Binary Intersymbol Interference Channels: Gallager Codes, Density Evolution, and Code Performance Bounds

Aleksandar Kavčić, *Member, IEEE*, Xiao Ma, and Michael Mitzenmacher, *Member, IEEE*

***Abstract*—We study the limits of performance of Gallager codes (low-density parity-check (LDPC) codes) over binary linear intersymbol interference (ISI) channels with additive white Gaussian noise (AWGN). Using the graph representations of the channel, the code, and the sum–product message-passing detector/decoder, we prove two error concentration theorems. Our proofs expand on previous work by handling complications introduced by the channel memory. We circumvent these problems by considering not just linear Gallager codes but also their cosets and by distinguishing between different types of message flow neighborhoods depending on the actual transmitted symbols. We compute the noise tolerance threshold using a suitably developed density evolution algorithm and verify, by simulation, that the thresholds represent accurate predictions of the performance of the iterative sum–product algorithm for finite (but large) block lengths. We also demonstrate that for high rates, the thresholds are very close to the theoretical limit of performance for Gallager codes over ISI channels. If C denotes the capacity of a binary ISI channel and if $\mathrm{C}_{\mathrm{i.i.d.}}$ denotes the maximal achievable mutual information rate when the channel inputs are independent and identically distributed (i.i.d.) binary random variables ($\mathrm{C}_{\mathrm{i.i.d.}} \leq \mathrm{C}$), we prove that the maximum information rate achievable by the sum–product decoder of a Gallager (coset) code is upper-bounded by $\mathrm{C}_{\mathrm{i.i.d.}}$. The last topic investigated is the performance limit of the decoder if the trellis portion of the sum–product algorithm is executed only once; this demonstrates the potential for trading off the computational requirements and the performance of the decoder.**

***Index Terms*—Bahl–Cocke–Jelinek–Raviv (BCJR)-once bound, channel capacity, density evolution, Gallager codes, independent and identically distributed (i.i.d.) capacity, intersymbol interference (ISI) channel, low-density parity-check (LDPC) codes, sum–product algorithm, turbo equalization.**

I. Introduction

If continuous channel inputs are allowed, the capacity of discrete-time intersymbol interference (ISI) channels with additive white Gaussian noise (AWGN) can be computed using the water-filling theorem [1], [2]. In many applications, the physics of the channel do not allow continuous input alphabets. A prime example of a two-level (binary) ISI channel is the saturation magnetic recording channel, because the magnetization domains can have only two stable phases [3]. Other examples include digital communication channels where the input alphabet is confined to a finite set [4].

The computation of the capacity of discrete-time ISI channels with a finite number of allowed signaling levels is an open problem. In the past, the strategy has been to obtain numeric [5] and analytic [6], [7] bounds on the capacity. Very often authors have concentrated on obtaining bounds on the achievable information rate when the inputs are independent and uniformly distributed (i.u.d.)—the so-called symmetric information rate [5]–[7]. Recently, a Monte Carlo method for numerically evaluating the symmetric information rate using the forward recursion of the Bahl–Cocke–Jelinek–Raviv (BCJR) algorithm [8] (also known as the Baum–Welch algorithm, the sum–product algorithm, or the forward–backward algorithm) has been proposed by Arnold and Loeliger [9], and independently by Pfister, Soriaga, and Siegel [10]. The same procedure can be used to numerically evaluate the i.i.d. capacity, which is defined as the maximal achievable information rate when the inputs are independent and identically distributed. This marks the first (arbitrarily close in the probability-1 sense) approximation to the exact result involving the channel capacity of a discrete-time ISI channel with binary inputs. Also, recently, tight lower [11] and upper [12], [13] bounds have been computed using Monte Carlo methods for Markov channel inputs. The remaining issue is to devise codes that will achieve the capacity (or at least the i.i.d. capacity).

The ability to achieve (near) channel capacity has recently been numerically demonstrated for various memoryless [14], [15] channels using Gallager codes, also known as low-density parity-check (LDPC) codes [16]. The theory of Gallager codes has vastly benefitted from the notion of codes on graphs first introduced by Tanner [17] and further expanded into a unifying theory of codes on graphs by Wiberg *et al.* [18] and Forney [19]. MacKay and Neal [20], [21] showed that there exist good Gallager codes with performances about 0.5 dB worse than turbo codes [22]. A major breakthrough was the construction of *irregular* Gallager codes [23], and the development of a method to analyze them for erasure channels [14], [24]. These methods were adapted to memoryless channels with continuous output alphabets (e.g., AWGN channels, Laplace channels, etc.) by Richardson and Urbanke [25], who also coined the term "den-

Manuscript received February 26, 2001; revised February 21, 2003. This work was supported by the National Science Foundation under Grants CCR-9904458 and CCR-0118701, and by the National Storage Industry Consortium. The material in this paper was presented in part at the IEEE International Symposium on Information Theory, Washington, DC, June 2001.

A. Kavčić and M. Mitzenmacher are with the Division of Engineering and Applied Sciences, Harvard University, Cambridge, MA 02138 USA (e-mail: kavcic@deas.harvard.edu; michaelm@eecs.harvard.edu).

X. Ma was with the Division of Engineering and Applied Sciences, Harvard University, Cambridge, MA USA. He is now with the Department of Electrical Engineering, City University of Hong Kong, Kowloon, Hong Kong (e-mail: xma@ee.cityu.hk).

Communicated by R. Urbanke, Associate Editor for Coding Techniques.

Digital Object Identifier 10.1109/TIT.2003.813563

sity evolution" for a tool to analyze the asymptotic performance of Gallager and turbo codes over these channels [26]. The usefulness of the tool was demonstrated by using it to optimize codes whose performance is proven to get very close to the capacity, culminating in a remarkable 0.0045-dB distance from the capacity of the memoryless AWGN channel reported by Chung *et al.* [27].

In this paper, we focus on developing the density evolution method for channels with binary inputs and ISI memory. The computed thresholds are used for lower-bounding the capacity, as well as for upper-bounding the average code performance. The main topics of this paper are: 1) concentration theorems for Gallager codes and the sum–product message-passing decoder over binary ISI channels; 2) a density evolution method for computing the thresholds of "zero-error" performance over these channels; 3) theorems establishing that the asymptotic performance of Gallager codes using the sum–product algorithm is upper-bounded by the symmetric information rate and the i.i.d. capacity; and 4) the computation of the BCJR-once bound, which is the limit of "zero-error" performance of the sum–product algorithms if the trellis portion of the algorithm is executed only once.

The paper is organized as follows. In Section II, we describe the channel model, introduce the various capacity and information rate definitions, and briefly describe the sum–product decoder [28]. In Section III, we introduce the necessary notation for handling the analysis of Gallager codes for channels with memory and prove two key concentration theorems. Section IV is devoted to describing the density evolution algorithm for channels with ISI memory. In Section V, computed thresholds are shown for regular Gallager codes. Section V also presents a theorem regarding the limit of achievable code rates using binary linear codes. In this section, we also develop the notion of the BCJR-once bound, which has a practical implication; namely, it is the limit of performance of the sum–product algorithm if the trellis portion of the algorithm is executed only once. This provides a concrete example of how we can trade off the computational load (by doing the expensive BCJR step only once) with the decoding performance. Section VI concludes the paper.

Basic Notation: Matrices are denoted by boldface upper case letters (e.g., $\boldsymbol{H}$). Column vectors are denoted by underlined characters, e.g., $\underline{a}$. Random variables (vectors) are typically denoted by upper case characters, while their realizations are denoted by lower case characters (e.g., a random vector $\underline{W}$ has a realization $\underline{w}$). The superscript T denotes matrix and vector transposition. If a column vector is $\underline{s} = [s_1, s_2, \ldots, s_n]^{\mathrm{T}}$, then a subvector collecting entries s_i, s_{i+1}, $\ldots$, s_j is denoted by $\underline{s}_i^j = [s_i, s_{i+1}, \ldots, s_j]^{\mathrm{T}}$. The notation $\Pr(\mathrm{event}_1)$ denotes the probability of event_1, while $\Pr(\mathrm{event}_1|\mathrm{event}_2)$ denotes the probability of event_1 given that event_2 occurred. The probability mass functions of discrete random variables will be denoted with the symbol "Pr," e.g., the probability mass function of a discrete random vector $\underline{X}$ evaluated at $\underline{x}$ will be denoted by $\Pr(\underline{X} = \underline{x})$, i.e., it is the probability that $\underline{X}$ takes the value $\underline{x}$. The probability density function (pdf) of a continuous random variable will be denoted by the symbol "f." For example, the pdf of a continuous random vector $\underline{Z}$ evaluated at the point $\underline{z}$ will be denoted by $f_{\underline{Z}}(\underline{z})$.

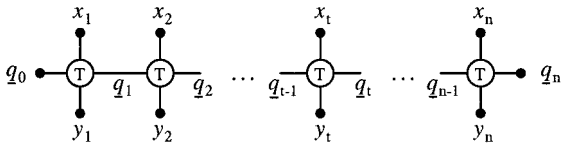

Fig. 1. Factor graph representation of the ISI channel.

II. The Channel, Gallager Codes and Decoding

A. Channel Model, Graph Representation and Capacity

Assume that we have a binary discrete-time ISI channel of finite length I, characterized by the channel response polynomial $h(D) = h_0 + h_1 D + \cdots + h_I D^I$, where $h_i \in \mathbb{R}$. The input x_t to the discrete-time channel at time $t \in \mathbb{Z}$ is a realization of a random variable X_t drawn from a binary alphabet $\mathcal{X} = \{-1, 1\}$. The output of the channel y_t is a realization of a random variable Y_t drawn from the alphabet $\mathcal{Y} = \mathbb{R}$. The channel's probabilistic law is captured by the equation

$$Y_t = \sum_{i=0}^{I} h_i X_{t-i} + N_t \tag{1}$$

where N_t is a zero-mean AWGN sequence with variance $\mathrm{E}[N_t^2] = \sigma^2$ whose realizations are $n_t \in \mathbb{R}$.

The channel in (1) is conveniently represented by a trellis [29], or, equivalently, by a graph where for each variable X_t there is a single *trellis node* [18], [19]. Define the state at time t as the vector that collects the input variables X_{t-I+1} through X_t, i.e., $\underline{Q}_t = \underline{X}_{t-I+1}^{t}$. The realization $\underline{q}_t$ of the random vector $\underline{Q}_t$ can take one of 2^I values. With this notation, we can factor the function

$$\Pr\left(\underline{X}_1^n = \underline{x}_1^n | \underline{Y}_1^n = \underline{y}_1^n, \underline{Q}_0 = \underline{q}_0\right) f_{\underline{Y}_1^n|\underline{Q}_0}\left(\underline{y}_1^n | \underline{Q}_0 = \underline{q}_0\right) = \prod_{t=1}^{n} \mathcal{F}\left(x_t, y_t, \underline{q}_{t-1}, \underline{q}_t\right) \tag{2}$$

where each factor is

$$\mathcal{F}\left(x_t, y_t, \underline{q}_{t-1}, \underline{q}_t\right) = f_{Y_t|X_t, \underline{Q}_{t-1}}\left(y_t | x_t, \underline{q}_{t-1}\right) \Pr\left(\underline{Q}_t = \underline{q}_t | \underline{Q}_{t-1} = \underline{q}_{t-1}\right). \tag{3}$$

This factorization is represented by the factor graph in Fig. 1. Each node of the graph (denoted by the letter "T") represents a factor (3), while each edge connected to the node represents a variable on which the factor depends. Edges terminated by a small filled circle (•) are half edges. Half edges may be considered terminals to which other graphs may be connected. For details on factor-graph representations, see [19], [28].

For the channel in (1), the capacity is defined as

$$\mathrm{C} = \lim_{n\to\infty} \frac{1}{n} \sup_{\Pr(\underline{X}_1^n = \underline{x}_1^n)} \mathrm{I}\left(\underline{X}_1^n; \underline{Y}_1^n\right) \tag{4}$$

where $\mathrm{I}(\underline{X}_1^n; \underline{Y}_1^n)$ is the mutual information[1] between the channel input and the output evaluated for a specific probability

[1]Some authors refer to $\mathrm{I}(\underline{X}_1^n; \underline{Y}_1^n)$ as the *average* mutual information (AMI), see, e.g., [1], [6], [7].

mass function $\Pr(\underline{X}_1^n = \underline{x}_1^n)$ of the channel input, where $\underline{x}_1^n \in \mathcal{X}^n$. Another quantity related to the mutual information is the maximum i.i.d. mutual information rate (the i.i.d. capacity), defined as

$$\mathrm{C_{i.i.d.}} = \lim_{n\to\infty} \frac{1}{n} \sup_{\Pr(\underline{X}_1^n = \underline{x}_1^n) = \prod_{t=1}^n \Pr(X = x_t)} \mathrm{I}(\underline{X}_1^n; \underline{Y}_1^n) \tag{5}$$

where the supremum is taken over all probability mass functions of i.i.d. random variables X_t, $1 \le t \le n$. Clearly, $\mathrm{C_{i.i.d.}} \le \mathrm{C}$.

We shall also use the symmetric information rate

$$\mathcal{I}_{\mathrm{i.u.d.}} = \lim_{n\to\infty} \left[\frac{1}{n} \mathrm{I}(\underline{X}_1^n; \underline{Y}_1^n) \Big|_{\Pr(\underline{X}_1^n = \underline{x}_1^n) = 2^{-n}} \right] \tag{6}$$

which is the information rate obtained when the input sequence is Bernoulli-$1/2$, i.e., when the inputs are i.u.d.

Conjecture 1: For the binary ISI channel modeled by (1), $\mathrm{C_{i.i.d.}} = \mathcal{I}_{\mathrm{i.u.d.}}$ holds.

Neither the capacity C nor the i.i.d. capacity $\mathrm{C_{i.i.d.}}$ are known in closed form. Only if the channel coefficients are $h_i = 0$ for $i \ge 1$ (i.e., if the channel does not have memory) do we have $\mathrm{C} = \mathrm{C_{i.i.d.}}$, in which case the capacity is known and can be evaluated via numerical integration [1], [6]. For channels with ISI memory, $\mathrm{C_{i.i.d.}}$ can be very accurately numerically evaluated (with probability 1) using the Arnold–Loeliger method [9]. These numerical evaluations also confirm (though they do not prove) that $\mathrm{C_{i.i.d.}} = \mathcal{I}_{\mathrm{i.u.d.}}$ for binary ISI channels.

B. Gallager Coset Codes

A Gallager code (also known as an LDPC code) is a linear block code whose parity-check matrix is sparse [16]. Here, we will extend this definition to include any coset of a linear block code with a sparse parity-check matrix. An information block is denoted by a $k \times 1$ vector $\underline{m} \in \{0, 1\}^k$. If a sparse $(n - k) \times n$ binary parity-check matrix is denoted by $\boldsymbol{H}$, then $\boldsymbol{G}(\boldsymbol{H})$ denotes the $n \times k$ generator matrix corresponding to $\boldsymbol{H}$ (with the property $\boldsymbol{H} \cdot \boldsymbol{G}(\boldsymbol{H}) = \boldsymbol{0}$). A Gallager coset code is specified by a parity-check matrix $\boldsymbol{H}$ and an $n \times 1$ coset-defining vector $\underline{r}$. The codeword is an $n \times 1$ vector

$$\underline{s} = [s_1, s_2, \ldots, s_n]^{\mathrm{T}} = [\boldsymbol{G}(\boldsymbol{H}) \cdot \underline{m}] \oplus \underline{r} \tag{7}$$

where $s_t \in \{0, 1\}$, and $\oplus$ denotes binary vector addition. The codeword $\underline{s}$ satisfies

$$\boldsymbol{H} \cdot \underline{s} = \underline{c} = [c_1, c_2, \ldots, c_{n-k}]^{\mathrm{T}} = \boldsymbol{H} \cdot \underline{r}. \tag{8}$$

The code is *linear* if and only if $\underline{c} = \underline{0}$; otherwise, the code is a *coset code* of a linear Gallager code.

It is convenient to represent a Gallager coset code by a bipartite graph [17], [19], [28]. The graph has two types of nodes: n variable nodes (one variable node for each entry in the vector $\underline{s}$) and $n - k$ check nodes (one check node for each entry in the vector $\underline{c}$). There is an edge connecting the ith check node and the jth variable node if the entry $\boldsymbol{H}(i, j)$ in the ith row and jth column of $\boldsymbol{H}$ is nonzero. Thus, each check node represents a parity-check equation $c_i = \bigoplus_{j:\boldsymbol{H}(i,j)\neq 0} s_j$, where the symbol $\bigoplus$ denotes binary addition. An example of a graph of a Gallager coset code is depicted in Fig. 2.

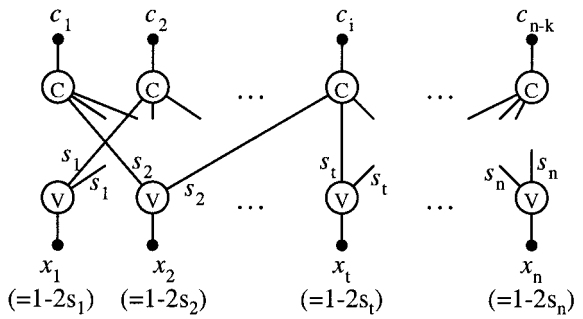

Fig. 2. Bipartite graph representation of a regular Gallager coset code $(L_{\max}, R_{\max}) = (2, 3)$.

The degree of a node is the number of edges connected to it. Two degree polynomials

$$\lambda(x) = \sum_{i=1}^{L_{\max}} \lambda_i x^{i-1} \quad \text{and} \quad \rho(x) = \sum_{i=1}^{R_{\max}} \rho_i x^{i-1}$$

are defined [23], where $L_{\max}$ and $R_{\max}$ are the maximal variable- and check-node degrees, respectively. If n_e represents the total number of edges in the graph, then the value λ_i represents the fraction of the n_e edges that are connected to variable nodes of degree i. Similarly, ρ_i represents the fraction of the n_e edges that are connected to check nodes of degree i. Clearly

$$\sum_{i=1}^{L_{\max}} \lambda_i = \sum_{i=1}^{R_{\max}} \rho_i = 1.$$

The design code rate[2] is

$$r = \frac{k}{n} = 1 - \frac{\int_0^1 \rho(x)\,\mathrm{d}x}{\int_0^1 \lambda(x)\,\mathrm{d}x}.$$

A *regular* Gallager coset code is a code for which $\lambda_{L_{\max}} = 1$ and $\rho_{R_{\max}} = 1$. The graph in Fig. 2 represents a regular Gallager coset code for which $(L_{\max}, R_{\max}) = (2, 3)$.

We define the ensemble $\mathcal{C}_n(\lambda(x), \rho(x))$ of Gallager coset codes as the set of all block codes that satisfy (7) and (8), whose codewords $\underline{s}$ are of dimension $n \times 1$, whose graph corresponding to the parity-check matrix $\boldsymbol{H}$ has variable and check degree polynomials $\lambda(x)$ and $\rho(x)$, respectively, and whose binary coset vector $\underline{r}$ can take any of 2^n values.

Before transmission over the channel (1), the variables $s_t \in \{0, 1\}$ are converted to variables $x_t \in \{-1, 1\}$ as

$$x_t = 1 - 2s_t. \tag{9}$$

Since there is a one-to-one correspondence between the vectors $\underline{x}$ and $\underline{s}$, the term codeword will be used interchangeably to describe either of the two vectors.

C. Sum–Product Decoding by Message Passing

In the literature, several methods exist for soft detection of symbols transmitted over ISI channels [8], [30]–[34]. There also exist several message-passing algorithms that decode codes on graphs [16], [17], [23], [25]. Here, we will adopt the algorithm

[2]The true code rate of a code defined by a graph will always be greater than or equal to the design code rate. In practice, they are often extremely close, so we do not distinguish between them throughout the paper.

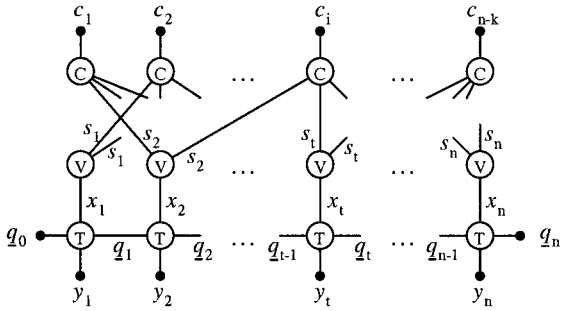

Fig. 3. Joint code/channel graph.

referred to in the coding literature as the "sum–product" algorithm [18], [28], but is also known as belief propagation [35], [36]. When applied specifically to ISI channels, the algorithm also takes the name "turbo equalization" [37]. For convenience in the later sections, we describe here the "windowed" version of the algorithm.

First, we join the channel factor graph (Fig. 1) with the code graph (Fig. 2) to get the joint channel/code graph depicted in Fig. 3. The exact schedule of the message-passing algorithm seems to have only very little effect on the convergence value but may affect the convergence speed. However, to do the analysis in Section III, we must adopt a message-passing schedule because the schedule affects the structure of the *message-flow neighborhood* defined in Section III. Here, we describe the scheduling choice presented in [38] often referred to as *turbo equalization* [37] due to the resemblance to turbo decoding [22].

Trellis-to-Variable Messages: Assume that the received vector is $\underline{y}_1^n$. In the ℓth round of the algorithm, we compute the trellis output messages $\mathcal{O}_t^{(\ell)}$, where the messages $e_t^{(\ell)}$ (these are available from the previous round of the message-passing decoding algorithm on the code subgraph of the joint channel/code graph) are considered as the extrinsic information (in the initial round $e_t^{(0)} = 0$). The output message is computed by running the "windowed" version of the BCJR algorithm. The windowed BCJR algorithm for computing the message $\mathcal{O}_t^{(\ell)}$ starts W trellis stages to the left and to the right of the tth trellis node. The forward-going and backward-going message vectors are started as $\underline{\alpha}_{W,t}^{(\ell)} = \underline{\beta}_{W,t}^{(\ell)} = 2^{-I}\underline{1}$, where $\underline{1}$ is an all-ones vector of size $2^I \times 1$. The computation of the message $\mathcal{O}_t^{(\ell)}$ follows the BCJR algorithm described in [8]; schematically depicted in Fig. 4. In the Appendix, this algorithm is reproduced for completeness.

Variable-to-Check Messages: Once the messages $\mathcal{O}_t^{(\ell)}$ are computed, we compute the messages going from the variable nodes to the check nodes. A detailed explanation of this computation can be found in [25], [27]. Here, we just state the result. Let the tth variable node be of degree L, i.e., it is connected to L check nodes. In the ℓth round, let $\mathcal{O}_t^{(\ell)}$ be the message arriving from the trellis node and let $u_m^{(\ell)}$ (where $1 \le m \le L$) denote the messages arriving from the check nodes (in the initial round, $u_m^{(0)} = 0$). The rule for computing the message $v_m^{(\ell+1)}$ is

$$v_m^{(\ell+1)} = \mathcal{O}_t^{(\ell)} + \sum_{\substack{j=1 \\ j \neq m}}^{L} u_j^{(\ell)} \tag{10}$$

and is depicted in Fig. 5.

Check-to-Variable Messages: The next step is to compute the messages going from the check nodes back to the variable nodes. Let the variable node be of degree R, i.e., it is connected to R variable nodes, and let it represent a parity-check equation for which $c_i \in \{0, 1\}$. In round ℓ, let $v_m^{(\ell+1)}$ (where $1 \le m \le R$) denote the messages arriving from the variable nodes to the check nodes. The rule for computing the message $u_m^{(\ell+1)}$ is

$$\tanh \frac{u_m^{(\ell+1)}}{2} = (-1)^{c_i} \prod_{\substack{k=1 \\ k \neq m}}^{R} \tanh \frac{v_k^{(\ell+1)}}{2} \tag{11}$$

and is depicted in Fig. 6. Here

$$\tanh(a) = (\mathrm{e}^{a} - \mathrm{e}^{-a})/(\mathrm{e}^{a} + \mathrm{e}^{-a}).$$

Variable-to-Trellis Messages: The last step required to complete a round of the message-passing sum–product agorithm is to compute the messages $e_t^{(\ell+1)}$ passed from the variable nodes to the trellis nodes. The rule for computing the message $e_t^{(\ell+1)}$ is

$$e_t^{(\ell+1)} = \sum_{j=1}^{L} u_j^{(\ell+1)} \tag{12}$$

and is depicted in Fig. 7.

The Full Message-Passing Algorithm: The algorithm is executed iteratively, where the stopping criterion can be chosen in a number of different ways [39]. Here we assume the simplest stopping criterion, i.e., conduct the iterations for exactly $\ell_{\max} \ge 1$ rounds. In short, the algorithm has the following form

- **Initialization**

1) receive channel outputs $y_1, y_2, \ldots, y_t, \ldots, y_n$;
2) for $1 \le t \le n$, set $e_t^{(0)} = 0$;
3) set all check-to-variable messages $u_m^{(0)} = 0$;
4) set $\ell = 0$.

- **Repeat while** $\ell < \ell_{\max}$

1) for $1 \le t \le n$ compute all trellis-to-variable messages $\mathcal{O}_t^{(\ell)}$ [Fig. 4 and the Appendix];
2) compute all variable-to-check messages $v_m^{(\ell+1)}$ [Fig. 5 and (10)];
3) compute all check-to-variable messages $u_m^{(\ell+1)}$ [Fig. 6 and (11)];
4) for $1 \le t \le n$ compute all variable-to-trellis messages $e_t^{(\ell+1)}$ [Fig. 7 and (12)];
5) increment ℓ by 1.

- **Decode**

1) for $1 \le t \le n$ decide $\hat{x}_t = \mathrm{sign}(\mathcal{O}_t^{(\ell_{\max}-1)} + e_t^{(\ell_{\max})})$, where we use $\mathrm{sign}(0) = 1$.

III. Concentration and the "Zero-Error" Threshold

In this section, we will prove that for i.u.d. information sequences, for almost all graphs and almost all cosets, the decoder behaves very closely to the expected behavior. When we say here that an information sequence is i.u.d., we mean that the channel input is a sequence of independent and uniformly distributed random variables. We will then conclude that there exists at least one graph and one coset for which the decoding probability of error can be made arbitrarily small on an i.u.d.

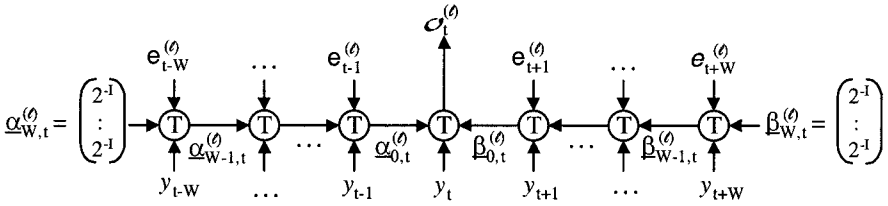

Fig. 4. Message-passing through the trellis—the "windowed" BCJR algorithm.

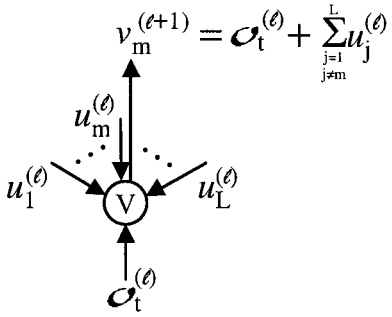

Fig. 5. Computation of messages from variable nodes to check nodes.

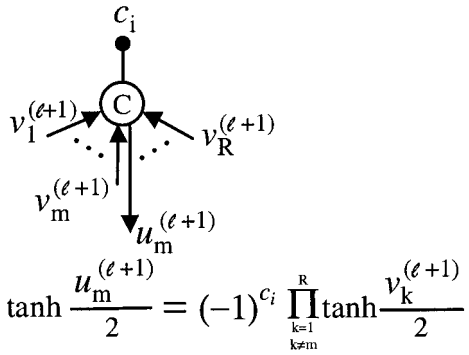

Fig. 6. Computation of messages from check nodes to variable nodes.

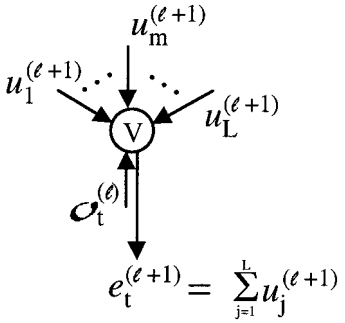

Fig. 7. Computation of messages from variable nodes to trellis nodes.

information sequence if the noise variance does not exceed a threshold. The proofs follow closely the ideas presented in [24], [25] for memoryless channels and rely heavily on results presented there. The main difference is that the channel under consideration here has an input-dependent memory. Therefore, we first must prove a concentration statement for every possible input sequence, and then show that the average decoder performance is closely concentrated around the decoder performance when the input sequence is i.u.d.

The section is organized as follows. In Section III-A, the basic notation is introduced. Section III-B gives the concentration result, while Section III-C defines the "zero-error" threshold and concludes that there exists a Gallager coset code that achieves an arbitrarily small probability of error if the noise variance is below the threshold.

A. Message-Flow Neighborhoods, Trees, and Error Probabilities

For clarity of presentation, we consider only regular Gallager codes, where every variable node has degree $L = L_{\max}$ and every check node has degree $R = R_{\max}$. In the joint code/channel graph (Fig. 3), consider an edge e that connects a variable node V_e to a check node C_e. In [25], Richardson and Urbanke define a directed neighborhood of depth d (distance d) of the edge e. Here, we cannot define a neighborhood based on the distance because the joint code/channel graph (Fig. 3) is not a bipartite graph. Instead, we define a *message-flow neighborhood* of depth ℓ (which equals the directed neighborhood if the graph is bipartite). Let $v_e^{(\ell+1)}$ be the message passed from the variable node V_e to the check node C_e in round ℓ. The message-flow neighborhood of depth ℓ of the edge e is a subgraph that consists of the two nodes C_e and V_e, the edge e, and all nodes and edges that contribute to the computation of the message $v_e^{(\ell+1)}$. In Fig. 8(a), a depth-1 message-flow neighborhood is depicted for the following parameters $(I, W, L, R) = (1, 1, 2, 3)$. The row of bits (binary symbols) "0101" given above the trellis section in Fig. 8(a) represent the binary symbols of the codeword $\underline{s}$ corresponding to the trellis nodes that influence the message flow. Since the channel has ISI memory of length I, there are exactly $2W + I + 1$ $(=4)$ binary symbols that influence the message flow. Fig. 8(b) is an equivalent short representation of the depth-1 neighborhood depicted in Fig. 8(a). A message-flow neighborhood of depth ℓ can now be obtained by branching out the neighborhood of depth 1. This is depicted in Fig. 9.

Since the channel has memory, the transmitted binary symbols do, in fact, influence the statistics of the messages in the message-flow neighborhood. We, therefore, must distinguish between neighborhoods of different *types*, where the type depends on the transmitted bits. The neighborhood type $\underline{\theta}$ is defined by the binary symbols that influence the message at the end (top) of the message-flow neighborhood. We simply index the types by the binary symbols in the neighborhood (with an appropriate, say lexicographic, ordering). For example, the message-flow neighborhood of depth ℓ in Fig. 9 is of type

$$\underline{\theta} = [0101, \ldots, 1111, \ldots, 0000, 1110, \ldots, 1001]^{\mathrm{T}}.$$

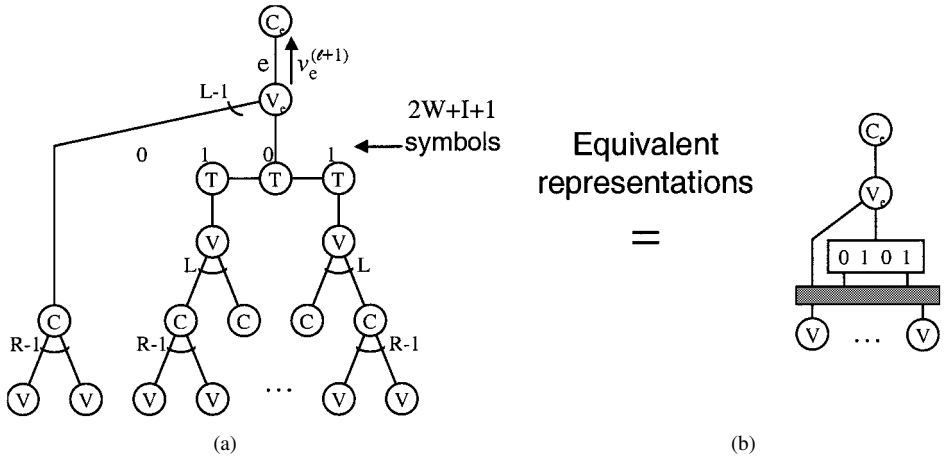

Fig. 8. Equivalent representations of a message flow neighborhood of depth 1. In this figure, $(I,\ W,\ L,\ R) = (1,\ 1,\ 2,\ 3)$.

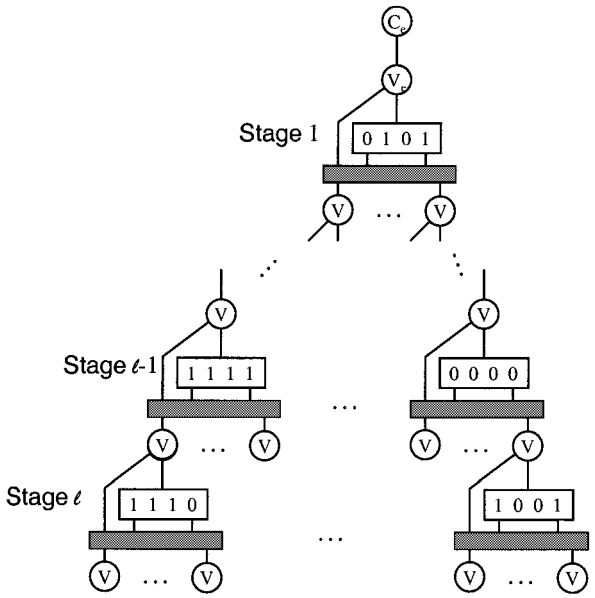

Fig. 9. Diagram of a message-flow neighborhood of depth ℓ. The neighborhood type is $\underline{\theta}^{\mathrm{T}} = [0101, \ldots, 1111, \ldots, 0000, 1110, \ldots, 1001]$.

There are as many possible types of message flow neighborhoods of depth ℓ as there are possible fillings of binary digits in Fig. 9. One can verify that for a regular Gallager code there are exactly $2^{N(\ell)}$ possible types of message-flow neighborhoods of depth ℓ, where

$$N(\ell) = (2W+I+1)\cdot\frac{(R-1)^{\ell}(2WL+L-1)^{\ell}-1}{(R-1)(2WL+L-1)-1}. \quad (13)$$

We index these neighborhoods as

$$\underline{\theta}_i \in \{0,\ 1\}^{N(\ell)}, \qquad \text{where } 1 \le i \le 2^{N(\ell)}.$$

A tree-like neighborhood, or simply a *tree* of depth ℓ is a message-flow neighborhood of depth ℓ in which all nodes appear only once. In other words, a tree of depth ℓ is a message-flow neighborhood that contains no loops. Just like message-flow neighborhoods, the trees of depth ℓ can be of any of the $2^{N(\ell)}$ types $\underline{\theta}_i \in \{0,\ 1\}^{N(\ell)}$, where $1 \le i \le 2^{N(\ell)}$.

Define $s_{\underline{\theta}}$ as the binary symbol corresponding to the message node V_e at the top of the message-flow neighborhood of type $\underline{\theta}$. In Fig. 8(a), the binary symbol $s_{\underline{\theta}}$ can be read as the symbol directly below the node V_e, i.e., $s_{\underline{\theta}} = 0$. The corresponding bipolar value of the symbol is $x_{\underline{\theta}} = 1 - 2s_{\underline{\theta}} = 1$. Define $\pi_{\underline{\theta}}^{(\ell)}$ as the probability that the tree of type $\underline{\theta}$ and depth ℓ delivers an incorrect message, i.e.,

$$\pi_{\underline{\theta}}^{(\ell)} = \Pr\left(v_e^{(\ell+1)} \cdot x_{\underline{\theta}} < 0 \,|\, \text{tree type } \underline{\theta}\right). \quad (14)$$

The probability in (14) is taken over all possible outcomes of the channel outputs when $\underline{\theta}$ is the tree type, i.e., when the binary symbols that define $\underline{\theta}$ are transmitted.

We define the probability $\Pr(\underline{\theta}|\underline{s})$ as the probability that a message-flow neighborhood (of a random edge) is of type $\underline{\theta}$ when the transmitted n-long sequence is $\underline{s}$ and the code graph is chosen uniformly at random from all possible graphs with degree polynomials $\lambda(x)$ and $\rho(x)$, i.e.,

$$\Pr(\underline{\theta}|\underline{s}) = \Pr(\text{neighborhood type} = \underline{\theta} \,|\, \text{transmitted sequence} = \underline{s}). \quad (15)$$

Note that the probability defined in (15) does not depend on the coset $\underline{r}$; also note that there always exists a vector $\underline{r}$ such that for any chosen parity-check matrix $\boldsymbol{H}$ the vector $\underline{s}$ is a codeword of the coset code specified by $\boldsymbol{H}$ and $\underline{r}$.

Next, define the *error concentration probability* when $\underline{s}$ is the transmitted sequence as

$$p^{(\ell)}(\underline{s}) = \sum_{i=1}^{2^{N(\ell)}} \pi_{\underline{\theta}_i}^{(\ell)} \Pr(\underline{\theta}_i|\underline{s}). \quad (16)$$

Define the *i.u.d. error concentration probability* $p_{\text{i.u.d.}}^{(\ell)}$ as the error concentration probability when all $2^{N(\ell)}$ neighborhood types $\underline{\theta}_i$, $1 \le i \le 2^{N(\ell)}$, are equally probable

$$p_{\text{i.u.d.}}^{(\ell)} = \sum_{i=1}^{2^{N(\ell)}} \pi_{\underline{\theta}_i}^{(\ell)} 2^{-N(\ell)}. \quad (17)$$

In the next subsection, we prove that for most graphs, if $\underline{s}$ is the transmitted codeword, then the probability of a variable-to-check message being erroneous after ℓ rounds of the message-passing decoding algorithm is highly concentrated around the value $p^{(\ell)}(\underline{s})$. Also, we prove that if the transmitted sequence is i.u.d., then the probability of a variable-to-check message being

erroneous after ℓ rounds of the message-passing decoding algorithm is highly concentrated around the value $p^{(\ell)}_{\text{i.u.d.}}$. To do that, we need the following result from [23]. Define $\mathrm{P}_{\overline{t}}$ as the probability that a neighborhood of depth ℓ is not a tree when a code graph is chosen uniformly at random from all possible graphs with degree polynomials $\lambda(x)$ and $\rho(x)$. In [23], it is shown that

$$\mathrm{P}_{\overline{t}} = \Pr(\text{neighborhood not a tree}) \leq \frac{\gamma}{n} \tag{18}$$

where γ is a constant independent of n.[3]

B. Concentration Theorems

Theorem 1: Let $\underline{s}$ be the transmitted codeword. Let $Z^{(\ell)}(\underline{s})$ be the random variable that denotes the number of erroneous variable-to-check messages after ℓ rounds of the message-passing decoding algorithm when the code graph is chosen uniformly at random from the ensemble of graphs with degree polynomials $\lambda(x)$ and $\rho(x)$. Let n_e be the number of variable-to-check edges in the graph. For an arbitrarily small constant $\varepsilon > 0$, there exists a positive number β, such that if $n > \frac{2\gamma}{\varepsilon}$, then

$$\Pr\left(\left|\frac{Z^{(\ell)}(\underline{s})}{n_e} - p^{(\ell)}_{\underline{s}}\right| \geq \varepsilon\right) \leq \mathrm{e}^{-\beta\varepsilon^2 n}. \tag{19}$$

Proof: The proof follows closely the proof of the concentration theorem for memoryless channels presented in [25]. First note that

$$\begin{aligned}
&\Pr\left(\left|\frac{Z^{(\ell)}(\underline{s})}{n_e} - p^{(\ell)}(\underline{s})\right| \geq \varepsilon\right) \\
&\quad\leq \Pr\left(\left|\frac{Z^{(\ell)}(\underline{s})}{n_e} - \frac{\mathrm{E}\left[Z^{(\ell)}(\underline{s})\right]}{n_e}\right| \geq \frac{\varepsilon}{2}\right) \\
&\qquad + \Pr\left(\left|\frac{\mathrm{E}\left[Z^{(\ell)}(\underline{s})\right]}{n_e} - p^{(\ell)}(\underline{s})\right| \geq \frac{\varepsilon}{2}\right).
\end{aligned} \tag{20}$$

The random variable $Z^{(\ell)}(\underline{s})$ depends on the deterministic sequence $\underline{s}$ and its probability space is the union of the ensemble of graphs with degree polynomials $\lambda(x)$, $\rho(x)$, and the ensemble of channel noise realizations (which uniquely define the channel outputs since $\underline{s}$ is known). Following [23], [25], we form a Doob edge-and-noise-revealing martingale and apply Azuma's inequality [40] to get

$$\Pr\left(\left|\frac{Z^{(\ell)}(\underline{s})}{n_e} - \frac{\mathrm{E}\left[Z^{(\ell)}(\underline{s})\right]}{n_e}\right| \geq \frac{\varepsilon}{2}\right) \leq 2\mathrm{e}^{-\beta\varepsilon^2 n} \tag{21}$$

where β depends only on $\lambda(x)$, $\rho(x)$, and ℓ.

Next, we show that the second term on the right-hand side of (20) equals 0 by using inequality (18). Again, this is adopted from [25], but adapted to a channel with ISI memory. We have

$$\begin{aligned}
\mathrm{E}\left[Z^{(\ell)}(\underline{s})\right] &\leq n_e(1-\mathrm{P}_{\overline{t}}) \sum_{i=1}^{2^{N(\ell)}} \pi^{(\ell)}_{\underline{\theta}_i} \Pr(\underline{\theta}_i|\underline{s}) + n_e\frac{\gamma}{n} \\
&\leq n_e \sum_{i=1}^{2^{N(\ell)}} \pi^{(\ell)}_{\underline{\theta}_i} \Pr(\underline{\theta}_i|\underline{s}) + n_e\frac{\gamma}{n} \\
&\leq n_e p^{(\ell)}(\underline{s}) + n_e\frac{\gamma}{n}
\end{aligned} \tag{22}$$

[3]Actually, in [23] this fact is shown for a bipartite graph, but the extension to joint code/channel graphs of Fig. 3 is straightforward.

and

$$\begin{aligned}
\mathrm{E}\left[Z^{(\ell)}(\underline{s})\right] &\geq n_e(1-\mathrm{P}_{\overline{t}}) \sum_{i=1}^{2^{N(\ell)}} \pi^{(\ell)}_{\underline{\theta}_i} \Pr(\underline{\theta}_i|\underline{s}) \\
&\geq n_e \sum_{i=1}^{2^{N(\ell)}} \pi^{(\ell)}_{\underline{\theta}_i} \Pr(\underline{\theta}_i|\underline{s}) - n_e\mathrm{P}_{\overline{t}} \sum_{i=1}^{2^{N(\ell)}} \Pr(\underline{\theta}_i|\underline{s}) \\
&\geq n_e p^{(\ell)}(\underline{s}) - n_e\frac{\gamma}{n}.
\end{aligned} \tag{23}$$

Combining (22) and (23), if $n > \frac{2\gamma}{\varepsilon}$, we get

$$\Pr\left(\left|\frac{\mathrm{E}\left[Z^{(\ell)}(\underline{s})\right]}{n_e} - p^{(\ell)}(\underline{s})\right| \geq \frac{\varepsilon}{2}\right) = 0. \tag{24}$$

□

Theorem 2: Let $\underline{S}$ be a random sequence of i.u.d. binary random variables (symbols) $S_1, S_2, \ldots, S_n$. Let $Z^{(\ell)}(\underline{S})$ be the random variable that denotes the number of erroneous variable-to-check messages after ℓ rounds of the message-passing decoding algorithm when the code graph is chosen uniformly at random from the ensemble of graphs with degree polynomials $\lambda(x)$ and $\rho(x)$, and when the transmitted sequence is $\underline{S}$. Let n_e be the number of variable-to-check edges in the graph. For an arbitrarily small constant $\varepsilon > 0$, there exists a positive number β', such that if $n > \frac{2\gamma}{\varepsilon}$, then

$$\Pr\left(\left|\frac{Z^{(\ell)}(\underline{S})}{n_e} - p^{(\ell)}_{\text{i.u.d.}}\right| \geq \varepsilon\right) \leq 4\mathrm{e}^{-\beta'\varepsilon^2 n}. \tag{25}$$

Proof: Using Theorem 1, we have the following:

$$\begin{aligned}
&\Pr\left(\left|\frac{Z^{(\ell)}(\underline{S})}{n_e} - p^{(\ell)}_{\text{i.u.d.}}\right| \geq \varepsilon\right) \\
&\quad= \sum_{j=1}^{2^n} 2^{-n} \Pr\left(\left|\frac{Z^{(\ell)}(\underline{s}_j)}{n_e} - p^{(\ell)}_{\text{i.u.d.}}\right| \geq \varepsilon\right) \\
&\quad\leq \sum_{j=1}^{2^n} 2^{-n} \Pr\left(\left|\frac{Z^{(\ell)}(\underline{s}_j)}{n_e} - p^{(\ell)}(\underline{s}_j)\right| \geq \frac{\varepsilon}{2}\right) \\
&\qquad + \sum_{j=1}^{2^n} 2^{-n} \Pr\left(\left|p^{(\ell)}(\underline{s}_j) - p^{(\ell)}_{\text{i.u.d.}}\right| \geq \frac{\varepsilon}{2}\right) \\
&\quad\leq \sum_{j=1}^{2^n} 2^{-n} \cdot 2\mathrm{e}^{-\beta\varepsilon^2 n/4} + \Pr\left(\left|p^{(\ell)}(\underline{S}) - p^{(\ell)}_{\text{i.u.d.}}\right| \geq \frac{\varepsilon}{2}\right) \\
&\quad= 2\mathrm{e}^{-\beta\varepsilon^2 n/4} + \Pr\left(\left|p^{(\ell)}(\underline{S}) - p^{(\ell)}_{\text{i.u.d.}}\right| \geq \frac{\varepsilon}{2}\right).
\end{aligned} \tag{26}$$

Next, recognize that if $\underline{S}$ is an i.u.d. random sequence, all neighborhood types are equally probable, i.e., $\Pr(\underline{\theta}_i|\underline{S}) = 2^{-N(\ell)}$. Using this, we prove that $\mathrm{E}[p^{(\ell)}(\underline{S})] = p^{(\ell)}_{\text{i.u.d.}}$

$$\begin{aligned}
\mathrm{E}\left[p^{(\ell)}(\underline{S})\right] &= \sum_{j=1}^{2^n} 2^{-n} p^{(\ell)}(\underline{s}_j) \\
&= \sum_{j=1}^{2^n} 2^{-n} \sum_{i=1}^{2^{N(\ell)}} \pi^{(\ell)}_{\underline{\theta}_i} \Pr(\underline{\theta}_i|\underline{s}_j) \\
&= \sum_{i=1}^{2^{N(\ell)}} \pi^{(\ell)}_{\underline{\theta}_i} \sum_{j=1}^{2^n} 2^{-n} \Pr(\underline{\theta}_i|\underline{s}_j)
\end{aligned}$$

$$= \sum_{i=1}^{2^{N(\ell)}} \pi_{\underline{\theta}_i}^{(\ell)} \Pr(\underline{\theta}_i | \underline{S})$$
$$= \sum_{i=1}^{2^{N(\ell)}} \pi_{\underline{\theta}_i}^{(\ell)} 2^{-N(\ell)} = p_{\text{i.u.d.}}^{(\ell)}.$$

Now form a Doob symbol-revealing martingale sequence M_0, $M_1, \ldots, M_n$

$$M_t = \mathrm{E}\left[p^{(\ell)}(\underline{S}) \mid S_1, S_2, \ldots, S_t\right]$$
$$M_0 = \mathrm{E}\left[p^{(\ell)}(\underline{S})\right] = p_{\text{i.u.d.}}^{(\ell)}$$
$$M_n = \mathrm{E}\left[p^{(\ell)}(\underline{S}) \mid \underline{S}\right] = p^{(\ell)}(\underline{S}).$$

If we can show that

$$|M_{t+1} - M_t| \leq \frac{\delta}{n} \tag{27}$$

where δ is a constant dependent on $\lambda(x)$, $\rho(x)$, and ℓ (but not dependent on n) then if we apply Azuma's inequality [40], we will have

$$\Pr\left(\left|p^{(\ell)}(\underline{S}) - p_{\text{i.u.d.}}^{(\ell)}\right| \geq \frac{\varepsilon}{2}\right) \leq 2\mathrm{e}^{-(8\delta^2)^{-1}\varepsilon^2 n}. \tag{28}$$

Then, by combining (28) and (26), for $\beta' = \min(\frac{\beta}{4}, \frac{1}{8\delta^2})$, we will get (25). So, all that needs to be shown is (27).

Consider two random variables $p^{(\ell)}(\hat{\underline{S}})$ and $p^{(\ell)}(\tilde{\underline{S}})$. The random vectors $\hat{\underline{S}}$ and $\tilde{\underline{S}}$ have the following properties: 1) the first t symbols of $\hat{\underline{S}}$ and $\tilde{\underline{S}}$ are deterministic and equal $\hat{\underline{S}}_1^t = \tilde{\underline{S}}_1^t = \underline{s}_1^t$; 2) The $(t+1)$th symbol of $\hat{\underline{S}}$ is the random variable S_t, while the $(t+1)$th symbol of $\tilde{\underline{S}}$ is fixed (non-random) $\tilde{S}_{t+1} = s_{t+1}$; 3) the remaining symbols $\hat{\underline{S}}_{t+2}^n$ and $\tilde{\underline{S}}_{t+2}^n$ are i.u.d. binary random vectors statistically independent of each other. Fixing the $(t+1)$th symbol, $\tilde{S}_{t+1} = s_{t+1}$ can affect at most a constant number (call this number κ) of message-flow neighborhoods of depth ℓ. The constant κ depends on $\lambda(x)$, $\rho(x)$, and ℓ, but it does not depend on n. Therefore, for any given neighborhood type $\underline{\theta}_i$, we have

$$\left|\Pr\left(\underline{\theta}_i | \hat{\underline{S}}\right) - \Pr\left(\underline{\theta}_i | \tilde{\underline{S}}\right)\right| \leq \frac{\kappa}{n_e}. \tag{29}$$

Using the notation $\lambda'(1) = \frac{\partial \lambda(x)}{\partial x}|_{x=1}$, we can verify that

$$n_e = [\lambda'(1) + 1]n.$$

Defining $\delta = \frac{2^{N(\ell)} \cdot \kappa}{\lambda'(1)+1}$, and using (29), we get

$$\left|p^{(\ell)}(\hat{\underline{S}}) - p^{(\ell)}(\tilde{\underline{S}})\right| \leq \sum_{i=1}^{2^{N(\ell)}} \left|\Pr\left(\underline{\theta}_i | \hat{\underline{S}}\right) - \Pr\left(\underline{\theta}_i | \tilde{\underline{S}}\right)\right|$$
$$\leq 2^{N(\ell)} \frac{\kappa}{n_e} = \frac{\delta}{n}. \tag{30}$$

Inequality (27) follows from (30). □

Corollary 2.1: Let $\underline{m}$ be any information block consisting of $k = rn$ binary digits. Let $C(\boldsymbol{H}, \underline{r})$ be a code chosen uniformly at random from the ensemble $\mathcal{C}_n(\lambda(x), \rho(x))$ of Gallager coset codes. Let $Z^{(\ell)}$ be a random variable representing the number of erroneous variable-to-check messages in round ℓ of the message-passing decoding algorithm on the joint channel/code graph of the code $C(\boldsymbol{H}, \underline{r})$. Then

$$\Pr\left(\left|\frac{Z^{(\ell)}}{n_e} - p_{\text{i.u.d.}}^{(\ell)}\right| \geq \varepsilon\right) \leq 4\mathrm{e}^{-\beta'\varepsilon^2 n}. \tag{31}$$

Proof: If $\boldsymbol{H}$ and $\underline{r}$ are chosen independently and uniformly at random, then the resulting codeword in (7) consists of i.u.d. binary symbols, and Theorem 2 applies directly. □

C. "Zero-Error" Threshold

The term "zero-error" threshold is a slight abuse because the decoding error can never be made equal to zero, but the concentration probability can be equal to zero in the limit as $\ell \to \infty$, and hence the probability of decoding error can be made arbitrarily small. As in [25], the "zero-error" noise standard deviation *threshold* σ^* is defined as

$$\sigma^* = \sup \sigma \tag{32}$$

where the supremum in (32) is taken over all noise standard deviations σ for which

$$\lim_{\ell \to \infty} p_{\text{i.u.d.}}^{(\ell)} = 0. \tag{33}$$

Corollary 2.2: Let $\underline{m}$ be an information block chosen uniformly at random from $2^k = 2^{rn}$ binary sequences of length k. There exists a code $C(\boldsymbol{H}, \underline{r})$ in the ensemble $\mathcal{C}_n(\lambda(x), \rho(x))$ of Gallager coset codes, such that for any $\sigma < \sigma^*$, the probability of error can be made arbitrarily low, i.e., if $Z^{(\ell)}$ is the number of erroneous variable-to-check messages in round ℓ of the message-passing decoding algorithm on the joint channel/code graph of the code $C(\boldsymbol{H}, \underline{r})$, then

$$\Pr\left(\frac{Z^{(\ell)}}{n_e} \geq 2\varepsilon \mid C(\boldsymbol{H}, \underline{r})\right) \leq 4\mathrm{e}^{-\beta'\varepsilon^2 n}. \tag{34}$$

Proof: Define an indicator random variable

$$\mathcal{I}\left(Z^{(\ell)}\right) = \begin{cases} 1, & \text{if } \left|\frac{Z^{(\ell)}}{n_e} - p_{\text{i.u.d.}}^{(\ell)}\right| \geq \varepsilon \\ 0, & \text{otherwise.} \end{cases} \tag{35}$$

From Corollary 2.1, for $\boldsymbol{H}$, $\underline{r}$, and $\underline{m}$ chosen uniformly at random we have $\mathrm{E}[\mathcal{I}(Z^{(\ell)})] \leq 4\mathrm{e}^{-\beta'\varepsilon^2 n}$. Since the expected value is lower than $4\mathrm{e}^{-\beta'\varepsilon^2 n}$, we conclude that there must exist at least one graph $\boldsymbol{H}$ and one coset-defining vector $\underline{r}$ such that for $\underline{m}$ chosen uniformly at random we have

$$\mathrm{E}[\mathcal{I}(Z^{(\ell)}) \mid C(\boldsymbol{H}, \underline{r})] \leq 4\mathrm{e}^{-\beta'\varepsilon^2 n}$$

i.e., there exists a graph $\boldsymbol{H}$ and a coset-defining vector $\underline{r}$ such that for $\underline{m}$ chosen uniformly at random

$$\Pr\left(\left|\frac{Z^{(\ell)}}{n_e} - p_{\text{i.u.d.}}^{(\ell)}\right| \geq \varepsilon \mid C(\boldsymbol{H}, \underline{r})\right) \leq 4\mathrm{e}^{-\beta'\varepsilon^2 n}. \tag{36}$$

The assumption $\sigma < \sigma^*$ guarantees

$$\lim_{\ell \to \infty} p_{\text{i.u.d.}}^{(\ell)} = 0.$$

Since $\lim_{\ell\to\infty} p_{\text{i.u.d.}}^{(\ell)} = 0$, it follows that for every $\varepsilon > 0$, there exists an integer $\ell(\varepsilon)$ such that for every $\ell \geq \ell(\varepsilon)$ we have $p_{\text{i.u.d.}}^{(\ell)} \leq \varepsilon$. Then, for $\ell \geq \ell(\varepsilon)$, we have

$$\Pr\left(\frac{Z^{(\ell)}}{n_e} \geq 2\varepsilon \mid C(\boldsymbol{H}, \underline{r})\right) \leq \Pr\left(\left|\frac{Z^{(\ell)}}{n_e} - p_{\text{i.u.d.}}^{(\ell)}\right| \geq \varepsilon \mid C(\boldsymbol{H}, \underline{r})\right). \quad (37)$$

The desired result (34) follows by combining (36) and (37). □

IV. Density Evolution and Threshold Computation

A. Density Evolution

Define $f_{V|\underline{\theta}}^{(\ell+1)}(\xi|\underline{\theta})$ as the pdf of the message $v_e^{(\ell+1)}$ obtained at the top of a depth-ℓ tree of type $\underline{\theta}$, see Fig. 8. With this notation, we may express the i.u.d. error concentration probability as

$$\begin{aligned} p_{\text{i.u.d.}}^{(\ell)} &= \sum_{i=1}^{2^{N(\ell)}} 2^{-N(\ell)} \pi_{\underline{\theta}_i}^{(\ell)} \\ &= \sum_{i=1}^{2^{N(\ell)}} 2^{-N(\ell)} \int_{-\infty}^{0} f_{V|\underline{\theta}_i}^{(\ell+1)}(\xi|\underline{\theta}_i) \cdot x_{\underline{\theta}_i} \cdot \mathrm{d}\xi \\ &= \int_{-\infty}^{0} \left[\sum_{i=1}^{2^{N(\ell)}} 2^{-N(\ell)} f_{V|\underline{\theta}_i}^{(\ell+1)}(\xi|\underline{\theta}_i) \cdot x_{\underline{\theta}_i}\right] \mathrm{d}\xi \\ &= \int_{-\infty}^{0} f_V^{(\ell+1)}(\xi)\, \mathrm{d}\xi. \end{aligned} \quad (38)$$

Here, $f_V^{(\ell+1)}(\xi)$ is the *average* pdf (averaged over all tree types) of the *correct* message from a variable node to a check node in round ℓ of the message-passing algorithm on a tree. We can obtain the pdf $f_V^{(\ell+1)}(\xi)$ in several different ways. Here, we perform the averaging in every round and enter a new round with an average pdf from the previous round, i.e., we *evolve* $f_V^{(\ell)}(\xi)$ into $f_V^{(\ell+1)}(\xi)$. This method was used in [14] for discrete messages and in [25] for continuous messages, where it was termed *density evolution*.

Denote by $f_{\mathcal{O}}^{(\ell)}(\xi)$ the *average* density (pdf) of a message $o_t^{(\ell)}$ in the ℓth round of the message-passing algorithm (averaged over all tree types), see Fig. 5. Let $f_U^{(\ell)}(\xi)$ denote the *average* pdf of a message $u_m^{(\ell)}$ in the ℓth round of the message-passing algorithm on a tree. Then the average density (pdf) $f_V^{(\ell+1)}(\xi)$ is given by

$$f_V^{(\ell+1)}(\xi) = f_{\mathcal{O}}^{(\ell)}(\xi) \otimes \left[\sum_{i=1}^{L_{\max}} \lambda_i \left(\bigotimes_{k=1}^{i-1} f_U^{(\ell)}(\xi)\right)\right] \quad (39)$$

where $\otimes$ stands for the convolution operation, and $\bigotimes_{k=1}^{i-1}$ denotes the convolution of $i-1$ pdfs. As shorthand, we use the following notation:

$$\lambda\left(f_U^{(\ell)}(\xi)\right) = \sum_{i=1}^{L_{\max}} \lambda_i \left(\bigotimes_{k=1}^{i-1} f_U^{(\ell)}(\xi)\right). \quad (40)$$

We also drop the function argument ξ since it is common for all convolved pdfs. Then (39) may be conveniently expressed as

$$f_V^{(\ell+1)} = f_{\mathcal{O}}^{(\ell)} \otimes \lambda\left(f_U^{(\ell)}\right). \quad (41)$$

Equation (41) denotes the evolution of the average density (pdf) through a variable node, Fig. 5.

To express the density evolution through a check node (Fig. 6), we require a variable change, resulting in a cumbersome change of measure. A convolution can then be defined in the new domain and an expression can be found for the density evolution through check nodes [25]. Here we do not pursue this rather complicated procedure because a numerical method for density evolution through check nodes can easily be obtained through a table lookup, for details see [27]. Here we simply denote this density evolution as

$$f_U^{(\ell+1)} = \sum_{i=1}^{R_{\max}} \rho_i \mathcal{E}_c^{i-1}\left(f_V^{(\ell+1)}\right) \quad (42)$$

where $\mathcal{E}_c^{i-1}(f_V^{(\ell+1)})$ is symbolic notation for the average message density obtained by evolving the density $f_V^{(\ell+1)}(\xi)$ through a check node of degree i. We further express (42) by the following notation:

$$f_U^{(\ell+1)} = \rho\left[\mathcal{E}_c\left(f_V^{(\ell+1)}\right)\right]. \quad (43)$$

Similar to (41), the average density (pdf) of messages $e_t^{(\ell+1)}$ (Fig. 7) is obtained using the convolution operator

$$f_E^{(\ell+1)} = \sum_{i=1}^{L_{\max}} \frac{\lambda_i}{i\int_0^1 \lambda(x)\,dx} \left(\bigotimes_{k=1}^{i} f_U^{(\ell+1)}(\xi)\right) = \overline{\lambda}\left(f_U^{(\ell+1)}\right). \quad (44)$$

Note that in this equation the degree distribution is averaged with respect to the nodes, rather than the edges. This explains the term $\lambda_i/(i\int_0^1 \lambda(x)\,dx)$, which is the fraction of variable nodes with degree i. The notation $\overline{\lambda}(f_U^{(\ell+1)})$ is symbolic shorthand. The step that is needed to close the loop of a single density evolution round is the evolution of the average density $f_E^{(\ell+1)}$ into the average density $f_{\mathcal{O}}^{(\ell+1)}$, i.e., the evolution of message densities through the trellis portion of the joint code/channel graph. We denote this step as

$$f_{\mathcal{O}}^{(\ell+1)} = \mathcal{E}_t\left(f_E^{(\ell+1)}, f_N\right) \quad (45)$$

where $\mathcal{E}_t$ is symbolic notation for *trellis evolution* and f_N denotes the pdf of the channel noise (in this case a zero-mean Gaussian with variance σ^2). Even though no closed-form solution for (45) is known, it can be calculated numerically using Monte Carlo techniques.

The density evolution is now given by

- **Initialization**

1) $f_N(\xi) = \frac{1}{\sqrt{2\pi\sigma^2}} \mathrm{e}^{-\frac{\xi^2}{2\sigma^2}}$;
2) set $f_V^{(0)}(\xi) = \delta(\xi)$ (where δ is the Dirac function).

- **For** $\ell = 0$ **to** $\ell_{\max} - 1$

1) $f_U^{(\ell)} = \rho\left[\mathcal{E}_c\left(f_V^{(\ell)}\right)\right]$;
2) $f_E^{(\ell)} = \overline{\lambda}\left(f_U^{(\ell)}\right)$;

3) $f_{\mathcal{O}}^{(\ell)} = \mathcal{E}_t\left(f_E^{(\ell)}, f_N\right)$;
4) $f_V^{(\ell+1)} = f_{\mathcal{O}}^{(\ell)} \otimes \lambda\left(f_U^{(\ell)}\right)$.

- **Compute**

1) $p_{\text{i.u.d.}}^{(\ell)} = \int_{-\infty}^{0} f_V^{(\ell+1)}(\xi)\,\mathrm{d}\xi$.

B. Threshold Computation

With the density evolution algorithm described in the previous subsection, the zero-error threshold σ^* can be evaluated (up to the numerical accuracy of the computation machine) as the maximal value of the noise variance σ for which $p_{\text{i.u.d.}}^{(\ell)} < \epsilon$, where ϵ is the numerical accuracy tolerance.

With a finite-precision machine, we must quantize the messages, resulting in a discrete probability mass function. For a sufficiently large number of quantization levels, the discrete probability mass functions are good approximations of continuous density functions (pdfs) $f_U^{(\ell)}$, $f_E^{(\ell)}$, $f_{\mathcal{O}}^{(\ell)}$, and $f_V^{(\ell)}$. In the for-loop of the density evolution algorithm in Section IV-A, steps 2) and 4) are straightforward convolutions (easily implemented numerically using the fast Fourier transform [41]). Step 1) of the for-loop can easily be implemented using a table lookup as explained in [27], or using a rather cumbersome change of measure explained in [25]. Actually, only step 3) of the for-loop needs further explanation. Since no closed-form solution is known for evolving densities through trellis sections, we employ a *Monte Carlo* approach to obtain a histogram that closely approximates $f_{\mathcal{O}}^{(\ell)}$. This has first been suggested in [26] for trellises of constituent convolutional codes of turbo codes. In [26], Richardson and Urbanke run the BCJR algorithm on a long trellis section when the input is the all-zero sequence. Here, since the channel has memory, the transmitted sequence must be a randomly chosen i.u.d. binary sequence. The length n of the sequence must be very long so that we can ignore the trellis boundary effects.

To implement step 3) of the for-loop in the density evolution algorithm in Section IV-A, for $1 \le t \le n$ we generate the symbols $x_t \in \{-1, 1\}$ independently and uniformly at random. They are then transmitted over the noisy ISI channel to get the channel output realization $\underline{y}_1^n$. We generate the extrinsic information $e_t^{(\ell)}$ for $1 \le t \le n$ as follows. For all $1 \le t \le n$, first create independent realizations $\tilde{e}_t^{(\ell)}$ according to the pdf (actually, histogram) $f_E^{(\ell)}$, and then set $e_t^{(\ell)} = \tilde{e}_t^{(\ell)} \cdot x_t$. For $1 \le t \le n$, we compute the *a priori* probability that the message symbol X_t equals 1 as

$$P_{1,t}^{(\ell)} = \frac{\exp\left[e_t^{(\ell)}\right]}{1 + \exp\left[e_t^{(\ell)}\right]}.$$

Using these prior probabilities and using y_t as the channel outputs, we run the BCJR algorithm [8] to compute the trellis outputs $o_t^{(\ell)}$. We then equate $f_{\mathcal{O}}^{(\ell)}$ to the histogram of the values $o_t^{(\ell)} \cdot x_t$, where $K \le t \le n - K$, and K is chosen large enough to avoid the trellis boundary effects. In [26], this technique is accelerated by forcing the consistency condition on the histogram. In ISI channels, however, consistency generally does not hold, so we must use a larger trellis section in the Monte Carlo simulation.

V. Achievable Rates of Gallager Codes

A. Achievable Rates of Binary Linear Codes Over ISI Channels

In Section II, we pointed out that $\mathcal{I}_{\text{i.u.d.}}$ is the limit (as $n \to \infty$) of the average mutual information between $\underline{X}_1^n$ and $\underline{Y}_1^n$ when $\underline{X}_1^n$ is an i.i.d. sequence with

$$\Pr(X_t = -1) = \Pr(X_t = +1) = \frac{1}{2}.$$

Since the input process, the channel, and, hence, the output process and the joint input–output process are all stationary and ergodic, one can adopt the standard random coding technique [1] to prove a coding theorem to assure that all rates $r < \mathcal{I}_{\text{i.u.d.}}$ are *achievable* (for the definition of *achievable*, see [2, p. 194]). We use the expression "standard random coding technique" to describe a method to generate the codebook, where codewords are chosen independently at random and the coded symbols are governed by the optimal input distribution. For a generic finite-state channel, see [1, Sec. 5.9] or [42] for a detailed description of the problem and the the proof of the coding theorem. For the channel in (1) with binary inputs, we present a somewhat stronger result involving *linear* codes.[4]

Theorem 3: Every rate $r < \mathcal{I}_{\text{i.u.d.}}$ is achievable; furthermore, the rate r can be achieved by linear block codes or their coset codes.

Proof: From [5], if the channel input is an i.u.d. (Bernoulli-1/2) sequence, we have

$$\mathcal{I}_{\text{i.u.d.}} = \lim_{\nu\to\infty} \frac{1}{\nu} \mathrm{I}\left(\underline{X}_1^{\nu}; \underline{Y}_1^{\nu}\right) = \lim_{\nu\to\infty} \frac{1}{\nu} \mathrm{I}\left(\underline{X}_1^{\nu}; \underline{Y}_1^{\nu} | \underline{q}_0\right)$$

where the second equality follows from the fact that the channel in (1) can be driven into any known state $\underline{q}_0$, with at most I inputs (where I is the ISI length). For any $\epsilon > 0$, there exists a positive integer N such that $\frac{I}{N} < \epsilon$ and

$$r < \frac{1}{N} \mathrm{I}\left(\underline{X}_1^N; \underline{Y}_1^N | \underline{q}_0\right) < \mathcal{I}_{\text{i.u.d.}}$$

where the starting state $\underline{q}_0$ is a known vector of I binary values, say $\underline{q}_0 = [+1, +1, \ldots, +1]^{\mathrm{T}}$. Now we consider the following transmission scheme. We transmit a binary vector $\underline{X}$, where before every block of N symbols we transmit the known sequence $\underline{q}_0$, i.e.,

$$\underline{X} = \begin{bmatrix} \underline{q}_0 \\ \underline{X}_1 \\ \underline{q}_0 \\ \underline{X}_2 \\ \vdots \\ \underline{q}_0 \\ \underline{X}_n \end{bmatrix}. \tag{46}$$

Clearly, from (46), for any $1 \le t \le n$, we have

$$\underline{q}_0 = \underline{X}_{(t-1)(N+I)+1}^{(t-1)(N+I)+I} \quad \text{and} \quad \underline{X}_t = \underline{X}_{(t-1)(N+I)+I+1}^{t(N+I)}.$$

[4]Here we use a different (and apparently simpler) proof methodology. However, the proof only applies to finite-state channels for which we can guarantee that we can achieve any state with a finite number of channel inputs (e.g., ISI channels with finite ISI memory); not for a general finite-state channel.

The symbols of the vector $\underline{X}$ are transmitted over the channel in (1) to obtain a vector $\underline{Y}$ at the channel output. Similar to the vector $\underline{X}$ in (46), we partition the vector $\underline{Y}$ as

$$\underline{Y} = \begin{bmatrix} \underline{\gamma}_1 \\ \underline{Y}_1 \\ \underline{\gamma}_2 \\ \underline{Y}_2 \\ \vdots \\ \underline{\gamma}_n \\ \underline{Y}_n \end{bmatrix}$$

where for any any $1 \leq t \leq n$, we have

$$\underline{\gamma}_t = \underline{Y}_{(t-1)(N+I)+1}^{(t-1)(N+I)+I} \quad \text{and} \quad \underline{Y}_t = \underline{Y}_{(t-1)(N+I)+I+1}^{t(N+I)}.$$

Clearly, we have a memoryless vector-channel as follows:

Input: $\underline{X}_t$ whose realization is a binary vector

$$\underline{x}_t \in \{+1, -1\}^N.$$

Output: $\underline{Y}_t$ whose realization is real vector $\underline{y}_t \in \mathbb{R}^N$.

The probability law of the vector channel is defined by the following conditional pdf:

$$f_{\underline{q}_0}(\underline{y}_t|\underline{x}_t) = f_{\underline{Y}_t|\underline{X}_t, \underline{Q}_0}(\underline{y}_t|\underline{x}_t, \underline{q}_0)$$

since the known sequence $\underline{q}_0$ is transmitted before every vector $\underline{X}_t$. This channel transition probability law is well defined [1], [42], hence, $\mathrm{I}(\underline{X}_t; \underline{Y}_t|\underline{q}_0)$ is also well defined. Note that the pdf $f_{\underline{q}_0}(\cdot|\cdot)$ is not dependent on t, which makes it possible to factor the joint pdf as

$$\begin{aligned} f_{\underline{Y}_1, \underline{Y}_2, \ldots, \underline{Y}_n|\underline{X}}\left(\underline{y}_1, \underline{y}_2, \cdots, \underline{y}_n \mid \underline{x}\right) &= \prod_{t=1}^{n} f_{\underline{Y}_t|\underline{X}_t, \underline{Q}_0}\left(\underline{y}_t \mid \underline{x}_t, \underline{q}_0\right) \\ &= \prod_{t=1}^{n} f_{\underline{q}_0}\left(\underline{y}_t \mid \underline{x}_t\right) \end{aligned}$$

showing that the vector channel is indeed memoryless. Further, quantize the output vector $\underline{Y}_t$ to get a quantized vector $\underline{\hat{Y}}_t = \mathrm{Quant}\,(\underline{Y}_t)$. Due to [1, Ch. 7], we can always find a quantizer to get a discrete channel such that the corresponding average mutual information $\frac{1}{N}\mathrm{I}(\underline{X}_t; \underline{\hat{Y}}_t|\underline{q}_0)$ is greater than the given rate r. Since ϵ is arbitrarily small, we can choose integers n and k such that

$$\begin{aligned} r &< \frac{kN}{n(N+I)} < \frac{k}{n} < \frac{1}{N}\mathrm{I}\left(\underline{X}_t; \underline{\hat{Y}}_t|\underline{q}_0\right) \\ &< \frac{1}{N}\mathrm{I}\left(\underline{X}_t; \underline{Y}_t|\underline{q}_0\right) < \mathcal{I}_{\mathrm{i.u.d.}}. \end{aligned}$$

Similar to [2, proof of Theorem 8.7.1, p. 198], we can prove that k/n is achievable for the obtained discrete memoryless channel. The reader should note that the random code we generated has $(2^N)^k$ codewords, which are statistically independent. The coded symbols are i.u.d., each with probability 2^{-N}. Every codeword consists of n vector symbols from $\{+1, -1\}^N$, say $(\underline{X}_1, \underline{X}_2, \ldots, \underline{X}_n)$. The transmitted block is $(\underline{q}_0, \underline{X}_1, \underline{q}_0, \underline{X}_2, \ldots, \underline{q}_0, \underline{X}_n)$ with length $n(N+I)$. So, the real code rate is $\frac{kN}{n(N+I)}$. The received sequence also has the same block length. However, from [2, proof of Theorem 8.7.1, p. 198], the decoding error probability can be made arbitrarily small even if we only use the typical-set decoding with respect to $\underline{X}_t$ and $\underline{\hat{Y}}_t$, which is not the full received sequence.

To prove the second part of this theorem, we should note that the error probability bound only depends on the statistical properties of the random codebook. We can generate a codebook by drawing codewords uniformly at random as in [1, Theorem 6.2.1, p. 206]. □

For binary ISI channels, define the capacity $\mathrm{C}_{\mathrm{blc}}$ as the supremum of rates r achievable by binary linear codes under any decoding algorithm. A consequence of Theorem 3 is

$$\mathcal{I}_{\mathrm{i.u.d.}} \leq \mathrm{C}_{\mathrm{blc}} \leq \mathrm{C}. \tag{47}$$

Formulating the exact relationship between $\mathcal{I}_{\mathrm{i.u.d.}}$, $\mathrm{C}_{\mathrm{i.i.d.}}$, $\mathrm{C}_{\mathrm{blc}}$, and C is still an open problem since to the best of our knowledge neither the literature nor the theorems presented in this paper answer this question. For example, it is our belief that the strict inequality $\mathrm{C}_{\mathrm{blc}} < \mathrm{C}$ must hold because binary linear codes cannot achieve spectral shaping required to match the spectral nulls of the code to the spectral nulls of the channel (see [43] for matched spectral null codes), but we cannot back up this statement with a proof. Further, we know (at least for some ISI channels) that $\mathrm{C}_{\mathrm{blc}} > \mathrm{C}_{\mathrm{i.i.d.}}$. An example can be constructed by concatenating an outer regular rate-$1/2$ Gallager code whose variable node degree is $L = L_{\max} = 3$ and check node degree is $R = R_{\max} = 6$, with an inner matched spectral null biphase code [43] of rate $1/2$. For this special construction, the resulting code is a linear (coset) code of rate–$1/4$. If we use this code for transmission over the dicode channel ($1 - D$ channel), though not explicitly shown here, we can compute that the zero-error threshold of message-passing decoding is above $\mathrm{C}_{\mathrm{i.i.d.}}$ (where for this channel it can be numerically shown using the algorithm in [9], [10] that $\mathcal{I}_{\mathrm{i.u.d.}} = \mathrm{C}_{\mathrm{i.i.d.}}$).

While the exact relationship between $\mathrm{C}_{\mathrm{blc}}$, $\mathrm{C}_{\mathrm{i.i.d.}}$, and C is still an open problem, we can prove a relationship between the zero-error threshold of the message-passing decoder of Gallager codes, and the values $\mathcal{I}_{\mathrm{i.u.d.}}$ and $\mathrm{C}_{\mathrm{i.i.d.}}$.

Proposition 1: Let r be the rate of a Gallager code and let σ^* be the threshold computed by density evolution using i.u.d. inputs. Then $r \leq \mathcal{I}_{\mathrm{i.u.d.}} \leq \mathrm{C}_{\mathrm{i.i.d.}}$, where $\mathcal{I}_{\mathrm{i.u.d.}}$ and $\mathrm{C}_{\mathrm{i.i.d.}}$ are evaluated at the noise standard deviation $\sigma = \sigma^*$.

Proof: According to the concentration theorem, the average probability of error (averaged over all random choices of the graph, the coset vector $\underline{r}$, and the information-bearing vector $\underline{m}$) can be made arbitrarily small if $\sigma < \sigma^*$. That means that there exists at least one graph that achieves an arbitrarily small average probability of decoding error (averaged over all random choices of the coset vector $\underline{r}$ and the information-bearing vector $\underline{m}$). Pick the parity-check matrix $\boldsymbol{H}$ corresponding to this graph as our code matrix. We design the following transmission scheme. The messages $\underline{m}$ are chosen uniformly at random and the coset vectors $\underline{r}$ are chosen also uniformly at random. The resulting transmitted sequence is i.i.d. with probability of each symbol 0.5, that is, the sequence is i.u.d. If the transmitted sequence is i.u.d., we cannot find a

code with rate higher than $\mathcal{I}_{\text{i.u.d.}}$ such that the decoding error is arbitrarily small. But since the decoding error (averaged over all messages $\underline{m}$ and all cosets $\underline{r}$) for the sum–product decoder of Gallager codes can be made arbitrarily small for $\sigma < \sigma^*$, we conclude that the code rate r must be smaller than the value for $\mathcal{I}_{\text{i.u.d.}}$ evaluated at $\sigma < \sigma^*$. Therefore,

$$r \leq \sup_{\sigma<\sigma^*} \mathcal{I}_{\text{i.u.d.}}(\sigma) = \mathcal{I}_{\text{i.u.d.}}(\sigma^*) \leq \text{C}_{\text{i.i.d.}}(\sigma^*). \qquad \square$$

B. Thresholds for Regular Gallager Codes as Lower Bounds on $\text{C}_{\text{i.i.d.}}$

The two proofs presented in the preceding subsection establish that the curve rate (r) versus threshold (σ^*) for a Gallager code over a binary ISI channel is upper-bounded by the curve $\mathcal{I}_{\text{i.u.d.}}$ versus σ, and upper-bounded by the curve $\text{C}_{\text{i.i.d.}}$ versus σ. Thus, we have a practical method for numerically lower-bounding $\text{C}_{\text{i.i.d.}}$. Furthermore, by virtue of specifying the degree polynomials $\lambda(x)$ and $\rho(x)$, we also characterize a code that can achieve this lower bound. This is a bounding method that is different from the closed-form bounds [6], [7] or Monte Carlo bounds [5] proposed in the past, where no bound-achieving characterization of the code is possible (except through random coding techniques which are impractical for implementations). Further, we compare the thresholds obtained by density evolution to the value $\text{C}_{\text{i.i.d.}}$ computed by the Arnold–Loeliger method [9], showing that the thresholds are very close to $\text{C}_{\text{i.i.d.}}$ in the high-code-rate regions $(0.7 \leq r \leq 1.0)$. This is exactly the region of practical importance in storage devices where high-rate codes for binary ISI channels are a necessity [3]. The codes studied in this paper do not provide tight bounds in the low-rate region, but the threshold bounds can be tightened by optimizing the degree polynomials $\lambda(x)$ and $\rho(x)$, see [44].

In this paper, we present thresholds only for regular Gallager codes[5] in the family (L, R), where $L = 3$ and R is allowed to vary in order to get a variable code rate $r = \frac{R-3}{R}$. This family of codes provides a curve r versus threshold that is very close to $\text{C}_{\text{i.i.d.}}$ for high code rates, but not for low code rates. To get tighter bounds in the low information rate regime, we would have to revert to irregular Gallager codes [14], [15], [44]. Table I tabulates the codes and their respective thresholds for the ISI channel $h(D) = \frac{1}{\sqrt{2}} - \frac{1}{\sqrt{2}}D$ with additive Gaussian noise (this channel was chosen for easy comparison to some previously published bounds [5]–[7]). The density evolution bounds (r versus SNR*) are plotted in Fig. 10. For comparison, the i.i.d. capacity numerically evaluated using the Arnold–Loeliger method [9] is also plotted in Fig. 10. (Note that for this channel, we can numerically verify that $\text{C}_{\text{i.i.d.}} = \mathcal{I}_{\text{i.u.d.}}$ at any signal-to-noise ratio (SNR) of interest.)

In [7], Shamai and Laroia introduce a curve for which they conjecture that it may be a lower bound on $\text{C}_{\text{i.i.d.}}$ (dash-dotted line in Fig. 10). Although the curve is only a conjecture, it is a very useful quick way to estimate $\text{C}_{\text{i.i.d.}}$ because it involves evaluating a one-dimensional integral, and also it seems to be a relatively accurate estimate of $\text{C}_{\text{i.i.d.}}$ as verified in [9]. For this reason, we compare the thresholds computed by density evolution to both $\text{C}_{\text{i.i.d.}}$ and to the Shamai–Laroia conjecture.

[5]Thresholds for irregular Gallager codes can also be obtained via density evolution.

TABLE I
THRESHOLDS FOR REGULAR GALLAGER CODES $(L = 3)$; CHANNEL $h(D) = \frac{1}{\sqrt{2}} - \frac{1}{\sqrt{2}}D$

code (L, R)	check-node degree R	code rate $r = \frac{R-3}{R}$	threshold σ^*	threshold SNR* [dB] SNR$^* = 20\log_{10}\frac{1}{\sigma^*}$
(3,3)	3	0.000	2.042	-6.201
(3,4)	4	0.250	1.196	-1.554
(3,5)	5	0.400	0.945	0.492
(3,6)	6	0.500	0.822	1.703
(3,8)	8	0.625	0.697	3.136
(3,10)	10	0.700	0.631	4.000
(3,15)	15	0.800	0.547	5.241
(3,30)	30	0.900	0.459	6.764
(3,60)	60	0.950	0.404	7.873
(3,150)	150	0.980	0.355	8.996

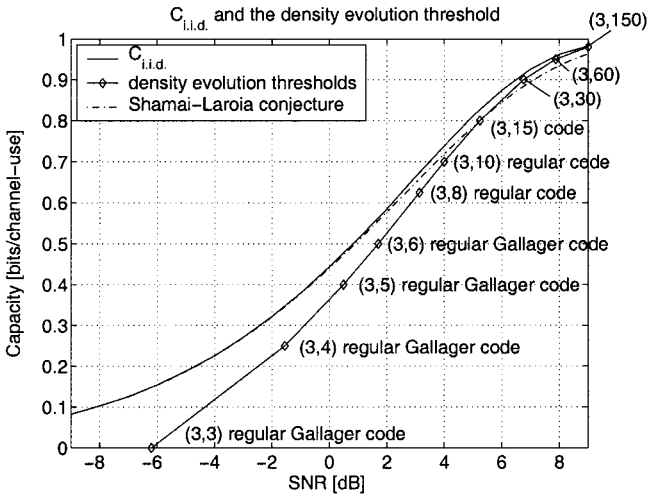

Fig. 10. The i.i.d. capacity $\text{C}_{\text{i.i.d.}}$ and thresholds for regular Gallager codes with message node degree $L = 3$.

Fig. 11 indicates the position of the threshold SNR* for two regular Gallager codes (the $(3, 6)$ code of rate $r = 0.5$ and the $(3, 30)$ code of rate $r = 0.9$) along with the SNR values for the Shamai–Laroia conjecture at these rates and the best known value for $\text{C}_{\text{i.i.d.}}$ computed by the Arnold–Loeliger method [9]. (Again, note that for this channel, numerical evaluations show $\text{C}_{\text{i.i.d.}} = \mathcal{I}_{\text{i.u.d.}}$.) Also shown in Fig. 11 are the SNR values for which simulated Gallager codes of lengths $n \in \{10^4, 10^5, 10^6\}$ achieved bit-error rates of 10^{-5}. First, observe that the thresholds accurately predict the limit of code performance as the block length n becomes very large. Next, observe that for the code $(3, 30)$ of rate $r = 0.9$, the threshold is tight (tighter than the Shamai–Laroia conjecture), establishing that regular Gallager codes are relatively good codes for high rates. For the code $(3, 6)$ of rate $r = 0.5$, the threshold is far away from the SNR values corresponding to $\text{C}_{\text{i.i.d.}}$ and the Shamai–Laroia conjecture, respectively, suggesting that good Gallager codes in the low-rate regime should be sought among irregular codes [14], [44].

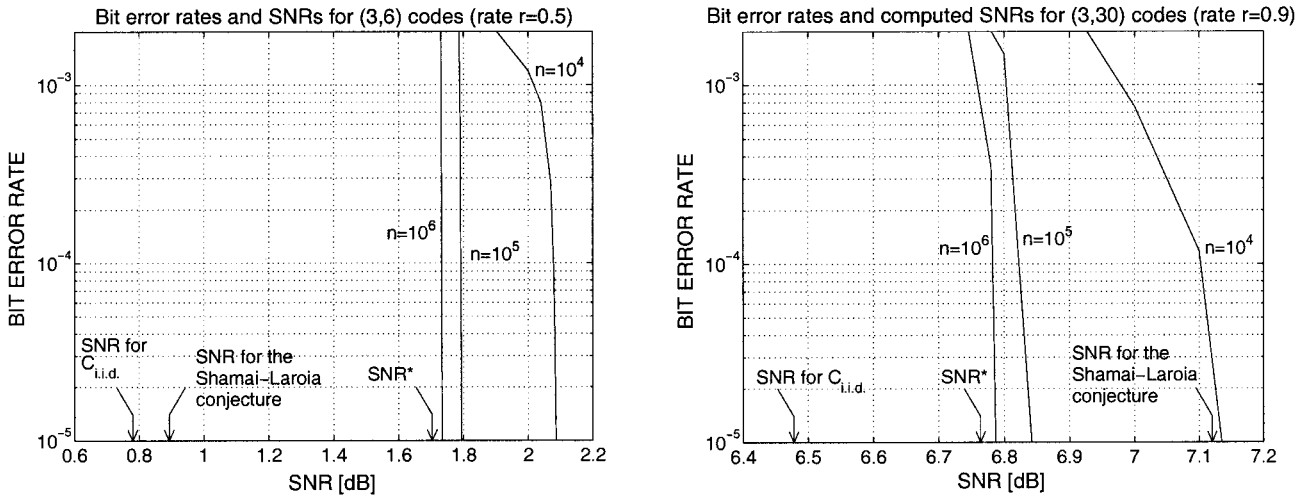

Fig. 11. Comparison of bit-error rate simulation results for finite-block-length Gallager codes of rates $r = 0.5$ and $r = 0.9$ to $\mathrm{C}_{i.i.d.}$ and to the density evolution thresholds and to the Shamai–Laroia conjectured bound.

C. The BCJR-Once Bound

Due to the high computational complexity of the BCJR algorihm, several authors suggest applying the BCJR step only once [33], [45] and subsequently iterating the message-passing decoding algorithm only within the code subgraph of the joint channel/code graph (see Fig. 3). Clearly, this strategy is suboptimal to fully iterating between the channel and the code subgraphs of the joint channel/code graph, but does provide substantial computational savings, which is of particular importance for on-chip implementations. The question that remains is how much does one lose in terms of achievable information rate when this strategy is applied. We develop next what we call the *BCJR-once* bound $\mathrm{C}_{\text{BCJR-once}}$ which answers this question.

Let $\underline{X}_1^n$ be a realization of a random channel input sequence $\underline{X}_1^n$. Let $\underline{y}_1^n$ be a realization of the channel output sequence $\underline{Y}_1^n$. Let $\mathcal{O}_t = \mathcal{O}_t^{(0)}$ be the random variable representing the message passed from the tth trellis node to the variable node in the first round of the sum–product algorithm (i.e., it is the output of the BCJR algorithm applied once in the first iteration of decoding). Denote the vector of realizations $o_1, \ldots, o_n$ by $\underline{o}_1^n$, which is a realization of a random vector $\underline{\mathcal{O}}_1^n$. We assume that the input sequence is i.u.d., and define the BCJR-once bound as

$$\mathrm{C}_{\text{BCJR-once}} = \lim_{n\to\infty} \frac{1}{n} \left[\sum_{t=1}^{n} \mathrm{I}(X_t; \mathcal{O}_t) \Big|_{\Pr(\underline{X}_1^n = \underline{x}_1^n) = 2^{-n}} \right]. \tag{48}$$

Two straightforward properties can be established for the BCJR-once bound.

Property 1:

$$\mathrm{C}_{\text{BCJR-once}} = \lim_{n\to\infty} \frac{1}{n} \left[\sum_{t=1}^{n} \mathrm{I}(X_t; \underline{Y}_1^n) \Big|_{\Pr(\underline{X}_1^n = \underline{x}_1^n) = 2^{-n}} \right].$$

Proof: The BCJR algorithm computes

$$o_t = \ln \frac{\Pr(X_t = 1 | \underline{Y}_1^n = \underline{y}_1^n)}{1 - \Pr(X_t = 1 | \underline{Y}_1^n = \underline{y}_1^n)}.$$

So, $\mathcal{O}_t$ is a sufficient statistic for determining X_t from $\underline{Y}_1^n$ (without knowledge of the code). Therefore (see, e.g., [2, p. 37]), $\mathrm{I}(X_t; \underline{Y}_1^n) = \mathrm{I}(X_t; \mathcal{O}_t)$. □

Property 2:

$$\mathrm{C}_{\text{BCJR-once}} \le \mathcal{I}_{\text{i.u.d.}} \le \mathrm{C}_{\text{i.i.d.}}.$$

Proof: Let $H(\underline{X}_1^n | \underline{Y}_1^n)$ denote the conditional entropy of $\underline{X}_1^n$ given $\underline{Y}_1^n$, and let $H(X_t)$ denote the entropy of X_t. From the independence bound [2, p. 28] it follows that

$$H(\underline{X}_1^n | \underline{Y}_1^n) \le \sum_{t=1}^{n} H(X_t | \underline{Y}_1^n).$$

If $\underline{X}_1^n$ is a vector of i.i.d. random variables, we have

$$\begin{aligned} \frac{1}{n} \mathrm{I}(\underline{X}_1^n; \underline{Y}_1^n) &= \frac{1}{n} \sum_{t=1}^{n} H(X_t) - H(\underline{X}_1^n | \underline{Y}_1^n) \\ &\ge \frac{1}{n} \sum_{t=1}^{n} [H(X_t) - H(X_t | \underline{Y}_1^n)] \\ &= \frac{1}{n} \sum_{t=1}^{n} \mathrm{I}(X_t; \underline{Y}_1^n). \end{aligned} \tag{49}$$

Evaluated when X_1^n is a Bernoulli-$1/2$ random (i.u.d.) sequence, the right-hand side of (49) is $\mathrm{C}_{\text{BCJR-once}}$, in the limit $n \to \infty$, and the left-hand side is $\mathcal{I}_{\text{i.u.d.}}$. □

Further, we have the following result for the BCJR-once bound $\mathrm{C}_{\text{BCJR-once}}$ if we "disregard the channel memory."

Proposition 2: Let the channel input X_t and the BCJR-once output $\mathcal{O}_t$ form a *memoryless* channel. For such a channel, any rate $r < \mathrm{C}_{\text{BCJR-once}}$ is achievable.

Proof: The proof follows the proof of Theorem 8.7.1 in [2, pp. 198–206]. □

In view of the definition (48) and Proposition 2, it is tempting to refer to $\mathrm{C}_{\text{BCJR-once}}$ as the BCJR-once *capacity* (or *rate*) instead of the term we chose—the BCJR-once *bound*. However, it is easy to show that the value $\mathrm{C}_{\text{BCJR-once}}$ is *not* a capacity (nor a rate) of a meaningful physical channel. This is because

the physical channel $X_t \Longrightarrow \mathcal{O}_t$ is not memoryless as assumed in Proposition 2, and we have

$$\mathrm{C}_{\text{BCJR-once}} < \lim_{n\to\infty} \frac{1}{n} \sup_{\Pr(\underline{X}_1^n = \underline{x}_1^n) = \prod_{t=1}^{n} \Pr(X = x_t)} \mathrm{I}(\underline{X}_1^n; \underline{\mathcal{O}}_1^n).$$

On the other hand, we believe it is appropriate to refer to $\mathrm{C}_{\text{BCJR-once}}$ as a *bound* because in Section V-D we show that the rate achievable by message-passing decoding of randomly constructed Gallager (coset) codes is upper-bounded by $\mathrm{C}_{\text{BCJR-once}}$.

The BCJR-once bound $\mathrm{C}_{\text{BCJR-once}}$ for the channel in (1) can be computed by i) running the BCJR algorithm on a very long trellis section, ii) collecting the outputs, iii) quantizing them, iv) forming a histogram for the symbol-to-symbol transition probabilities, and v) computing the mutual information of a memoryless channel whose transition probabilities equal those computed by the histogram. Another way is to devise a method similar to the Arnold–Loeliger method for computing $\mathcal{I}_{\text{i.u.d.}}$ (see [9]). First, we note that for i.u.d. input symbols, $\frac{1}{n} H(\underline{X}_1^n) = 1$. Thus, the problem of computing $\mathrm{C}_{\text{BCJR-once}}$ reduces to the problem of computing

$$\lim_{n\to\infty} \frac{1}{n} \sum_{t=1}^{n} H(X_t | \underline{Y}_1^n) = \lim_{n\to\infty} \frac{1}{n} \sum_{t=1}^{n} \mathrm{E}[\mathcal{H}(\Pr(X_t = 1 | \underline{Y}_1^n))] \quad (50)$$

where $\mathcal{H}(p)$ is the binary entropy function defined as $\mathcal{H}(p) = -p \log_2 p - (1-p)\log_2(1-p)$. For a given channel output realization $\underline{y}_1^n$, the BCJR algorithm computes $\Pr(X_t = 1 | \underline{y}_1^n)$. So, we can estimate (50) by generating an n-long i.u.d. input sequence, transmitting it over the channel and running the BCJR algorithm on the observed channel output $\underline{y}_1^n$ to get $\Pr(X_t = 1 | \underline{y}_1^n)$ for every $1 \le t \le n$. The estimate

$$\hat{\mathrm{C}}_{\text{BCJR-once}} = 1 - \frac{1}{n} \sum_{t=1}^{n} \mathcal{H}\left(\Pr\left(X_t = 1 | \underline{y}_1^n\right)\right)$$

converges with probability 1 to $\mathrm{C}_{\text{BCJR-once}}$ as $n \to \infty$.

The BCJR-once bound $\mathrm{C}_{\text{BCJR-once}}$ (computed in the manner described above for $n = 10^6$) is depicted as the dashed curve in Fig. 12 (the same figure also shows three other curves: 1) the curve for $C_{\text{i.i.d.}}$ as computed by the Arnold-Loeliger method, 2) the thresholds presented in Section V-B, and 3) the BCJR-once thresholds for regular Gallager codes which are presented next in Section V-D).

D. BCJR-Once Thresholds for Gallager Codes

Just as we performed density evolution for the full sum–product algorithm over the joint channel/code graph, we do the same for the *BCJR-once* version of the decoding algorithm. The only difference here is in the shape of the depth-ℓ message flow neighborhood, while the general method remains the same. Denote by $\sigma^*_{\text{BCJR-once}}$ the noise tolerance threshold for the BCJR-once sum–product algorithm for a Gallager-code/ISI-channel combination. The threshold $\sigma^*_{\text{BCJR-once}}$ can be computed by density evolution on a tree-like message-flow neighborhood assuming that the trellis portion of the sum–product algorithm is executed only in the first decoding round.

Proposition 3: Let r be the rate of a Gallager code and let $\sigma^*_{\text{BCJR-once}}$ be the BCJR-once noise tolerance threshold (computed by density evolution using i.u.d. inputs). Then $r \le \mathrm{C}_{\text{BCJR-once}}$, where $\mathrm{C}_{\text{BCJR-once}}$ is the BCJR-once bound evaluated at the noise standard deviation $\sigma = \sigma^*_{\text{BCJR-once}}$.

Proof: The threshold $\sigma^*_{\text{BCJR-once}}$ is computed using the density evolution method described in Section IV-A, where the trellis evolution step is executed only in the first round. Thus, the threshold $\sigma^*_{\text{BCJR-once}}$ is computed as the threshold of a Gallager code of rate r on a memoryless channel, whose channel law (conditional pdf of channel output given the channel input) is given by

$$f_{\mathcal{O}_t|X_t}(o_t | X_t = 1) = \lim_{W\to\infty} f_{\mathcal{O}_t^{[W]}|X_t}(o_t | X_t = 1).$$

Here $\mathcal{O}_t^{[W]}$ is the output of the windowed BCJR algorithm when the window size is W, and clearly, due to the channel symmetry

$$f_{\mathcal{O}_t|X_t}(o_t | X_t = -1) = f_{\mathcal{O}_t|X_t}(-o_t | X_t = 1).$$

As evident from the density averaging in the trellis portion of the density evolution, the function $f_{\mathcal{O}_t^{[W]}|X_t}(o_t | X_t = 1)$ is the average conditional pdf of $\mathcal{O}_t^{[W]}$, taken over all conditional pdfs of $\mathcal{O}_t^{[W]}$ conditioned on $\underline{X}_{t-W-I}^{t+W}$ under the constraint $X_t = 1$, i.e.,

$$f_{\mathcal{O}_t^{[W]}|X_t}\left(o_t^{[W]} | X_t = 1\right) = 2^{-(2W+I)} \sum_{\text{all } \underline{x}_{t-W-I}^{t+W};\, x_t = 1} f_{\mathcal{O}_t^{[W]}|\underline{X}_{t-W-I}^{t+W}}\left(o_t | \underline{X}_{t-W-I}^{t+W} = \underline{x}_{t-W-I}^{t+W}\right).$$

For this channel, when the noise standard deviation is σ, the channel information rate is $\mathcal{I}_W(\sigma) = \mathrm{I}(\mathcal{O}_t^{[W]}; X_t)$, where $\Pr(X_t = 1) = 1/2$. Similar to the proof of Proposition 1, we use a Gallager code where the coset vector is chosen uniformly at random in each block transmission. For this Gallager code of rate r, the transmitted symbols are i.u.d. From the concentration theorem, we have that if $\sigma \le \sigma^*_{\text{BCJR-once}}$, then the probability of decoding error is arbitrarily small. Since the probability of error can be made arbitrarily small, r must satisfy $r \le \mathcal{I}_W(\sigma^*_{\text{BCJR-once}})$. Now, let $W \to \infty$, and we get

$$r \le \lim_{W\to\infty} \mathcal{I}_W(\sigma^*_{\text{BCJR-once}}) = \mathrm{C}_{\text{BCJR-once}}(\sigma^*_{\text{BCJR-once}}). \quad \square$$

Again, we choose the family of regular Gallager codes with a constant variable node degree $L = 3$ and a varying check node degree R. The channel is $h(D) = \frac{1}{\sqrt{2}} - \frac{1}{\sqrt{2}} D$ with AWGN. The BCJR-once thresholds are given in Table II, and the corresponding plot is given in Fig. 12. Fig. 12 shows the BCJR-once bound derived in Section V-C. It can be seen that the regular Gallager codes have the capability to achieve the BCJR-once bound at high information rates if the BCJR-once version of the message-passing decoding algorithm is applied. For comparison, Fig. 12 also shows the curve for $\mathrm{C}_{\text{i.i.d.}}$ as computed by the Arnold–Loeliger method [9]. (It can be numerically verified for this channel that $\mathcal{I}_{\text{i.u.d.}} = \mathrm{C}_{\text{i.i.d.}}$.) The figure shows that the BCJR-once bound is very close to $\mathrm{C}_{\text{i.i.d.}}$ at low SNRs, but is about 1 dB away from $\mathrm{C}_{\text{i.i.d.}}$ at higher information rates. The difference between the full sum–product threshold curve

TABLE II
BCJR-ONCE THRESHOLDS FOR REGULAR GALLAGER CODES ($L = 3$); CHANNEL $h(D) = \frac{1}{\sqrt{2}} - \frac{1}{\sqrt{2}}D$

code (L, R)	check-node degree R	code rate $r = \frac{R-3}{R}$	threshold $\sigma^*_{\text{BCJR-once}}$	$\text{SNR}^*_{\text{BCJR-once}}$ [dB] $20\log_{10}\left(1/\sigma^*_{\text{BCJR-once}}\right)$	distance to σ^* [dB]
(3,3)	3	0.000	2.001	-6.024	0.177
(3,4)	4	0.250	1.110	-0.906	0.468
(3,5)	5	0.400	0.850	1.412	0.920
(3,6)	6	0.500	0.729	2.746	1.043
(3,8)	8	0.625	0.614	4.237	1.101
(3,10)	10	0.700	0.556	5.099	1.099
(3,15)	15	0.800	0.486	6.268	1.027
(3,30)	30	0.900	0.414	7.661	0.987
(3,60)	60	0.950	0.369	8.660	0.787
(3,150)	150	0.980	0.328	9.683	0.687

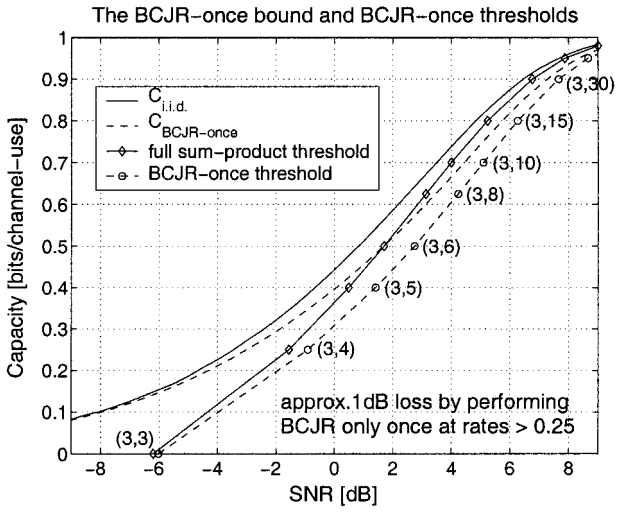

Fig. 12. The BCJR-once bound $\text{C}_{\text{BCJR-once}}$ and the BCJR-once thresholds for regular Gallager codes with $L = 3$ compared to capacity $\text{C}_{\text{i.i.d.}}$ and full sum–product thresholds (computed in Section V-B).

and the BCJR-once threshold curve also seems to be closely approximated by the difference between $\text{C}_{\text{i.i.d.}}$ and $\text{C}_{\text{BCJR-once}}$. We thus conclude that, say at rate $r = 0.9$, we can expect to see a loss of 1 dB if we execute the BCJR algorithm only once at the very beginning of the sum–product decoding algorithm (as opposed to executing the trellis sum–product algorithm in every iteration of the decoder).

VI. CONCLUSION

In this paper, we have developed a density evolution method for determining the asymptotic performance of Gallager codes over binary ISI channels in the limit $n \to \infty$, where n is the block length. We proved two concentration theorems: 1) for a particular transmitted sequence and 2) for a random transmitted sequence of i.u.d. symbols. The noise tolerance threshold was defined as the supremum of noise standard deviations for which the probability of decoding error tends to zero as the number of rounds of the decoding algorithm tends to infinity. We also established that the code rate r versus the noise tolerance threshold traces a curve that is upper-bounded by the i.i.d. capacity of binary ISI channels. We have computed the thresholds for regular Gallager codes with three ones per column of the parity-check matrix over the dicode channel $(h(D) = \frac{1}{\sqrt{2}} - \frac{1}{\sqrt{2}}D)$ and showed that they get very close to the limit of i.i.d. capacity in the high code rate region. For low code rates, regular Gallager codes do not perform close to the i.i.d. capacity. A good low-rate code should, therefore, be sought in the space of irregular Gallager codes. We showed via Monte Carlo simulations that codes with increasing code lengths n approach closely the threshold computed by density evolution.

We also explored the limits of performance of Gallager codes if a slightly more practical sum–product algorithm is utilized. Since the computational bottleneck in the sum–product algorithm for ISI channels is the trellis portion of the algorithm, it is computationally advantageous to run the trellis portion of the algorithm only once at the beginning of the first decoding iteration, i.e., the "BCJR-once" version of the algorithm. This algorithm suffers from a performance loss compared to the full sum–product algorithm. We computed the maximal achievable rate of the BCJR-once sum–product algorithm and showed that for the dicode channel $(h(D) = \frac{1}{\sqrt{2}} - \frac{1}{\sqrt{2}}D)$, the asymptotic performance loss at high rates is about 1 dB, while for low rates, the loss is minimal. Approximately, the same difference (at most 1.1 dB) was observed for the thresholds computed for the full sum–product algorithm and the BCJR-once version.

We conclude the paper by pointing out some remaining challenges in coding for binary ISI channels. While $\text{C}_{\text{i.i.d.}}$ can now be numerically evaluated [9], [10], the computation of the capacity remains a challenge. A method for lower-bounding C by extending the memory of the source is presented in [9], suggesting that at high rates, C and $\text{C}_{\text{i.i.d.}}$ are close to each other (at low rates, C and $\text{C}_{\text{i.i.d.}}$ differ substantially). In channels of practical interest, i.e., channels with signal-dependent noise [11], due to the signal-dependent nature of the noise C and $\text{C}_{\text{i.i.d.}}$ may not be close to each other even in the high-rate region. There is a need for a practical tool for computing the lower bound on C by optimizing the trellis transition probabilities of an extended-memory source [9]. Another challenging problem is to move the performance thresholds of practical codes beyond $\text{C}_{\text{i.i.d.}}$. A viable strategy may be to somehow combine the beneficial spectral-shaping characteristics of matched spectral null codes [43] with Gallager's low-density parity-check constraints, but it is not clear how to achieve this and still have a relatively simple encoder/decoder. Even for linear binary codes (i.e., non-spectral-shaping codes) the optimization of irregular Gallager codes to achieve $\text{C}_{\text{i.i.d.}}$ is also a challenging problem.

APPENDIX

We briefly describe the windowed BCJR algorithm mainly for completeness of the text. Our description uses a compact matrix notation. For a conventional description, we refer the reader to [8]. The notation in this appendix refers to Fig. 4. We start with the messages $e_t^{(\ell)}$ available from the code portion of the graph and y_t available from the channel output, where $1 \le t \le n$, and n is the codeword length. First, set

$$P_{1,t}^{(\ell)} = \frac{\exp\left[e_t^{(\ell)}\right]}{1+\exp\left[e_t^{(\ell)}\right]}$$
$$P_{-1,t}^{(\ell)} = 1 - P_{1,t}^{(\ell)}. \tag{51}$$

Then, for every t, form a diagonal matrix $\boldsymbol{D}_t^{(\ell)}$ of size $2^I \times 2^I$, where I is the ISI length of the channel. Enumerate the states of the finite-state machine with numbers 1 through 2^I. Set the ith diagonal element of $\boldsymbol{D}_t^{(\ell)}$ as

$$\boldsymbol{D}_t^{(\ell)}(i,\,i) = \begin{cases} P_{-1,t}^{(\ell)}, & \text{if } i\text{th state is reached when the} \\ & \text{channel input at time } t \text{ is } -1 \\ P_{1,t}^{(\ell)}, & \text{if } i\text{th state is reached when the} \\ & \text{channel input at time } t \text{ is } 1. \end{cases}$$

Next, for every t, form a matrix $\boldsymbol{T}_t$ of size $2^I \times 2^I$, with the entry in the intersection of the ith row and jth column given by

$$\boldsymbol{T}_t(i,j) = \begin{cases} 0, & \text{if no trellis branch} \\ & \text{connects states } i \text{ and } j \\ \exp\left[-\frac{(y_t - \xi[i,j])^2}{2\sigma^2}\right], & \text{otherwise} \end{cases}$$

where $\xi[i,\,j]$ is the noiseless channel output when the finite-state machine corresponding to the ISI channel transitions from state i to state j. Now, for each t form the two vectors of size $2^I \times 1$

$$\underline{\alpha}_{W,t}^{(\ell)} = \left[2^{-I},\, 2^{-I},\, \ldots,\, 2^{-I}\right]^{\mathrm{T}}$$
$$\underline{\beta}_{W,t}^{(\ell)} = \left[2^{-I},\, 2^{-I},\, \ldots,\, 2^{-I}\right]^{\mathrm{T}}.$$

For every t, compute

$$\underline{\alpha}_{0,t}^{(\ell)} = \left(\boldsymbol{D}_{t-1}^{(\ell)}\boldsymbol{T}_{t-1}^{\mathrm{T}}\right)\left(\boldsymbol{D}_{t-2}^{(\ell)}\boldsymbol{T}_{t-2}^{\mathrm{T}}\right) \cdots \left(\boldsymbol{D}_{t-W}^{(\ell)}\boldsymbol{T}_{t-W}^{\mathrm{T}}\right)\underline{\alpha}_{W,t}^{(\ell)} \tag{52}$$

$$\underline{\beta}_{0,t}^{(\ell)} = \left(\boldsymbol{D}_{t+1}^{(\ell)}\boldsymbol{T}_{t+1}^{\mathrm{T}}\right)^{\mathrm{T}}\left(\boldsymbol{D}_{t+2}^{(\ell)}\boldsymbol{T}_{t+2}^{\mathrm{T}}\right)^{\mathrm{T}} \cdots \left(\boldsymbol{D}_{t+W}^{(\ell)}\boldsymbol{T}_{t+W}^{\mathrm{T}}\right)^{\mathrm{T}}\underline{\beta}_{W,t}^{(\ell)}. \tag{53}$$

For each t, compute the vector $\underline{b}_t^{(\ell)}$ as

$$\underline{b}_t^{(\ell)} = \underline{\beta}_{0,t}^{(\ell)} \odot \left(\boldsymbol{T}_t^{\mathrm{T}} \cdot \underline{\alpha}_{0,t}^{(\ell)}\right) \tag{54}$$

where $\odot$ denotes the Hadamard (i.e., element-wise) product of two vectors. Denote by $\sum_{-1} \underline{b}_t^{(\ell)}$ the sum of the elements of $\underline{b}_t^{(\ell)}$ that correspond to the states that are reached if the channel input is -1, i.e., the element $b_t^{(\ell)}(i)$ is included in the sum if state i is reached when the channel input is -1. Similarly, denote by $\sum_{1} \underline{b}_t^{(\ell)}$ the sum of the elements of $\underline{b}_t^{(\ell)}$ that correspond to the states that are reached if the channel input is 1. Then, the message to the code portion of the graph is computed as

$$\mathcal{O}_t^{(\ell)} = \ln \frac{\sum\limits_{1} \underline{b}_t^{(\ell)}}{\sum\limits_{-1} \underline{b}_t^{(\ell)}}.$$

The windowed BCJR algorithm described in this appendix is not the most economical method (in terms of memory). Our aim was to give a compact description for completeness of the text. In practice, to achieve a numerically stable method, the multiplications in (52) and (53) need to be normalized such that the vector obtained by successive multiplication from the left all have the property that the sum of their elements equal to 1 [8]. For other implementations of the windowed BCJR algorithm, see [46]–[48].

ACKNOWLEDGMENT

The authors would like to thank Sae-Young Chung for extremely helpful discussions on implementations of the density evolution algorithm, Dieter Arnold for providing the latest results on capacity computations long before they appeared in print, and Xiaowei Jin and Nedeljko Varnica for helping with the simulations of Gallager codes of finite block lengths.

REFERENCES

[1] R. G. Gallager, *Information Theory and Reliable Communication*. New York: Wiley, 1968.
[2] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. New York: Wiley, 1991.
[3] K. A. S. Immink, P. H. Siegel, and J. K. Wolf, "Codes for digital recorders," *IEEE Trans. Inform. Theory*, vol. 44, pp. 2260–2299, Oct. 1998.
[4] J. G. Proakis, *Digital Communications*, 4th ed. New York: McGraw-Hill, 2000.
[5] W. Hirt, "Capacity and information rates of discrete-time channels with memory," Ph.D. dissertation, Swiss Federal Institute of Technology (ETH), Zurich, Switzerland, 1988.
[6] S. Shamai (Shitz), L. H. Ozarow, and A. D. Wyner, "Information rates for a discrete-time Gaussian channel with intersymbol interference and stationary inputs," *IEEE Trans. Inform. Theory*, vol. 37, pp. 1527–1539, Nov. 1991.
[7] S. Shamai (Shitz) and R. Laroia, "The intersymbol interference channel: Lower bounds on capacity and channel precoding loss," *IEEE Trans. Inform. Theory*, vol. 42, pp. 1388–1404, Sept. 1996.
[8] 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. IT-20, pp. 284–287, Sept. 1974.
[9] D. Arnold and H.-A. Loeliger, "On the information rate of binary-input channels with memory," in *Proc. IEEE Int. Conf. Communications 2001*, Helsinki, Finland, June 2001.
[10] H. D. Pfister, J. B. Soriaga, and P. H. Siegel, "On the achievable information rates of finite state ISI channels," in *Proc. IEEE Global Communications Conf. 2001*, San Antonio, TX, Nov. 2001, pp. 2992–2996.
[11] A. Kavčić, "On the capacity of Markov sources over noisy channels," in *Proc. IEEE Global Communications Conf.*, San Antonio, TX, Nov. 2001, pp. 2997–3001.
[12] P. Vontobel and D. M. Arnold, "An upper bound on the capacity of channels with memory and constraint input," in *Proc. IEEE Information Theory Workshop*, Cairns, Australia, Sept. 2001.
[13] S. Yang and A. Kavčić, "Markov sources achieve the feedback capacity of finite-state machine channels," in *Proc. IEEE Int. Symp. Information Theory*, Lausanne, Switzerland, July 2002, p. 361.
[14] M. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. Spielman, "Improved low-density parity-check codes using irregular graphs," *IEEE Trans. Inform. Theory*, vol. 47, pp. 585–598, Feb. 2001. Also in *Proc. 30th Annu. ACM Symp. Theory of Computing*, pp. 249–258, 1998.

[15] T. Richardson, A. Shokrollahi, and R. Urbanke, "Design of capacity-approaching low-density parity-check codes," *IEEE Trans. Inform. Theory*, vol. 47, pp. 619–637, Feb. 2001.
[16] R. G. Gallager, *Low-Density Parity-Check Codes*. Cambridge, MA: MIT Press, 1962.
[17] B. M. Tanner, "A recursive approach to low complexity codes," *IEEE Trans. Inform. Theory*, vol. IT-27, pp. 533–547, Sept. 1981.
[18] N. Wiberg, H.-A. Loeliger, and R. Kötter, "Codes and iterative decoding on general graphs," *Europ. Trans. Commun.*, vol. 6, pp. 513–526, Sept. 1995.
[19] G. D. Forney, Jr., "Codes on graphs: Normal realizations," *IEEE Trans. Inform. Theory*, vol. 47, pp. 520–548, Feb. 2001.
[20] D. J. C. MacKay and R. M. Neal, "Near Shannon limit performance of low-density parity-check codes," *Electron. Lett.*, vol. 32, pp. 1645–1646, 1996.
[21] D. J. C. MacKay, "Good error-correcting codes based on very sparse matrices," *IEEE Trans. Inform. Theory*, vol. 45, pp. 399–431, Mar. 1999.
[22] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo-codes," in *Proc. IEEE Int. Conf. Communications*, Geneva, Switzerland, May 1993, pp. 1064–1070.
[23] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. A. Spielman, "Improved low density parity check codes using irregular graphs and belief propagation," in *Proc. IEEE Int. Symp. Information Theory*, Cambridge, MA, Aug. 1998, p. 117.
[24] M. Luby, M. Mitzenmacher, and M. A. Shokrollahi, "Analysis of random processes via and-or tree evaluation," in *Proc. 9th Annu. ACM-SIAM Symp. Discrete Algorithms*, 1998, pp. 364–373.
[25] T. Richardson and R. Urbanke, "The capacity of low-density parity check codes under message-passing decoding," *IEEE Trans. Inform. Theory*, vol. 47, pp. 599–618, Feb. 2001.
[26] ——, "Thresholds for turbo codes," in *Proc. IEEE Int. Symp. Information Theory*, Sorrento, Italy, June 2000, p. 317.
[27] S.-Y. Chung, G. D. Forney, Jr., T. Richardson, and R. Urbanke, "On the design of low-density parity-check codes within 0.0045 dB of the Shannon limit," *IEEE Commun. Lett.*, vol. 5, pp. 58–60, Feb. 2001.
[28] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Trans. Inform. Theory*, vol. 47, pp. 498–519, Feb. 2001.
[29] G. D. Forney, Jr., "Maximum-likelihood sequence estimation of digital sequences in the presence of intersymbol interference," *IEEE Trans. Inform. Theory*, vol. IT-18, pp. 363–378, Mar. 1972.
[30] J. Hagenauer and P. Hoeher, "A Viterbi algorithm with soft-decision outputs and its applications," in *Proc. IEEE Global Telecommunications Conf.*, Dallas, TX, Nov. 1989, pp. 1680–1686.
[31] Y. Li, B. Vucetic, and Y. Sato, "Optimum soft-output detection for channels with intersymbol interference," *IEEE Trans. Inform. Theory*, vol. 41, pp. 704–713, May 1995.
[32] M. P. C. Fossorier, F. Burkert, S. Lin, and J. Hagenauer, "On the equivalence between SOVA and max-log-MAP decodings," *IEEE Commun. Lett.*, vol. 2, pp. 137–139, May 1998.
[33] Z.-N. Wu and J. M. Cioffi, "Low-complexity iterative decoding with decision-aided equalization for magnetic recording channels," *IEEE J. Select. Areas Commun.*, vol. 19, pp. 699–708, Apr. 2001.
[34] M. Tüchler, R. Kötter, and A. Singer, "Iterative correction of ISI via equalization and decoding with priors," in *Proc. IEEE Int. Symp. Information Theory*, Sorrento, Italy, June 2000, p. 100.
[35] J. Pearl, *Probabilistic Reasoning and Intelligent Systems*. San Mateo, CA: Morgan Kaufmann, 1988.
[36] R. J. McEliece, D. J. C. MacKay, and J.-F. Cheng, "Turbo decoding as an instance of Pearl's belief propagation algorithm," *IEEE J. Select. Areas Commun.*, vol. 16, pp. 140–152, Feb. 1998.
[37] J. Douillard, M. Jezequel, C. Berrou, A. Picart, P. Didier, and A. Glavieux, "Iterative correction of intersymbol interference: Turbo-equalization," *Europ. Trans. Commun.*, vol. 6, pp. 507–511, Sept. 1995.
[38] J. Fan, A. Friedmann, E. Kurtas, and S. McLaughlin, "Low density parity check codes for magnetic recording," in *Proc. Allerton Conf. Communications and Control*, 1999.
[39] D. J. C. MacKay. Gallager codes—Recent results. [Online]. Available: http://wol.ra.phy.cam.ac.uk/.
[40] R. Motwani and P. Raghavan, *Randomized Algorithms*. Cambridge, U.K.: Cambridge Univ. Press, 1995.
[41] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*. Englewoods Cliffs, NJ: Prentice-Hall, 1989.
[42] A. Lapidoth and E. I. Telatar, "The compound channel capacity of a class of finite-state channels," *IEEE Trans. Inform. Theory*, vol. 44, pp. 973–983, May 1998.
[43] R. Karabed and P. H. Siegel, "Matched spectral-null codes for partial response channels," *IEEE Trans. Inform. Theory*, vol. 37, pp. 818–855, May 1991.
[44] N. Varnica and A. Kavčić, "Optimized LDPC codes for partial response channels," in *Proc. IEEE Int. Symp. Information Theory*, Lausanne, Switzerland, July 2002, p. 197.
[45] T. Souvignier, A. Friedmann, M. Öberg, P. Siegel, R. E. Swanson, and J. K. Wolf, "Turbo codes for PR4: Parallel versus serial concatenation," in *Proc. IEEE Int. Conf. Communications*, Vancouver, BC, Canada, June 1999, pp. 1638–1642.
[46] S. Benedetto, D. Divsalaar, G. Montorsi, and F. Pollaara, "Algorithm for continuous decoding of turbo codes," *Electron. Lett.*, vol. 32, pp. 314–315, Feb. 1996.
[47] A. Viterbi, "An intuitive justification and a simplified implementation of the MAP decoder for convolutional codes," *IEEE J. Select. Areas Commun.*, vol. 16, pp. 260–264, Feb. 1998.
[48] B. Bai, X. Ma, and X. Wang, "Novel algorithm for continuous decoding of turbo codes," in *IEE Proc. Commun.*, vol. 46, Oct. 1999, pp. 314–315.