Error Floor Approximation for LDPC Codes in the AWGN Channel

Brian K. Butler, *Senior Member, IEEE*, and Paul H. Siegel, *Fellow, IEEE*

***Abstract*— This paper addresses the prediction of error floors of low-density parity-check codes transmitted over the additive white Gaussian noise channel. Using a linear state-space model to estimate the behavior of the sum-product algorithm (SPA) decoder in the vicinity of trapping sets (TSs), we study the performance of the SPA decoder in the log-likelihood ratio (LLR) domain as a function of the LLR saturation level. When applied to several widely studied codes, the model accurately predicts a significant decrease in the error floor as the saturation level is allowed to increase. For nonsaturating decoders, however, we find that the state-space model breaks down after a small number of iterations due to the strong correlation of LLR messages. We then revisit Richardson's importance-sampling methodology for estimating error floors due to TSs when those floors are too low for Monte Carlo simulation. We propose modifications that account for the behavior of a nonsaturating decoder and present the resulting error floor estimates for the Margulis code. These estimates are much lower, significantly steeper, and more sensitive to iteration count than those previously reported.**

***Index Terms*— Absorbing set, belief propagation (BP), error floor, linear analysis, low-density parity-check (LDPC) code, Margulis code, near-codeword, sum-product algorithm (SPA) decoding, trapping set.**

I. Introduction

A very important class of modern codes, the class of low-density parity-check (LDPC) codes, was first published by Gallager in 1962 [1], [2]. LDPC codes are linear block codes described by a sparse parity-check matrix. Decoding algorithms for LDPC codes are generally iterative. The renaissance of interest in these codes began with work by MacKay and Neal [3] and Wiberg *et al.* [4], [5] in the late 1990s. Progress has been rapid, with information-theoretic channel capacity essentially reached for some channel models. Additionally, standardization has been completed for commercial applications of LDPC codes, *e.g.*, DVB-S2 for video broadcast via satellite and IEEE 802.3an for 10 Gbit/s Ethernet.

As a function of channel quality, the error rate performance of an LDPC code under iterative decoding is typically divided into two regions. The first region, called the *waterfall* region, occurs at poorer channel quality, close to the decoding threshold, and is characterized by a rapid drop in error rate as channel quality improves. The second region, called the *error floor* region, is the focus of the present paper. The error floor appears at higher channel quality and is characterized by a more gradual decrease in error rate as channel quality improves. For message-passing iterative decoders operating on a Tanner graph representing the code, the error floor is largely attributed to the effect of small structures, often referred to as trapping sets (TSs), within the Tanner graph. The understanding of the LDPC code error floor has progressed significantly during the past 15 years, but a number of important challenges, such as the accurate prediction of the error floor for an arbitrary LDPC code, remain.

For the binary erasure channel (BEC), it is well known that graphical structures, known as *stopping sets*, limit the performance of message-passing iterative decoders as channel conditions improve [6]. These sets have a simple combinatorial description and can be enumerated to accurately predict the observed error floors.

For other memoryless channels, the cause of error floors is less well understood, and there is no simple method for predicting the channel quality and error-rate level of their onset. This situation has slowed the adoption of LDPC codes in certain applications, such as magnetic recording. In other examples, such as the DVB-S2 standard, an additional layer of coding has been used to ensure a low error floor.

For Margulis-type codes, MacKay and Postol found that the error floors observed when transmitting over the additive white Gaussian noise (AWGN) channel with sum-product algorithm (SPA) decoding were associated with substructures in the Tanner graph that they called *near-codewords* [7]. Shortly thereafter, Richardson wrote a seminal paper on the error floors of memoryless channels [8]. For a specified decoder operating on a Tanner graph, he used the term *trapping sets* (TSs) to denote the sets of variable nodes responsible for decoding failures in the error floor region. Both near-codewords and TSs are characterized by a pair of parameters, (a, b), associated with their corresponding subgraphs, where a is the number of variable nodes and b is the number of odd-degree check nodes. The (a, b) parameters of error-floor-causing structures are typically small.

For the binary symmetric channel (BSC), Nguyen *et al.* [9], presented LDPC code construction techniques to avoid the

Manuscript received February 14, 2012; revised December 26, 2013; accepted February 4, 2014. Date of publication October 17, 2014; date of current version November 18, 2014. This work was supported in part by the Center for Magnetic Recording Research, University of California at San Diego, La Jolla, CA, USA, and in part by the National Science Foundation under Grant CCF-0829865 and Grant CCF-1116739. This paper was presented at the 49th Annual Allerton Conference on Communication, Control, and Computing in 2011.

B. K. Butler was with the Department of Electrical and Computer Engineering, University of California at San Diego, La Jolla, CA 92093 USA. He is now an independent Consultant (e-mail: butler@ieee.org).

P. H. Siegel is with the Department of Electrical and Computer Engineering, University of California at San Diego, La Jolla, CA 92093 USA (e-mail: psiegel@ucsd.edu).

Communicated by P. O. Vontobel, Associate Editor for Coding Techniques.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIT.2014.2363832

most harmful TSs they identified in codes with variable-node degree $d_v = 3$. Further work on TSs includes algorithms to enumerate the occurrence of small TSs in specific codes [10]–[14] and the characterization of TS parameters in random ensembles of LDPC codes [15], [16]. Dolecek *et al.* used hardware-oriented SPA decoders to study very low error floors and discovered an elegant combinatorial description for the TSs dominating the performance in the error floor region, which they called *absorbing sets* [17].

Several other papers have proposed techniques to combat error floors. With respect to these, we note that [18] slows decoder convergence using averaged decoding, [19] selectively biases the messages from check nodes, [20] employs informed scheduling of nodes for updating, [21] adds check equations to the parity-check matrix, and [22] replaces traditional SPA with a "difference-map" message-passing algorithm. With similar objectives, [23] examined several techniques, including a bi-mode decoder which adds the steps of erasing suspect symbols followed by erasure correction, a bit-pinning decoder which utilizes an outer algebraic code, and several generalized-LDPC decoders. Several authors found empirically that increasing the saturation level of the log-likelihood ratios (LLRs) in the SPA decoder lowers the error floor [24]–[27].

In [8], Richardson emphasized error-floor analysis techniques for the AWGN channel. He detailed a methodology—a variant of importance sampling—to estimate a TS's impact on the error rate. His technique was shown to be accurate at predicting the error floors when the TSs were known. Roughly speaking, error floors can be measured down to frame error rates of about 10^{-8} in Monte Carlo computer simulation and about 10^{-10} in hardware simulation, depending on code complexity and the computational resources available. Richardson's method allows us to reach orders of magnitude lower in characterizing the error floor.

Following the pioneering work of Richardson, several authors have proposed analytical techniques to predict error floors for the AWGN channel. In his Ph.D. thesis [28], Sun developed a model of elementary TS behavior based on a linear state-space model of decoder dynamics on the TS, using density evolution (DE) to approximate decoder behavior outside of the TS [28], [29]. This work has gained little attention, even though it reaches a conclusion contrary to the prevailing view. Specifically, Sun's model shows no error floor for elementary TSs (excluding codewords) in regular, infinite-length LDPC codes if the variable-degree of the code is at least three and the magnitude of the decoder's LLRs are allowed to grow very large. Sun is able to make similar claims for irregular LDPC codes under certain conditions. In these cases, Sun shows that the graph outside of the TS will eventually correct the TS errors; in his example, using a $(3, 1)$ TS, this correction became likely at a mean magnitude of LLRs of about 10^{10} after about 40 iterations.

Independently from Sun, Schlegel and Zhang [11] developed an improved state-space model that incorporated time-variant gains and channel errors from outside of the specified TS. They applied the model to the dominant $(8, 8)$ TS of the IEEE 802.3an code, and then compared the analytically predicted error floors to those found by FPGA-based simulation and importance sampling. Their results indicated that errors external to the TS have very little impact and that the error floor can be predicted quite accurately.

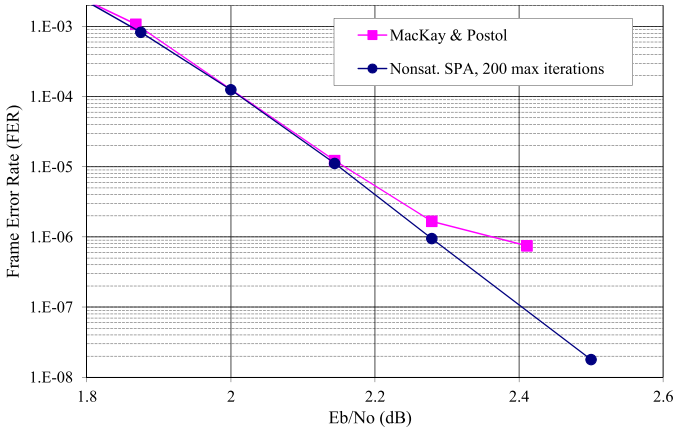


Fig. 1. FER vs. E_b/N_0 for $(2640, 1320)$ Margulis LDPC code in AWGN.

Brevik and O'Sullivan developed a new mathematical description of decoder convergence on elementary TSs using a model of the probability-domain SPA decoder [30]. While considering just the channel inputs, their model finds regions of convergence to the correct codeword, regions of convergence to the TS, and regions of non-convergence typified by an oscillatory behavior. Finally, we note that Xiao *et al.* [31] have developed a technique to estimate the error floor of quantized soft-decision decoders and applied it successfully to a variety of codes over the binary-input AWGN channel.

In this paper, we investigate the effect of LLR saturation upon the error floors of LDPC codes with SPA decoding over the AWGN channel. The original motivation for this study was the observation that simulated error floors can be lowered quite dramatically, sometimes by several orders of magnitude, when care is taken to implement the SPA decoder in such a way that LLR saturation is avoided. (Recent independent work by Zhang and Schlegel confirms this observation [32].)

Fig. 1 illustrates the improvement in performance that nonsaturating decoders can provide. The curves represent the frame error rate (FER) reported in [7] for the $(2640, 1320)$ Margulis code and the corresponding results obtained from our simulation of a nonsaturating SPA decoder. The error floor found in [7] starts at an FER of about 10^{-6} with an E_b/N_0 of 2.4 dB. The dominant errors corresponded to near-codewords. (It should be noted that others have reported even higher error floors for this code, with an FER of 10^{-6} appearing at an E_b/N_0 of 2.8 dB in [8] and [23].) Our simulation, on the other hand, shows no evidence of an error floor. The lowest simulated point at an FER of 1.8×10^{-8} represents 154 error events observed in 8600 hours of floating-point simulation, with a maximum of 200 iterations per decoded frame. Just one of the error events was a $(14, 4)$ near-codeword, one of two structures identified in previous studies as the dominant causes of errors in the error floor region. Thus, we see very little indication that an error floor is developing, suggesting that any error floor would manifest itself only at a substantially lower FER.

Our investigation relies upon a linear state-space model of SPA decoder behavior on elementary TSs. This model, which is an extension of the models introduced in Sun [28], [29] and Schlegel and Zhang [11], is used to study the effect of LLR saturation on the error floors of binary LDPC codes with fixed variable-node degree over the AWGN channel. The dynamics of the model shed light on the empirically observed error floor properties described above. More specifically, we compare analytically predicted error performance to the results of Monte Carlo computer simulation for four specific LDPC codes, and we find that the model estimates the error floor quite accurately for three of the four codes.

On the other hand, we show that the linear state-space model does not accurately reflect the error-rate performance of a nonsaturating SPA decoder in the case of a finite-length code. In particular, we observe that, after several iterations, the LLR distributions used by the model diverge from those found in simulation due to strongly correlated LLRs with large variances at unsatisfied check nodes. We surmise that, in contrast, the model performs fairly well for saturating decoders because LLR saturation drives the LLR variances at unsatisfied check nodes to zero before reaching the point in the iterative decoding where the LLR correlations become very significant.

As an alternative to the linear system approach, we propose a methodology for estimating the error floor of a nonsaturating SPA decoder using a variation on Richardson's importance-sampling technique. When applied to the Margulis code, the methodology yields error floor estimates that are much lower, significantly steeper, and more sensitive to iteration count than those previously reported.

Section II introduces the general terminology related to graphs, SPA decoders, and TSs. In particular, we propose an implementation of the SPA decoding algorithm that avoids message saturation without suffering from numerical overflow problems, which we use to provide our performance benchmarks. Section III develops the state-space model of decoder dynamics on a TS that we will be using for the analytical results of this paper. It also develops approximations to the dominant eigenvalue required to characterize the system, using connections to graph theory. Section IV develops the probability that the state-space model fails to converge to the correct decoding solution and the related error-rate estimates for the error floor. Section V demonstrates the accuracy of the state-space model's predictions of saturated decoder performance by comparing them to simulation results for four LDPC codes. Section VI discusses the use of the Gaussian approximation for DE to model the unsatisfied-check LLR values to determine error floor bounds of nonsaturating decoders. Section VII applies a modified version of Richardson's technique to estimate the error floor of a nonsaturating SPA decoder. Finally, in Section VIII, we draw our conclusions.

Appendix A presents a numerical survey of the model parameters for several large classes of potential absorbing sets. Appendix B develops a special case in support of the state-space model, while Appendix C extends the model's applicability.

II. Preliminaries

This section provides the background material necessary to present the main results. We assume that the reader is familiar with linear algebra, including the fundamental results of Perron–Frobenius theory of nonnegative matrices [33]–[36]. Section II-A presents terminology from graph theory and bounds on the spectral radii of multigraphs and digraphs. In Section II-B, we introduce LDPC codes, the AWGN channel, and SPA decoding. Section II-C describes the LLR saturation issues commonly encountered in SPA decoding. The final subsection defines TSs and absorbing sets. Note that, throughout the paper, we present vectors as column vectors.

A. General Graph Theory

An *undirected graph* $G = (V, E)$ consists of a finite set of vertices V and a finite collection of edges E. Each edge is an unordered pair of vertices $\{v_i, v_j\}$ such that the edge *joins* vertices v_i and v_j. Given the edge $\{v_i, v_j\}$, we say that vertex v_i and vertex v_j are *neighbors*. We use $\mathcal{N}(v_i)$ to denote the set of vertices which are the neighbors of vertex v_i. A vertex and an edge are *incident* with one another if the vertex is contained in the edge description.

A *self-loop* is an edge joining a vertex to itself. *Parallel edges* are multiple inclusions of an edge in the edge collection. We do not give further consideration to graphs with self-loops. Loopless graphs are commonly known as *multigraphs*, which may contain parallel edges.

The *order* of a graph is the number of vertices and the *size* is the number of edges. The degree $d(v_i)$ of vertex v_i is the number of edges incident with v_i. A *regular* graph is a graph whose vertices are all of equal degree.

In an undirected graph G, a *walk* between two vertices is an alternating sequence of incident vertices and edges. The vertices and edges in a walk need not be distinct. The number of edges in a walk is its *length*. The vertices v_i and v_j are said to be *connected* if the graph contains a walk of any length from v_i to v_j, noting that every vertex is considered connected to itself. A graph is said to be *connected* if every pair of vertices is connected, otherwise the graph is said to be *disconnected*. The vertices of a disconnected graph may be partitioned into connected components.

An edge is called *unique* if it appears only once in the edge collection of the graph. A walk that *backtracks* is a walk in which a unique edge appears twice or more in-a-row in the walk.

A *closed walk* is a walk that begins and ends on the same vertex. A *cycle* is a closed walk of length at least two with no repeated edges or vertices (except the initial and final vertex). The *girth* of a graph with cycles is the length of its shortest cycle. If every vertex in a multigraph has at least degree two, then the graph contains one or more cycles.

A *leaf* is a vertex of degree one. A *tree* is a connected graph without cycles. Trees with at least two vertices have leaves. We call an undirected graph *leafless* if it does not contain leaves.

We will say that two graphs are *identical* if they have equal vertex sets and equal edge collections. Two graphs are said

to be *isomorphic* if there exists between their vertex sets a one-to-one correspondence having the property that whenever two vertices are joined by k edges in one graph, the corresponding two vertices are joined by k edges in the other graph. This correspondence relationship is called an *isomorphism*. The isomorphism is a relabeling of the vertices that preserves the graph's structure.

The *adjacency matrix* $\mathbf{A}(G)$ of any multigraph G of order n is the $n \times n$ symmetric matrix whose (i, j) entry indicates the number of edges joining v_i and v_j in G. The number of walks of length p between vertices v_i and v_j, in the graph G, is the (i, j) entry of $\mathbf{A}(G)^p$. Two graphs, G_1 and G_2, are isomorphic if and only if $\mathbf{A}(G_1)$ is a symmetric permutation of $\mathbf{A}(G_2)$, *i.e.*, $\mathbf{A}(G_1) = \mathbf{P}\,\mathbf{A}(G_2)\mathbf{P}^T$, for some permutation matrix $\mathbf{P}$ [34], [37].

The *incidence matrix* $\mathbf{N}(G)$ of any multigraph G of order n and size m is the $n \times m$ $(0, 1)$-matrix whose (i, j) entry is 1 if and only if the ith vertex is incident with the jth edge of G. It then follows that the multigraph's adjacency matrix may be expressed as

$$\mathbf{A}(G) = \mathbf{N}(G)\mathbf{N}(G)^T - \mathbf{D}_G, \tag{1}$$

where $\mathbf{D}_G = \mathrm{diag}\,(d(v_1), d(v_2), \ldots, d(v_n))$.

A *bipartite graph* $B = (V, C, E)$ is a special case of an undirected graph in which the graph's vertices may be partitioned into two disjoint sets V and C. Each edge $e \in E$ of a bipartite graph joins a vertex from V to a vertex from C.

The *line graph* $\mathcal{L}(G)$ of a multigraph $G = (V, E)$ is the graph whose vertices are the edges of G. Two vertices of $\mathcal{L}(G)$ are neighbors if and only if their corresponding edges in G have a vertex (or two) in common. In fact, two parallel edges connect vertices in $\mathcal{L}(G)$ if and only if their corresponding edges in G are parallel. For the line graph $\mathcal{L}(G)$, we find

$$\mathbf{A}(\mathcal{L}(G)) = \mathbf{N}(G)^T\mathbf{N}(G) - 2\,\mathbf{I} \quad \text{and} \tag{2}$$

$$\mathrm{size}\,(\mathcal{L}(G)) = \frac{1}{2}\sum_{v_i \in V} d(v_i)\,(d(v_i) - 1)\,. \tag{3}$$

The *spectral radius* $\rho(G)$ of a multigraph $G = (V, E)$ is defined to be the spectral radius[1] of the matrix $\mathbf{A}(G)$, and it is bounded by

$$\frac{1}{|V|}\sum_{v_i \in V} d(v_i) \le \rho(G) \le \max_{v_i \in V} d(v_i). \tag{4}$$

Note that $d(v_i)$ is just the ith row (or column) sum of $\mathbf{A}(G)$. Since the matrix $\mathbf{A}(G)$ is symmetric, the lower bound follows by applying the Rayleigh quotient to $\mathbf{A}(G)$ with an all-one vector [33, pp. 176–181]. The upper bound, which follows from the fact that the matrix $\mathbf{A}(G)$ is nonnegative, is due to Frobenius [33, p. 492]. The lower bound holds with equality if and only if the all-one vector is an eigenvector of $\mathbf{A}(G)$ corresponding to the eigenvalue $\rho(G)$. If G is connected, then the upper bound holds with equality if and only if G is regular. When G is connected these conditions are equivalent.

[1]The *spectral radius* $\rho(\mathbf{M})$ of the matrix $\mathbf{M}$ is the maximum modulus of the eigenvalues of $\mathbf{M}$. When $\mathbf{M}$ is a nonnegative matrix, $\rho(\mathbf{M})$ itself is an eigenvalue and $\rho(\mathbf{M})$ may also be referred to as the "dominant eigenvalue."

A *directed graph* or *digraph* $D = (Z, A)$ consists of a set of vertices Z and a collection of directed edges or *arcs* A. Each arc is an ordered pair of vertices (z_i, z_j) such that the arc is directed from vertex z_i to vertex z_j. For arc (z_i, z_j), we call the first vertex z_i its *initial vertex* and the second vertex z_j its *terminal vertex*. We will focus on *simple digraphs* which exclude self-loops and parallel arcs.

In a digraph D, a *directed walk* is an alternating sequence of vertices and arcs from z_i to z_j in D such that every arc a_k in the sequence is preceded by its initial vertex and is followed by its terminal vertex. A digraph D is said to be *strongly connected* if for any ordered pair of distinct vertices z_i and z_j there is a directed walk in D from z_i to z_j. For example, the digraph in Fig. 4d is strongly connected.

The *adjacency matrix* $\mathbf{A}(D)$ of any simple digraph D is the $(0, 1)$-matrix whose (i, j) entry is 1 if and only if (z_i, z_j) is an arc of D.

The *outdegree* $d^+(z_i)$ of vertex z_i is the number of arcs in digraph D with initial vertex z_i. Likewise, the *indegree* $d^-(z_j)$ of vertex z_j is the number of arcs with terminal vertex z_j. For the digraph $D = (Z, A)$, we note that $\mathrm{size}\,(D) = \sum_{z_i \in Z} d^+(z_i)$. A *regular* digraph is a digraph whose vertices are all of equal indegree and outdegree. The *spectral radius* $\rho(D)$ of a digraph D is defined to be $\rho(\mathbf{A}(D))$, and it is bounded in the following lemma.

Lemma 1: Let the ith vertex z_i of digraph $D = (Z, A)$ have outdegree $d^+(z_i)$. Then the spectral radius $\rho(D)$ of D is bounded above and below by

$$\min_{z_i \in Z} d^+(z_i) \le \rho(D) \le \max_{z_i \in Z} d^+(z_i).$$

If D is strongly connected, then the inequalities are strict unless the digraph has regular outdegree. Analogous statements hold for the indegrees.

Proof: This follows from the classic bounds of Frobenius by merely noting the equivalence between the outdegrees of a digraph and the row-sums of the associated adjacency matrix. See [33, p. 492] and [35, pp. 8 and 22]. ■

The interested reader is referred to [36]–[39] for a more complete treatment of the subject. Our use of graph theory has parallels to [30].

B. LDPC Codes, the AWGN Channel, and SPA Decoding

An LDPC code $\mathcal{C}$ is defined by the null space of a parity-check matrix $\mathbf{H}$, whose entries are elements of a particular field. The column vector $\mathbf{c}$ is a codeword of $\mathcal{C}$, if $\mathbf{c}$ satisfies $\mathbf{Hc} = \mathbf{0}$. A given code can be described by many different parity-check matrices.

A matrix $\mathbf{H}$ over $\mathbb{F}_2$ describes a bipartite graph $B = (V, C, E)$, called the *Tanner graph* of $\mathbf{H}$, in which the vertices are known as variable nodes V and check nodes C. Tanner graphs of binary codes do not have parallel edges and bipartite graphs cannot have self-loops. The adjacency matrix $\mathbf{A}(B)$ of the Tanner graph B is

$$\mathbf{A}(B) = \begin{bmatrix} \mathbf{0} & \mathbf{H}^T \\ \mathbf{H} & \mathbf{0} \end{bmatrix}.$$

A d_v-*variable-regular* Tanner graph is a Tanner graph whose variable nodes all have equal degree d_v, and a d_c-*check-regular* Tanner graph is a Tanner graph whose check nodes all have equal degree d_c. A (d_v, d_c)-*regular* Tanner graph is both d_v-variable-regular and d_c-check-regular.

Given the Tanner graph $B = (V, C, E)$ and any subset of its variable nodes $\mathcal{S} \subseteq V$, we let $B_{\mathcal{S}}$ represent the *induced subgraph* of $\mathcal{S}$ and $\mathcal{N}(\mathcal{S})$ denote the union of check nodes which are neighbors of the nodes of $\mathcal{S}$. That is, $B_{\mathcal{S}} = (\mathcal{S}, \mathcal{N}(\mathcal{S}), E_{\mathcal{S}})$ is the bipartite graph containing the variable-node set $\mathcal{S}$, the check-node set $\mathcal{N}(\mathcal{S})$, and the set of edges $E_{\mathcal{S}} \subseteq E$ which are incident with the vertices of $\mathcal{S}$. We will frequently refer to these induced subgraphs as Tanner subgraphs with parity-check submatrix $\mathbf{H}_{\mathcal{S}}$, where the submatrix $\mathbf{H}_{\mathcal{S}}$ is formed by selecting the columns of $\mathbf{H}$ as indexed by the members of the set $\mathcal{S}$ and removing any resulting all-zero rows.

Assumption 1: We are only concerned with LDPC codes over the binary field $\mathbb{F}_2$ and with binary antipodal signaling over the AWGN channel.

If the Tanner graph had no cycles, the SPA decoder would be optimal with respect to minimizing the symbol error rate. In any "good" finite-length code, the Tanner graph will indeed have cycles [40], but we generally find that the SPA decoder does quite well. We implement our decoder simulation in the LLR-domain, as it is close to the approximations typically used in building hardware. Since we will assume that the reader is familiar with SPA decoding of LDPC codes over the AWGN channel ([41, § 5.4] and [42]), the remainder of this subsection is presented merely to establish notation.

We define the channel SNR to be $1/\sigma^2 = 2RE_b/N_0$, and we denote the intrinsic channel LLR for the ith received symbol by $\lambda^{[i]}$. During the first half of the lth iteration, the decoder computes $\lambda_l^{[i \leftarrow j]}$, the message to be sent from check node j to the neighboring variable node i, according to the check-node update rule

$$\lambda_l^{[i \leftarrow j]} = 2 \tanh^{-1} \left(\prod_{k \in \mathcal{N}(j) \setminus i} \tanh \frac{\lambda_{l-1}^{[k \rightarrow j]}}{2} \right), \tag{5}$$

where $\lambda_{l-1}^{[k \rightarrow j]}$ is the message sent from variable node k to check node j during the previous iteration. In (5), $\mathcal{N}(j) \setminus i$ denotes the set of variable nodes neighboring check node j, excluding variable node i. In the second half of the iteration, the decoder computes the variable-to-check-node messages in the usual manner,

$$\lambda_l^{[i \rightarrow j]} = \lambda^{[i]} + \sum_{k \in \mathcal{N}(i) \setminus j} \lambda_l^{[i \leftarrow k]}.$$

Finally, we use $\tilde{\lambda}_l^{[i]}$ to denote the soft-output LLR of the ith symbol, which is the sum of the intrinsic channel LLR $\lambda^{[i]}$ and all of the extrinsic LLRs $\lambda_l^{[i \leftarrow j]}$ from check nodes $j \in \mathcal{N}(i)$ at the completion of the lth iteration.

C. SPA Decoder With and Without Saturation

Direct implementation of check-node update rule (5) in a computer simulation leads to numerical problems when LLR magnitudes get sufficiently large. According to the IEEE Standard 754 [43], double-precision floating-point (*i.e.*, 64-bit) computer computations maintain 53 bits of precision, with the remaining bits used to represent the sign and exponent. As a result, the value of $\tanh(\lambda/2)$ is rounded to ± 1 whenever the LLR magnitude satisfies $|\lambda| > 55 \ln 2 \approx 38.123$. Since the evaluation of the $\tanh^{-1}$ function at an argument of ± 1 will cause a numerical overflow, one must either limit the magnitude of LLRs when computing the check-node update in (5) or find an alternative formulation of the update rule that avoids such overflow problems. In the former case, we refer to the magnitude-limiting decoder as a "saturating" SPA decoder. An examination of error floor results in the published literature suggests that LLR saturation is commonly employed, at times without explicit mention.

Hardware implementations of SPA decoders, which must represent LLR values with a limited number of bits, typically impose LLR saturation at magnitudes less than $55 \ln 2$. This leads to the situation in which hardware designs produce error floors at error-rate levels only somewhat greater than typical floating-point simulation with saturation. To our knowledge, efforts to explore beyond these saturation limits have primarily appeared only in the very recent literature, including [32], [44], and [45].

The nonsaturating SPA decoder simulation we have used is based on a pairwise check-node reduction [46], [47] which includes a small approximation [48]. Its implementation in double-precision floating-point contains no numerical issues until LLR magnitudes reach approximately 1.79×10^{308}. For a detailed examination of the numerical issues of the SPA decoder in several domains of floating-point computation, see [45].

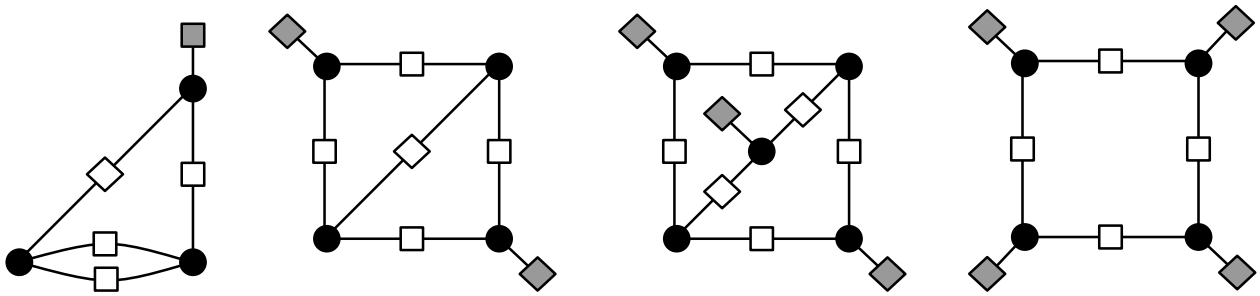
D. Trapping Sets and Absorbing Sets

We briefly review the notion of TSs and related terminology.

Definition 1 (Richardson [8]): A *trapping set* (TS) is an error-prone set of variable nodes $\mathcal{T}$ that depends upon the decoder's input space and decoding algorithm. Let $\tilde{\lambda}_l^{[i]}(\mathbf{r})$ denote the decoder's ith soft-output symbol at the lth iteration given the input vector $\mathbf{r}$. We say that symbol i is *eventually correct* if there exists L such that, for all $l \geq L$, $\tilde{\lambda}_l^{[i]}(\mathbf{r})$ has the correct sign. If the set of symbols that are not eventually correct is not the empty set, we call it a TS and denote it by $\mathcal{T}(\mathbf{r})$, or sometimes more simply by $\mathcal{T}$ when the explicit reference to the input $\mathbf{r}$ is not required. A TS $\mathcal{T}$ is called an (a, b) TS if it contains $a = |\mathcal{T}|$ variable nodes and the induced subgraph $B_{\mathcal{T}}$ contains exactly b check nodes of odd degree.

The TS definition includes the set of variable nodes corresponding to the support of a valid LDPC codeword, unlike the definition of a near-codeword. Henceforth, we will use the term *trapping set* (TS) to mean the TSs that arise from transmission over the AWGN channel and saturated LLR-domain SPA decoding.

Definition 2: A Tanner subgraph is called *elementary* if all the check nodes are of degree one or two [49]. A TS $\mathcal{T}$ is called elementary if the induced subgraph $B_{\mathcal{T}}$ is an elementary Tanner subgraph.



(a) $a = 3$, $b = 1$ (b) $a = 4$, $b = 2$ (c) $a = 5$, $b = 3$ (d) $a = 4$, $b = 4$

Fig. 2. Four elementary Tanner subgraphs of a code with $d_{\mathrm{v}} = 3$. Odd-degree check nodes are shaded.

Elementary TSs are of significant interest for two reasons. First, as observed in [8], [15], [23], and [49]–[51], the majority of TSs contributing to the error floors of belief propagation decoders are elementary TSs. Second, tractable linear state-space models can be used to model the behavior of SPA decoding on elementary TSs.

Another important class of TSs are absorbing sets. Extensive performance simulations by Dolecek *et al.* [17] have shown that absorbing sets represent the dominant errors in the error floor regions of saturating SPA decoders.

Definition 3 (Dolecek et al. [17]): Let $\mathcal{S}$ be a subset of the variable nodes V of the Tanner graph $B = (V, C, E)$ and let $\mathcal{S}$ induce a subgraph having a set of odd-degree check nodes $\mathcal{N}_{\mathrm{o}}(\mathcal{S})$ and a set of even-degree check nodes $\mathcal{N}_{\mathrm{e}}(\mathcal{S})$. If $\mathcal{S}$ has cardinality a and induces a subgraph with $|\mathcal{N}_{\mathrm{o}}(\mathcal{S})| = b$, in which each node in $\mathcal{S}$ has strictly fewer neighbors in $\mathcal{N}_{\mathrm{o}}(\mathcal{S})$ than neighbors in $\mathcal{N}_{\mathrm{e}}(\mathcal{S})$, then we say that $\mathcal{S}$ is an (a, b) *absorbing set*. If, in addition, each node in $V \setminus \mathcal{S}$ has strictly fewer neighbors in $\mathcal{N}_{\mathrm{o}}(\mathcal{S})$ than neighbors in the set $C \setminus \mathcal{N}_{\mathrm{o}}(\mathcal{S})$, then we say that $\mathcal{S}$ is an (a, b) *fully absorbing set*.

Note that the definition of the subclass of fully absorbing sets imposes conditions on variable nodes outside of the TS. It is straightforward to see that fully absorbing sets support uncorrectable error patterns when decoding with the bit-flipping algorithm [17]: if the bits corresponding to the variable nodes in the absorbing set are received in error, there are more neighboring check nodes working to reinforce than there are working to correct every incorrect value when the bit-flipping update rule is applied.

Example 1: Four elementary Tanner subgraphs are shown in Fig. 2 for a code of variable-degree three. The variable nodes in each of these graphs form an absorbing set.

III. State-Space Model

In this section we justify and refine the linear state-space model, introduced in [28] and [29], that will be used to analyze the behavior of an LLR-domain SPA decoder of an LDPC code on an elementary TS. The aim is to shed light on the dynamics responsible for incorrectly decoding the variable nodes in the TS while correctly decoding the variable nodes outside of it. Throughout the discussion we assume, without loss of generality, that the all-zero codeword was sent.

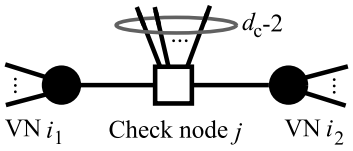


Fig. 3. Illustration of a check node of degree two in the subgraph and degree d_{c} in the Tanner graph.

Assumption 2: All TSs of interest are elementary and contain more than one variable node. We do not consider TSs containing a single variable node as they would not have states in the state-space model. (For an examination of trapping-like behavior of single degree-1 variable nodes, see [52, p. 24].)

First, we review the failure state of an elementary TS. Let the TS $\mathcal{S} \subset V$ induce the Tanner subgraph $B_{\mathcal{S}}$. In later iterations the state is reached where the variable nodes of TS $\mathcal{S}$ are in error and the variable nodes outside the TS (*i.e.*, $V \setminus \mathcal{S}$) are correct. In this state, the subgraph's check nodes of degree one are unsatisfied and those of degree two are satisfied.

There are two vector inputs to the state-space model. The elements of the input vector $\boldsymbol{\lambda}_l^{(\mathrm{ex})}$ are the messages from the degree-one check nodes at iteration l. To model this input vector, we will use density evolution and simulation techniques, described in later sections. The other input vector to be used in variable-node updates at every iteration is the intrinsic information $\boldsymbol{\lambda}$ provided by the channel. For the AWGN channel model, each element of $\boldsymbol{\lambda}$ is an independent and identically distributed (i.i.d.) Gaussian random variable. Both of these inputs will be treated as column vectors of LLRs; for an (a, b) TS, the vector $\boldsymbol{\lambda}$ has a entries and the vector $\boldsymbol{\lambda}_l^{(\mathrm{ex})}$ has b entries.

A. Check-Node Gain Model

Sun's linear state-space model included the asymptotic approximation that the output message on every edge of every degree-two (*i.e.*, ultimately satisfied) check node is equal to the input message on the other edge [28]. This relies on the assumption that LLR values sent by variable nodes outside of $B_{\mathcal{S}}$ are strongly positive LLR values, *i.e.*, they have both the correct sign and large magnitude. Schlegel and Zhang enhanced the model by applying a multiplicative gain g_l at the degree-two check nodes, at iteration l, where $0 < g_l \leq 1$ [11]. This gain factor models the effect of the $d_{\mathrm{c}} - 2$ external variable nodes upon the magnitude of LLR messages as they pass through the degree-two check nodes in $B_{\mathcal{S}}$. (The relevant

configuration of nodes and edges is illustrated in Fig. 3.) After several iterations these gains approach 1.

Assumption 3: We will assume that the external inputs to the subgraph's degree-two check nodes are statistically independent. This is generally a reasonable assumption to make as the effect of the gains is most significant during early iterations. When, after a sufficient number of iterations, the inputs to the check node have become substantially correlated, the gains are approximately unity.

We now briefly review the check-node gain model of Schlegel and Zhang [11].[2] The incorporation of the gains turns the state-space model into a state-space model which is time-varying. While we will generally include the gains in the model, we will find it useful shortly to temporarily remove the gains to apply linear time-invariant (LTI) transform theory which allows us to examine the ultimate stability of SPA decoding on TSs. Referring to Fig. 3, define the gain of the jth check node during the lth iteration to be

$$g_l^{[j]} \triangleq \prod_{k \in \mathcal{N}(j) \setminus \{i_1, i_2\}} \tanh\left(\frac{\lambda_{l-1}^{[k \to j]}}{2}\right). \tag{6}$$

The expected value of the check-node gain over all realizations of the channel noise vector $\mathbf{n}$ and all degree-d_c check nodes is denoted by $\bar{g}_l(d_c)$. By Assumption 3, we can write this as

$$\bar{g}_l(d_c) = \mathbb{E}_{\mathbf{n},j,k}\left[\tanh\left(\frac{\lambda_{l-1}^{[k \to j]}}{2}\right)\right]^{d_c-2}, \tag{7}$$

where $\mathbb{E}_{\mathbf{n},j,k}$ denotes the average over all noise realizations, all check nodes j with $d(c_j) = d_c$, and all $k \in \mathcal{N}(j)$.

The multiplicative gain model for the degree-2 check-node output can be justified by first rewriting the check-node update rule (5) as

$$\begin{aligned}\lambda_l^{[i_1 \leftarrow j]} &= 2\,\tanh^{-1}\left[\tanh\left(\frac{\lambda_{l-1}^{[i_2 \to j]}}{2}\right) g_l^{[j]}\right] \\ &\triangleq f(\lambda_{l-1}^{[i_2 \to j]}),\end{aligned} \tag{8}$$

where $g_l^{[j]}$ is as defined in (6), $\lambda_{l-1}^{[i_2 \to j]}$ is the input LLR from variable node i_2 to check node j, and $\lambda_l^{[i_1 \leftarrow j]}$ is the output LLR from check node j to variable node i_1.

The function $f(\lambda)$ in (8) satisfies

$$\left.\frac{\partial f(\lambda)}{\partial \lambda}\right|_{\lambda=0} = g_l \quad \text{and} \quad \left.\frac{\partial^2 f(\lambda)}{\partial \lambda^2}\right|_{\lambda=0} = 0,$$

so the Taylor series approximation of order 2 is given by

$$\lambda_l^{[i_1 \leftarrow j]} \approx g_l^{[j]} \lambda_{l-1}^{[i_2 \to j]},$$

i.e., the input LLR multiplied by the gain. The approximation is good in the interval $|\lambda_l^{[i_1 \leftarrow j]}| \le 2\tanh^{-1}(g_l^{[j]})$. Note that the exact expression for $\lambda_l^{[i_1 \leftarrow j]}$ in (8) saturates at $\pm 2\tanh^{-1}(g_l^{[j]})$, whereas the approximation is linear. Nevertheless, since we expect the LLRs outside of the TS to grow quite rapidly with increasing iterations, we believe this to be an adequate approximation for our purposes.

The model is further simplified by replacing each gain $g_l^{[j]}$ with the mean gain of (7), *i.e.*,

$$\lambda_l^{[i_1 \leftarrow j]} \approx \bar{g}_l(d(c_j))\, \lambda_{l-1}^{[i_2 \to j]}.$$

We have tried introducing a gain variance to the model, but found the effect to be very small.

We now extend the check-node gain model to include polarity inversions of the degree-two check-node output messages caused by erroneous LLR messages sent from variable nodes outside of the TS. Such inversions occur primarily during early decoder iterations. In reference to Fig. 3, an inversion occurs when an odd number of the $d_c - 2$ messages sent from outside of a TS along edges incident with a degree-two check node within the TS are erroneous. In such a situation, the message sent by the check node to variable node i_1 will have the inverse polarity of the message that was sent to the check node by variable node i_2, and vice versa. Schlegel and Zhang [11] accounted for inversions by injecting a stochastic cancellation signal into the state update equations, resulting in a small increase in their predicted error rates. We now present an alternative method based upon a modification of the mean check-node gains that, in contrast, produces a reduction in our predicted error rates.

During the first iteration, messages from variable nodes may contain inversions due to channel errors. The probability that a specific input message to a check node is erroneous during iteration $l = 1$ is just the uncoded symbol error rate, which for AWGN is

$$P_{e,1} = \mathrm{Q}\left(\sqrt{\frac{2RE_b}{N_0}}\right). \tag{9}$$

Thus, by Assumption 3, the probability of a polarity inversion through a specific check node is given at iteration l by

$$\begin{aligned}P_{\mathrm{inv},l} &= \sum_{k\ \mathrm{odd}} \binom{d_c-2}{k} P_{e,l}^k (1-P_{e,l})^{d_c-2-k} \\ &= \frac{1-(1-2P_{e,l})^{d_c-2}}{2},\end{aligned} \tag{10}$$

which is the probability of an odd number of errors in the $d_c - 2$ input messages at iteration l [2, p. 38].

For subsequent iterations we can use density evolution [53], [54] to predict an effective E_b/N_0 at the output of the variable nodes and then apply (9) and (10). When there is an inversion through the check node, the output message magnitude will likely be very low, as suggested in [11]. Hence, we will model random check-node inversions by randomly setting the check-node gain to zero with probability $P_{\mathrm{inv},l}$. We now define the *modified mean gain* of the check node as

$$\bar{g}_l' = \bar{g}_l(1 - P_{\mathrm{inv},l}). \tag{11}$$

For an LDPC code with regular check node degree d_c, the scalar gain $\bar{g}_l'$ of (11) may be applied at each iteration l to model the external influence on the subgraph's

[2] We correct an error in [11] related to the dependence of the gain expression on the iteration count.

degree-two check nodes. For an LDPC code with irregular check degree, we have a few options. If the check node degree spread is small, we may generalize (7) and (11) by also taking the expectation over the degree distribution of all the degree-two check nodes in the subgraph. Alternatively, we may maintain several versions of gain $\bar{g}_l'$, one for each check degree that appears in the subgraph.

Assumption 4: We assume that multiplication by the modified mean gain $\bar{g}_l'$ reflects the LLR update process at all degree-two check nodes in the subgraph induced by the TS.

One possible criticism of the proposed gain model with inversions is the potential for double-counting: first, inversions from outside the TS will reduce the check-node gain according to (7); then, the same inversions will further reduce the check-node gain according to (11). Heuristically, one may consider this as just adding weight to the significance of these inversions. We justify our approach by noting that the arguments underlying its derivation parallel those in [11] and, moreover, it is supported by the empirical results presented in Section V.

B. Linear State-Space Model

Assumption 5: We consider only codes described by variable-regular Tanner graphs with variable-node degree $d_{\mathrm{v}} \geq 3$. (Sun presents asymptotic results for irregular codes in [28].)

Assumption 6: We assume that the messages generated within the TS have little impact on decoder behavior in the remainder of the Tanner graph. This is justified by the fact that the number of unsatisfied check nodes which form the main interface between the TS and the rest of the Tanner graph is small.

The state-space modeling equations are shown below [11], [28]. The column vectors $\boldsymbol{\lambda}_l^{(\mathrm{ex})}$ and $\boldsymbol{\lambda}$, defined at the start of Section III, are the model inputs. The column vector $\tilde{\boldsymbol{\lambda}}_l$, whose entries are the LLR-domain soft-output decisions at iteration l, is the output.

$$\mathbf{x}_0 = \mathbf{B}\boldsymbol{\lambda}$$

$$\mathbf{x}_l = \bar{g}_l'\mathbf{A}\mathbf{x}_{l-1} + \mathbf{B}\boldsymbol{\lambda} + \mathbf{B}_{\mathrm{ex}}\boldsymbol{\lambda}_l^{(\mathrm{ex})} \quad \text{for } l \geq 1 \tag{12}$$

$$\tilde{\boldsymbol{\lambda}}_l = \bar{g}_l'\mathbf{C}\mathbf{x}_{l-1} + \boldsymbol{\lambda} + \mathbf{D}_{\mathrm{ex}}\boldsymbol{\lambda}_l^{(\mathrm{ex})} \quad \text{for } l \geq 1 \tag{13}$$

The central part of the state-space model is the updating of the state vector $\mathbf{x}_l$. The elements of $\mathbf{x}_l$ are the LLR messages sent along the edges from the subgraph's variable nodes toward the degree-two check nodes. The state is updated once per full iteration. The vector $\bar{g}_l'\mathbf{x}_{l-1}$ represents the LLR messages produced by the degree-two check nodes during the first half of iteration l, and $\bar{g}_l'\mathbf{A}\mathbf{x}_{l-1}$ represents their contribution to the variable-node update during the second half of iteration l.

The number of edges in the variable-regular Tanner subgraph with a variable nodes and variable-node degree d_{v} equals ad_{v}, so the number of states in the state-space model of the subgraph is $m = ad_{\mathrm{v}} - b$. For an elementary subgraph, these m edges are incident with the subgraph's degree-two check nodes, forcing m to be an even integer.

The $m \times a$ matrix $\mathbf{B}$ and the $m \times b$ matrix $\mathbf{B}_{\mathrm{ex}}$ are used to map input vectors $\boldsymbol{\lambda}$ and $\boldsymbol{\lambda}_l^{(\mathrm{ex})}$, respectively, to the appropriate entries of the state vector to match the set of variable-node update equations. Like all the matrices appearing in the state-space equations, they are $(0, 1)$-matrices. Since every variable node has exactly one element from $\boldsymbol{\lambda}$ participating, $\mathbf{B}$ has a single 1 in every row, and is zero otherwise. The number of 1 entries per row of $\mathbf{B}_{\mathrm{ex}}$ corresponds to the number of degree-one check nodes neighboring the corresponding variable node, which may be zero.

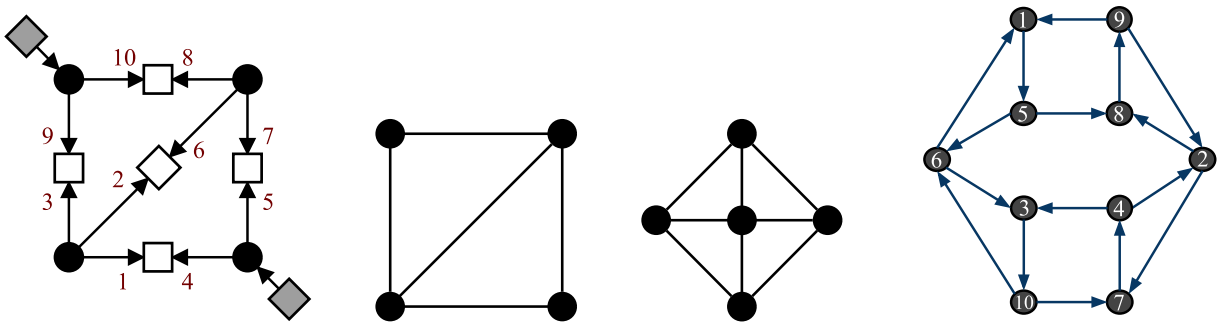
The $m \times m$ matrix $\mathbf{A}$, which we call the *system matrix*, describes the dependence of the state update upon the prior state vector. Each nonzero entry $[\mathbf{A}]_{ij} = 1$ indicates that the jth edge of the prior iteration contributes to the variable-node update computation of the ith edge. Thus, the matrix $\mathbf{A}^T$ may be taken as the adjacency matrix associated with a simple digraph of order m which describes the state-variable update relationships. For d_{v}-variable-regular codes, the ith row sum of $\mathbf{A}$ and the ith row sum of $\mathbf{B}_{\mathrm{ex}}$ will add up to $d_{\mathrm{v}} - 1$ for every row i.

Finally, the $a \times m$ *output matrix* $\mathbf{C}$ and the $a \times b$ matrix $\mathbf{D}_{\mathrm{ex}}$ are used to map $\mathbf{x}_{l-1}$ and $\boldsymbol{\lambda}_l^{(\mathrm{ex})}$ entries, respectively, to the corresponding entry of the soft-output decision vector $\tilde{\boldsymbol{\lambda}}_l$. The ith row sum of $\mathbf{C}$ and the ith row sum of $\mathbf{D}_{\mathrm{ex}}$ will add up to the variable-node degree d_{v} for every row i.

If there is a single degree-one check node neighboring every variable node in the subgraph, such as in the $(8, 8)$ TS of the code specified in IEEE 802.3an, then (12) degenerates to the case where $\mathbf{B} = \mathbf{B}_{\mathrm{ex}}$ and $\mathbf{B}$ has regular column sums as derived in [11]. Furthermore, this degenerate case produces only system matrices $\mathbf{A}$ which have uniform row and column sums which results in a dominant eigenvalue $\rho(\mathbf{A}) = d_{\mathrm{v}} - 2$. Thus, our development will be significantly more general than [11].

Again, note that this model of the behavior of elementary TSs is linear. One drawback to any linear system approach is that saturation cannot be applied to the state variables. Thus, we will model saturation effects by limiting the range of the linear system inputs.

Example 2: For the $(4, 2)$ TS with $d_{\mathrm{v}} = 3$, we first assign integer labels to all the edges incident with degree-two check nodes, as shown in Fig. 4a. Then, we set up the matrices that appear in the linear system equations. The edge numbers in Fig. 4a reflect the order in which the corresponding messages will appear in the state vector $\mathbf{x}_l$. The system matrix $\mathbf{A}$ describes the updating of the state vector in a full iteration of the SPA decoder; for example, since edge 1 depends only on edges 6 and 9 during one iteration, the entries in the first row of $\mathbf{A}$ have the value 1 in columns 6 and 9, and value 0 elsewhere. All the matrices associated with the TS of Fig. 4a are given below. Note that $\mathbf{A}^T$ is the adjacency matrix corresponding to



(a) Tanner subgraph, $B_{\mathcal{S}}$ (b) Multigraph, G (c) Line graph, $\mathcal{L}(G)$ (d) Simple digraph, D

Fig. 4. Several graphical descriptions of the (4, 2) TS with model's states numbered. Arrows have been added to the Tanner subgraph in (a) only to emphasize the direction of state variable flow. The spectral radii of G, $\mathcal{L}(G)$, and D are 2.5616, 3.2316, and 1.5214, respectively.

the simple digraph shown in Fig. 4d.

$$\mathbf{A} = \begin{bmatrix} 0&0&0&0&0&1&0&0&1&0 \\ 0&0&0&1&0&0&0&0&1&0 \\ 0&0&0&1&0&1&0&0&0&0 \\ 0&0&0&0&0&0&1&0&0&0 \\ 1&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&1&0&0&0&0&1 \\ 0&1&0&0&0&0&0&0&0&1 \\ 0&1&0&0&1&0&0&0&0&0 \\ 0&0&0&0&0&0&0&1&0&0 \\ 0&0&1&0&0&0&0&0&0&0 \end{bmatrix}$$

$$\mathbf{B} = \begin{bmatrix} 1&0&0&0 \\ 1&0&0&0 \\ 1&0&0&0 \\ 0&1&0&0 \\ 0&1&0&0 \\ 0&0&1&0 \\ 0&0&1&0 \\ 0&0&1&0 \\ 0&0&0&1 \\ 0&0&0&1 \end{bmatrix} \quad \mathbf{B}_{\text{ex}} = \begin{bmatrix} 0&0 \\ 0&0 \\ 0&0 \\ 1&0 \\ 1&0 \\ 0&0 \\ 0&0 \\ 0&0 \\ 0&1 \\ 0&1 \end{bmatrix}$$

$$\mathbf{C} = \begin{bmatrix} 0&0&0&1&0&1&0&0&1&0 \\ 1&0&0&0&0&0&1&0&0&0 \\ 0&1&0&0&1&0&0&0&0&1 \\ 0&0&1&0&0&0&0&1&0&0 \end{bmatrix} \quad \mathbf{D}_{\text{ex}} = \begin{bmatrix} 0&0 \\ 1&0 \\ 0&0 \\ 0&1 \end{bmatrix}$$

If we remove the gains $\bar{g}_l'$ from the linear state-space equations, we have a linear time-invariant (LTI) system to which we can associate a transfer function. Let the unilateral z-transforms of the input vector $\boldsymbol{\lambda}_l^{(\text{ex})}$ and the soft-output vector $\tilde{\boldsymbol{\lambda}}_l$ be denoted by $\boldsymbol{\Lambda}_{\text{ex}}(z)$ and $\tilde{\boldsymbol{\Lambda}}(z)$, respectively. (The unilateral z-transform of a discrete-time sequence $\{x_k\}_{k=0}^{\infty}$ is $X(z) = \sum_{k=0}^{\infty} x_k z^{-k}$ [55]. This definition extends naturally to discrete-time vector sequences.) Note that the other input vector $\boldsymbol{\lambda}$ is not a function of the iteration number.

In the transform domain, the input-output relationship of the system translates to

$$\tilde{\boldsymbol{\Lambda}}(z) = \left[\mathbf{C}\left(z\mathbf{I} - \mathbf{A}\right)^{-1}\mathbf{B} + \mathbf{I}\right]\frac{\boldsymbol{\lambda}}{1 - z^{-1}} + \left[\mathbf{C}\left(z\mathbf{I} - \mathbf{A}\right)^{-1}\mathbf{B}_{\text{ex}} + \mathbf{D}_{\text{ex}}\right]\boldsymbol{\Lambda}_{\text{ex}}(z). \quad (14)$$

This equation makes evident the importance of the eigenvalues of $\mathbf{A}$ in the analysis of the system. As the roots of the characteristic equation $\det(\mathbf{A} - \mu\mathbf{I}) = 0$, they correspond directly to the poles of the discrete-time LTI system in (14). For the TSs addressed in this paper, the dominant poles lie on or outside the unit circle. Hence, the LTI system is marginally stable or, more often, unstable.

We wish to take the recursive state-update equation (12) and develop it into an expression without feedback. The first two iterations are easy to express as

$$\mathbf{x}_1 = \bar{g}_1'\mathbf{A}\mathbf{B}\boldsymbol{\lambda} + \mathbf{B}\boldsymbol{\lambda} + \mathbf{B}_{\text{ex}}\boldsymbol{\lambda}_1^{(\text{ex})}$$

and

$$\mathbf{x}_2 = \bar{g}_1'\bar{g}_2'\mathbf{A}^2\mathbf{B}\boldsymbol{\lambda} + \bar{g}_2'\mathbf{A}\mathbf{B}\boldsymbol{\lambda} + \mathbf{B}\boldsymbol{\lambda} + \bar{g}_2'\mathbf{A}\mathbf{B}_{\text{ex}}\boldsymbol{\lambda}_1^{(\text{ex})} + \mathbf{B}_{\text{ex}}\boldsymbol{\lambda}_2^{(\text{ex})}.$$

Generalizing to an arbitrary iteration $l > 0$, we have

$$\mathbf{x}_l = \mathbf{A}^l\mathbf{B}\boldsymbol{\lambda}\prod_{j=1}^{l}\bar{g}_j' + \sum_{i=1}^{l}\mathbf{A}^{l-i}\left(\mathbf{B}\boldsymbol{\lambda} + \mathbf{B}_{\text{ex}}\boldsymbol{\lambda}_i^{(\text{ex})}\right)\prod_{j=i+1}^{l}\bar{g}_j'. \quad (15)$$

C. Graphical Assumptions and Interrelationships

In Assumptions 2 and 5, we have already imposed two restrictions on the TSs and induced subgraphs that we consider in our analysis. In this subsection, we impose some additional constraints and also describe interrelationships among several useful graphical descriptions of TSs: the Tanner subgraph, its corresponding undirected multigraph, the line graph of the multigraph, and the simple digraph that corresponds directly to the state update model. Finally, we present some useful bounds and estimates for the dominant eigenvalue of the system matrix $\mathbf{A}$ associated with the TS.

Let $B_{\mathcal{S}}$ be an elementary Tanner subgraph within the Tanner graph of a d_{v}-variable-regular code, and let $\mathbf{H}_{\mathcal{S}}$ be the corresponding parity-check submatrix. The undirected multigraph $G = (V, E)$ derived from the Tanner subgraph is defined as follows. Each variable node in $B_{\mathcal{S}}$ becomes a vertex, $v \in V$, in G. The degree-two check nodes in $B_{\mathcal{S}}$ become the edges of G, each joining the pair of vertices corresponding to its neighboring variable nodes in $B_{\mathcal{S}}$. The degree-one check nodes in $B_{\mathcal{S}}$ are ignored.

The multigraph G provides a more basic description of the elementary TS $B_{\mathcal{S}}$, as illustrated in Figs. 4a and 4b. Either one of these graphs is sufficient to characterize the iteration-to-iteration dependencies among the state variables in the model. For the multigraph $G = (V, E)$ of order a corresponding to $B_{\mathcal{S}}$, we have

$$\mathbf{A}(G) = \mathbf{H}_{\mathcal{S}}^T \mathbf{H}_{\mathcal{S}} - d_{\mathrm{v}} \mathbf{I}$$

and

$$\text{size}\,(G) = |E| = \frac{a d_{\mathrm{v}} - b}{2} = \frac{1}{2} \sum_{v_i \in V} d(v_i). \tag{16}$$

The final equality in (16) follows from Euler's handshaking lemma, which states that the sum of all degrees of a graph must be twice its number of edges. Tanner graphs, with cycles of length k, map to multigraphs with cycles of length $k/2$. Thus, cycles in $B_{\mathcal{S}}$ of length 4, the smallest length permitted in a Tanner graph, will generate parallel edges in G. Even though 4-cycles in the Tanner graph are often avoided by code designers, they are allowed in this paper, except within Appendix A.

Example 3: The $(4, 2)$ Tanner subgraph shown in Figs. 2b and 4a maps to the multigraph of Fig. 4b. Note that the $(3, 1)$ Tanner subgraph of Fig. 2a will map to a multigraph with parallel edges.

Assumption 7: Tanner subgraphs of interest and their associated multigraphs are connected. Those TSs described by disconnected Tanner subgraphs can instead be analyzed component by component. The main reason for this simplification is to reduce the number of cases in which the state update expression contains a reducible system matrix.

Assumption 8: We further assume that the variable nodes within the Tanner subgraphs contain no more than $d_{\mathrm{v}} - 2$ neighboring degree-one check nodes. Equivalently, we assume that the associated multigraphs are leafless; that is, they do not contain vertices of degree one. For consideration of multigraphs with leaves, see Appendix C.

A leaf in the multigraph corresponds to a variable node in the Tanner subgraph with one degree-two check node and $d_{\mathrm{v}} - 1$ degree-one check nodes as neighbors. Thus, we require here that every variable node in the Tanner subgraph must have at least two neighboring degree-two check nodes. The multigraphs allowed by Assumptions 7 and 8 will be connected, leafless graphs, containing one or more cycles.

In [28], TSs whose subgraphs satisfied the conditions prescribed above were called *simple* TSs. For Tanner graphs with fixed variable-node degree $d_{\mathrm{v}} = 3$, the TSs consistent with Assumption 8 are equivalent to absorbing sets. When $d_{\mathrm{v}} \geq 4$, however, the conditions in Assumption 8 are less restrictive than those in the definition of absorbing sets because we allow more degree-one check nodes per variable node.

Finally, we describe how to create the simple digraph $D = (Z, A)$ from an undirected multigraph $G = (V, E)$ which satisfies Assumptions 7 and 8. The construction is illustrated by reference to Figs. 4b and 4d. Each edge of a connected leafless multigraph G corresponds to two vertices in D, representing the two directions of LLR message flow in the original Tanner graph. That is, the number of vertices m in D is simply

$$m = \text{order}\,(D) = 2 \cdot \text{size}\,(G) = a d_{\mathrm{v}} - b.$$

The digraph D is a representation of the TS's message updating process in one full iteration of the SPA decoder. Let edge $e_i = \{v_j, v_k\} \in E$ map to vertices $z_i, z_{i'} \in Z$, with z_i representing the direction of message flow along e_i from v_j to v_k and $z_{i'}$ representing the opposite direction. No arcs in D join z_i to $z_{i'}$. Arcs initiating in z_i are directed to vertices corresponding to other edges in G flowing out of v_k, and arcs terminating in z_i are directed from vertices corresponding to other edges in G flowing into v_j. Hence,

$$d^+(z_i) = d(v_k) - 1 = d^-(z_{i'}) \tag{17}$$

and

$$d^-(z_i) = d(v_j) - 1 = d^+(z_{i'}). \tag{18}$$

In [30], the authors refer to D as the *flow graph* of G arising from the SPA and to $\mathbf{A}(D)$ as the *directed edge matrix* of G. The latter term is from [56].

We now describe a second version of the construction of the simple digraph. From the multigraph G, shown in Fig. 4b, we create the line graph $\mathcal{L}(G)$ shown in Fig. 4c as described in Section II-A. Next, we perform a special directed lifting of $\mathcal{L}(G)$ to form the simple digraph D, shown in Fig. 4d. This lifting by a factor of two replaces each vertex with a pair of vertices and each edge with a pair of arcs, oriented in opposite directions. Thus, if the (i, j) entry of $\mathbf{A}(\mathcal{L}(G))$ is w it is replaced with a 2×2 submatrix containing w ones, where the line graph limits w to the values 0, 1, and 2. When the 2×2 submatrix which replaces the (i, j) entry of $\mathbf{A}(\mathcal{L}(G))$ is determined, the selection of the corresponding (j, i) submatrix follows a set of rules of correspondence, listed in Table I, which ensure that the arcs are oriented in opposite directions of flow with respect to the edges of $\mathcal{L}(G)$. Thus, the subdiagonal elements can be easily determined from the superdiagonal elements.

TABLE I
SUBMATRIX CORRESPONDENCE RULES OF THE DIRECTED LIFTING OF THE LINE GRAPH (FOR $i = j$, ONLY THE ALL-ZERO MATRIX IS ALLOWED)

Entry (i, j)	**Entry** (j, i)
$\left(\begin{smallmatrix}1&0\\0&0\end{smallmatrix}\right)$	$\left(\begin{smallmatrix}0&0\\0&1\end{smallmatrix}\right)$
$\left(\begin{smallmatrix}0&1\\0&0\end{smallmatrix}\right)$	$\left(\begin{smallmatrix}0&1\\0&0\end{smallmatrix}\right)$
$\left(\begin{smallmatrix}0&0\\1&0\end{smallmatrix}\right)$	$\left(\begin{smallmatrix}0&0\\1&0\end{smallmatrix}\right)$
$\left(\begin{smallmatrix}1&0\\0&1\end{smallmatrix}\right)$	$\left(\begin{smallmatrix}1&0\\0&1\end{smallmatrix}\right)$
$\left(\begin{smallmatrix}0&1\\1&0\end{smallmatrix}\right)$	$\left(\begin{smallmatrix}0&1\\1&0\end{smallmatrix}\right)$

The exact lifting required to construct digraph D has additional restrictions to preserve proper SPA message flow. All directed walks in D will correspond to walks in G. All walks in G that do not backtrack will correspond to directed walks in D. Backtracking is prohibited here due to the structure of the SPA decoder as expressed in the message update rules,

which exclude an edge's own incoming LLR from its outgoing LLR computation.

The adjacency matrix of D is $\mathbf{A}(D)$, which is an $m \times m$ $(0, 1)$-matrix. The system matrix $\mathbf{A}$ used in the state-space model (12) is equal to $\mathbf{A}(D)^T$ as D describes the flow of messages within the model. This connection will be convenient when discussing the properties of $\mathbf{A}$.

Example 4: Figs. 2a, 2b, 2c, 2d, and 5c satisfy Assumptions 7 and 8. Of these, only Fig. 2d has a reducible $\mathbf{A}(D)$ matrix and it may be analyzed using techniques explained in Appendix B. The graphs of Figs. 5a and 5b do not satisfy Assumption 8 and have reducible $\mathbf{A}(D)$ matrices. In the case of Fig. 5a, the graph may first be analyzed without the leaf, and then with the leaf restored as discussed in Appendix C. The graph in Fig. 5b has all zero eigenvalues as no amount of leaf removal creates a satisfactory structure. Thus, it is not analyzed in this paper.

We now discuss bounds on the spectral radius of the graphs presented in this subsection and use them to estimate the dominant eigenvalues of $\mathbf{A}(D)$ and, thus, of $\mathbf{A}$. Using (4) and (16), we may bound the spectral radius of the multigraph $G = (V, E)$ as

$$d_{\mathrm{v}} - b/a \leq \rho(G) \leq \max_{v_i \in V} d(v_i). \tag{19}$$

Since, by Assumption 7, G is connected, the inequalities in (19) hold with equality if and only if G is regular.

For a multigraph G, the spectral radius of its line graph is given by

$$\begin{aligned} \rho(\mathcal{L}(G)) &= \rho\big(\mathbf{N}(G)^T\mathbf{N}(G) - 2\mathbf{I}\big) \\ &= \rho(\mathbf{A}(G) + \mathbf{D}_G) - 2. \end{aligned} \tag{20}$$

This follows from (1) and (2), and the fact that, for any real matrix $\mathbf{M}$, the two products $\mathbf{M}^T\mathbf{M}$ and $\mathbf{M}\mathbf{M}^T$ have identical nonzero eigenvalues.

Now, when G is regular of degree k, we have $\rho(G) = k$ and $k = d_{\mathrm{v}} - b/a$, where b/a must be an integer. Additionally, (20) implies that $\rho(\mathcal{L}(G)) = 2k - 2$ for the line graph which is itself regular [37, p. 36]. Furthermore, we find that the corresponding digraph D is regular with indegree $k - 1$ and outdegree $k - 1$, implying that its spectral radius must be $\rho(D) = k - 1 = d_{\mathrm{v}} - 1 - b/a$. This value corresponds to Schlegel and Zhang's approximation to the spectral radius of an absorbing set [11],

$$\rho(D) \approx d_{\mathrm{v}} - 1 - b/a. \tag{21}$$

Alternatively, one can view (21) as arising from the lower bound of (19), which is fairly tight, and the fact that $\rho(D) = \rho(G) - \Delta$, where Δ is equal to or slightly greater than 1. This difference between $\rho(D)$ and $\rho(G)$ stems from the vertex-degree relationships in (17) and (18). We empirically examine the accuracy of (21) in Appendix A.

We can derive an alternative approximation using a bound on $\rho(\mathcal{L}(G))$. From (16), the order of $\mathcal{L}(G)$ is $(ad_{\mathrm{v}} - b)/2$. For the following calculations, we assume that no variable node has more than one neighboring unsatisfied check node. This is always the case for absorbing sets with $d_{\mathrm{v}} = 3$ or 4, and likely true for other TSs of interest. The condition implies that G contains b nodes of degree $d_{\mathrm{v}} - 1$ and $a - b$ nodes of degree d_{v}, from which we get an approximation to (3),

$$\text{size}\,(\mathcal{L}(G)) \approx (d_{\mathrm{v}} - 1)(ad_{\mathrm{v}} - 2b)/2,$$

and, using (4), an approximate lower bound on the spectral radius of $\mathcal{L}(G)$,

$$\rho(\mathcal{L}(G)) \gtrapprox 2\left(d_{\mathrm{v}} - 1 - \frac{b - b/d_{\mathrm{v}}}{a - b/d_{\mathrm{v}}}\right).$$

Finally, since the directional lifting technique used to derive $\mathbf{A}(D)$ from $\mathbf{A}(\mathcal{L}(G))$ roughly halves each row sum and column sum, we form the new approximation

$$\rho(D) \approx d_{\mathrm{v}} - 1 - \frac{b - b/d_{\mathrm{v}}}{a - b/d_{\mathrm{v}}}, \tag{22}$$

whose accuracy is also examined in Appendix A. Note that when b/a equals 0 or 1, the approximations in (21) and (22) are equal.

IV. Dominant Eigen Values and Probability Model

This section will simplify (15), the state vector expression, to the point that we can easily incorporate probability distributions and predict failure rates for specific TSs.

A. Utilizing Dominant Eigenvalues

For the analysis of the model, we need to simplify the powers of the system matrix $\mathbf{A}$ that appear in (15). This section generalizes the development by Schlegel and Zhang [11] with a focus on the dominant eigenvalues of the nonnegative $m \times m$ matrix $\mathbf{A}$ and its left eigenvectors. This approach will avoid using the assumption made by Sun, which was that $\mathbf{A}$ is diagonalizable [28]. In our extensive search of potential TSs, shown in Appendix A, we found that a small fraction of the TSs have system matrices which cannot be diagonalized.

Let $\mu_k \in \mathbb{C}$ be an eigenvalue of matrix $\mathbf{A}$, and let $\mathbf{w}_k^*$ be the left eigenvector associated with μ_k, such that $\mathbf{w}_k^*\mathbf{A} = \mu_k\mathbf{w}_k^*$, where $\mathbf{w}_k^*$ denotes the conjugate transpose of column vector $\mathbf{w}_k$. Then, by induction, for any positive integer i,

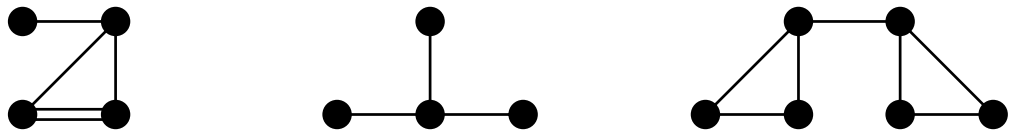
$$\mathbf{w}_k^*\mathbf{A}^i = \mu_k^i\mathbf{w}_k^*. \tag{23}$$

Left-multiplying (15) by $\mathbf{w}_k^*$ and using (23) to simplify, we derive the scalar quantity

$$\begin{aligned} \mathbf{w}_k^*\mathbf{x}_l = {} & \mu_k^l\mathbf{w}_k^*\mathbf{B}\boldsymbol{\lambda}\prod_{j=1}^{l}\bar{g}_j' \\ & + \sum_{i=1}^{l}\mu_k^{l-i}\mathbf{w}_k^*\left(\mathbf{B}\boldsymbol{\lambda} + \mathbf{B}_{\mathrm{ex}}\boldsymbol{\lambda}_i^{(\mathrm{ex})}\right)\prod_{j=i+1}^{l}\bar{g}_j'. \end{aligned}$$

We now shift to using a specific left eigenvector of the nonnegative matrix $\mathbf{A}$, the left eigenvector $\mathbf{w}_1^T$ associated with the eigenvalue of maximum modulus r. Classifying the $m \times m$ nonnegative matrix $\mathbf{A}$ into the following two cases, the theory of nonnegative matrices allows us to make certain statements regarding $\mathbf{w}_1$ and r [33]–[36]:

1) Let the nonnegative matrix $\mathbf{A}$ be irreducible. There is a simple eigenvalue r, such that $r = \rho(\mathbf{A})$ and



(a) cycles and a leaf (b) tree (c) bridged cycles

Fig. 5. Three undirected multigraphs.

$r > 0$. The left eigenvector $\mathbf{w}_1^T$ associated with r is positive. There are no other nonnegative left eigenvectors of $\mathbf{A}$, except for $\mathbf{w}_1^T$ multiplied by a positive scalar.

2) Let the nonnegative matrix $\mathbf{A}$ be reducible and let the block upper triangular matrix $\mathbf{A}'$ be a symmetric permutation of $\mathbf{A}$. The submatrices $\mathbf{A}'_i$ along the block diagonal are either irreducible square matrices or the 1-by-1 zero matrix. The eigenvalues of $\mathbf{A}$ are thus the union of the eigenvalues of the submatrices $\mathbf{A}'_i$. Appendix B shows that the reducible cases allowed by Assumptions 7 and 8 have $r = \rho(\mathbf{A}) = 1$ which may be associated with the all-one left eigenvector $\mathbf{w}_1^T$.

Thus, given our prior assumptions, we may associate a positive left eigenvector $\mathbf{w}_1^T$ with the eigenvalue r, and this r is real with magnitude greater than or equal to that of all other eigenvalues.

In [11], the scalar quantity β was formed by summing the states and if found to be less than zero, the TS was determined to be in error. In generalizing that quantity, we define the scalar error indicator $\beta_l \triangleq \mathbf{w}_1^T \mathbf{x}_l$. Consider β_l to be the scaled projection of the state vector $\mathbf{x}_l$ onto the positive vector $\mathbf{w}_1$. Since the state vector $\mathbf{x}_l$ contains the internal messages of the TS, the projection onto a positive vector indicates whether the TS's messages are generally in the positive (correct) direction or negative (incorrect) direction.

While β_l appears to be approximately indicative of the output bit decisions of the variable nodes, a more careful examination of the model's soft-output in (13) will reveal the importance of β_l with regard to the output decisions. The first term of (13) includes $\mathbf{x}_{l-1}$ whose elements, the state variables, will tend to all move toward positive or negative infinity for cases in which the system is inherently unstable, $r > 1$. The second term of (13), the intrinsic channel LLRs, is constant for each channel realization and does not change with iteration count. The final term of (13) includes the elements of $\boldsymbol{\lambda}_l^{(\mathrm{ex})}$ which will reach saturated magnitudes in decoders with saturating LLRs. Under these conditions it is easy to see that the elements of $\mathbf{x}_{l-1}$ will dominate the soft-output expression (13). Additionally, we note that the row sums of the output matrix $\mathbf{C}$ in (13) are larger than the row sums of the system matrix $\mathbf{A}$ in (12). This implies that computing $\tilde{\boldsymbol{\lambda}}_l$ is much like computing $\mathbf{x}_l$ but with even more weight applied to the terms containing $\mathbf{x}_{l-1}$. Thus, we argue that, even for nonsaturating decoders, the sign of β_l is indicative of the sign of the elements of $\tilde{\boldsymbol{\lambda}}_l$ in the model.

Example 5: The nonnegative system matrices $\mathbf{A}$ describing the state update operations of Figs. 2a and 2b are both primitive,[3] with $r \approx 1.6956$ and $r \approx 1.5214$, respectively. The nonnegative system matrix $\mathbf{A}$ describing Fig. 2c is imprimitive[4] with index of imprimitivity $h = 4$ and $r = \sqrt{2}$. One may find that $h = 4$ by visual inspection of Fig. 2c, noting that every cycle length of message flow (measured in full iterations) is divisible by four. The nonnegative system matrix $\mathbf{A}$ describing Fig. 2d is reducible with $r = 1$.

Our error indicator expression simplifies if we rescale β_l by a positive constant to $\beta'_l \triangleq \beta_l / \left(r^l \prod_{j=1}^{l} \bar{g}'_j \right)$, so

$$\beta'_l = \mathbf{w}_1^T \mathbf{B}\boldsymbol{\lambda} + \sum_{i=1}^{l} \frac{\mathbf{w}_1^T \left(\mathbf{B}\boldsymbol{\lambda} + \mathbf{B}_{\mathrm{ex}} \boldsymbol{\lambda}_i^{(\mathrm{ex})} \right)}{r^i \prod_{j=1}^{i} \bar{g}'_j}. \tag{24}$$

This expression is similar to, but more general than (1) in [11], which was derived for a specific degenerate TS as described in Section III-B.

B. Modeling the Probability of Error

The expression for β'_l in (24) is a linear combination of stochastic vectors $\boldsymbol{\lambda}$ and $\boldsymbol{\lambda}_l^{(\mathrm{ex})}$. Since elements of $\boldsymbol{\lambda}$ are i.i.d. Gaussian, linear operations on $\boldsymbol{\lambda}$ will produce a Gaussian distribution. Several authors [11], [53] have used the approximation that the check-node output LLRs, such as $\boldsymbol{\lambda}_l^{(\mathrm{ex})}$, are Gaussian, too. The central limit theorem implies that the distribution of a linear combination of several independent check-node output messages with the elements of $\boldsymbol{\lambda}$ will be nearly Gaussian, even if the check-node output LLRs are only approximately Gaussian.

Assumption 9: We assume that β'_l has a Gaussian distribution.

Under this assumption, the probability of the failure event, $\xi(\mathcal{S})$, corresponding to TS $\mathcal{S}$, at iteration l, is then simply

$$\Pr\{\xi(\mathcal{S})\} = \Pr\left\{\beta'_l < 0\right\} = \mathrm{Q}\left(\frac{\mathbb{E}[\beta'_l]}{\sqrt{\mathrm{VAR}[\beta'_l]}} \right). \tag{25}$$

If $\{\mathcal{S}_i\}$ enumerates all potential TSs, the union bound provides an upper bound on the error floor of the block error rate

$$P_{\mathrm{f}} \lessapprox \sum_i \Pr\{\xi(\mathcal{S}_i)\}. \tag{26}$$

The block error rate of (26) is also commonly called the frame error rate (FER) or codeword error rate.

[3] If $\mathbf{A}$ is an irreducible nonnegative matrix and has only one eigenvalue with modulus equal to $\rho(\mathbf{A})$, then $\mathbf{A}$ is said to be *primitive*.

[4] If $\mathbf{A}$ has $h > 1$ eigenvalues with modulus equal to $\rho(\mathbf{A})$, then $\mathbf{A}$ is said to be *imprimitive*. The value h is known as the *index of imprimitivity* or period.

Next, we wish to express the error floor as an information bit error rate (BER). Letting $\hat{a}_i$ represent the number of information bits associated with the TS failure $\xi(\mathcal{S}_i)$ for the specific encoding technique used, we may express the BER union bound as

$$P_{\mathrm{b}} \lessapprox \sum_i \frac{\hat{a}_i}{k} \Pr\{\xi(\mathcal{S}_i)\}, \tag{27}$$

where k is the number of information bits per codeword.

We now assume a systematic encoding, *i.e.*, the k information bits appear explicitly in the codeword. If the $\hat{a}_i$ information bit locations are assumed to be independent of the a_i bit locations in the TS $\mathcal{S}_i$ then the expected number of information bit errors associated with the TS failure $\xi(\mathcal{S}_i)$ is $\mathbb{E}[\hat{a}_i] = a_i k/n$. In this case, the approximate inequality in (27) may be simplified to

$$P_{\mathrm{b}} \lessapprox \sum_i \frac{a_i}{n} \Pr\{\xi(\mathcal{S}_i)\}. \tag{28}$$

We present (28) as an alternative to a similar expression in [11] which used k in the denominator.

C. Codewords

In the case of elementary TSs that are also codeword supports, we have $b = 0$, and $\mathbf{B}_{\mathrm{ex}}$ is not used since there are no unsatisfied check nodes. We find that the failure probability (25) of the codeword simplifies to

$$\Pr\{\xi(\mathcal{S}_i)\} = \mathrm{Q}\left(\sqrt{\frac{2RE_{\mathrm{b}}}{N_0}}\,\frac{\sum_{k=1}^{a}(\mathbf{w}_1^T\mathbf{B})_k}{\sqrt{\sum_{k=1}^{a}(\mathbf{w}_1^T\mathbf{B})_k^2}}\right). \tag{29}$$

This reduces further as the eigensystem of the system matrix $\mathbf{A}$ for codewords is rather simple. Every row sum and column sum of $\mathbf{A}$ is $d_{\mathrm{v}} - 1$, so the spectral radius is $r = d_{\mathrm{v}} - 1$. The positive left eigenvector $\mathbf{w}_1^T$ associated with r is proportional to the all-one row-vector, *i.e.*, $\mathbf{w}_1^T \propto [1, 1, \ldots 1]$. Also, $\mathbf{B}$ has a uniform row sum of 1 and a column sum of d_{v}. Therefore, $(\mathbf{w}_1^T\mathbf{B})_k = d_{\mathrm{v}}\ \forall\ k \in \{1, \ldots, a\}$. Noting that a is just the Hamming weight w_{H} of the codeword corresponding to $\mathcal{S}_i$, which we denote by $w_{\mathrm{H}}(\mathcal{S}_i)$, the expression in (29) can be written in the more recognizable

$$\Pr\{\xi(\mathcal{S}_i)\} = \mathrm{Q}\left(\sqrt{\frac{2RE_{\mathrm{b}}}{N_0} w_{\mathrm{H}}(\mathcal{S}_i)}\right),$$

independent of iteration count l.

D. Non-Codewords

To further understand the general behavior of (25) as a function of the iteration l, we take a closer look at the numerator and denominator terms. Letting $m_{\lambda(\mathrm{ex})}^{(l)}$ be the expected value of each entry of $\boldsymbol{\lambda}_l^{(\mathrm{ex})}$, we can write the numerator of the Q-function argument as

$$\mathbb{E}[\beta_l'] = m_\lambda\left(1 + \sum_{i=1}^{l}\frac{1}{r^i\prod_{j=1}^{i}\bar{g}_j'}\right)\sum_{k=1}^{a}(\mathbf{w}_1^T\mathbf{B})_k + \sum_{i=1}^{l}\frac{m_{\lambda(\mathrm{ex})}^{(i)}}{r^i\prod_{j=1}^{i}\bar{g}_j'}\sum_{k=1}^{b}(\mathbf{w}_1^T\mathbf{B}_{\mathrm{ex}})_k. \tag{30}$$

Assumption 10: We will assume that the entries of $\boldsymbol{\lambda}$ and $\boldsymbol{\lambda}_l^{(\mathrm{ex})}$ are statistically independent of each other at a given iteration l, as is often done in density evolution (DE) studies. Further, we assume that the entries of $\boldsymbol{\lambda}_l^{(\mathrm{ex})}$ are independent from iteration to iteration, as was implicitly assumed in [11].

Letting σ_l^2 be the variance of each entry of the vector $\boldsymbol{\lambda}_l^{(\mathrm{ex})}$, we can write the squared-denominator of the Q-function argument as

$$\mathrm{VAR}[\beta_l'] = 2m_\lambda\left(1 + \sum_{i=1}^{l}\frac{1}{r^i\prod_{j=1}^{i}\bar{g}_j'}\right)^2\sum_{k=1}^{a}(\mathbf{w}_1^T\mathbf{B})_k^2 + \sum_{i=1}^{l}\frac{\sigma_i^2}{(r^i\prod_{j=1}^{i}\bar{g}_j')^2}\sum_{k=1}^{b}(\mathbf{w}_1^T\mathbf{B}_{\mathrm{ex}})_k^2. \tag{31}$$

We wish to understand the asymptotic behavior of (25) as the number of iterations goes toward infinity. To this end, we examine the convergence of the partial sums in (30) and (31) that depend on the iteration number l.

We first note that the divergence of the first partial sum in (30) would not necessarily cause (25) to diverge because its effect would be offset by the divergence of the the first partial sum in (31). We can apply the Ratio Test to examine the convergence properties of the second partial sum in (30). Given a real sequence $\{a_i\}_{i=1}^{\infty}$, we define $\rho = \lim_{i\to\infty}|a_{i+1}/a_i|$. The Ratio Test states that the series $\sum_{i=1}^{\infty} a_i$ converges absolutely if $\rho < 1$ and diverges if $\rho > 1$. With respect to the second partial sum within (30), we see that

$$\rho = \lim_{i\to\infty}\left|\frac{m_{\lambda(\mathrm{ex})}^{(i+1)} r^i\prod_{j=1}^{i}\bar{g}_j'}{m_{\lambda(\mathrm{ex})}^{(i)} r^{i+1}\prod_{j=1}^{i+1}\bar{g}_j'}\right| = \lim_{i\to\infty}\left|\frac{m_{\lambda(\mathrm{ex})}^{(i+1)}}{m_{\lambda(\mathrm{ex})}^{(i)} r}\right|. \tag{32}$$

The right-most expression follows by noting that the gains $\bar{g}_l'$ approach 1 rapidly as the iteration count l increases. Thus, if the mean unsatisfied-check LLR $m_{\lambda(\mathrm{ex})}^{(i)}$ dominates r^i asymptotically (*i.e.*, $m_{\lambda(\mathrm{ex})}^{(i)} > Cr^i$ for every positive constant C and sufficiently large i), then $\rho > 1$ and (30) will grow without bound.

If we can be further assured that (31) does not grow as fast as the square of (30), then the entire argument of the Q-function will grow without bound, driving the failure rate of the potential elementary TS toward zero. In that case, a sufficient number of iterations and sufficient LLR dynamic range for $\boldsymbol{\lambda}_l^{(\mathrm{ex})}$ growth are the requirements for the model to achieve as low an error floor as desired.

Other models of error floor behavior [11], [28] have used the Gaussian approximation to LLR densities [53], which implies $\sigma_l^2 = 2m_{\lambda(\mathrm{ex})}^{(l)}$. Moreover, using a numerical version of DE, Fu found that $\sigma_l^2 < 2m_{\lambda(\mathrm{ex})}^{(l)}$ as the LLRs get large [57]. With such an LLR variance, *i.e.*, $\sigma_l^2 \le c\, m_{\lambda(\mathrm{ex})}^{(l)}$ for some positive c, we find that the entire argument to the Q-function grows without bound in the cases that satisfy $m_{\lambda(\mathrm{ex})}^{(i)} > Cr^i$. We will find in the following sections that this latter condition is true under certain specified assumptions. However, if the variance grows as the square of the mean, then the argument to the Q-function reaches a finite limit and a nonzero error floor is produced.

TABLE II
DDE NUMERICAL RESULTS BY ITERATION FOR (3, 6)-REGULAR CODE AT AN E_b/N_0 OF 2.8 dB ($m_\lambda = 3.8109$) WITH LLR SATURATION SET TO ± 25 AT CHECK-NODE OUTPUT. LLR PDFS DISCRETIZED TO PMFS WITH A RESOLUTION OF 50/2047

l	$m_{\lambda(\mathrm{ex})}^{(l)}$	σ_l^2	$\bar{g}_l$
1	0.669	1.47	0.3242
2	1.315	2.81	0.4898
3	2.08	4.24	0.6243
4	3.11	5.94	0.7472
5	4.66	8.05	0.8586
6	7.21	10.74	0.9446
7	11.66	14.22	0.9891
8	19.67	16.59	0.9995
9	24.91	0.47	1.0000
10	25.00	0.00	1.0000

This is a very important issue that we will revisit later in the paper.

E. Bounds on Spectral Radius of the System Matrix A

We now need some bounds on r, the spectral radius of the system matrix $\mathbf{A}$, in order to compare $m_{\lambda(\mathrm{ex})}^{(i)}$ to r^i.

Theorem 2: Consider a variable-regular LDPC code with variable-degree $d_v \geq 3$. Let the TS $\mathcal{S}$ induce an elementary connected subgraph $B_{\mathcal{S}}$ with $a \geq 2$ variable nodes and $b > 0$ degree-one check nodes. Further, let the associated undirected multigraph G be leafless. Then, the adjacency matrix $\mathbf{A}(D)$ of the associated simple digraph D has spectral radius r such that $1 < r < d_v - 1$ when $\mathbf{A}(D)$ is irreducible and $r = 1$ when $\mathbf{A}(D)$ is reducible.

Proof: From (17), the maximum possible outdegree of D is $d_v - 1$. Since G cannot be d_v-regular due the presence of degree-one check nodes in the subgraph, D cannot be $(d_v - 1)$-outdegree regular. Now, by applying Lemma 1, we find that $1 \leq r < d_v - 1$ when $\mathbf{A}$ is irreducible. We defer to Appendix B the case in which the matrix $\mathbf{A}$ is reducible, which is the only allowed case in which $r = 1$. ∎

V. MODELING OF SATURATING DECODERS

In this section, we apply the prediction model to saturating decoders. We present techniques for generating the required inputs to the state-space model: the mean and variance of unsatisfied, degree-one check nodes, as well as the mean gain $\bar{g}_l$ of the degree-two check nodes. We then compare predicted error floors to results of computer simulation for four representative LDPC codes.

We first describe two numerical methods for computing the mean and variance of the unsatisfied, degree-one check nodes. The first method is discretized density evolution (DDE), a quantized version of density evolution (DE) [54]. DDE uses a probability mass function (pmf) that closely approximates the continuous probability density function (pdf) of node input messages and output messages. By operating directly on pmfs, it permits us to model the behavior of LLRs as they saturate. At a specified E_b/N_0 value, we use it to capture the mean and variance of the LLRs at each iteration, as needed for insertion into the model. Table II shows numerical results for 10 iterations for a (3, 6)-regular code and an LLR saturation level of ± 25. The table also shows the mean check-node gain, which was computed according to (7).

TABLE III
SPA SIMULATION RESULTS BY ITERATION FOR THE (3, 6)-REGULAR, (2640, 1320) MARGULIS CODE AT AN E_b/N_0 OF 2.8 dB ($m_\lambda = 3.8109$) WITH LLR SATURATION SET TO ± 25 AT CHECK-NODE OUTPUT

l	$m_{\lambda(\mathrm{ex})}^{(l)}$	σ_l^2	$\bar{g}_l$
1	0.675	1.49	0.3245
2	1.324	2.88	0.4897
3	2.09	4.36	0.6219
4	3.14	6.19	0.7425
5	4.73	8.83	0.8505
6	7.37	13.65	0.9338
7	12.08	25.82	0.9793
8	19.13	33.94	0.9956
9	23.66	11.86	0.9995
10	24.83	1.31	1.0000

The second method collects the required statistics by running the LLR-domain SPA decoder with early termination[5] turned off. As above, for a specified E_b/N_0 value, we capture at each iteration the mean and variance of the check-node output LLRs and compute the mean check-node gain as in (7). Table III shows the results obtained by simulating 100 frames of the (3, 6)-regular, (2640, 1320) Margulis code, with the same saturation level as above.

The mean values in the two tables are in close agreement, as might be expected. However, starting at iteration six, the LLR variance obtained from the SPA simulation becomes significantly larger than that produced by the DDE technique. This is caused by the cycles in the Margulis code. We would expect that the onset of this variance divergence would occur later in a code with a larger girth. Interestingly, despite the differences in the variance values they generated, the two techniques produced nearly identical error floor predictions. The modeling results shown in the figures that follow were obtained using the SPA simulation technique.

We note that our approach differs from that presented in [11]. There, only the mean LLRs from a (continuous) DE analysis are recorded, and the computation of the variance is based on the consistent Gaussian distribution assumption [53]. The consistent Gaussian approximation[6] implies that $\sigma_l^2 = 2\,m_{\lambda(\mathrm{ex})}^{(l)}$. We have purposely avoided the consistency assumption since the effect of LLR saturation will invalidate it, as can be seen in Table II.

The first code we examine is a (640, 192) quasi-cyclic (QC) LDPC code with $d_v = 5$ and irregular check degree, taken from [23]. The dominant TS for a saturating LLR-domain SPA decoder is the (5, 5) elementary TS shown in [23]. The system matrix $\mathbf{A}$ for this TS is primitive and has maximum

[5] *Early termination* refers to the discontinuation of further iterations once a valid codeword has been decoded. It is frequently employed to conserve average decoding complexity.

[6] The relationship between the mean and variance of the LLRs was printed erroneously in [11] as $m = 2\sigma^2$.

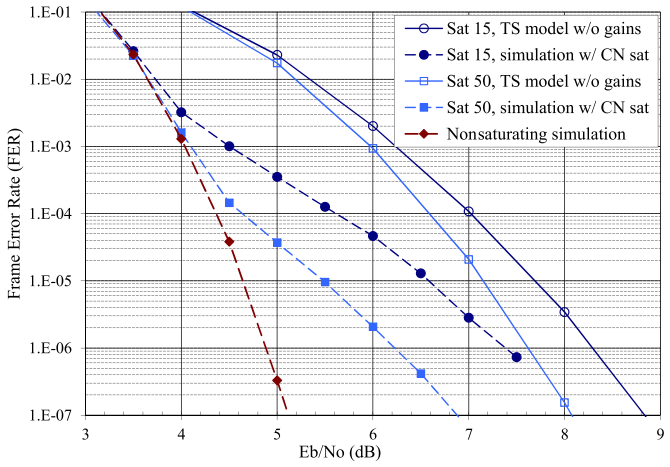


Fig. 6. FER vs. E_b/N_0 in dB for the (640, 192) QC code. Error floor model of (5, 5) TS used unity gains and no check-node inversions.

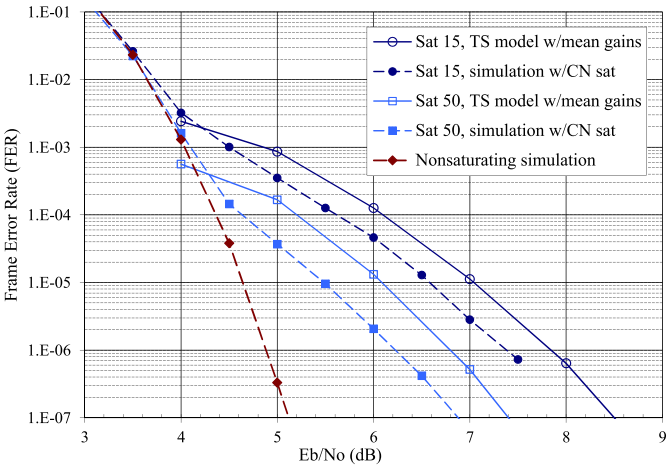


Fig. 7. FER vs. E_b/N_0 in dB for the (640, 192) QC code. Error floor model of (5, 5) TS used mean gains and no check-node inversions.

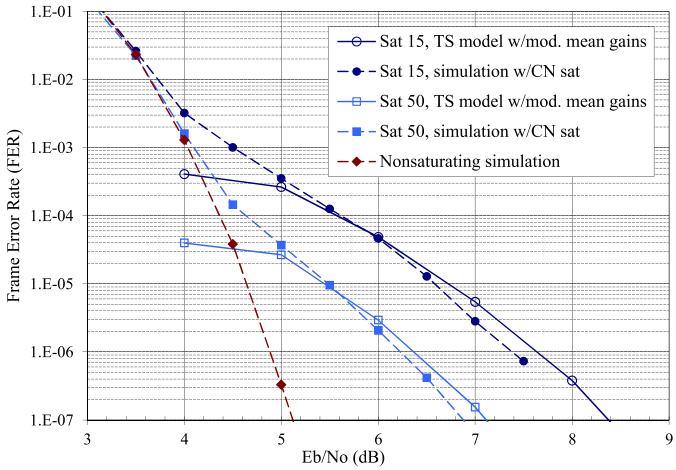


Fig. 8. FER vs. E_b/N_0 in dB for the (640, 192) QC code. Error floor model of (5, 5) TS used modified mean gains (11) with check-node inversions.

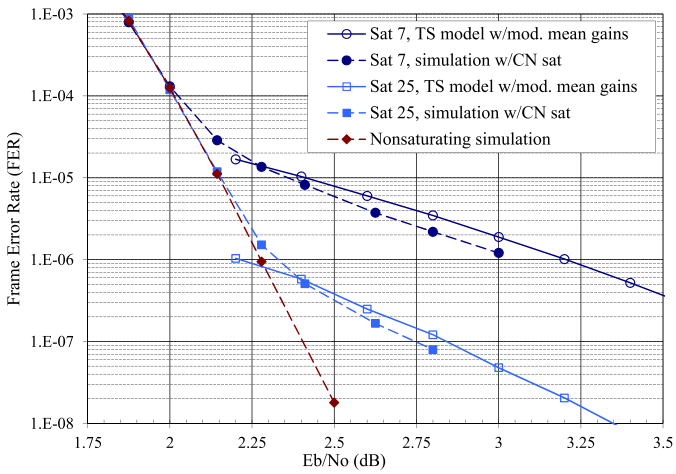


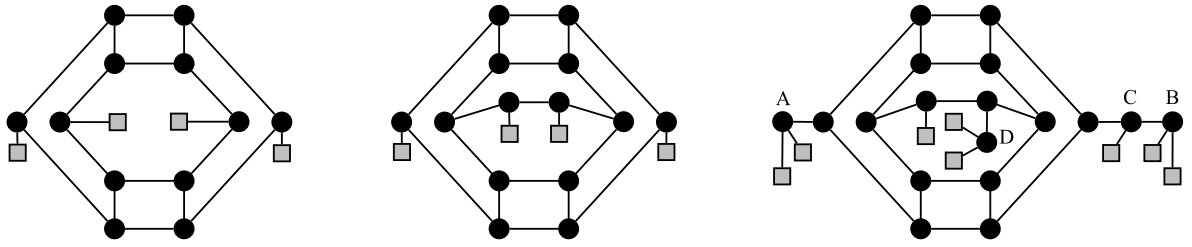
Fig. 9. FER vs. E_b/N_0 in dB for (2640, 1320) Margulis code. Error floor model of (12, 4) TS used modified mean gains (11) with check-node inversions.

eigenvalue $r = 3$. The multiplicity of the TS in the code is 64 and the corresponding sets of variable nodes are mutually disjoint. Figs. 6, 7 and 8 show FER results from simulation and error floor predictions at two levels of LLR saturation, 15 and 50. The error floor prediction model showed very fast convergence—a behavior that we attribute to the large eigenvalue r—so we stopped the model after 14 iterations.

The FER simulation results shown in these figures are produced by a nonsaturating LLR-domain SPA decoder, implemented in double-precision floating-point arithmetic. The computations are organized such that saturation will not occur (see §II-C). The nonsaturating decoder never failed on the (5, 5) TSs, in contrast to the results in [23]. For the saturating decoders, we introduced LLR saturation into the simulations at a point corresponding to the output of the complete check-node update. In all of the simulations, we assumed that the all-zero codeword was transmitted, and any decoding result other than the all-zero codeword was treated as a frame error. The maximum number of iterations was 200.

The predicted error floors in Fig. 6 were obtained using the union bound in (26), but with check-node gains forced to unity in the TS modeling. For the results shown in Fig. 7, the mean gains computed according to (7) were applied to the model. The prediction results in Fig. 8 show the effect of using the modified mean gains in (11), thereby capturing the effect of polarity inversions within the degree-two check nodes. Comparing the figures, we see that the mean gains provide a significant improvement in the model's accuracy, to within 0.5 dB of simulation, and the addition of the polarity inversion model further enhances the model's accuracy, to within ± 0.1 dB of simulation.

A heuristic explanation for the observed reduction in the predicted error floors when check-node gains are introduced has to do with incorrect growth in LLR values during the early iterations of the model. With unity gains, incorrect channel inputs may propagate rapidly among the highly interconnected states of the TS. When smaller gains are used, this LLR increase is delayed for several iterations, giving the unsatisfied check nodes a better chance to "correct" the system. As noted previously, during early model iterations the polarity inversions through the degree-two check nodes effectively lower the check-node gain. Therefore, as we would expect, introducing the inversions lowers the predicted error floor.



(a) $(12, 4)$ (b) $(14, 4)$ (c) $(18, 8)$

Fig. 10. Three elementary TSs of the Margulis code (with $d_v = 3$) are shown with degree-one check nodes shaded and with degree-two check nodes omitted. The two on the left are absorbing sets and dominate the error floor of the saturated SPA decoder. The one on the right is not an absorbing set, but was a TS collected during nonsaturated SPA decoding.

The SPA decoder results presented in [23] fall slightly above the curves in Figs. 6–8 that correspond to a saturation level of 15 . The simulated FER of our nonsaturating SPA decoder is three orders of magnitude lower than that of the SPA decoder in [23] at $E_b/N_0 = 5$ dB, and the difference grows with increasing E_b/N_0.

The second code we analyzed is the $(2640, 1320)$ Margulis LDPC code. Recall that this is a $(3, 6)$-regular code with dominant TSs reported to be the $(12, 4)$ and $(14, 4)$ elementary TSs [7], [8], [23]. The system matrices $\mathbf{A}$ for these two TSs are both imprimitive, with $h = 2$, and have maximum eigenvalues $r \approx 1.6956$ and $r \approx 1.7606$, respectively. We ran the model for 16 iterations. The multiplicity of each of these TSs is 1320 with significant overlap between the variable nodes of the different TSs.

In fact, each $(14, 4)$ TS contains all 12 variable nodes and 2 of the unsatisfied check nodes of a $(12, 4)$ TS, as can be seen in Fig. 10. Thus, when we compute the failure probability of one $(12, 4)$ TS, it largely includes the failure probability of the $(14, 4)$ TS that contains it. Therefore, in Fig. 9, we show only the error floors predicted by the model for the $(12, 4)$ TSs. The results were obtained using the modified mean gains, with the polarity inversion model applied at every iteration. The error floors predicted by the state-space model overestimated the simulated error floors by 0.15 dB or less in the E_b/N_0 range of the simulations.

The third code we considered is the Reed-Solomon-based binary $(2048, 1723)$ LDPC code [58] included in the 802.3an standard. It is a $(6, 32)$-regular code with dominant TSs reported to be the $(8, 8)$ elementary TS [11], [27]. The multiplicity of the $(8, 8)$ TS is 14 272 [11]. The system matrix $\mathbf{A}$ for this TS is primitive, with a maximum eigenvalue $r = 4$. The state-space model for this TS converges in only 3 iterations when the LLR saturation is set to 15, and it requires only a few more iterations at higher saturation limits. We conservatively ran the model for 16 iterations, including the polarity inversion model at every iteration. Fig. 11 shows the BER error floors predicted by the model using modified mean gains, under the assumption that no other TSs contribute significantly to the floor.

Since a standard Monte Carlo simulation of such a low error floor would take a prohibitively large amount of CPU time, we used Richardson's semi-analytical technique to estimate the floor [8]. The calculation considered only the $(8, 8)$ TSs.

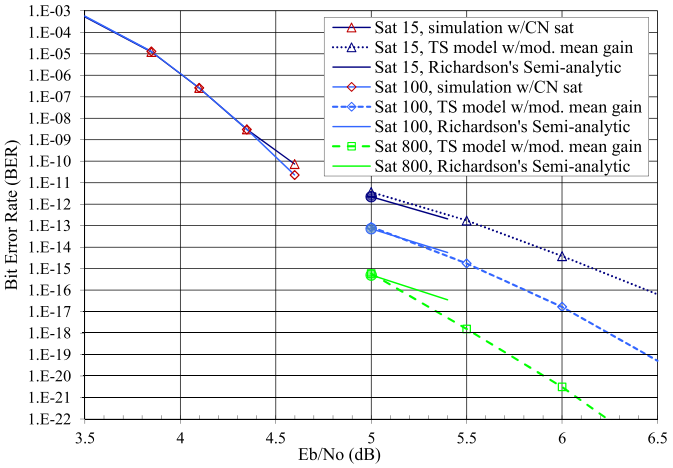


Fig. 11. BER vs. E_b/N_0 in dB for $(2048, 1723)$ LDPC code of 802.3an with simulation, Richardson's semi-analytical error floor estimation, and the error floor prediction model using modified mean gains.

At an E_b/N_0 of 5.0 dB, we obtained the BERs represented by the solid circles in Fig. 11. We used Richardson's extrapolation method to extend these results to 5.4 dB, as shown by the solid lines in the figure. For the LLR saturation levels of 15, 100, and 800, the application of Richardson's technique required 0.9, 9.5, and 1300 hours of CPU time, respectively. In contrast, the state-space model used less than 2 minutes of CPU time to collect the LLR statistics from 1000 frames that were used to generate each error floor prediction point. At $E_b/N_0 = 5.0$ dB, the model's predictions are within 0.1 dB of the error rates obtained using the semi-analytical technique.

The fourth code we considered is the $(3, 5)$-regular, $(155, 64)$ QC-LDPC code introduced by Tanner *et al.* [59]. The dominant TS is reported to be an $(8, 2)$ elementary TS, with multiplicity 465 [32], although there are several other small TSs.[7] The system matrix $\mathbf{A}$ for this TS is primitive, with a maximum eigenvalue $r \approx 1.7870$. Fig. 12 shows that the FER predicted by the model for the $(8, 2)$ TS using modified mean gains agrees with simulation to within 0.2 dB at an LLR saturation level of 7. At this level of saturation, an error floor is indeed evident in the simulation results. For E_b/N_0 values above 4.86 dB, we found that 69.0% of the

[7]Taking into account the TSs of this code, Declercq *et al.* [60] have developed quantized decoders that significantly reduce the error floor when this code is used over the BSC.

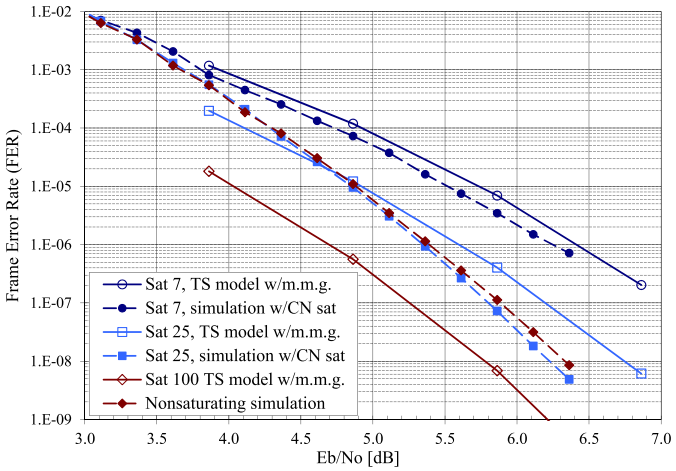


Fig. 12. FER vs. E_b/N_0 in dB for (155, 64) QC-LDPC code with simulation and model predicted floor of the (8, 2) TS using modified mean gains.

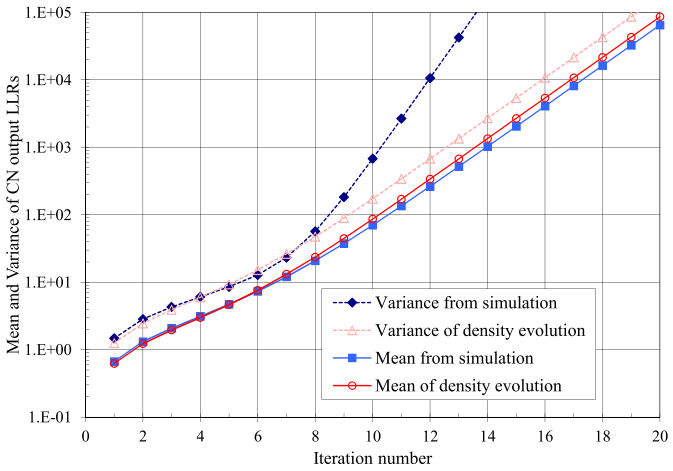


Fig. 13. Check-node output LLR mean and variance versus iteration number for the Margulis code at E_b/N_0 of 2.8 dB.

simulation failures were associated with the (8, 2) TSs, and most of the remaining failures could be attributed to other small TSs. However, at an LLR saturation level of 25, the error floor region is less pronounced in the FER simulation; in fact, only 6.8% of the simulated frame failures occurred on (8, 2) TSs. In this case, the state-space model overestimated TS failures by about 0.9 dB at 10^{-7} FER. The figure also shows that the error-rate curve for the nonsaturating decoder is slightly higher than that produced when the saturation level is 25. In fact, further simulations with a saturation level of 100 yielded results that were nearly identical to those of the nonsaturating decoder. At that saturation level, only a small fraction of the overall errors—roughly 7%—were due to the (8, 2) TSs. Moreover, about 55% of the frame errors had 11 or more erroneous bits. Some of these unusual comparative results may be connected to the fact that the waterfall region of this code has a very gentle slope. Finally, we note that in [32], BER estimates for this code were generated using an (8, 2) TS model and by importance sampling in the vicinity of the (8, 2) TSs with saturation levels of 10 and 100. The results were in close agreement with one another, suggesting that errors seen with a saturation level of 100 would be largely due to (8, 2) TSs. This is inconsistent with the results found in our simulation studies.

We chose to apply the state-space model to these four LDPC codes because they are structurally quite diverse in nature and their error floor performance under SPA decoding has been previously studied. The TSs in these codes have a wide range of multiplicities, with different degrees of overlap, and different graph-theoretic properties. In particular, the Margulis code has two dominant TSs, both with imprimitive $\mathbf{A}$ matrices, while the other codes are dominated by single TSs with primitive $\mathbf{A}$ matrices. Despite the differences among these codes, we found that the state-space model with modified mean gains estimated the error floors observed in simulations reasonably well and quite efficiently.

We saw that there are situations in which LLR saturation even at the fairly high levels of 25, 50 and 100 produce noticeable error floors in both simulation and model prediction. We also noted that the variances of the messages from degree-one check nodes of the induced subgraph of a TS—denoted by σ_i^2 in (31)—have a small impact on the overall results.

VI. Modeling of Nonsaturating Decoders

An original motivation for this study was the observation that, for the Margulis code, error-rate results generated by simulation of a nonsaturating decoder did not exhibit the error-floor behavior seen in previously published error-rate curves. In this section, we use the state-space model to investigate this phenomenon. We first apply density evolution (DE) to generate the relevant model inputs corresponding to nodes outside of the TS. Then, we refer back to the divergence condition of (32) to see if the model predicts that error floors are produced.

DE assumes that the Tanner graph has no cycles and that the block length of the code is infinite. For our purposes, this is equivalent to assuming independence among incoming LLRs, as in Assumption 10.

SPA decoding of LDPC codes exhibits a threshold phenomenon as the block length and the number of iterations tend to infinity. The error rate drops very dramatically as the SNR exceeds the *decoding threshold*, a value which can be found using DE [53]. As we are interested in the behavior of the error floor in this work, we assume that the channel SNR is always above the decoding threshold.

In [28], Sun asymptotically characterized the growth of the mean extrinsic check-node output, $m_{\lambda(\mathrm{ex})}$, from iteration $l-1$ to l, showing that

$$m_{\lambda(\mathrm{ex})}^{(l)} = (d_v - 1) m_{\lambda(\mathrm{ex})}^{(l-1)} + \text{“some small value terms.”} \quad (33)$$

In [61], we refine Sun's analysis and develop bounds on the mean extrinsic check-node LLR required, as a function of SNR, such that the growth rate of $m_{\lambda(\mathrm{ex})}^{(l)}$ exceeds that of the internal TS LLRs for all future iterations. In either DE analysis, in the high E_b/N_0 regime,

$$m_{\lambda(\mathrm{ex})}^{(l)} > Cr^l$$

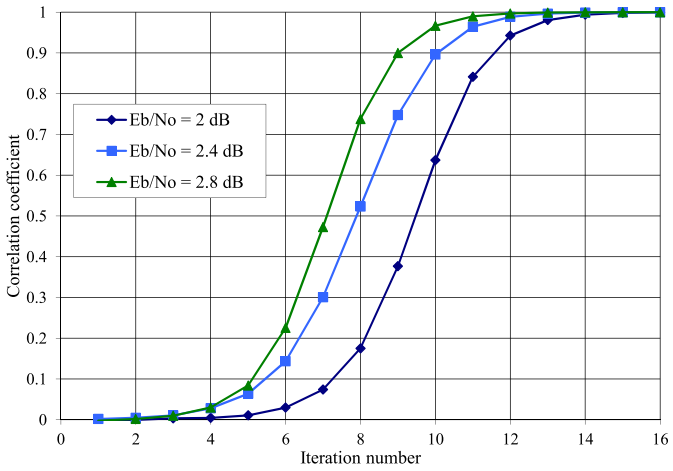


Fig. 14. Correlation coefficient among the four LLR check-node output metrics used as prediction model inputs versus iteration number for $(12, 4)$ TSs of the Margulis code as measured in LLR-domain SPA simulation.

for every positive constant C, any r such that $1 \leq r < d_v - 1$, and sufficiently large l.

According to (32), this growth rate comparison implies that every potential (non-codeword) elementary trapping-set error can be corrected by a nonsaturating LLR-domain SPA decoder after a sufficient number of iterations, provided we can rely upon DE as a model for LLR growth outside of the TS for variable-regular ($d_v \geq 3$) LDPC codes. (Note that $d_v \geq 3$ appears here because $d_v = 2$ would not produce exponential growth in (33).)

In Fig. 13 we compare the mean check-node output LLRs produced by the SPA decoder simulation of the $(2640, 1320)$ Margulis code to those generated by the application of DE to the ensemble of $(3, 6)$-regular LDPC codes. In the decoder simulation, the all-zero codeword is transmitted over the AWGN channel and early termination of the decoder is disabled. We note that the mean LLR from the simulation closely approximates the mean LLR predicted by DE during the first 20 iterations. The variance of the simulated check-node output LLRs is in agreement with DE for the first seven iterations, but then deviates significantly. Through the first seven iterations, the variance is approximately twice the mean, as the Gaussian approximation to DE would predict. By the ninth iteration, however, the simulated variance is better approximated by the square of the mean. As discussed in Section IV, such quadratic growth of the variance relative to the mean can result in an error floor, with the argument of the Q-function in (25) approaching a finite value as the number of iterations increases.

The cause of the departure of the simulated variance's behavior from that predicted by DE is the positive correlation that develops among the LLRs after several decoder iterations. For the $(12, 4)$ TSs of the Margulis code, Fig. 14 shows the average of the off-diagonal values from the 4×4 matrix of correlation coefficients of the four LLR check-node output metrics used as inputs to the state-space model, as a function of the iteration number. Significant positive correlation can already be seen in the sixth iteration and increases rapidly. Thus, the assumptions used to justify the expression for $\mathrm{VAR}[\beta'_l]$ in (31) do not hold in this case. For the $(640, 192)$ QC code discussed in Section V, we found similar behavior of the variance and LLR correlation. In fact, the onset came after even fewer iterations, most likely due the smaller girth of the code relative to that of the Margulis code (6 vs. 8).

A further source of inaccuracy in the probability of error model (25) is the implicit assumption that the skewness, *i.e.*, the third standardized moment, of β'_l is zero, a consequence of its assumed Gaussianity. In our simulation of the Margulis code, we measured skewness of β'_l in the range of 0.6 to 1.8. This deficiency in the modeling assumptions can also have a significant effect on error floor predictions.

The problems discussed above limit the usefulness of the model in predicting the performance of the nonsaturating SPA decoder beyond the seventh iteration for the Margulis code. However, in the next section, we describe a modification of Richardson's semi-analytic technique that may offer a way to more accurately estimate error floors of nonsaturating decoders.

VII. Modification of Richardson's Semi-Analytical Technique for Nonsaturating Decoders

In this section, we briefly review Richardson's semi-analytical technique for estimating error floors caused by TSs [8]. We then describe modifications intended to improve the effectiveness of the technique when applied to nonsaturating SPA decoders. Numerical results are presented for the Margulis code.

Richardson's semi-analytical technique is a type of importance sampling; it essentially averages the results of several importance sampling experiments. (See [32], [50], and [62] for the use of importance sampling to estimate the error rates of particular TSs.) Richardson's technique biases the noise in the direction of the TS by a varying amount s. Richardson effectively utilizes an orthogonal transform on the i.i.d. Gaussian noise samples within the TS to represent its noise with a new set of basis vectors in which each dimension is independent Gaussian noise. This allows him to treat the bias amount s as a Gaussian random variable, while performing Monte Carlo simulation on the other $n - 1$ dimensions of the noise with variance σ^2.

Richardson fixes the bias amount s to several negative values and runs simulations, finding the conditional TS failure probability $\Pr\{\xi_{\mathcal{T}}|s\}$ for each s value. Then, the overall TS failure probability $\Pr\{\xi_{\mathcal{T}}\}$ may be computed using the law of total probability, *i.e.*,

$$\Pr\{\xi_{\mathcal{T}}\} = \int \Pr\{\xi_{\mathcal{T}}|s\} p_S(s)\, ds. \tag{34}$$

Due to the high degree of overlap among TSs, when counting failures for $\Pr\{\xi_{\mathcal{T}}|s\}$, Richardson only counts a frame as a failure if the set of symbols which are not eventually correct exactly matches the symbol locations of $\mathcal{T}$, the specific TS under test.

The dominant TSs of the $(2640, 1320)$ Margulis LDPC code are the $(12, 4)$ and $(14, 4)$ elementary TSs, shown in Figs. 10a and 10b [7], [8], [23]. In the case of the $(14, 4)$ TS of the Margulis code, the not-eventually-correct symbols exactly

match the symbols in $\mathcal{T}$ when testing the saturated decoder at a rate of about 3 in 4 failures or greater (at $s = -1.35$). Note that this rate drops substantially as s approaches -1. However, for our nonsaturating decoder, this ratio became less than 1 in 10^5 (at $s = -1.35$), indicating that the not-eventually-correct condition is not effective when saturation is eliminated. This ratio was so low when running Richardson's technique for $\mathcal{T}$ set to a $(12, 4)$ TS without saturation that we could not measure it. When saturation is present, failures are generally forced to occur on absorbing sets, and this is often the set under test when $s \leq -1.25$. In the absence of saturation, the failures occur on a wide variety of Tanner subgraphs. Most of the failing subgraphs during nonsaturating simulation are not absorbing sets and most are larger than the original a variables under test. In fact, the two smallest failing structures that we captured at high SNR were the $(14, 4)$ absorbing set of Fig. 10b and the $(18, 8)$ subgraph shown in Fig. 10c. Since the set of general Tanner subgraphs within the Margulis code is vast [14], computing the error floor estimate for each type of subgraph with such a low capture rate would be impractical. An alternative approach is needed.

One possible approach would be to ignore the not-eventually-correct criterion and simply count all failures, at the risk of overestimating the error floor. Instead, we propose to estimate the error floor by introducing saturation in the final iterations of failed frames, so that they may fail on absorbing sets. For example, iterations with LLR saturation would force corrections to the labeled variable nodes of Fig. 10c, as the unsatisfied check nodes outnumber the satisfied checks, driving the failed structure to the $(14, 4)$ absorbing set contained within. Since this approach reduces the number of incorrect bits per failure, it perhaps has meaningful applications. It appears effective, as we see that most failures are reduced to small absorbing sets in practice. We prefer this strategy because we are interested in establishing a lower bound on the floor for the nonsaturating case.

As there may be many variable nodes to correct to get to the absorbing set contained within the failed structure, we allow for 20 iterations of saturated SPA decoding before beginning the "eventually correct" logic which runs for 12 additional iterations. We chose to run the saturation phase at an LLR limit of ± 25.

Fig. 15 shows results for saturating and nonsaturating decoders. All simulations were run for a maximum of 200 iterations, with the exception of the nonsaturating semi-analytical simulations which were run for 32 additional iterations as described above. In all cases, the semi-analytical technique was applied at $E_b/N_0 = 2.8$ dB, and the results were extrapolated locally as suggested in [8].

Richardson's extrapolation assumes that the conditional failure probability $\Pr\{\xi_{\mathcal{T}}|s\}$ is insensitive to SNR changes. Additionally, note that the extrapolations appear to produce parallel lines in the log-log plot (*i.e.*, the logarithm of the error rate versus the E_b/N_0 measured in decibels, which is also logarithmic). This parallel line effect is due to the nature of the integration in (34). The conditional failure probability within the integrand is largely about 5 standard deviations of s below the mean value of s for the cases presented here. Thus, small changes to the standard deviation of s in (34) produce a multiplicative effect on the estimated error rate, regardless of the decoder's specific saturation level.

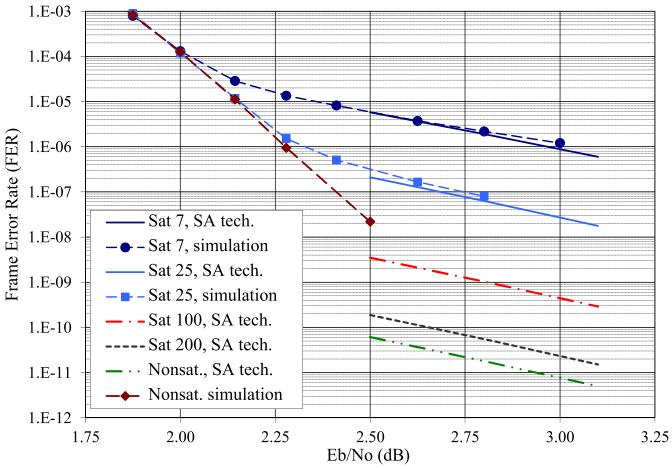


Fig. 15. FER vs. E_b/N_0 in dB for Margulis LDPC Code in AWGN. The semi-analytical (SA) technique only estimates errors due to the $(12, 4)$ and $(14, 4)$ TSs at $E_b/N_0 = 2.8$ dB.

For the nonsaturating decoder, Fig. 15 shows an estimated FER of 2×10^{-11} at $E_b/N_0 = 2.8$ dB. This is slightly lower than the FER of 6×10^{-11} found when the decoder saturates the check-node output LLR magnitudes at 200. (In making this error floor prediction, we have assumed that no other TSs become dominant in these conditions.)

Several additional observations are worth noting. First, increasing the maximum number of iterations of the nonsaturating decoder simulation to 1000 reduces the error floor by about a factor of 3.

Second, we have observed that when saturation levels are increased, the error floor is more sensitive to SNR than Richardson's extrapolation suggests. We have applied the semi-analytical techniques of this section to estimate the combined FER contributions of the $(12, 4)$ and $(14, 4)$ TSs to be 4×10^{-17} FER at $E_b/N_0 = 4$ dB. This shows that the error floor falls significantly faster than Richardson's extrapolation predicts, since the conditional failure probability $\Pr\{\xi_{\mathcal{T}}|s\}$ appears to decrease significantly with increasing SNR. From the DE expressions, at higher SNR, the beneficial LLR growth starts earlier and grows faster giving the code beyond the TS a higher probability to correct a channel error along a TS for a nonsaturating decoder. However, an SPA decoder with limited LLR dynamic range will not leverage these effects that depend on SNR. This effect is also illustrated in Fig. 11 for the greatest saturation limit.

Third, we have observed a shift in which TSs dominate the error floor as saturation levels are increased. Table IV shows the ratio of the error floor contributions of the $(12, 4)$ TS to the contributions of the $(14, 4)$ TS at $E_b/N_0 = 2.8$ dB. We note that when saturation limits are low the $(12, 4)$ TS dominates the error floor, but when the limits are raised the $(14, 4)$ TS dominates. There are two potential reasons for this. The first is that the $(12, 4)$ TS, with its smaller maximum eigenvalue value, is more sensitive to saturation

TABLE IV
RATIO OF (12, 4) TS ERROR FLOOR CONTRIBUTION TO (14, 4) FOR THE MARGULIS CODE AT $E_b/N_0 = 2.8$ dB USING RICHARDSON'S SEMI-ANALYTICAL TECHNIQUE

Decoder	Ratio
Saturated SPA, at ± 7 CN out	$4.6:1$
Richardson's 5-bit [8]	$3.3:1$
Saturated SPA, at ± 15 CN out	$3.3:1$
Saturated SPA, at ± 25 CN out	$1.7:1$
Saturated SPA, at ± 50 CN out	$0.7:1$
Saturated SPA, at ± 100 CN out	$0.6:1$
Saturated SPA, at ± 200 CN out	$0.46:1$
Nonsaturating SPA decoder	$0.4:1$

level—that is, as the saturation limits are raised an initially erroneous (12, 4) TS becomes increasing likely to be corrected by the extrinsic LLRs, more so than an an initially erroneous (14, 4) TS. The second potential reason is that we noticed that a growing fraction of (12, 4) TS failures are subsumed by the (14, 4) absorbing sets which contain them as LLR limits are raised. Additionally, we notice that the (14, 4) TSs dominate even more as the SNR is increased for a nonsaturating decoder.

Finally, we would like to know if we are actually seeing an error floor in the nonsaturating SPA simulation results. Extrapolating the FER of the waterfall region of Fig. 15 out to $E_b/N_0 = 4.0$ dB using a constant log-log slope yields an FER of about 5×10^{-20}. This is significantly smaller than the FER contribution of the (12, 4) and (14, 4) TSs, which we estimated above to be 4×10^{-17}. Therefore, preliminary evidence predicts that we will indeed see an error floor, but it would begin at a much lower FER and would have about twice the slope as a function of SNR in a log-log plot when compared to the previously reported error floor estimate for the Margulis code [7], [8], [23]. Specifically, comparing to prior SPA simulation results, our error floor estimate is about 5 orders of magnitude lower at $E_b/N_0 = 2.8$ dB and about 8 orders of magnitude lower at 4.0 dB.

VIII. CONCLUSION

We have reexamined the error floor levels previously published for LDPC codes over the AWGN channel and their root cause ([7], [8], and [23]). We have shown that the error floor levels were caused by the interaction of non-codeword TSs and the effects of constraining highly-certain messages in a belief propagation decoder, such as the LLR-domain SPA decoder. It is likely that the saturation of LLRs in prior work, which was sometimes implicit, was used to avoid numerical implementation issues that we have noted. We have shown that when care is taken in processing those messages the error floor level is lowered by many orders of magnitude for several specific LDPC codes.

We have reviewed and refined a linear state-space model of TS behavior which predicts the level of the error floor. We have added some clarity to the situation of two distinctly different applications of this state-space model. On one hand, Sun introduced the model and proved that error floors do not exist when LLRs are not limited and density evolution produces valid LLR statistics [28]. On the other hand, Schlegel and Zhang improved upon the model and predicted nonzero error floors due to the dominant (8, 8) TSs of the 802.3an code [11].

With respect to Sun's work, we are in agreement with Sun's conclusion as applied to variable-regular LDPC codes without cycles. We have shown empirically that the assumptions in the probability of error of the state-space model do not hold for codes with cycles. For these codes, we have shown that the effects of positively correlated LLRs when allowed to grow (without saturation) drive the ratio of the mean to standard deviation to a finite value possibly implying an error floor. This error floor level is difficult to estimate due to the positively correlated LLRs and the non-Gaussian distribution of our error indicator.

We have also been able to put Schlegel and Zhang's work into a larger context. Their error rate results showing a nonzero error floor for a specific TS were for a decoder that saturated (*i.e.*, "clipped") the LLR values. Their addition of mean gains to the model made a substantial improvement to error floor prediction as we have shown. Our realization of the state-space model is more general than theirs, in that ours does not require the graphical regularity such that the dominant root of the model, $\rho(\mathbf{A})$, is limited to the integral value $d_v - 2$. Additionally, we have simplified considerably the incorporation of polarity inversion by the degree-two check nodes into the model.

We have presented two new methods to collect the statistics needed to drive the model. We have shown that the reason that the model predicts saturated performance relatively well is that by the iteration count at which LLRs become substantially correlated, their variance is driven to zero by the effect of saturation. This means that the model derived for the case with no cycles actually works rather well for the case with cycles when there is a significant saturation effect present. In our experiments the model with modified mean gains estimated the FER floor of floating-point simulation to within 0.2 dB for several different LDPC codes.

Also, we have reached an improved understanding of the dynamics of TS decoding failure in SPA decoders. Chen *et al.* in [25] made the insightful observation, "…that the error floor is caused by the combined effects of short cycles in the graph representation of the code and of clipping." To further understand this relationship, we have revisited Richardson's semi-analytic error floor estimation technique to measure the error floors of the Margulis code without the presence of "clipping." We estimated that the total error floor contribution of the dominant TSs is reduced by 5 to 8 orders of magnitude and this error floor contribution has twice the slope as a function of SNR (in a log-log plot) than prior results with saturating decoders. This leads us to speculate that channel errors corresponding to absorbing sets can be frequently corrected when the beneficial unsatisfied check LLRs are allowed to eventually overpower the TS's own detrimental LLR growth. This reasoning also helps to explain the observed slope deviations from Richardson's error-floor extrapolation technique as LLR limits are substantially raised.

Additionally, when LLR saturation is removed from the decoder, we have observed that channel errors corresponding to TSs are not "stable" structures as reported by Ryan and Lin, who stated that TS failures converge very rapidly, often in less than ten iterations (§15.3.1 of [41]). Ryan and Lin's observation is attributable to decoders with metric saturation. When the messages reach the saturation point, the dynamics of the SPA decoder degenerate to that of a bit flipping decoder. We have observed that, for the Margulis code and a nonsaturating decoder, the TS failure error rate was reduced by a factor of three by increasing the number of maximum iterations from 200 to 1000.

The results make clear that when presenting error floor results the documentation of any LLR saturation is important.

Future work may include extensions to more channel models, investigations into non-elementary TSs, and an accounting of the correlations among LLRs. We hope that several of the techniques covered in this paper will be useful for variable-irregular LDPC codes, but recognize that challenges await.

APPENDIX A
TABLES OF ABSORBING SET STRUCTURE

As opposed to studying the TSs that dominate several specific codes, in this appendix we present an overview of all subgraphs that may become error-prone trapping sets which satisfy specific conditions for particular large classes of codes. This can offer some perspective on the structural parameters that have been discussed in this paper such as: a, b, the index of imprimitivity h, and the spectral radius r.

The variable-regular codes we examine are assumed to be described by Tanner graphs that are 4-cycle-free. We group the information into tables according to their variable-degree d_{v}. The further conditions we put on the subgraphs of interest are described in Assumptions 2 and 7 and Definition 3: the subgraphs must be elementary, connected and absorbing, respectively. We will characterize subgraphs as stronger or weaker according to the relative magnitude of their spectral radius, *i.e.*, a "stronger" TS has a relatively greater value of r.

Instead of working on Tanner graphs directly, we would prefer to work with their corresponding multigraphs which are less complex. The following lemma shows that there is a one-to-one mapping between both sets.

Lemma 3: There exists a bijective map between the set of d_{v}-variable-regular elementary Tanner subgraphs (up to a relabeling of the check nodes) and the set of multigraphs $G = (V, E)$ with vertex degrees upper bounded by d_{v}, i.e., $d(v_i) \leq d_{\mathrm{v}} \forall\, v_i \in V$.

Proof: Given the elementary Tanner subgraph $B_{\mathcal{S}} = (\mathcal{S}, \mathcal{N}(\mathcal{S}), E_{\mathcal{S}})$, each variable node $v \in \mathcal{S}$ becomes a vertex $v \in V$ of G, that is $V = \mathcal{S}$, and the degree-one check nodes are discarded. The check nodes of degree two become the edges of G, each joining a pair of vertices.

Given the multigraph G, each vertex $v \in V$ of G becomes a variable node $v \in \mathcal{S}$ of $B_{\mathcal{S}}$. So, again $V = \mathcal{S}$. Each edge of G is replaced by a degree-two check node (with an arbitrary label) and two edges in $B_{\mathcal{S}}$. Finally, degree-one check nodes are attached with single edges to variable nodes as needed until every variable node has d_{v} neighbors. ■

Rather than find every single graph that satisfies our parameters we can simplify the search considerably. As we are just interested in a graph's structure as opposed to its labeling we need only identify one of the graphs among several that are isomorphic to the others. In fact, we are partitioning the graphs into equivalence classes induced by graphical isomorphism.

Furthermore, we will limit our search to *simple graphs*, which have neither self-loops nor parallel edges. By limiting the search to simple graphs, we eliminate the multigraphs of girth 2 and their corresponding Tanner subgraphs containing 4-cycles.

Computational graph theory tools such as "geng" [63] can very quickly generate non-isomorphic simple graph descriptions satisfying sets of parameters such as ours. We run this tool once for each row of the tables. Each time we configure it to find undirected graphs of order a, size $(ad_{\mathrm{v}} - b)/2$, and with vertex degrees in the interval $[\lceil d_{\mathrm{v}}/2 \rceil, d_{\mathrm{v}}]$. The first two conditions were developed in the beginning of Section III-C, while the third condition follows from the definition of absorbing sets with respect to the multigraph representation. We then process each graph in a custom tool that converts it to the adjacency matrix of the associated directed graph to find its spectral radius r and index of imprimitivity h.

Table V shows the graphs found corresponding to the set of 4-cycle-free codes with $d_{\mathrm{v}} = 3$. The graphical equivalence classes found are divided into rows by their (a, b) parameters listed in the first column. The second column of each row presents the number of equivalence classes found that satisfy the (a, b) parameters and all our other assumptions. The third column shows the maximum index of imprimitivity h over the (a, b) equivalence classes. The fourth and fifth columns show the minimum and maximum spectral radius r over the (a, b) equivalence classes. To save space we have left out the weakest (a, b) pairs with $r_{\max} \leq 1.3$. Tables VI and VII present similar results based on codes with $d_{\mathrm{v}} = 4$ and 5, respectively, omitting the (a, b) pairs with $r_{\max} \leq 2.6$ for $d_{\mathrm{v}} = 5$.

These tables present enough information to comment on the approximations to the spectral radius r developed in Section III-C. The extreme relative estimation errors are summarized in Table VIII. We find empirically that the approximation of (21) does not overestimate the spectral radius, but generally underestimates it. For 90% of the $d_{\mathrm{v}} = 3$ equivalence classes, this approximation is no more than 6.0% below the true value of the spectral radius. The approximation of (22) generally produces larger estimates of the spectral radius, sometimes overestimating the true value. For the $d_{\mathrm{v}} = 3$ equivalence classes shown in Table V, the mean approximation error is -3.5% for (21) and $+1.3\%$ for (22). For both estimators, we measure the standard deviation of the relative error to be about 2.0%. Since we find the correlation coefficient between the relative errors of these two estimates

TABLE V
CONNECTED, ELEMENTARY ABSORBING SETS FROM THE SET OF 4-CYCLE-FREE $d_v = 3$ VARIABLE-REGULAR CODES, WITH $r_{max} > 1.3$

(a, b)	Num.	h_{max}	r_{min}	r_{max}
4,0	1	1	2	2
4,2	1	1	1.521	1.521
5,1	1	1	1.829	1.829
5,3	2	4	1.414	1.424
6,0	2	2	2	2
6,2	4	2	1.696	1.729
6,4	4	2	1.348	1.361
7,1	4	1	1.883	1.888
7,3	10	2	1.599	1.665
7,5	6	2	1.298	1.316
8,0	5	2	2	2
8,2	19	2	1.780	1.870
8,4	25	2	1.521	1.622
9,1	19	1	1.911	1.929
9,3	63	2	1.696	1.851
9,5	52	2	1.463	1.592
10,0	19	2	2	2
10,2	113	2	1.829	1.911
10,4	198	2	1.629	1.841
10,6	109	4	1.414	1.570
11,1	114	1	1.929	1.947
11,3	482	2	1.758	1.899
11,5	536	2	1.571	1.836
11,7	197	2	1.379	1.555
12,0	85	2	2	2
12,2	835	2	1.861	1.940
12,4	1,892	2	1.696	1.894
12,6	1,373	2	1.521	1.833
12,8	351	2	1.348	1.545
13,1	839	1	1.941	1.961
13,3	4,541	2	1.799	1.934
13,5	6,374	2	1.645	1.891
13,7	3,159	2	1.481	1.831
13,9	581	2	1.321	1.537
14,0	509	2	2	2
14,2	7,589	2	1.883	1.954
14,4	21,434	2	1.745	1.931
14,6	19,587	2	1.599	1.889
14,8	6,879	2	1.446	1.830
14,10	931	4	1.298	1.532

TABLE VI
CONNECTED, ELEMENTARY ABSORBING SETS FROM THE SET OF 4-CYCLE-FREE $d_v = 4$ VARIABLE-REGULAR CODES

(a, b)	Num.	h_{max}	r_{min}	r_{max}
4,4	1	1	2	2
5,0	1	1	3	3
5,2	1	1	2.629	2.629
5,4	1	1	2.219	2.219
6,0	1	1	3	3
6,2	2	1	2.697	2.710
6,4	3	1	2.355	2.367
6,6	2	2	2	2
7,0	2	1	3	3
7,2	7	1	2.744	2.762
7,4	11	2	2.449	2.480
7,6	4	1	2.159	2.160
8,0	6	2	3	3
8,2	28	2	2.778	2.805
8,4	50	2	2.525	2.585
8,6	28	2	2.272	2.296
8,8	5	2	2	2
9,0	16	1	3	3
9,2	126	1	2.805	2.850
9,4	285	2	2.584	2.728
9,6	177	2	2.355	2.429
9,8	27	1	2.126	2.141
10,0	59	2	3	3
10,2	719	2	2.826	2.886
10,4	1,915	2	2.629	2.821
10,6	1,404	2	2.422	2.648
10,8	298	2	2.219	2.313
10,10	19	2	2	2
11,0	265	1	3	3
11,2	4,721	1	2.843	2.903
11,4	14,569	2	2.667	2.852
11,6	12,458	2	2.478	2.788
11,8	3,231	2	2.295	2.458
11,10	208	1	2.104	2.127

to be 0.732, we may reduce the standard deviation and bias by combining them. In summary, it appears that (22) and the combined estimator are marginally better estimators than (21).

As discussed in Section III-C, the approximation (22) assumes that the TS contains mostly zero or one unsatisfied check node per variable node, which is not always true in Table VII (*i.e.*, $d_v = 5$). When the assumption does not hold, the approximation of (22) can underestimate the spectral radius significantly, as demonstrated by the extreme value within Table VIII. In this case, the -20.2% error occurred over the relatively weak (10, 16) TSs. If we limit the processing of Table VII to its rows such that $b \leq a$, *i.e.*, the strong TSs, then the extreme relative error for this estimator would be reduced to -12.9%, as shown in the final row of Table VIII.

APPENDIX B
REDUCIBLE SYSTEM MATRICES $\mathbf{A}$

In this appendix we address the case of reducible system matrices $\mathbf{A}$. We will find that the only reducible system matrices allowed by the prior assumptions in Section III-C are associated with multigraphs which are cycle graphs, which we now define.

A *cycle graph*, denoted C_n, is a graph of order $n \geq 2$, in which the distinct vertices of the set $\{v_1, v_2, \ldots, v_n\}$ are joined by the edges from the collection $\{\{v_1, v_2\}, \ldots, \{v_{n-1}, v_n\}, \{v_n, v_1\}\}$. A cycle graph is a single cycle.

Example 6: The multigraph corresponding to the Tanner subgraph of Fig. 2d is the cycle graph of order 4, C_4.

Lemma 4: A $(0, 1)$-matrix is irreducible if and only if the associated digraph is strongly connected.

Proof: See [36, p. 78]. ■

Lemma 5: Let G be a connected multigraph of order $n \geq 2$. Then G is isomorphic to the cycle graph C_n if and only if every vertex has degree two.

Proof: Omitted. ■

Theorem 6: Consider a multigraph G which satisfies Assumptions 7 and 8. G is either a cycle graph or has an associated $\mathbf{A}(D)$ matrix that is irreducible.

Proof: Multigraphs with any vertices of degree one are not allowed by Assumption 8. Connected multigraphs with all vertices of degree two are cyclic by Lemma 5. So now we will show any possible remaining connected multigraphs must have an irreducible adjacency matrix $\mathbf{A}(D)$.

The remaining multigraphs must have at least one vertex of degree greater than two. Also, all vertex degrees must sum

TABLE VII
CONNECTED, ELEMENTARY ABSORBING SETS FROM THE SET OF 4-CYCLE-FREE $d_v = 5$ VARIABLE-REGULAR CODES, WITH $r_{max} > 2.6$

(a, b)	Num.	h_{max}	r_{min}	r_{max}
5,5	1	1	3	3
5,7	1	1	2.629	2.629
6,0	1	1	4	4
6,2	1	1	3.693	3.693
6,4	2	1	3.360	3.403
6,6	4	1	3	3.112
6,8	5	1	2.697	2.767
7,1	1	1	3.875	3.875
7,3	5	1	3.596	3.669
7,5	14	1	3.307	3.427
7,7	23	1	3	3.130
7,9	25	1	2.744	2.827
8,0	3	1	4	4
8,2	16	1	3.775	3.827
8,4	68	1	3.522	3.673
8,6	165	1	3.271	3.465
8,8	252	2	3	3.208
8,10	232	2	2.778	2.918
8,12	124	2	2.525	2.619
9,1	28	1	3.904	3.905
9,3	276	1	3.693	3.769
9,5	1,151	2	3.464	3.665
9,7	2,541	2	3.243	3.521
9,9	3,284	2	3	3.289
9,11	2,541	2	2.805	3.071
9,13	1,150	2	2.584	2.751
10,0	60	2	4	4
10,2	1,188	2	3.822	3.870
10,4	8,435	2	3.626	3.789
10,6	27,706	2	3.421	3.810
10,8	49,991	2	3.220	3.717
10,10	53,884	2	3	3.440
10,12	35,721	2	2.826	3.191
10,14	14,308	2	2.629	3.009
10,16	3,224	2	2.422	2.652

TABLE VIII
EXTREME RELATIVE ESTIMATION ERRORS FOR THE SPECTRAL RADIUS

Originating Table	With respect to (21) and		With respect to (22) and	
	r_{min}	r_{max}	r_{min}	r_{max}
V ($d_v = 3$)	0.0%	−21.9%	6.1%	−16.4%
VI ($d_v = 4$)	0.0%	−12.0%	2.1%	−9.4%
VII ($d_v = 5$)	0.0%	−13.9%	1.0%	−20.2%[a]
VII with $b \leq a$	0.0%	−13.9%	1.0%	−12.9%

[a]Discovered over the relatively weak $(10, 16)$ TSs.

to an even number, as every edge joins two vertices. This yields a connected multigraph with at least two cycles. Since a walk without backtracking through a connected multigraph with two cycles is able to reverse direction, every edge may be visited in either direction. Thus, when the edges of G are expanded to be vertices of the digraph D, D will be strongly connected. Since D it is strongly connected, its associated adjacency matrix $\mathbf{A}(D)$ will be irreducible by Lemma 4. ∎

Of course, since $\mathbf{A}(D)$ is irreducible so must be $\mathbf{A}(D)^T$, which we use as the system matrix $\mathbf{A}$ in our state-space model. Therefore, the remainder of this appendix need only address the properties of cycle graphs. Relative to the variable-regular codes addressed in this paper, the cycle graph C_a corresponds to an (a, b) TS with $a \geq 2$ and $b = (d_v - 2)a$.

The expansion of the edges of cycle graph $G = C_a$ to the digraph D will produce two disconnected directed cycles of length a, one associated with the clockwise non-backtracking walk of G and one with the counter-clockwise non-backtracking walk of G. Thus, $\mathbf{A}(D)$ has a symmetric permutation that is $\mathbf{P}\,\mathbf{A}(D)\,\mathbf{P}^T = \left[\begin{smallmatrix} \mathbf{A}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_2 \end{smallmatrix}\right]$, in which $\mathbf{A}_1$ and $\mathbf{A}_2$ are each $a \times a$ irreducible permutation matrices that implement a circular shift by one position. All the eigenvalues of $\mathbf{A}_1$ and $\mathbf{A}_2$ lie on the unit circle. The dominant eigenvalues of $\mathbf{A}_1$ and $\mathbf{A}_2$ are $\rho(\mathbf{A}_1) = \rho(\mathbf{A}_2) = 1$ and their corresponding eigenvectors are the all-one vectors. Every eigenvector of $\mathbf{A}_1$ becomes an eigenvector of $\mathbf{P}\ \mathbf{A}(D)\ \mathbf{P}^T$ by appending a zeros.

It is now easy to show that $\mathbf{w}_1^T = [1, 1, \ldots 1]$ is a left eigenvector of $\mathbf{A}$ corresponding to the dominant eigenvalue of $\mathbf{A}$, which is $r = 1$. However, when $r = 1$ the first partial sums within (30) and (31) do not absolutely converge with increasing iterations. We now address this issue.

For any TS with an associated reducible matrix $\mathbf{A}$ satisfying Assumptions 7 and 8, we can also show that all the column sums of $\mathbf{B}$ and $\mathbf{B}_{ex}$ are two. With these values of r, $\mathbf{w}_1^T$, and the column sums, (30) simplifies to

$$\mathbb{E}[\beta_l'] = 2a\,m_\lambda \left(1 + \sum_{i=1}^{l} \frac{1}{\prod_{j=1}^{i} \bar{g}_j'}\right) + 2a\,(d_v - 2) \sum_{i=1}^{l} \frac{m_{\lambda(ex)}^{(i)}}{\prod_{j=1}^{i} \bar{g}_j'} \tag{35}$$

and (31) simplifies to

$$\mathrm{VAR}[\beta_l'] = 8a\,m_\lambda \left(1 + \sum_{i=1}^{l} \frac{1}{\prod_{j=1}^{i} \bar{g}_j'}\right)^2 + 4a\,(d_v - 2) \sum_{i=1}^{l} \frac{\sigma_i^2}{(\prod_{j=1}^{i} \bar{g}_j')^2}. \tag{36}$$

This lack of convergence poses no problem to finding the ratio of $\mathbb{E}[\beta_l']$ in (35) to the square root of $\mathrm{VAR}[\beta_l']$ in (36), which is what we require for our prediction model. In fact, this ratio simplifies significantly if we can assume that the LLRs are saturated. In the saturated decoding case, the partial sum with σ_i^2 in (36) provides little contribution and we ignore it. Then,

$$\lim_{l\to\infty} \frac{\mathbb{E}[\beta_l']}{\sqrt{\mathrm{VAR}[\beta_l']}} = \sqrt{\frac{a}{2\,m_\lambda}} \left(m_\lambda + (d_v - 2) \cdot \lim_{l\to\infty} m_{\lambda(ex)}^{(l)}\right). \tag{37}$$

We note that in (37), the value at which the extrinsic LLR magnitudes saturate, $\lim_{l\to\infty} m_{\lambda(ex)}^{(l)}$, plays an important role in determining the probability of failure for this type of TS in a variable-regular LDPC code.

APPENDIX C
MULTIGRAPHS WITH LEAVES AND BRANCHES

We now expand the set of multigraphs addressed in this work to include those with leaves and branches, which were previously eliminated by Assumption 8. Thus, the variable

nodes in the associated Tanner subgraphs will be permitted to have $d_v - 1$ neighboring degree-one check nodes. In [28], Sun referred to the graphs addressed in this appendix as "TSs with external data nodes."

We will describe the elementary Tanner graphs of this appendix from the perspective of their associated multigraphs using the mapping described in Section III-C. Let a *base graph* be a multigraph that satisfies Assumptions 2, 7, and 8. Such a graph contains one or more cycles. A *leaf* is a vertex of degree one. A *branch* is a vertex of degree two or more outside of the base graph, and is not contained in any cycles.

Example 7: The TS in Fig. 5a contains one leaf on a base graph and the TS in Fig. 5b contains just three leaves and no base graph. The $(18, 8)$ Tanner subgraph of Fig. 10c has leaves A, B, and D and branch C on a base graph that is the $(14, 4)$ TS of Fig. 10b.

Assumption 11 (Replacement for Assumption 8): Multigraphs of interest must contain a base graph and may contain leaves and branches.

We require a base graph because without one the eigenvalues would all be zero, as will become apparent shortly. Our new assumption allows the variable nodes within the associated Tanner subgraphs to have as many as $d_v - 1$ neighboring degree-one check nodes.

Theorem 7: Let $G = (V, E)$ be a multigraph satisfying Assumptions 2, 7, and 11, containing base graph G_B, n leaves and m branches, with $n \geq 1$ an $m \geq 0$. Let $D = (Z, A)$ and D_B be the digraphs associated (as described in Section III-C) with G and G_B, respectively. Then, the adjacency matrix $\mathbf{A}(D)$ is reducible and the set of its unique eigenvalues is the union of the eigenvalues of $\mathbf{A}(D_B)$ and zero.

Proof: Each additional leaf in G adds one edge to G. Let the edge $e_i \in E$ join leaf v_k to vertex v_j in G, where v_j is not a leaf. The edge e_i maps to vertices $z_i, z_i' \in Z$ in D. These vertices are not strongly connected to the digraph as $d^+(z_i) = 0$ and $d^-(z_i') = 0$, and hence $\mathbf{A}(D)$ must be reducible by Lemma 4.

Every leaf in G creates an all-zero column in $\mathbf{A}(D)$ due to $d^-(z_i') = 0$ and an all-zero row due to $d^+(z_i) = 0$. The all-zero columns may be symmetrically permuted to the left (*i.e.*, their corresponding vertices are relabeled with the lowest possible values) and the all-zero rows to the bottom to form

$$\mathbf{P}\,\mathbf{A}(D)\,\mathbf{P}^T = \begin{bmatrix} \mathbf{0} & \mathbf{Y}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{X} & \mathbf{Y}_2 \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}, \tag{38}$$

where the block diagonal contains square submatrices. The $n \times n$ zero matrices at both ends of the block diagonal correspond to the n leaves in G.

The eigenvalues of $\mathbf{A}(D)$ are the roots of the characteristic equation $\det(\mathbf{A}(D) - \mu\mathbf{I}) = 0$, which simplifies to $\det(\mathbf{X} - \mu\mathbf{I})\mu^{2n} = 0$ by use of the expansion by minors along every zero row and column. Thus, the eigenvalues of $\mathbf{A}(D)$ are the eigenvalues of $\mathbf{X}$ and $2n$ zeros. In case G contains branches, the removal of one layer of leaves exposes a new layer of leaves and the operations in this paragraph must be repeated until we reduce $\mathbf{X}$ to be a symmetric permutation of $\mathbf{A}(D_B)$, which will be irreducible except for the case addressed in Appendix B. ■

By showing that the nonzero eigenvalues are preserved by the addition of leaves and branches, it is simple to argue that Theorem 2, which bounds the spectral radius of $\mathbf{A}(D)$, still holds if Assumption 8 is replaced by Assumption 11.

The only weakness with respect to our prior development is that we can no longer assume that the left eigenvector $\mathbf{w}_1^T$ is positive, as $\mathbf{A}(D)$ may now be reducible. In practice, we have found that the new entries added to $\mathbf{w}_1^T$ by the addition of leaves and branches are half zero and half positive. This may be proved by applying the Subinvariance Theorem in [35, p. 23] to (38). The presence of zero entries in $\mathbf{w}_1^T$ weakens our prior claims on the error indicator $\beta_l \triangleq \mathbf{w}_1^T \mathbf{x}_l$. Now, the error indicator is most effective on the variable nodes corresponding to the base graph and less effective on the variable nodes corresponding to the branches and leaves.

Finally, we do not see much practical motivation to predict the error floors of graphs with branches and leaves. We would prefer to run predictions on the base graphs contained within. Since the graphs with branches and leaves have the same spectral radius as their base graph they are no more likely to fail, in theory. In fact, they should be less likely to fail as more channel values are involved and more unsatisfied check nodes are working to correct the error pattern associated with the graph.

ACKNOWLEDGMENT

The authors would like to thank Aravind Iyengar, Christian Schlegel, Roxana Smarandache, and Xiaojie Zhang for their helpful discussions and encouragement. They would like to thank Xiaojie Zhang for verifying several of our low error floor results in the AWGN channel using an independent implementation of the nonsaturating SPA decoder. They would also like to thank Yang Han and William Ryan for providing the parity-check matrix of the $(640, 192)$ QC code used in Section V. Finally, the authors thank the Associate Editor, Pascal Vontobel, for the many suggestions which improved this paper.

REFERENCES

[1] R. G. Gallager, "Low-density parity-check codes," *IRE Trans. Inf. Theory*, vol. 8, no. 1, pp. 21–28, Jan. 1962.

[2] R. G. Gallager, *Low-Density Parity-Check Codes*. Cambridge, MA, USA: MIT Press, 1963.

[3] D. J. C. MacKay and R. M. Neal, "Near Shannon limit performance of low density parity check codes," *Electron. Lett.*, vol. 32, no. 18, pp. 1645–1646, Aug. 1996.

[4] N. Wiberg, "Codes and decoding on general graphs," Ph.D. dissertation, Dept. Elect. Eng., Linköping Univ., Linköping, Sweden, Apr. 1996.

[5] N. Wiberg, H.-A. Loeliger, and R. Kötter, "Codes and iterative decoding on general graphs," *Eur. Trans. Telecommun.*, vol. 6, no. 5, pp. 513–525, Sep./Oct. 1995.

[6] C. Di, D. Proietti, I. E. Telatar, T. J. Richardson, and R. L. Urbanke, "Finite-length analysis of low-density parity-check codes on the binary erasure channel," *IEEE Trans. Inf. Theory*, vol. 48, no. 6, pp. 1570–1579, Jun. 2002.

[7] D. J. C. MacKay and M. S. Postol, "Weaknesses of Margulis and Ramanujan–Margulis low-density parity-check codes," *Electron. Notes Theoretical Comput. Sci.*, vol. 74, pp. 97–104, Oct. 2003.

[8] T. Richardson, "Error-floors of LDPC codes," in *Proc. 41st Annu. Allerton Conf.*, Monticello, IL, USA, Oct. 2003, pp. 1426–1435.

[9] D. V. Nguyen, S. K. Chilappagari, M. W. Marcellin, and B. Vasić, "On the construction of structured LDPC codes free of small trapping sets," *IEEE Trans. Inf. Theory*, vol. 58, no. 4, pp. 2280–2302, Apr. 2012.

[10] C.-C. Wang, S. R. Kulkarni, and H. V. Poor, "Finding all small error-prone substructures in LDPC codes," *IEEE Trans. Inf. Theory*, vol. 55, no. 5, pp. 1976–1999, May 2009.

[11] C. Schlegel and S. Zhang, "On the dynamics of the error floor behavior in (regular) LDPC codes," *IEEE Trans. Inf. Theory*, vol. 56, no. 7, pp. 3248–3264, Jul. 2010.

[12] S. Abu-Surra, D. Declercq, D. Divsalar, and W. E. Ryan, "Trapping set enumerators for specific LDPC codes," in *Proc. Inf. Theory Appl. Workshop (ITA)*, San Diego, CA, USA, Feb. 2010, pp. 1–5.

[13] X. Zhang and P. H. Siegel, "Efficient algorithms to find all small error-prone substructures in LDPC codes," in *Proc. IEEE Global Telecommun. Conf.*, Houston, TX, USA, Dec. 2011, pp. 1–6.

[14] M. Karimi and A. H. Banihashemi, "Efficient algorithm for finding dominant trapping sets of LDPC codes," *IEEE Trans. Inf. Theory*, vol. 58, no. 11, pp. 6942–6958, Nov. 2012.

[15] O. Milenkovic, E. Soljanin, and P. Whiting, "Asymptotic spectra of trapping sets in regular and irregular LDPC code ensembles," *IEEE Trans. Inf. Theory*, vol. 53, no. 1, pp. 39–55, Jan. 2007.

[16] S. Abu-Surra, D. Divsalar, and W. E. Ryan, "Enumerators for protograph-based ensembles of LDPC and generalized LDPC codes," *IEEE Trans. Inf. Theory*, vol. 57, no. 2, pp. 858–886, Feb. 2011.

[17] L. Dolecek, Z. Zhang, V. Anantharam, M. J. Wainwright, and B. Nikolić, "Analysis of absorbing sets and fully absorbing sets of array-based LDPC codes," *IEEE Trans. Inf. Theory*, vol. 56, no. 1, pp. 181–201, Jan. 2010.

[18] S. Laendner and O. Milenkovic, "Algorithmic and combinatorial analysis of trapping sets in structured LDPC codes," in *Proc. Int. Conf. Wireless Netw., Commun. Mobile Comput.*, vol. 1. Maui, HI, USA, Jun. 2005, pp. 630–635.

[19] Z. Zhang, V. Anantharam, M. J. Wainwright, and B. Nikolić, "An efficient 10GBASE-T Ethernet LDPC decoder design with low error floors," *IEEE J. Solid-State Circuits*, vol. 45, no. 4, pp. 843–855, Apr. 2010.

[20] A. I. Vila Casado, M. Griot, and R. D. Wesel, "LDPC decoders with informed dynamic scheduling," *IEEE Trans. Commun.*, vol. 58, no. 12, pp. 3470–3479, Dec. 2010.

[21] E. Sharon, O. Fainzilber, and S. Litsyn, "Decreasing error floor in LDPC codes by parity-check matrix extensions," in *Proc. IEEE Int. Symp. Inf. Theory*, Seoul, Korea, Jun./Jul. 2009, pp. 374–378.

[22] J. S. Yedidia, Y. Wang, and S. C. Draper, "Divide and concur and difference-map BP decoders for LDPC codes," *IEEE Trans. Inf. Theory*, vol. 57, no. 2, pp. 786–802, Feb. 2011.

[23] Y. Han and W. E. Ryan, "Low-floor decoders for LDPC codes," *IEEE Trans. Commun.*, vol. 57, no. 6, pp. 1663–1673, Jun. 2009.

[24] J. Thorpe, "Low-complexity approximations to belief propagation for LDPC codes," pp. 1–6, Oct. 2002. [Online]. Available: http://www.systems.caltech.edu/~jeremy/research/papers/

[25] J. Chen and M. P. C. Fossorier, "Density evolution for BP-based decoding algorithms of LDPC codes and their quantized versions," in *Proc. IEEE Global Telecommun. Conf.*, vol. 2. Taipei, Taiwan, Nov. 2002, pp. 1378–1382.

[26] J. Zhao, F. Zarkeshvari, and A. H. Banihashemi, "On implementation of min-sum algorithm and its modifications for decoding low-density parity-check (LDPC) codes," *IEEE Commun. Lett.*, vol. 53, no. 4, pp. 549–554, Apr. 2005.

[27] Z. Zhang, L. Dolecek, B. Nikolić, V. Anantharam, and M. J. Wainwright, "Design of LDPC decoders for improved low error rate performance: Quantization and algorithm choices," *IEEE Trans. Wireless Commun.*, vol. 8, no. 11, pp. 3258–3268, Nov. 2009.

[28] J. Sun, "Studies on graph-based coding systems," Ph.D. dissertation, Dept. Elect. Eng., Ohio State Univ., Columbus, OH, USA, 2004.

[29] J. Sun, O. Y. Takeshita, and M. P. Fitz, "Analysis of trapping sets for LDPC codes using a linear system model," in *Proc. 42nd Annu. Allerton Conf.*, Monticello, IL, USA, Sep./Oct. 2004, pp. 1701–1702.

[30] J. O. Brevik and M. E. O'Sullivan, "The sum-product algorithm for degree-2 check nodes and trapping sets," pp. 1–26, Nov. 2014. [Online]. Available: http://arxiv.org/abs/1411.2169

[31] H. Xiao, A. H. Banihashemi, and M. Karimi, "Error rate estimation of low-density parity-check codes decoded by quantized soft-decision iterative algorithms," *IEEE Trans. Commun.*, vol. 61, no. 2, pp. 474–484, Feb. 2013.

[32] S. Zhang and C. Schlegel, "Controlling the error floor in LDPC decoding," *IEEE Trans. Commun.*, vol. 61, no. 9, pp. 3566–3575, Sep. 2013.

[33] R. A. Horn and C. R. Johnson, *Matrix Analysis*. Cambridge, U.K.: Cambridge Univ. Press, 1985.

[34] C. D. Meyer, *Matrix Analysis and Applied Linear Algebra*. Philadelphia, PA, USA: SIAM, 2000.

[35] E. Seneta, *Non-negative Matrices and Markov Chains*. New York, NY, USA: Springer-Verlag, 1981.

[36] H. Minc, *Nonnegative Matrices*. New York, NY, USA: Wiley, 1988.

[37] R. A. Brualdi and H. J. Ryser, *Combinatorial Matrix Theory*. Cambridge, U.K.: Cambridge Univ. Press, 1991.

[38] R. J. Trudeau, *Introduction to Graph Theory*. Kent, OH, USA: Kent State Univ. Press, 1976.

[39] D. Cvetković, P. Rowlinson, and S. Simić, *An Introduction to the Theory of Graph Spectra*. Cambridge, U.K.: Cambridge Univ. Press, 2010.

[40] T. Etzion, A. Trachtenberg, and A. Vardy, "Which codes have cycle-free Tanner graphs?" *IEEE Trans. Inf. Theory*, vol. 45, no. 6, pp. 2173–2181, Sep. 1999.

[41] W. Ryan and S. Lin, *Channel Codes: Classical and Modern*. Cambridge, U.K.: Cambridge Univ. Press, 2009.

[42] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 498–519, Feb. 2001.

[43] *IEEE Standard for Floating-Point Arithmetic*, IEEE Standard 754-2008, Aug. 2008.

[44] X. Zhang and P. H. Siegel, "Quantized iterative message passing decoders with low error floor for LDPC codes," *IEEE Trans. Commun.*, vol. 62, no. 1, pp. 1–14, Jan. 2014.

[45] B. K. Butler and P. H. Siegel, "Numerical issues affecting LDPC error floors," in *Proc. IEEE Global Telecommun. Conf.*, Anaheim, CA, USA, Dec. 2012, pp. 3225–3231.

[46] J. Chen, A. Dholakia, E. Eleftheriou, M. P. C. Fossorier, and X.-Y. Hu, "Reduced-complexity decoding of LDPC codes," *IEEE Trans. Commun.*, vol. 53, no. 8, pp. 1288–1299, Aug. 2005.

[47] A. Anastasopoulos, "A comparison between the sum-product and the min-sum iterative detection algorithms based on density evolution," in *Proc. IEEE Global Telecommun. Conf.*, vol. 2. San Antonio, TX, USA, Nov. 2001, pp. 1021–1025.

[48] G. Richter, G. Schmidt, M. Bossert, and E. Costa, "Optimization of a reduced-complexity decoding algorithm for LDPC codes by density evolution," in *Proc. IEEE Int. Conf. Commun.*, vol. 1. Seoul, Korea, May 2005, pp. 642–646.

[49] S. Ländner, T. Hehn, O. Milenkovic, and J. B. Huber, "The trapping redundancy of linear block codes," *IEEE Trans. Inf. Theory*, vol. 55, no. 1, pp. 53–63, Jan. 2009.

[50] C. A. Cole, S. G. Wilson, E. K. Hall, and T. R. Giallorenzi, "A general method for finding low error rates of LDPC codes," pp. 1–30, 2006. [Online]. Available: http://arxiv.org/abs/cs/0605051

[51] Y. Zhang and W. E. Ryan, "Toward low LDPC-code floors: A case study," *IEEE Trans. Commun.*, vol. 57, no. 6, pp. 1566–1573, Jun. 2009.

[52] J. Hamkins, "Performance of low-density parity-check coded modulation," Jet Propul. Lab., Pasadena, CA, USA, Tech. Rep. 42-184, Feb. 2011.

[53] S.-Y. Chung, T. J. Richardson, and R. L. Urbanke, "Analysis of sum-product decoding of low-density parity-check codes using a Gaussian approximation," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 657–670, Feb. 2001.

[54] S.-Y. Chung, G. D. Forney, Jr., T. J. 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, no. 2, pp. 58–60, Feb. 2001.

[55] T. Kailath, *Linear Systems*. Upper Saddle River, NJ, USA: Prentice-Hall, 1980.

[56] H. M. Stark and A. A. Terras, "Zeta functions of finite graphs and coverings," *Adv. Math.*, vol. 121, no. 1, pp. 124–165, Jul. 1996.

[57] M. Fu, "On Gaussian approximation for density evolution of low-density parity-check codes," in *Proc. IEEE Int. Conf. Commun.*, vol. 3. Istanbul, Turkey, Jun. 2006, pp. 1107–1112.

[58] I. Djurdjevic, J. Xu, K. Abdel-Ghaffar, and S. Lin, "A class of low-density parity-check codes constructed based on Reed–Solomon codes with two information symbols," *IEEE Commun. Lett.*, vol. 7, no. 7, pp. 317–319, Jul. 2003.

[59] R. M. Tanner, D. Sridhara, A. Sridharan, T. E. Fuja, and D. J. Costello, Jr., "LDPC block and convolutional codes based on circulant matrices," *IEEE Trans. Inf. Theory*, vol. 50, no. 12, pp. 2966–2984, Dec. 2004.

[60] D. Declercq, L. Danjean, E. Li, S. K. Planjery, and B. Vasić, "Finite alphabet iterative decoding (FAID) of the (155,64,20) Tanner code," in *Proc. 6th Int. Symp. Turbo Codes*, Brest, France, Sep. 2010, pp. 11–15.

[61] B. K. Butler and P. H. Siegel, "LDPC code density evolution in the error floor region," pp. 1–5, Sep. 2014. [Online]. Available: http://arxiv.org/abs/1409.5783

[62] L. Dolecek, P. Lee, Z. Zhang, V. Anantharam, B. Nikolić, and M. J. Wainwright, "Predicting error floors of structured LDPC codes: Deterministic bounds and estimates," *IEEE J. Sel. Areas Commun.*, vol. 27, no. 6, pp. 908–917, Aug. 2009.

[63] B. D. McKay. (Nov. 4, 2009). *Nauty User's Guide (Version 2.4)*. [Online]. Available: http://cs.anu.edu.au/~bdm/nauty/

Brian K. Butler (S'90–M'90–SM'01) received the B.S. degree in engineering from Harvey Mudd College, Claremont, CA, in 1989, the M.S. degree in electrical engineering from Stanford University, Stanford, CA, in 1990, and the Ph.D. degree in electrical and computer engineering from the University of California, San Diego (UCSD), in 2013. At UCSD, he was affiliated with the Center for Magnetic Recording Research. His research interests include coding, modulation, and the design of wireless systems and receivers.

He was with QUALCOMM, Inc., San Diego, CA, during the summer of 1989 and from 1990 until 2008. At QUALCOMM, he worked on a variety of CDMA cellular and other wireless systems, including their product designs. He performed system simulation, design verification, algorithm design, and field testing of the early CDMA cellular system. He designed and led portions of the air-interface and the ASIC devices on the GLOBALSTAR CDMA-LEO satellite project. He contributed to the CDMA air-interface designs, phone designs, and the ASIC designs over many generations. From 1996 to 2005, he held the position of Vice President of Engineering in the ASIC division, leading the communication systems engineering department and the modem technology team. He led systems engineering on and was co-project-engineer of the first 3G-1x product and the 1x-EV-DV prototype developments. From 2005 to 2008, he contributed as a VP in QUALCOMM Corporate R&D, including as project engineer of the 4G-LTE (3GPP E-UTRA) effort.

Dr. Butler holds 39 issued U.S. patents and several foreign.

Paul H. Siegel (M'82–SM'90–F'97) received the S.B. and Ph.D. degrees in mathematics from the Massachusetts Institute of Technology (MIT), Cambridge, in 1975 and 1979, respectively.

He held a Chaim Weizmann Postdoctoral Fellowship at the Courant Institute, New York University. He was with the IBM Research Division in San Jose, CA, from 1980 to 1995. He joined the faculty of the School of Engineering at the University of California, San Diego, in July 1995, where he is currently Professor of Electrical and Computer Engineering. He is affiliated with the Center for Magnetic Recording Research, where he holds an endowed chair and served as Director from 2000 to 2011. His primary research interests lie in the areas of information theory and communications, particularly coding and modulation techniques, with applications to digital data storage and transmission.

Prof. Siegel was a member of the Board of Governors of the IEEE Information Theory Society from 1991 to 1996 and from 2009 to 2011. He was re-elected for another three-year term in 2012. He served as Co-Guest Editor of the May 1991 Special Issue on "Coding for Storage Devices" of the IEEE TRANSACTIONS ON INFORMATION THEORY. He served the same TRANSACTIONS as Associate Editor for Coding Techniques from 1992 to 1995, and as Editor-in-Chief from July 2001 to July 2004. He was also Co-Guest Editor of the May/September 2001 two-part issue on *The Turbo Principle: From Theory to Practice* of the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS.

Prof. Siegel was corecipient, with R. Karabed, of the 1992 IEEE Information Theory Society Paper Award and shared the 1993 IEEE Communications Society Leonard G. Abraham Prize Paper Award with B. H. Marcus and J. K. Wolf. With J. B. Soriaga and H. D. Pfister, he received the 2007 Best Paper Award in Signal Processing and Coding for Data Storage from the Data Storage Technical Committee of the IEEE Communications Society. He holds several patents in the area of coding and detection, and was named a Master Inventor at IBM Research in 1994. He is a member of Phi Beta Kappa and the National Academy of Engineering.