coefficients (i.e., the channel memory length is M). Let $\boldsymbol{h} \equiv [h_M, h_{M-1}, \ldots, h_1, h_0]^{\mathrm{T}}$ be the vector of channel coefficients, and define $\underline{\boldsymbol{x}}_j \equiv [x_{j-M}, x_{j-M+1}, \ldots, x_j]^{\mathrm{T}}$. The jth entry of $\boldsymbol{r}$ in (1) is then given by

$$\begin{aligned} r_j &= h_M x_{j-M} + h_{M-1} x_{j-M+1} + \ldots + h_1 x_{j-1} + h_0 x_j + n_j \\ &= \boldsymbol{h}^T \underline{\boldsymbol{x}}_j + n_j . \end{aligned} \tag{12}$$

Now, we define

$$\boldsymbol{G} \equiv \begin{bmatrix} \boldsymbol{0}_{M\times 1} & \boldsymbol{I}_{M\times M} \\ 0_{1\times 1} & \boldsymbol{0}^T_{1\times M} \end{bmatrix}_{L\times L} \text{ and } \boldsymbol{f} \equiv \begin{bmatrix} \boldsymbol{0}_{M\times 1} \\ 1 \end{bmatrix}_{L\times 1}, \tag{13}$$

where $\boldsymbol{I}$ is a unit matrix and $\boldsymbol{0}$ a zero matrix, and the subscripts of $\boldsymbol{I}$ and $\boldsymbol{0}$ indicate their sizes. It is easy to see that

$$\underline{\boldsymbol{x}}_j = \underline{\boldsymbol{y}}_j + \boldsymbol{f} x_j \tag{14}$$

where

$$\underline{\boldsymbol{y}}_j = \boldsymbol{G}\underline{\boldsymbol{x}}_{j-1} . \tag{15}$$

The factor graph representation of (12), (14) and (15) is shown in the dashed box in Fig. 2.We can use this as a building block to form a complete factor graph

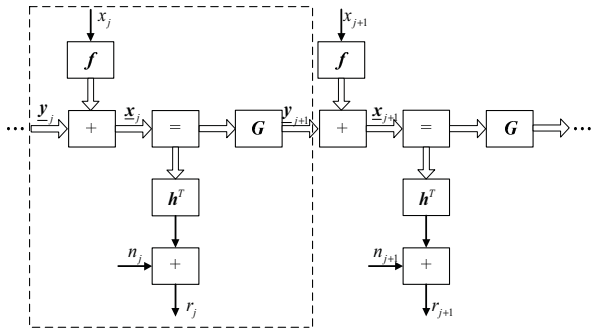

Fig. 2. The factor graph representation of (1) for an ISI channel.

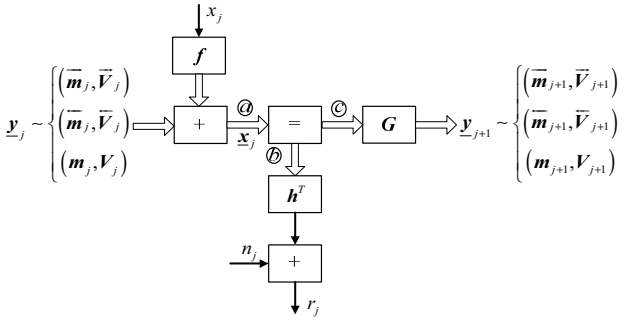

Fig. 3. Message passing for the *j*th building block.

representation of (1), as shown in Fig. 2.

B. Message Passing on a Factor Graph

For a companion Gaussian system, all the variables in Fig. 2 are Gaussian. The message passing technique developed in [11] can be applied to this graph. With this technique, we compute the so-called "messages" which are the parameters, i.e., mean vectors and covariance matrices (or their inverses), characterizing the distribution of the variables involved. The basic principle follows the discussion in [11]. In what follows, we will explain some details, since different implementation methods may result in quite different complexities.

C. Notations

We can adopt either the two-way schedule or the flooding schedule [14] to accomplish the message passing procedure. For notational convenience, the arrows are used to indicate the directions of messages. For example, we will use $\vec{\boldsymbol{m}}_j^a$ and $\vec{\boldsymbol{V}}_j^a$ to indicate the mean and covariance of the variable vector at point "*a*" in Fig. 3 for the forward process, and use $\overleftarrow{\boldsymbol{m}}_j^a$ and $\overleftarrow{\boldsymbol{V}}_j^a$ to indicate its mean and covariance for the backward process. Other messages such as $\vec{\boldsymbol{m}}_j^c$, $\vec{\boldsymbol{V}}_j^c$, $\overleftarrow{\boldsymbol{m}}_j^c$ and $\overleftarrow{\boldsymbol{V}}_j^c$, are defined similarly.

In both schedules, we will compute the out-bound messages $\{\vec{\boldsymbol{m}}_{j+1}, \vec{\boldsymbol{V}}_{j+1}, \overleftarrow{\boldsymbol{m}}_j, \overleftarrow{\boldsymbol{V}}_j\}$ for the *j*th building block (as shown in Fig. 3), based on the in-bound messages $\{\vec{\boldsymbol{m}}_j, \vec{\boldsymbol{V}}_j, \overleftarrow{\boldsymbol{m}}_{j+1}, \overleftarrow{\boldsymbol{V}}_{j+1}\}$.

D. Preparation

We first find the messages at point "*b*" in the *j*th building block shown in Fig. 3, which is part of Fig. 2. According to the update rules [11] for the sum constraint and matrix multiplication, the upward messages at point "*b*" are calculated as

$$\boldsymbol{W}_j^b = \frac{1}{\sigma^2}\boldsymbol{h}\boldsymbol{h}^T, \quad \boldsymbol{m}_j^b = \frac{r_j}{\sigma^2}\left(\boldsymbol{W}_j^b\right)^{\#}\boldsymbol{h}, \tag{16}$$

where "$\#$"denotes Moore-Penrose pseudo-inverse. These messages represent the information about $\underline{\boldsymbol{x}}_j$ after observing r_j. They will be shared by both forward and backward processes below, and so we will not use arrows to indicate their directions.

E. Forward Process

We assume that $\{\vec{\boldsymbol{m}}_j, \vec{\boldsymbol{V}}_j\}$ for $\underline{\boldsymbol{y}}_j$ (see Fig. 3) are available. They represent the information provided by the sub-graph left to the *j*th building block. In the forward process we compute $\{\vec{\boldsymbol{m}}_{j+1}, \vec{\boldsymbol{V}}_{j+1}\}$ for $\underline{\boldsymbol{y}}_{j+1}$ based on $\{\vec{\boldsymbol{m}}_j, \vec{\boldsymbol{V}}_j\}$, $\{\bar{x}_j, v_j\}$ and $\{\boldsymbol{m}_j^b, \boldsymbol{W}_j^b\}$, where v_j is the variance of x_j. From (15), $\vec{\boldsymbol{V}}_j$ and $\vec{\boldsymbol{m}}_j$ have the following structures:

$$\vec{\boldsymbol{V}}_j = \begin{bmatrix} \boldsymbol{V} & \boldsymbol{0} \\ \boldsymbol{0}^T & 0 \end{bmatrix}, \quad \vec{\boldsymbol{m}}_j = \begin{bmatrix} \boldsymbol{m} \\ 0 \end{bmatrix}, \tag{17}$$

where $\boldsymbol{V}$ is an $M\times M$ matrix, $\boldsymbol{0}$ a length-M zero column vector, and $\boldsymbol{m}$ a length-M column vector. Based on the update rule [11] for sum constraint (for the block marked by "+"), the messages at point "*a*" are computed as

$$\vec{\boldsymbol{V}}_j^a = \begin{bmatrix} \boldsymbol{V} & \boldsymbol{0} \\ \boldsymbol{0} & v_j \end{bmatrix}, \quad \vec{\boldsymbol{m}}_a^j = \begin{bmatrix} \boldsymbol{m} \\ \bar{x}_j \end{bmatrix}. \tag{18}$$

Next, according to the update rule [11] for the equality constraint (for the block marked by "="), we can obtain the messages at point "*c*" as

$$\begin{aligned} \vec{\boldsymbol{V}}_j^c &= \left((\vec{\boldsymbol{V}}_j^a)^{-1} + \boldsymbol{W}_j^b\right)^{-1} \\ &= \vec{\boldsymbol{V}}_j^a - \frac{\vec{\boldsymbol{V}}_j^a \boldsymbol{h}\boldsymbol{h}^T \vec{\boldsymbol{V}}_j^a}{\sigma^2 + \boldsymbol{h}^T \vec{\boldsymbol{V}}_j^a \boldsymbol{h}}, \end{aligned} \tag{19}$$

$$\begin{aligned} \vec{\boldsymbol{m}}_j^c &= \vec{\boldsymbol{V}}_j^c \left((\vec{\boldsymbol{V}}_j^a)^{-1}\vec{\boldsymbol{m}}_j^a + \boldsymbol{W}_j^b \boldsymbol{m}_j^b\right) \\ &= \vec{\boldsymbol{m}}_j^a + \left(\frac{r_j}{\sigma^2} - \frac{\left(\boldsymbol{h}^T \vec{\boldsymbol{m}}_j^a + (r_j/\sigma^2)\boldsymbol{h}^T \vec{\boldsymbol{V}}_j^a \boldsymbol{h}\right)}{\sigma^2 + \boldsymbol{h}^T \vec{\boldsymbol{V}}_j^a \boldsymbol{h}}\right)\vec{\boldsymbol{V}}_j^a \boldsymbol{h}. \end{aligned} \tag{20}$$

In the above derivation, we have used the matrix inversion lemma [16] and the flowing property of the Moore-Penrose pseudo-inverse

$$\boldsymbol{A}\boldsymbol{A}^T\left(\boldsymbol{A}\boldsymbol{A}^T\right)^{\#}\boldsymbol{A} = \boldsymbol{A}. \tag{21}$$

We partition $\vec{\boldsymbol{V}}_j^c$ and $\vec{\boldsymbol{m}}_j^c$ into the following forms:

$$\vec{\boldsymbol{V}}_j^c = \begin{bmatrix} \mu & \boldsymbol{u}^T \\ \boldsymbol{u} & \boldsymbol{U} \end{bmatrix}, \quad \vec{\boldsymbol{m}}_j^c = \begin{bmatrix} \delta \\ \boldsymbol{g} \end{bmatrix}, \tag{22}$$

where $\boldsymbol{U}$ is an $M\times M$ matrix, μ, δ two scalars, and $\boldsymbol{u}$, $\boldsymbol{g}$ two length-M column vectors. Then, based on the update rule [11] for matrix multiplication, we obtain

$$\vec{\boldsymbol{V}}_{j+1} = \begin{bmatrix} \boldsymbol{U} & \boldsymbol{0} \\ \boldsymbol{0}^T & 0 \end{bmatrix}, \quad \vec{\boldsymbol{m}}_{j+1} = \begin{bmatrix} \boldsymbol{g} \\ 0 \end{bmatrix}. \tag{23}$$

The computational complexity for the forward process in terms of flops is estimated as follows. (A flop involves one multiplication and one addition operation). Generating $\vec{\boldsymbol{V}}_j^a \boldsymbol{h}$ in (19) and (20) requires L^2 flops.

Computing $\vec{\boldsymbol{V}}_j^a \boldsymbol{h}\boldsymbol{h}^T \vec{\boldsymbol{V}}_j^a$ requires about $0.5L^2$ multiplications (since it is a symmetric matrix). The matrix subtraction in (19) requires about $0.5L^2$ additions. Other operations involved are all in order L, and are neglected. Hence, the forward process takes about $1.5L^2$ flops per bit. (Note that each building block in Fig. 3 corresponds to an input bit).

F. Backward Process

Next, consider computing $\{\overleftarrow{\boldsymbol{m}}_j, \overleftarrow{\boldsymbol{V}}_j\}$ based on $\{\overleftarrow{\boldsymbol{m}}_{j+1}, \overleftarrow{\boldsymbol{V}}_{j+1}\}$, $\{\bar{x}_j, v_j\}$ and $\{\boldsymbol{m}_j^b, \boldsymbol{W}_j^b\}$. Partition $\overleftarrow{\boldsymbol{V}}_{j+1}$ and $\overleftarrow{\boldsymbol{m}}_{j+1}$ into the following forms:

$$\overleftarrow{\boldsymbol{V}}_{j+1} = \begin{bmatrix} \boldsymbol{P} & \boldsymbol{p} \\ \boldsymbol{p}^T & \rho \end{bmatrix}, \quad \overleftarrow{\boldsymbol{m}}_{j+1} = \begin{bmatrix} \boldsymbol{z} \\ \tau \end{bmatrix}, \tag{24}$$

where $\boldsymbol{P}$ is an $M\times M$ matrix, ρ, τ two scalars, and $\boldsymbol{p}$, $\boldsymbol{z}$ two length-M column vectors. Define $\overleftarrow{\boldsymbol{W}}_{j+1}$ as the inverse of $\overleftarrow{\boldsymbol{V}}_{j+1}$:

$$\overleftarrow{\boldsymbol{W}}_{j+1} = \left(\overleftarrow{\boldsymbol{V}}_{j+1}\right)^{-1} = \begin{bmatrix} \boldsymbol{Q} & \boldsymbol{q} \\ \boldsymbol{q}^T & \eta \end{bmatrix}, \tag{25}$$

where $\boldsymbol{Q}$ is an $M\times M$ matrix, η a scalar, and $\boldsymbol{q}$ a length-M column vector. According to the matrix inversion lemma [16], we have

$$\boldsymbol{Q}^{-1} = \boldsymbol{P} - \frac{1}{\rho}\boldsymbol{p}\boldsymbol{p}^T, \quad \boldsymbol{q} = -\frac{1}{\rho}\boldsymbol{Q}\boldsymbol{p}. \tag{26}$$

Based on the update rule for the matrix multiplication, we can compute the backward messages at point "*c*" as

$$\overleftarrow{\boldsymbol{W}}_j^c = \begin{bmatrix} 0 & \boldsymbol{0}^T \\ \boldsymbol{0} & \boldsymbol{Q} \end{bmatrix}, \quad \overleftarrow{\boldsymbol{m}}_j^c = \left(\overleftarrow{\boldsymbol{W}}_j^c\right)^{\#} \begin{bmatrix} 0 \\ \boldsymbol{Q}\boldsymbol{z} + \tau\boldsymbol{q} \end{bmatrix}. \tag{27}$$

Then, according to the update rule for the equality constraint, we can calculate the backward messages at point "*a*" as

$$\begin{aligned}
\overleftarrow{\boldsymbol{V}}_j^a &= \left(\overleftarrow{\boldsymbol{W}}_j^c + \boldsymbol{W}_j^b\right)^{-1} \\
&= \begin{bmatrix} \frac{1}{h_0^2}\left(\sigma^2 + \boldsymbol{h}_1^T\boldsymbol{Q}^{-1}\boldsymbol{h}_1\right) & -\frac{1}{h_0}\boldsymbol{h}_1^T\boldsymbol{Q}^{-1} \\ -\frac{1}{h_0}\boldsymbol{Q}^{-1}\boldsymbol{h}_1 & \boldsymbol{Q}^{-1} \end{bmatrix} \\
&= \begin{bmatrix} \frac{1}{h_0^2}\left(\sigma^2 + \boldsymbol{h}_1^T(\boldsymbol{P} - \frac{1}{\rho}\boldsymbol{p}\boldsymbol{p}^T)\boldsymbol{h}_1\right) & -\frac{1}{h_0}\boldsymbol{h}_1^T(\boldsymbol{P} - \frac{1}{\rho}\boldsymbol{p}\boldsymbol{p}^T) \\ -\frac{1}{h_0}(\boldsymbol{P} - \frac{1}{\rho}\boldsymbol{p}\boldsymbol{p}^T)\boldsymbol{h}_1 & \boldsymbol{P} - \frac{1}{\rho}\boldsymbol{p}\boldsymbol{p}^T \end{bmatrix},
\end{aligned} \tag{28}$$

$$\begin{aligned}
\overleftarrow{\boldsymbol{m}}_j^a &= \overleftarrow{\boldsymbol{V}}_j^a\left(\boldsymbol{W}_j^b\boldsymbol{m}_j^b + \overleftarrow{\boldsymbol{W}}_j^c\overleftarrow{\boldsymbol{m}}_j^c\right) \\
&= \overleftarrow{\boldsymbol{V}}_j^a\left(\frac{r_j}{\sigma^2}\boldsymbol{h} + \begin{bmatrix} 0 \\ \boldsymbol{Q}\boldsymbol{z} + \tau\boldsymbol{q} \end{bmatrix}\right) \\
&= \begin{bmatrix} \frac{1}{h_0}\left(r_j - \boldsymbol{h}_1^T(\boldsymbol{z} - \frac{\tau}{\rho}\boldsymbol{p})\right) \\ \boldsymbol{z} - \frac{\tau}{\rho}\boldsymbol{p} \end{bmatrix},
\end{aligned} \tag{29}$$

where $\boldsymbol{h} = [h_0 \ \boldsymbol{h}_1^T]^T$. In the derivation of (28) and (29), eqn. (26) is used. Finally, according to the update rule of the sum constraint, we have

$$\overleftarrow{\boldsymbol{V}}_j = \overleftarrow{\boldsymbol{V}}_j^a + \begin{bmatrix} 0 & & \\ & \ddots & \\ & & v_j \end{bmatrix}, \quad \overleftarrow{\boldsymbol{m}}_j = \overleftarrow{\boldsymbol{m}}_j^a - \begin{bmatrix} 0 \\ \vdots \\ \bar{x}_j \end{bmatrix}. \tag{30}$$

For complexity analysis, the dominant part of the backward process is to generate $\boldsymbol{p}\boldsymbol{p}^T$ and $\boldsymbol{h}_1^T\left(\boldsymbol{P} - (1/\rho)\boldsymbol{p}\boldsymbol{p}^T\right)$ in (28), costing about $1.5L^2$ flops per bit. (Note that $\boldsymbol{p}\boldsymbol{p}^T$ is symmetric.)

G. Output Process

The output messages are obtained by combining the forward and backward messages. For example, at point "*a*" shown in Fig. 3, the forward and backward messages are $\{\vec{\boldsymbol{V}}_j^a, \vec{\boldsymbol{m}}_j^a\}$ and $\{\overleftarrow{\boldsymbol{V}}_j^a, \overleftarrow{\boldsymbol{m}}_j^a\}$ respectively. The output messages can be calculated as

$$\begin{aligned}
\boldsymbol{V}_j^{out} &= \left((\vec{\boldsymbol{V}}_j^a)^{-1} + (\overleftarrow{\boldsymbol{V}}_j^a)^{-1}\right)^{-1}, \\
\boldsymbol{m}_j^{out} &= \boldsymbol{V}_j^{out}\left((\vec{\boldsymbol{V}}_j^a)^{-1}\vec{\boldsymbol{m}}_j^a + (\overleftarrow{\boldsymbol{V}}_j^a)^{-1}\overleftarrow{\boldsymbol{m}}_j^a\right).
\end{aligned} \tag{31}$$

The diagonal elements of $\boldsymbol{V}_j^{out}$ and elements of $\boldsymbol{m}_j^{out}$ are the *a posteriori* variances and means of x_{j-M}, …, x_j respectively. Note that $\vec{\boldsymbol{V}}_j^a$, $\overleftarrow{\boldsymbol{V}}_j^a$ are all symmetric, so computing (31) requires about $2.5L^3$ flops. (Inverting a symmetric matrix with size $L\times L$ takes about $(5/6)L^3$ flops). We only need to compute the output messages once every L building blocks. Therefore, the output process requires about $2.5L^2$ flops per bit.

H. Overall Complexity

From the discussion above, the overall message passing algorithm takes about $5.5L^2$ flops per bit. As a comparison, the extending window approach proposed in [8] for evaluating (5) requires about $L\times N_1$ flops, where N_1 is the non-causal filter length. Normally, to ensure good performance, it requires that $N_1 \geq 2L$. The sliding window approach proposed in [6] has a much higher complexity. Thus the complexity of the proposed factor graph approach is comparable to other alternatives. However, as discussed below, the proposed method has a noticeable advantage for parallel processing. (The existing methods [6]-[8] are serial by nature.)

I. The Serial and Parallel Schedules

The factor graph in Fig. 2 contains no loop and has a tree structure. We can thus apply either the two-way schedule or the flooding schedule [14] in the message passing algorithm.

With the two-way schedule, the building blocks in the factor graph compute the messages according to a sequential order. When a serial processor is used, the two-way schedule is an optimal method for a tree-structured factor graph, since it can find the optimal solution using a minimum number of operations. However, due to its serial nature, the two-way schedule may not be efficient if multiple processors are available.

With the flooding schedule, every building block in the factor graph computes the messages simultaneously and passes them to its neighbors. This is most suitable for parallel implementation. We observed that a full flooding approach has a poor convergence property. Alternatively, we can adopt a partially parallel approach, in which the overall graph is partitioned into a number of sub-graphs,

each containing several building blocks shown in Fig. 3. All the processors operate in parallel, and each processor processes the building blocks in a sub-graph. (Within a sub-graph the building blocks are processed in a serial manner.) In this way, we can achieve a good compromise between parallelism and convergence speed.

IV. Numerical Results

Consider a system employing a rate-1/2 nonsystematic convolution code with generators $(23, 35)_8$. The information block length is 16384. The encoded bit stream is scrambled by a random generated interleaver before it is transmitted in BPSK format. A severely distorted 5-tap channel (taken from [15]) with coefficients [0.227 0.460 0.688 0.460 0.227] (L = 5) is used and assumed to be known at the receiver. In the (partially) parallel schedule, each sub-graph consists of 5 building blocks.

Fig. 4 compares the performance of the factor graph methods based on the two-way and parallel schedules. The two-way schedule has faster convergence speed in the first few iterations, but the difference becomes marginal after 20 iterations. If parallel processors are available, the parallel schedule may be preferred since it can take the advantage of multiple processors and reduce the execution time of each iteration.

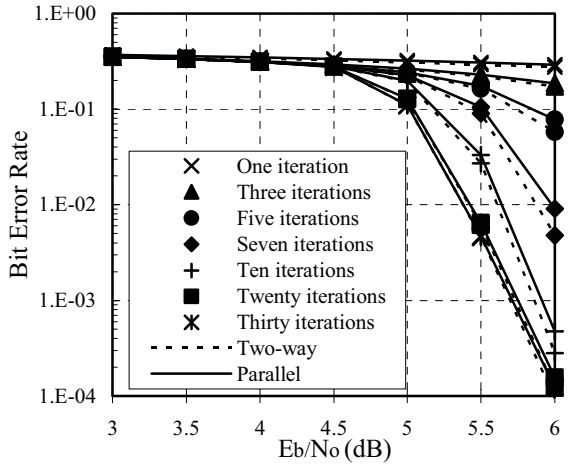

Fig. 4. Performance comparison between the parallel and two-way schedules.

V. Conclusions

We have proposed a factor graph approach to turbo equalization. The proposed is a full-window approach with low complexity. It provides an option for parallel implementation. This feature is particularly useful for practical implementation, since all the existing algorithms, such as those discussed in [6]-[8], are serial by nature.

Appendix

From (1), the covariance of $\boldsymbol{r}$ and $\boldsymbol{x}$ is

$$Cov(\boldsymbol{x},\boldsymbol{r}) = \boldsymbol{V}\boldsymbol{H}^{\boldsymbol{T}}. \tag{32}$$

Substituting (32) into (7) and using matrix inversion lemma, we can write the *a posteriori* mean of x_j as (the jth row of (7))

$$\begin{aligned}
\bar{x}_j^{post} &= E\left(x_j \mid \boldsymbol{r}\right) \\
&= \bar{x}_j + v_j \boldsymbol{h}_j^T \boldsymbol{R}^{-1}\left(\boldsymbol{r} - \boldsymbol{H}\bar{\boldsymbol{x}}\right) \\
&= \bar{x}_j + v_j \boldsymbol{h}_j^T \left(\boldsymbol{R}_j + v_j \boldsymbol{h}_j \boldsymbol{h}_j^T\right)^{-1}\left(\boldsymbol{r} - \boldsymbol{H}\bar{\boldsymbol{x}}\right) \\
&= \bar{x}_j + v_j \boldsymbol{h}_j^T \left(\boldsymbol{R}_j^{-1} - \frac{\boldsymbol{R}_j^{-1}\boldsymbol{h}_j \boldsymbol{h}_j^T \boldsymbol{R}_j^{-1}}{\boldsymbol{h}_j^T \boldsymbol{R}_j^{-1}\boldsymbol{h}_j + 1/v_j}\right)\left(\boldsymbol{r} - \boldsymbol{H}\bar{\boldsymbol{x}}\right) \\
&= \frac{\bar{x}_j / v_j + \boldsymbol{h}_j^T \boldsymbol{R}_j^{-1}\left(\boldsymbol{r} - \boldsymbol{H}\bar{\boldsymbol{x}} + \boldsymbol{h}_j \bar{x}_j\right)}{1/v_j + \boldsymbol{h}_j^T \boldsymbol{R}_j^{-1}\boldsymbol{h}_j}.
\end{aligned} \tag{33}$$

The *a posteriori* variance of x_j is given by [13]

$$\begin{aligned}
v_j^{post} &= E\left(\left(x_j - \bar{x}_j^{post}\right)^2\right) \\
&= v_j - \boldsymbol{Cov}(x_j, \boldsymbol{r})\boldsymbol{R}^{-1}\boldsymbol{Cov}(\boldsymbol{r}, x_j) \\
&= v_j - v_j^2 \boldsymbol{h}_j^T \boldsymbol{R}^{-1}\boldsymbol{h}_j \\
&= v_j - v_j^2 \boldsymbol{h}_j^T \left(\boldsymbol{R}_j^{-1} - \frac{\boldsymbol{R}_j^{-1}\boldsymbol{h}_j \boldsymbol{h}_j^T \boldsymbol{R}_j^{-1}}{\boldsymbol{h}_j^T \boldsymbol{R}_j^{-1}\boldsymbol{h}_j + 1/v_j}\right)\boldsymbol{h}_j \\
&= \frac{1}{1/v_j + \boldsymbol{h}_j^T \boldsymbol{R}_j^{-1}\boldsymbol{h}_j}.
\end{aligned} \tag{34}$$

The theorem follows from (5), (33) and (34). □

Acknowledgements

This work was fully supported by a grant from the Research Grant Council of the Hong Kong SAR, China [Project No. CityU 1314/04E]. The authors wish to thank Mr. Lihai Liu and Mr. Xiaojun Yuan for their invaluable help.

References

[1] C. Douillard, M. Jezequel, C. Berrou, A. Picart, P. Didier, and A. Glarieux, "Iterative correction of intersymbol interference: Turbo-equalization," *Eur. Trans. Telecommun.*, vol. 6, pp. 507-511, Sept./Oct. 1995.

[2] G. Bauch and V. Franz, "A comparison of soft-in/soft-out algorithms for 'Turbo detection'," in *Proc. Int. Conf. Telecomm.*, June 1998, pp. 259-263.

[3] P. Magniez, P. Duhamel, A. Roumy, and I. Fijalkow, "Turbo-equalization applied to trellis-coded-modulation," in *Proc. IEEE VTC99-Fall*, Amsterdam, The Netherlands, Sept. 1999, pp. 2556-2560.

[4] A. Glavieux, C. Laot, and J. Labat, "Turbo equalization over a frequency selective channel," in *Proc. Int. Symp. Turbo codes*, Brest, France, Sept 1997, pp. 96-102.

[5] D. Reynolds and X. Wang, "Low complexity turbo-equalization for diversity channels," *Signal Processing*, vol. 81, pp. 989-995, May 2001.

[6] M. Tüchler, R. Kotter, and A. Singer, "Turbo equalization: principles and new results," *IEEE Trans. Commun.*, vol. 50, pp. 754-767, May 2002.

[7] M. Tüchler, A. Singer, and R. Kotter, "Minimum mean squared error equalization using *a priori* information," *IEEE Trans. Signal Processing,* vol. 50, pp. 673-683, Mar. 2002.

[8] Lihai Liu and Li Ping, "An extending window MMSE turbo equalization algorithm," *IEEE Signal Processing Lett.*, vol. 11, pp. 891-894, Nov. 2004.

[9] Lihai Liu, W. K. Leung and Li Ping, "Simple chip-by-chip multiuser detection for CDMA systems," *IEEE Vehicular Tech. Conf., VTC'03,* pp. 2157-2161.

[10] G. D. Forney, Jr., "Codes on graph: Normal realizations," *IEEE Trans. Inform. Theory*, vol. 47, no.2, pp.520-548, Feb. 2001.

[11] H, -A. Loeliger, "An introduction to factor graphs," *IEEE Signal Processing Mag.*, pp. 28-41, Jan. 2004.

[12] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo codes," *Proc. 1993 Int. Conf. Comm.*, pp. 1064-1070.

[13] Steven M. Kay, *Fundamentals of statistical signal processing*, Prentice-Hall PTR, 1993.

[14] F. R. Kschischang, and B. J. Frey, "Iterative decoding of compound codes by probability propagation in graphical models," *IEEE J. Select. Areas Commun.*, pp. 219-230, Feb. 1998.

[15] J. G. Proakis, *Digitial communications*, 3rd ed. New York: MCGraw-Hill, 1995.

[16] R. A. Horn and C. R. Johnson, *Matrix Analysis*, Cambridge University Press, 1990.