TMO Progress Report 42-144 February 15, 2001

Iterative Turbo Decoder Analysis Based on Density Evolution

D. Divsalar,[1] S. Dolinar,[1] and F. Pollara[1]

We track the density of extrinsic information in iterative turbo decoders by actual density evolution, and also approximate it by consistent Gaussian density functions. The approximate model is verified by experimental measurements. We view the evolution of these density functions through an iterative decoder as a nonlinear dynamical system with feedback. Iterative decoding of turbo codes and of serially concatenated codes is analyzed by examining whether a signal-to-noise ratio (SNR) for the extrinsic information keeps growing with iterations. We define a noise figure for the iterative decoder, such that the turbo decoder will converge to the correct codeword if the noise figure is bounded by a number below 0 dB. By decomposing the code's noise figure into individual curves of output SNR versus input SNR corresponding to the individual constituent codes, we gain many new insights into the performance of the iterative decoder for different constituents. Many mysteries of turbo codes are explained based on this analysis. For example, we show why certain codes converge better with iterative decoding than do more powerful codes, which are suitable only for maximum-likelihood decoding. The roles of systematic bits and of recursive convolutional codes as constituents of turbo codes are crystallized. The analysis is generalized to serial concatenations of mixtures of complementary outer and inner constituent codes. Design examples are given to optimize mixture codes to achieve low iterative decoding thresholds on the signal-to-noise ratio of the channel.

I. Introduction

Concatenated coding schemes consist of the combination of two or more simple constituent encoders and interleavers. The parallel concatenation known as a "turbo code" [1] has been shown to yield remarkable coding gains close to the theoretical limits, yet admitting a relatively simple iterative decoding technique. Also, serial concatenation of interleaved codes [2] may offer superior performance to parallel concatenation at very low bit-error rates. In both coding schemes, the core of the iterative decoding structure is a soft-input soft-output (SISO) a posteriori probability (APP) module [7].

The analysis of iterative decoders for concatenated codes with short blocks is an unsolved problem. However, for very large block sizes, such analysis is possible under certain assumptions. An asymptotic (as block size goes to infinity) analysis of iterative decoding can be based on the method of density

[1] Communications Systems and Research Section.

The research described in this publication was carried out by the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

evolution proposed by Richardson and Urbanke [8]; see also [10,11]. Their analysis tracks the probability density function of the extrinsic information messages as this density evolves from iteration to iteration. They used this method to compute iterative decoding thresholds for low-density parity check (LDPC) codes over a binary input additive white Gaussian noise (AWGN) channel. In our article, we apply the density evolution method to analyze the (asymptotic) performance of iterative decoders for turbo codes and turbo-like serially concatenated codes.

Wiberg in his dissertation [3] showed that the extrinsic information in iterative decoding can be approximated by a Gaussian density function. This approximation was used by Chung et al. [9] to obtain a threshold on minimum bit signal-to-noise ratio, E_b/N_0, for LDPC codes. El Gamal, in his dissertation [4] and with Hammons [5], considered the SISO module in turbo decoders as a signal-to-noise ratio (SNR) transformer, and also suggested a method for analyzing the overall turbo decoder. Prior to El Gamal, Hagenauer and Hoeher [20] proposed a similar SNR transformer analysis for the soft output Viterbi algorithm (SOVA), and Alexander et al. [21] proposed a similar noise variance trace method to analyze iterative multiuser detection. In [14], ten Brink developed a method for analyzing the convergence of the decoder based on the evolution of mutual information. Our methods in this article are similar to all of these approaches, and also to analysis of the turbo decoder algorithm by Agrawal and Vardy [25] and of turbo decoder thresholds by Richardson and Urbanke [26], Vialle and Boutros [27], and Chung and Forney [28]. Our main contribution is to apply these analyses to gain new insights into designing new turbo-like code structures and to explain the workings of old ones.

In this article, we first evaluate iterative decoding for turbo and turbo-like codes using actual density evolution and then compare these results with analyses obtained using a Gaussian approximation to the density function for the extrinsic information messages. For the Gaussian approximation, we have two choices for modeling the Gaussian density: (1) based on the empirically determined mean and variance as independent parameters or (2) based on the empirical mean only, assuming that the variance is determined by the consistency condition proposed by Richardson et al. [10]. We found that the second Gaussian model gave closer agreement with the results obtained from actual density evolution, so this is the model that we used for our approximate analysis based on Gaussian density evolution.

For any parallel or serially concatenated turbo or turbo-like code having two component codes, we compute input and output SNRs at each iteration for the two component decoders, using actual density evolution or the consistent Gaussian approximation. We also define, as in low-noise amplifiers, a noise figure for the overall decoder. We argue that if the noise figure of the iterative decoder is bounded by a number strictly lower than 0 dB then the iterative decoder converges to the correct codeword. Another method is to plot the output SNR versus the input SNR for one component decoder, and the input SNR versus the output SNR for the other component decoder. If the two curves do not intersect, then the iterative decoder converges. We apply this methodology both to explain conditions of decoder convergence and to resolve some mysteries associated with the iterative decoding of turbo codes. Next, by focusing on the input–output SNR characteristics of individual component codes, we devised some new serially concatenated codes constructed by mixing different inner or outer component codes with complementary SNR characteristics. Finally, we derive analytic expressions for the SNR characteristics of arbitrary 2-state recursive convolutional component codes (rate 1 or rate 1/2) when used as an inner code, an outer code, or a middle code in a serial concatenation.

II. The Density Evolution Model and the Gaussian Approximation

Consider a parallel turbo code as shown in Fig. 1 or a serially concatenated convolutional code as shown in Fig. 2. Iterative decoders for these codes are based on two SISO modules shown in Figs. 1 and 2 and described in detail in [7].

The iterative decoder for either code construction can be viewed as a nonlinear dynamical feedback system. Extrinsic information messages $\{\lambda_i\}$ are passed from one constituent decoder to the other. The

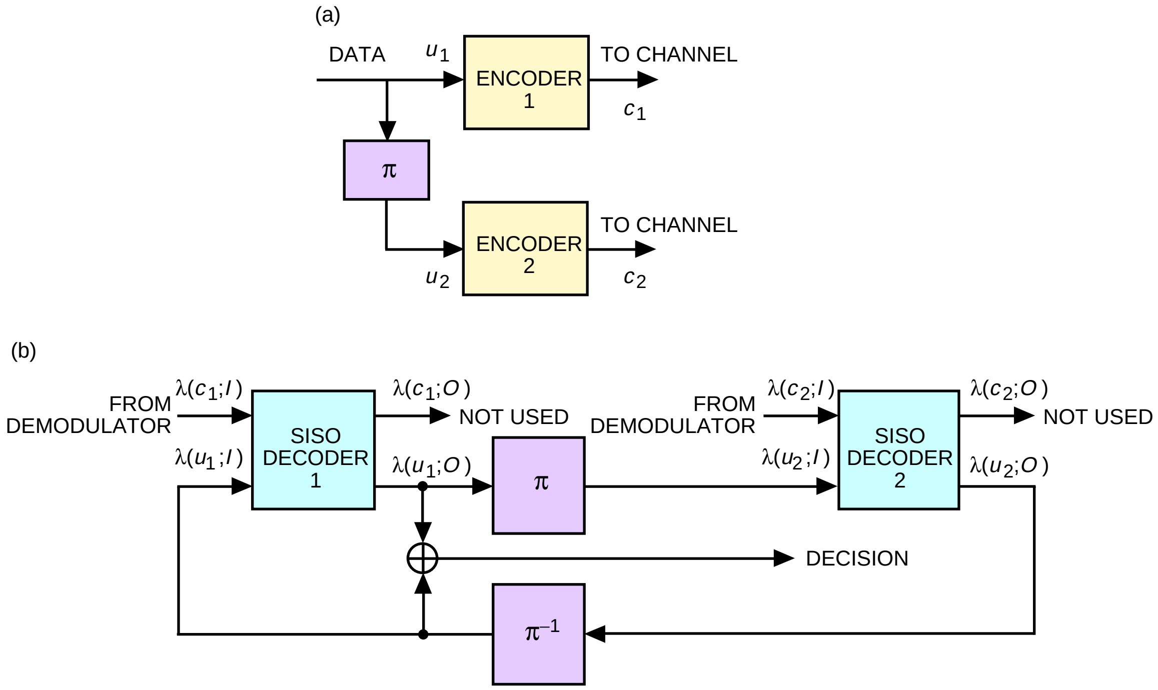

Fig. 1. Structure for (a) encoding and (b) iteratively decoding parallel concatenated convolutional codes (turbo codes).

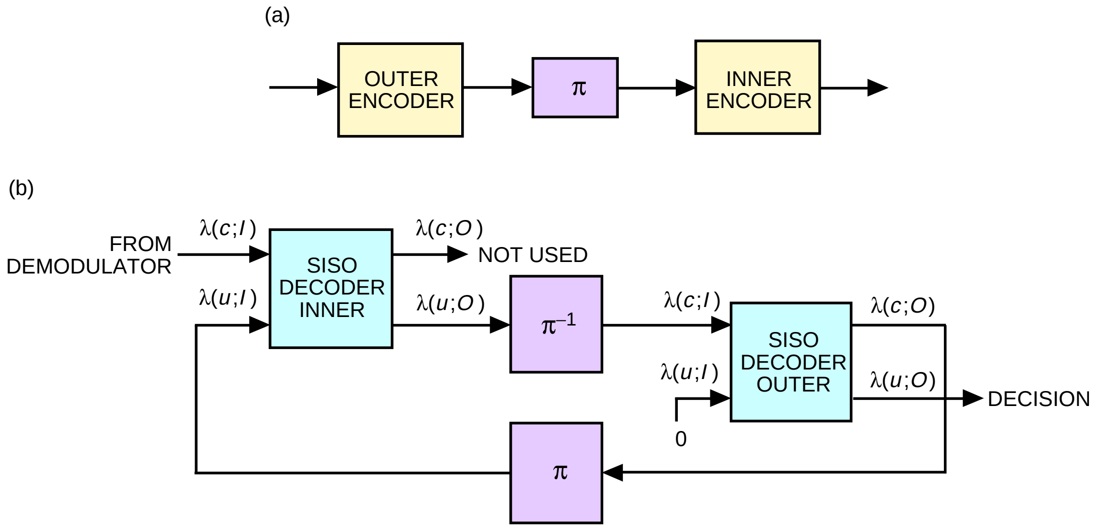

Fig. 2. Structure for (a) encoding and (b) iteratively decoding serially concatenated convolutional codes.

message λ_i measures the log-likelihood ratio for the ith bit based on input messages $\{\lambda_j\}$ from all other bits except the ith. For analysis purposes, we assume that the all-zero codeword is transmitted (corresponding to transmission of $+1$'s on the channel), so, for each i, a positive value of extrinsic information $\lambda_i > 0$ constitutes favorable evidence toward determining the true value of the ith bit.

When the interleaver π in Fig. 1 or 2 is very large and random, the extrinsic information messages λ_i are independent and identically distributed, with probability density function $f(\lambda)$. As shown in [10], this density function is *consistent*, i.e., $\lambda = \log[f(\lambda)/f(-\lambda)]$. Its mean, $\mu = \mathrm{E}(\lambda)$, is the *discrimination* between the two densities $f(\lambda)$ and $f(-\lambda)$. The error probability, $\epsilon = \mathrm{Prob}\{\lambda < 0\}$, for a bitwise decoding decision based solely on λ can be evaluated as $\epsilon = \mathrm{E}\{1/(1+e^{|\lambda|})\}$ when the density $f(\lambda)$ is consistent. Similarly, for a consistent density $f(\lambda)$, the mutual information I between the ith information bit and the ith extrinsic information value λ_i can be evaluated as $I = \mathrm{E}\{\log_2[2/(1+e^{-\lambda})]\}$.

We computed histograms of the extrinsic information λ_{in} and λ_{out} at the input and output of a SISO module for a systematic 8-state rate-1/2 turbo-like code shown in Figs. 3 and 4. For this code, the information bits are repeated three times, with one copy sent directly to the channel and two copies sent through an interleaver to a punctured recursive convolutional code with forward and backward generator polynomials $1 + D + D^3$ and $1 + D^2 + D^3$, respectively. The unpunctured version of this constituent code is denoted $(1+D+D^3)/(1+D^2+D^3)$, or more compactly in octal form as 13/15 (interpreting the least-significant bit in the octal representation to be the coefficient of D^0). The puncturing removes every second code symbol (puncturing pattern 1010), resulting in a rate-2 constituent code and an overall code rate of 1/2.

As shown in these figures, the (empirical) probability densities $f(\lambda_{\mathrm{in}})$ and $f(\lambda_{\mathrm{out}})$ evolve with successive decoder iterations from narrow densities concentrated near $\lambda = 0$ to broader Gaussian-shaped densities with increasing means as the iterations continue. We have observed that, for a wide variety of turbo and turbo-like code constructions, the evolution of densities shown in Figs. 3 and 4 is typical. Except for some skewness in the early iterations, the probability density function $f(\lambda)$ can be approximated by a Gaussian density.

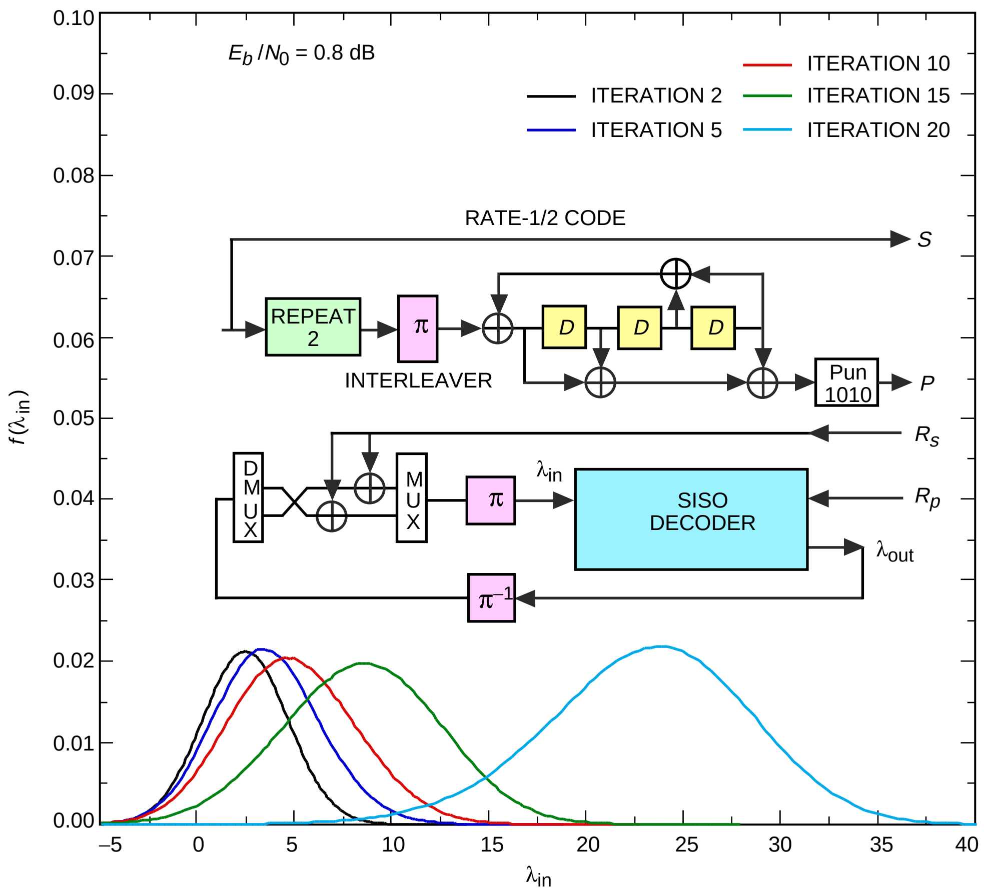

Fig. 3. Evolution with iterations of empirically measured histograms of a SISO module's input extrinsic information, λ_{in}, for a systematic 8-state, rate-1/2 turbo-like code.

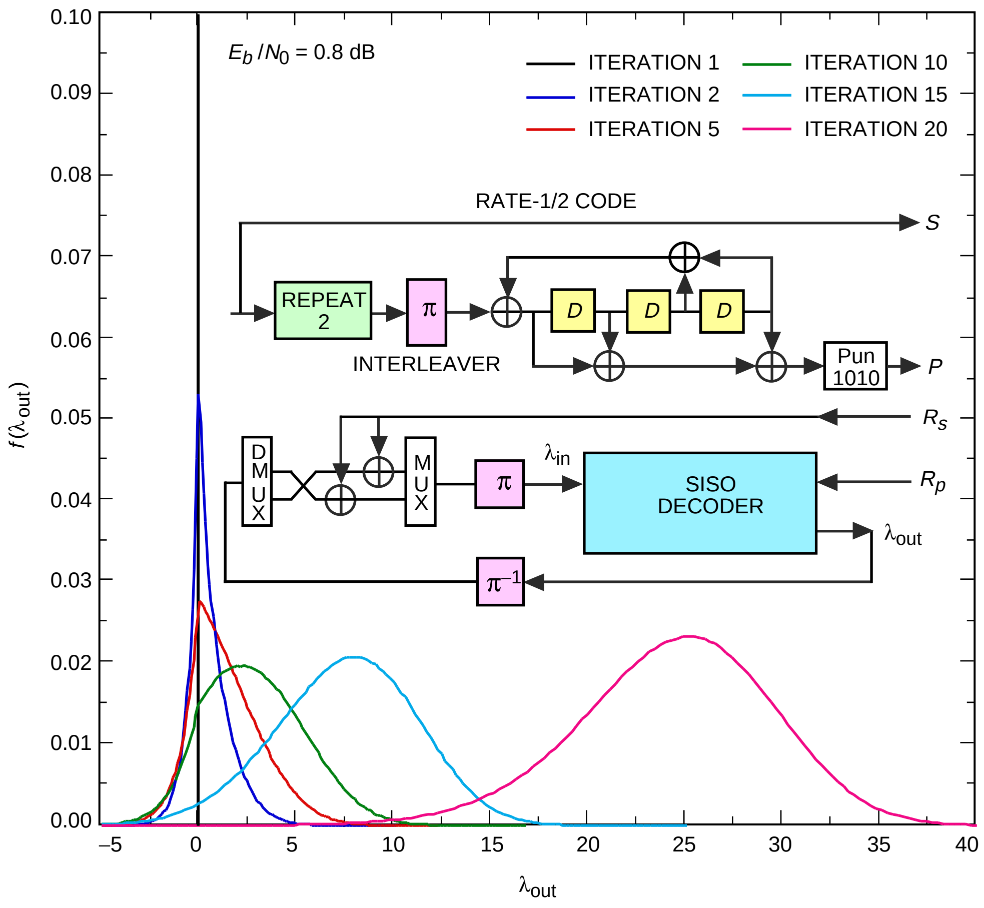

Fig. 4. Evolution with iterations of empirically measured histograms of a SISO module's output extrinsic information, λ_{out}, for a systematic 8-state, rate-1/2 turbo-like code.

If $f(\lambda)$ is approximated by a Gaussian density function, then its statistics depend on only two parameters, its mean, $\mu = \mathrm{E}(\lambda)$, and its variance, $\sigma^2 = \mathrm{Var}(\lambda)$. A signal-to-noise ratio for this random variable can be defined as $\mathrm{SNR} \triangleq \mu^2/\sigma^2$. A high value of SNR implies that $f(\lambda)$ is easily discriminated from $f(-\lambda)$. Note that this SNR is a signal-to-noise ratio for the quality of the extrinsic information developed during the iterative decoding process. The usual signal-to-noise ratio measuring the quality of the channel symbols is denoted conventionally in this article as E_s/N_0, or E_b/N_0 when normalized per information bit.

If $f(\lambda)$ is both Gaussian and consistent, then $\sigma^2 = 2\mu$ and $\mathrm{SNR} = \mu/2$. However, since the Gaussian density is not a perfect approximation to the true density, $f(\lambda)$, the empirically measured variance σ^2 does not bear this exact relationship to the empirically measured mean. Thus, we get different answers depending on whether the evolution of SNR with iterations is computed as $\mathrm{SNR} = \mu^2/\sigma^2$ or $\mathrm{SNR} = \mu/2$. In our experience, the second formula (enforcing the consistency condition in the definition of SNR) gives a better prediction of decoder convergence. The first formula (defining SNR based on mean and variance measured independently) generally gives predictions that are too pessimistic at low values of SNR and too optimistic at high values.

III. Dynamic Models for Decoder Convergence

Consider the input and output SNRs for each decoder at each iteration, as shown in Fig. 5. These are denoted $\mathrm{SNR1}_{\mathrm{in}}$, $\mathrm{SNR1}_{\mathrm{out}}$, $\mathrm{SNR2}_{\mathrm{in}}$, and $\mathrm{SNR2}_{\mathrm{out}}$. A nonzero E_b/N_0 from the channel enables decoder 1 to produce a nonzero $\mathrm{SNR1}_{\mathrm{out}}$ for the output extrinsic information despite starting with $\mathrm{SNR1}_{\mathrm{in}} = 0$. For a given value of E_b/N_0, the output SNR of each decoder is a nonlinear function of its input SNR, denoted by G_1 for decoder 1 and G_2 for decoder 2, as shown in Fig. 5. We have $\mathrm{SNR1}_{\mathrm{out}} = G_1(\mathrm{SNR1}_{\mathrm{in}}, E_b/N_0)$ and $\mathrm{SNR2}_{\mathrm{out}} = G_2(\mathrm{SNR2}_{\mathrm{in}}, E_b/N_0)$. Also, $\mathrm{SNR2}_{\mathrm{in}} = \mathrm{SNR1}_{\mathrm{out}}$, and thus $\mathrm{SNR2}_{\mathrm{out}} = G_2(G_1(\mathrm{SNR1}_{\mathrm{in}}, E_b/N_0), E_b/N_0)$. This general model can be applied to both parallel and serial concatenations, as illustrated in the next two main sections. For serial concatenations, decoder 1 and decoder 2 correspond to the inner and outer constituent codes, respectively, and no channel observations are input to decoder 2 unless the code is a hybrid code that sends some bits directly from the outer code to the channel.

We evaluated the functions G_1 and G_2 empirically by Monte Carlo simulation, using two different statistical models to generate the input λ's. The actual density evolution model generates input λ's directly from the histogram of output λ's from the previous decoder. The approximate Gaussian density evolution model generates input λ's from a consistent Gaussian density with mean μ and variance 2μ, where μ is a varied parameter. In each case, the input and output SNRs are computed from the actual λ-histograms as $\mathrm{E}\{\lambda\}/2$, which is the appropriate measure of SNR if the λ-distribution is both Gaussian and consistent.

In these evaluations, we used large blocks of 2000 bits or longer and discarded λ's computed within 200 bits of the edges of the block before computing λ-histograms or SNRs.

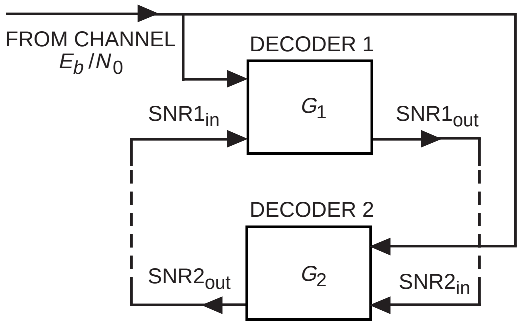
FROM CHANNEL
E_b/N_0
DECODER 1
G_1
$\mathrm{SNR1}_{\mathrm{in}}$
$\mathrm{SNR1}_{\mathrm{out}}$
DECODER 2
G_2
$\mathrm{SNR2}_{\mathrm{out}}$
$\mathrm{SNR2}_{\mathrm{in}}$

Fig. 5. Analysis of turbo coding as a nonlinear dynamical system with feedback using density evolution.

A. Full-Iteration Dynamic Model

The decoder's convergence can be assessed by measuring the change in the SNR of the extrinsic information from one iteration to the next. A noise figure $F = \mathrm{SNR1}_{\mathrm{in}}/\mathrm{SNR2}_{\mathrm{out}}$ can be defined for the turbo decoder at each iteration as the ratio of the input SNR of decoder 1 at the beginning of the iteration to the output SNR of decoder 2 at the end of the iteration (which becomes the input SNR to decoder 1 at the start of the next iteration). If the noise figure at a given iteration is less than 1, this indicates an improvement in the SNR of the extrinsic information from the beginning of the iteration to the end. If the noise figure is bounded by a number lower than 1 for the entire range of input SNR to decoder 1, then the SNR of the extrinsic information messages will increase without bound (if the block size is infinite) and the turbo decoder will converge to the correct codeword. We used all the assumptions made by Richardson and Urbanke [8] for very large block sizes (essentially, when the block size and the number of iterations go to infinity but the number of iterations is much less than roughly the log of the block size corresponding to the girth of the graph representing the overall code, then the effects of cycles on performance can be ignored).

These claims can be justified by a concentration theorem [18,11] similar to that used in [8–11] to prove iterative-decoding decoding thresholds for LDPC codes. The concentration theorem says that the average bit-error probability is concentrated around the ensemble average of the bit-error probability over all possible graphs representing a given code, or over all interleavers in the case of turbo codes, when the block size goes to infinity. This convergence is exponential in the block size, and, as the block size goes to infinity, the graphs representing the code can be considered loop-free (locally tree-like). Such an assumption for turbo codes was argued in [11], based on the decay of dependencies of messages that are far apart from each other on the trellis (similar to the concept of finite-length trace back in Viterbi decoding).

We show in Fig. 6 an example of how the convergence properties of the overall decoder can be determined from its noise figure. The code in this case is a simple serially concatenated repeat-and-accumulate (RA) code [19], tested at three values of E_b/N_0 just above the iterative decoding threshold. The noise figure in each case rises with iterations from an initial value of 0 to a maximum just below 1, indicating that decoder convergence is achieved. However, as E_b/N_0 is reduced toward the threshold value, there is a dramatic increase in the number of iterations required to get past the region where the noise figure is just barely less than 1. This is the zone where the decoder is making very slow progress toward convergence (called the iterative decoding tunnel in the next section). After getting past this zone, the decoder in each case converges quickly.

B. Half-Iteration Dynamic Model

Equivalently, we can test decoder convergence by tracking the evolution of the extrinsic information's SNR from half-iteration to half-iteration. The analytical method is to plot the output SNR of decoder 1 versus its input SNR, and the input SNR of decoder 2 versus its output SNR, as shown in Fig. 7. In

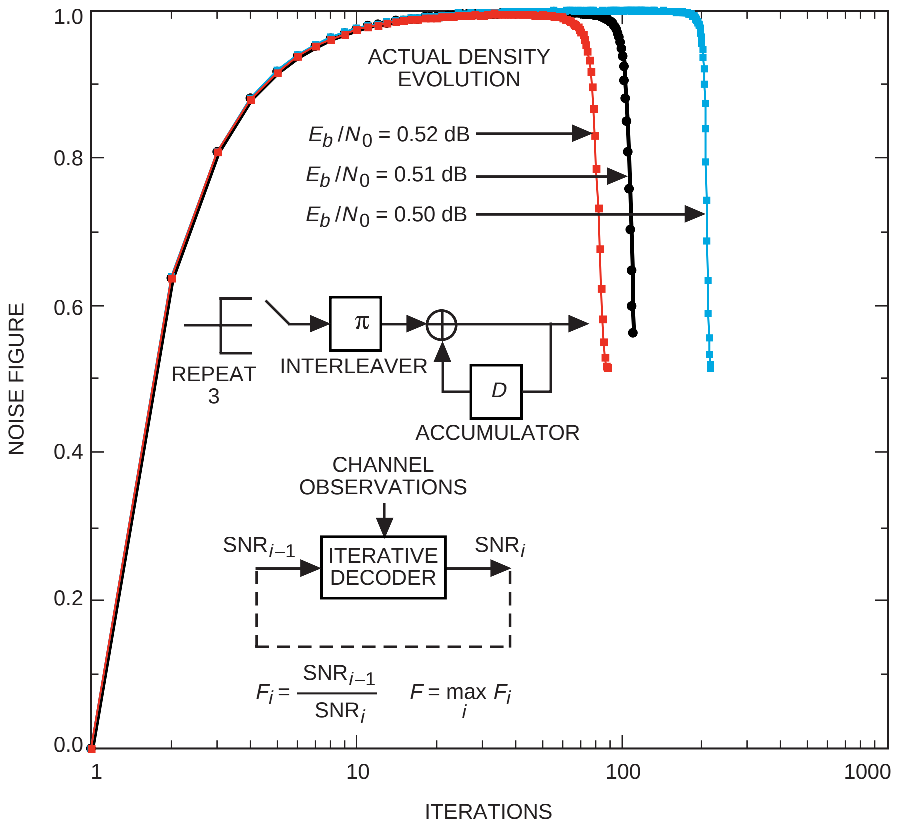

Fig. 6. Noise figure for a rate-1/3 repeat-and accumulate (RA) code.

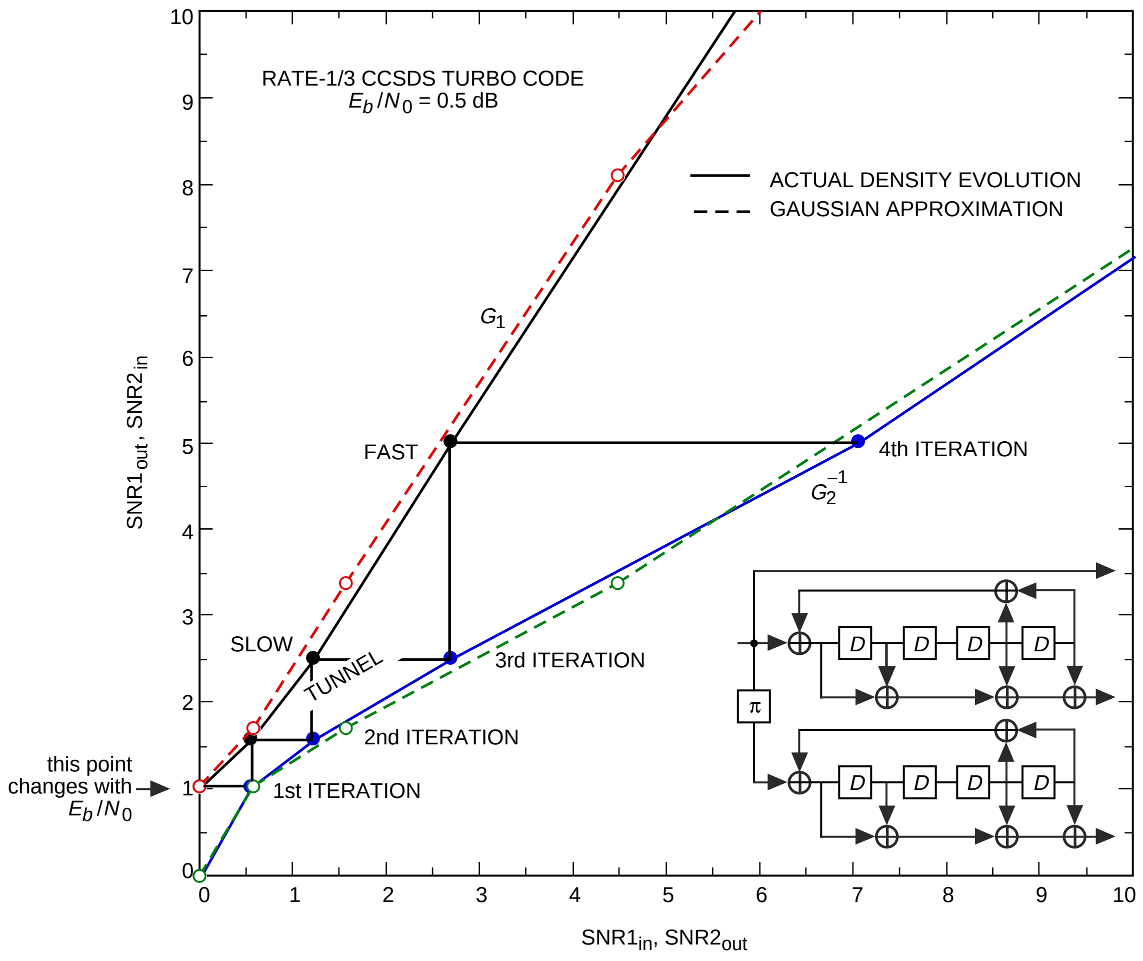

Fig. 7. Iterations and convergence of a turbo decoder.

this figure, we considered a rate-1/3 Consultative Committee for Space Data Systems (CCSDS) turbo code [12] consisting of two 16-state systematic recursive convolutional codes. Encoder 1 is rate 1/2, and encoder 2 is rate 1, as its systematic bits are punctured to make the overall code rate 1/3. The upper pair of curves corresponds to the input–output function G_1 for decoder 1, and the lower pair of curves corresponds to G_2^{-1} for decoder 2.

In Fig. 7, the input–output functions G_1 and G_2^{-1} are computed under both the actual density evolution model and the Gaussian approximation for comparison. The curve for G_1 or G_2^{-1} obtained from actual density evolution is just a sequence of discrete points; in the figure, these points are joined by linear interpolation to give an estimate of the value of G_1 or G_2^{-1} at intermediate points. The curve for G_1 or G_2^{-1} obtained from the Gaussian approximation can be smoothed to any desired degree by testing sufficiently close values of the parametric input.

Figure 7 graphically shows the progress of the decoder's iterations. The improvement in the SNR of the extrinsic information, and the corresponding improvement in the decoder's bit-error rate, follows a staircase path reflecting at right angles between the curves corresponding to G_1 and G_2^{-1}. The steps in this staircase are large when the bounding curves are far apart and small when they are close together. Where the curves are closest together, the improvement in bit-error rate slows down, as many iterations are required to bore through the narrow iterative decoding tunnel between the curves. If the iterative decoder successfully passes through the tunnel, convergence becomes very rapid as the two curves get farther apart at higher SNRs. This means that as the block size goes to infinity the bit-error rate goes to zero as the number of iterations increases.

The initial displacement of the G_1 curve for $\text{SNR1}_{\text{in}} = 0$ is dependent on the E_b/N_0 due to the channel observations. If we reduce E_b/N_0 from the value of 0.5 dB used in Fig. 7, then at some point the two curves will just touch each other. That value of E_b/N_0 represents the iterative decoding threshold. The iterative decoding tunnel will be closed at the SNR where the two curves touch, and the staircase path will not go past this point. The bit-error rate will settle to a nonzero value determined by this finite SNR. Conversely, if E_b/N_0 is greater than this threshold, the decoder converges and the bit-error rate goes to zero as the iterations increase.

This article exclusively uses $\text{SNR} = \text{E}\{\lambda\}/2$, which can be interpreted either as a signal-to-noise ratio for consistent Gaussian extrinsics or as half the discrimination between $f(\lambda)$ and $f(-\lambda)$ for any consistent density $f(\lambda)$. Similar analysis with similar conclusions about decoder convergence can be conducted using alternative averages over the λ statistics, such as the probability of error ϵ and the mutual information I mentioned earlier. In these two cases, the input–output characteristics are confined to a finite interval $[0, 1]$, and decoder convergence corresponds to $\epsilon \to 0$ or $I \to 1$, rather than $\text{SNR} \to \infty$.

IV. The Analysis Method for Parallel Concatenated (Turbo) Codes

The half-iteration dynamic model using separate SNR_{out} versus SNR_{in} characteristic curves for the two constituent codes can help to explain many mysteries of parallel concatenated (turbo) codes and provide more insight into the selection of good constituent codes.

A. The Role of Systematic Bits

There has not been an adequate explanation of why the systematic bits in turbo codes should be transmitted in order for the decoder to converge. A maximum-likelihood decoder does not require these bits; indeed, it is possible to construct more powerful turbo codes without transmission of the systematic bits. An explanation for the role of the systematic bits can be surmised from Fig. 8, which analyzes the SNR_{out} versus SNR_{in} characteristic for a parallel concatenation of identical octal 5/7, rate-1, 4-state recursive convolutional codes, with or without an additional set of systematic bits sent to the channel. The comparison is done at a channel symbol signal-to-noise ratio E_s/N_0 of -4 dB. Even though the received symbols corresponding to parity bits from the octal 5/7 code have nonzero E_s/N_0, if we don't send the systematic bits, the SNR of the extrinsic information at the first iteration will be zero. This causes the curves for the two constituent codes to intersect at $\text{SNR} = 0$, and the decoder never makes any progress toward converging to the correct codeword. However, if the systematic bits are transmitted, e.g., using the octal (1,5/7) code, the curve for code 1 moves upward and the SNR of the extrinsics at the first iteration will be high.

Now the secondary question is why this is happening. Consider any rate-1 recursive convolutional code (obtained by puncturing the systematic bits of a rate-1/2 systematic recursive convolutional code). If the degree of the feedforward polynomial is greater than or equal to 1, then an input bit equals the modulo-2 sum of nearly half of the output parity bits in the entire block. In this case, it can be shown analytically that the SNR of the input bits goes to zero as the block size goes to infinity. However, if the degree of the feedforward polynomial is zero, then the input equals the modulo-2 sum of only a few output parity bits, depending on the degree of the feedback polynomial. This results in a nonzero SNR for the input bits at the first iteration.

An example of such a rate-1 recursive convolutional code is an accumulator (differential encoder), for which the feedforward polynomial is 1 (degree is zero). Consider a parallel turbo code constructed from a differential encoder and a rate-1 recursive 16-state convolutional code, as shown in Fig. 9. For this code, there is a substantial nonzero SNR1_{out} for decoder 1 at the end of the first iteration. In such cases, the

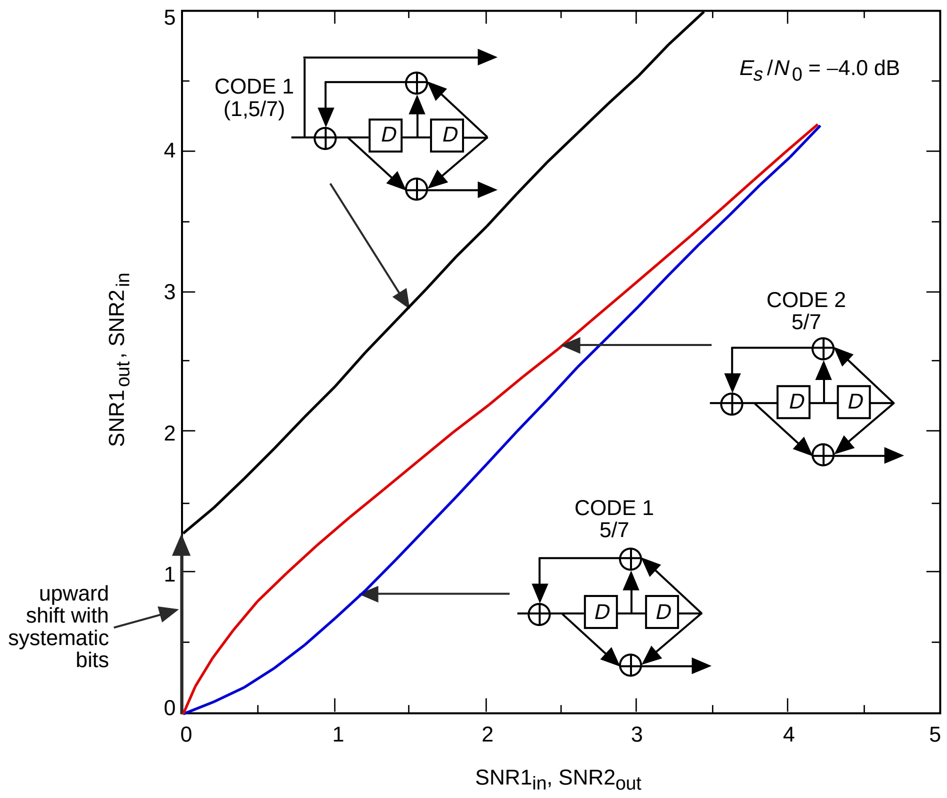

Fig. 8. The role of systematic bits in turbo codes.

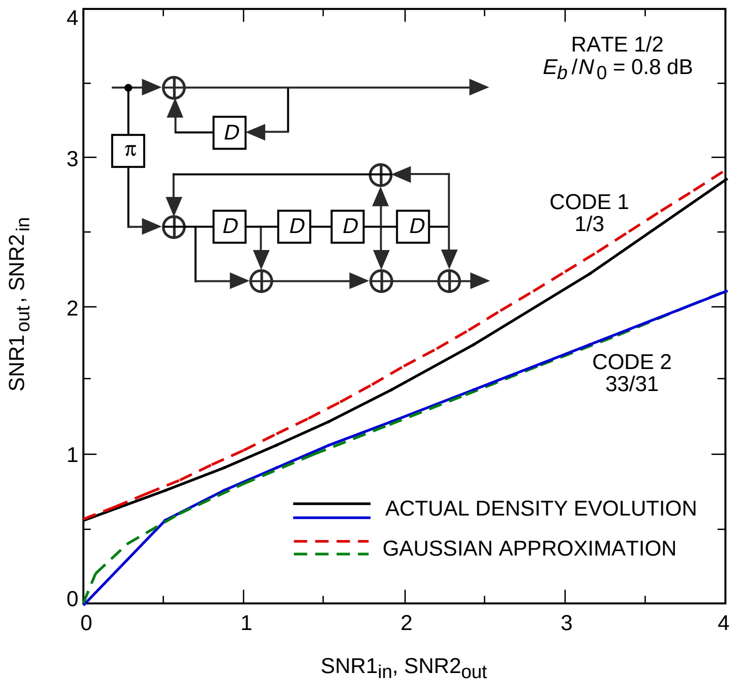

Fig. 9. Example of not sending the systematic bits.

decoder will converge as long as E_b/N_0 is high enough to keep open a narrow iterative decoding tunnel between the two curves.

There is a more detailed way to argue that a nontrivial feedforward polynomial causes the extrinsic information at the first iteration to be zero. Suppose that the code's input bits $\{x_k\}$ and output bits $\{y_k\}$ are related by $\sum_i x_{k-i}p_i = \sum_i y_{k-i}q_i$, corresponding to a rate-1 recursive code with feedforward polynomial coefficients $\{p_i\}$ and feedback polynomial coefficients $\{q_i\}$. Assume without loss of generality that $p_j = 1$ for some j. Then, using the algebra introduced by Hagenauer [17], we have the following equation relating the extrinsic information $\{\lambda_k\}$ associated with the input bits $\{x_k\}$ and the channel information $\{\lambda_k^c\}$ associated with the coded bits $\{y_k\}$:

$$\tanh\left(\frac{\lambda_{k-j}}{2}\right) = \prod_{i\neq j}\left[\tanh\left(\frac{\lambda_{k-i}}{2}\right)\right]^{p_i}\prod_i\left[\tanh\left(\frac{\lambda_{k-i}^c}{2}\right)\right]^{q_i}$$

Unless $p_i = 0$ for all $i \neq j$, the right-hand side is zero at the first iteration because all of the initial extrinsics, $\{\lambda_{k-i}\}$, are zero. Thus, given an input SNR of zero, the output SNR will also be zero. For a code with only one nonzero feedforward component, the iterations will start with a nonzero SNR, because in that case there are no $\tanh(\cdot)$ factors on the right side of this equation coming from zero-SNR extrinsics. By taking derivatives of the equation above, we can establish the slope of the SNR characteristic for the first iteration and use this slope as one of the tools for code design.

One may ask if this same conclusion is true when we use the forward–backward sum-product algorithm on the trellis representation of a rate-1 recursive convolutional code. This is easy to show analytically for a rate-1 recursive convolutional code with full-degree feedback and feedforward polynomials. If we start with uniform state distributions at the beginning and end of a block for the calculation of α and β in the forward and backward algorithm [6,7], the distribution of α and β remains uniform. Due to the symmetry of trellis edges for input bits 0 and 1, the output extrinsic information will be zero. If the feedback polynomial is not full-degree or if the feedforward polynomial with at least two nonzero components is not full-degree, then, by averaging the transition matrix representing the trellis section, with transition probabilities depending on the parity bits on the edges leaving or entering the states, we will have a nonuniform state distribution but with groups of two or more states having the same value that again results in extrinsics that approach zero on average.

We note that the above arguments do not hold for a recursive convolutional code with rate less than one. In this case, the SNR of the extrinsics at the first iteration can be nonzero. However, if we compare, say, rate-1/2 systematic and nonsystematic recursive codes (the latter obtained by puncturing the systematic bits of a rate-1/3 convolutional encoder), the SNR of the extrinsics at the first iteration for the systematic code is significantly higher than for the nonsystematic code. However, the nonsystematic code's SNR characteristic has a higher slope as the input SNR is increased. The basic mechanism for all of these conclusions is that, when the feedforward polynomial has only one nonzero component, then the sequence of channel observations gives direct information about the sequence of states, and nonzero extrinsic information can be inferred about each input bit by applying the feedback polynomial coefficients to the state sequence. On the other hand, when the feedforward polynomial has more than one nonzero component, nothing can be inferred about the state sequence from the channel observations unless there are more channel bits than input bits, i.e., the code rate is less than 1.

Examples of rate-1/2 recursive nonsystematic codes that produce a nonzero SNR at the first iteration are those proposed by Collins et al. [24]. Consider a rate-1/2 recursive nonsystematic code $([p_1(D)/q(D)], [p_2(D)/q(D)])$, where $p_1(D)$ and $p_2(D)$ do not have any common factors with $q(D)$. If $p_1(D) + p_2(D) = D^i$ for some $i \geq 0$, then the code produces a nonzero SNR output at the first iteration. Figure 10 shows the $\mathrm{SNR}_{\mathrm{out}}$ versus $\mathrm{SNR}_{\mathrm{in}}$ curve for one such code, octal (3/7,13/7), alongside the corresponding curve for a rate-1/2 recursive systematic code, octal (1,5/7).

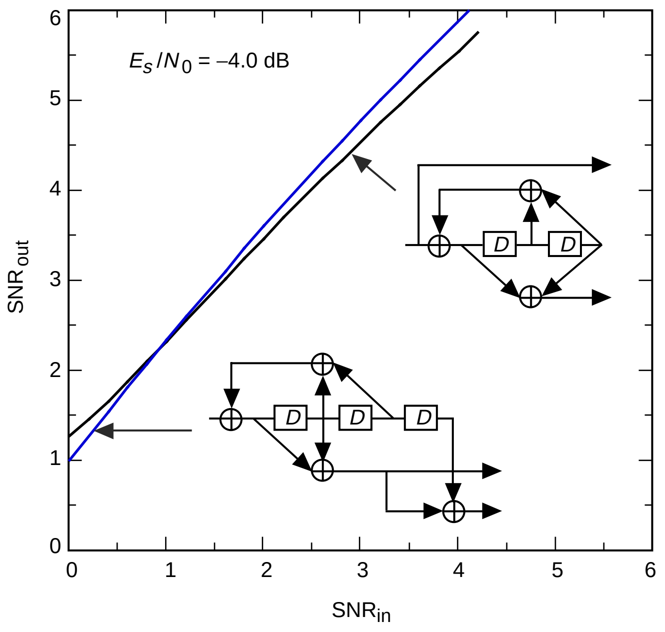

Fig. 10. A rate-1/2 recursive nonsystematic code that starts with nonzero SNR at the first iteration.

B. The Role of Recursive Convolutional Constituent Codes

Next we explain the role of recursive convolutional codes by considering what happens when the component codes are nonrecursive. As shown in Fig. 11, in this case the two curves will always cross each other. The curves for codes 1 and 2 in this figure have the opposite convexity from those seen in the previous figures for recursive convolutional constituent codes. Iterations start with a substantially nonzero SNR1_{out} due to the channel information, but SNR improvements at successive iterations are eventually trapped at the point where the two curves cross. Further iterations will not improve the bit-error rate beyond an error floor determined by this SNR.

For a high enough SNR, the curves for recursive convolutional codes approach a straight-line asymptote with slope 1, whereas those for nonrecursive codes flatten out to zero slope. One may wonder, for the case of recursive codes, what is the mechanism by which highly reliable extrinsic information keeps generating additional extrinsic information, despite the fact that the weak channel symbols seem almost irrelevant compared to the strong a priori information associated with the high-SNR extrinsics? The explanation is the same as the explanation for why recursive constituent codes make stronger turbo codes than nonrecursive constituents, namely, that errors with information weight 1 correspond to codewords with infinite coded weight. Thus, even weak channel symbols, amassed over an infinite block, are sufficient to rule out the possibility of a single isolated information bit error. High-SNR extrinsic information for a given bit i corresponds to a tiny a priori probability of bit error ε_i. But since the channel information rules out single errors, bit i cannot be in error unless at least one other bit j is also wrong, with tiny probability ε_j. Thus, the input extrinsic information at bit i is $\log[(1-\varepsilon_i)/\varepsilon_i]$, and bit i gets new extrinsic information that amounts to $\log[(1-\varepsilon_j)/\varepsilon_j]$. If the high-SNR extrinsics are uniform over the code block, this implies that the SNR increases at a 1:1 slope for high SNR.[2]

[2] For finite block size, the output SNR for large input SNR eventually saturates, and its slope goes to zero. The saturation level depends on type of code, block size, and channel E_s/N_0.

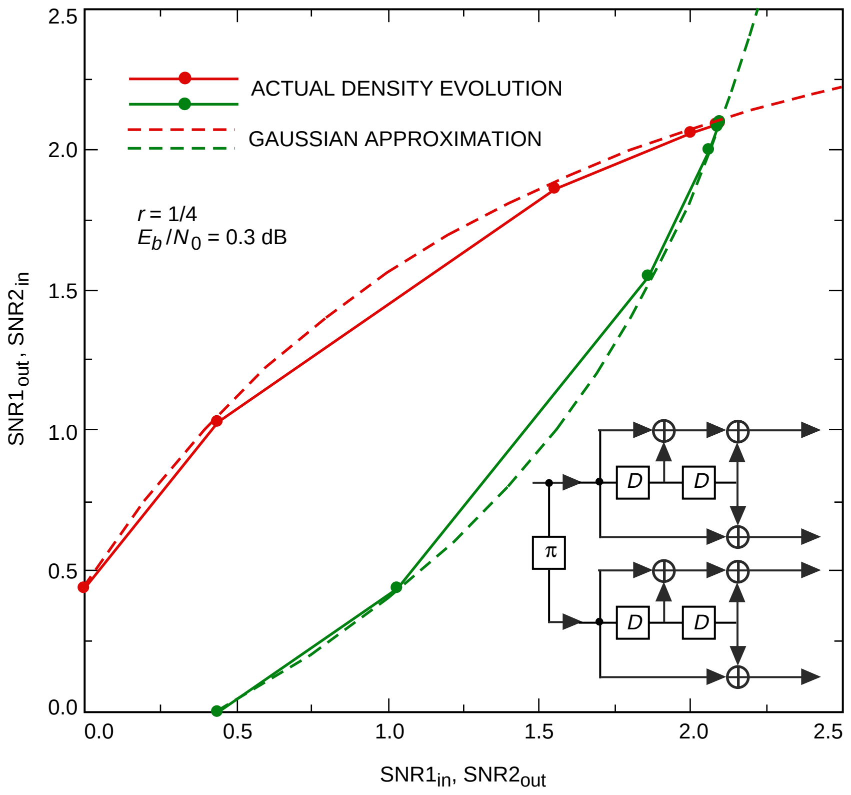

Fig. 11. Convergence of non-recursive encoders.

C. The Role of Primitive Polynomials

Figure 12 elucidates the role of a primitive feedback polynomial. In this figure, we show (using actual density evolution only) the SNR$_{\text{out}}$ versus SNR$_{\text{in}}$ curves for two different rate-1/3 4-state turbo codes operating at $E_b/N_0 = 1.2$ dB. One code is the same code considered in Fig. 8, for which the feedback polynomial is primitive (octal 7). The other code uses a nonprimitive feedback polynomial (octal 5). We see that for high SNRs the two curves for the code using primitive feedback diverge from each other until they eventually approach their asymptotic 1:1 slope, after which they maintain a large separation. On the other hand, the curves for the code using nonprimitive feedback converge toward each other before straightening to their asymptotic slope, so decoder convergence is much slower for this code than for the other code. Also, in this case, the critical iterative decoding tunnel is most constrictive at high values of SNR rather than low values, and the effective iterative decoding threshold for this code will be determined by this narrowing of the tunnel at high SNR even if the curves never actually cross each other.

D. The Role of State Complexity

Figure 13 shows the role of different state complexities. The SNR$_{\text{out}}$ versus SNR$_{\text{in}}$ curves are plotted for rate-1/3 turbo codes with 4-state and 16-state constituents (again using actual density evolution only). We see that the 16-state code (with polynomials given by octal 33/31) has a clear advantage in speed of convergence in the SNR region beyond the iterative decoding tunnel. On the other hand, the sharper curvature of the SNR curves for the individual 16-state codes can narrow the iterative decoding tunnel and require a slightly higher decoding threshold.

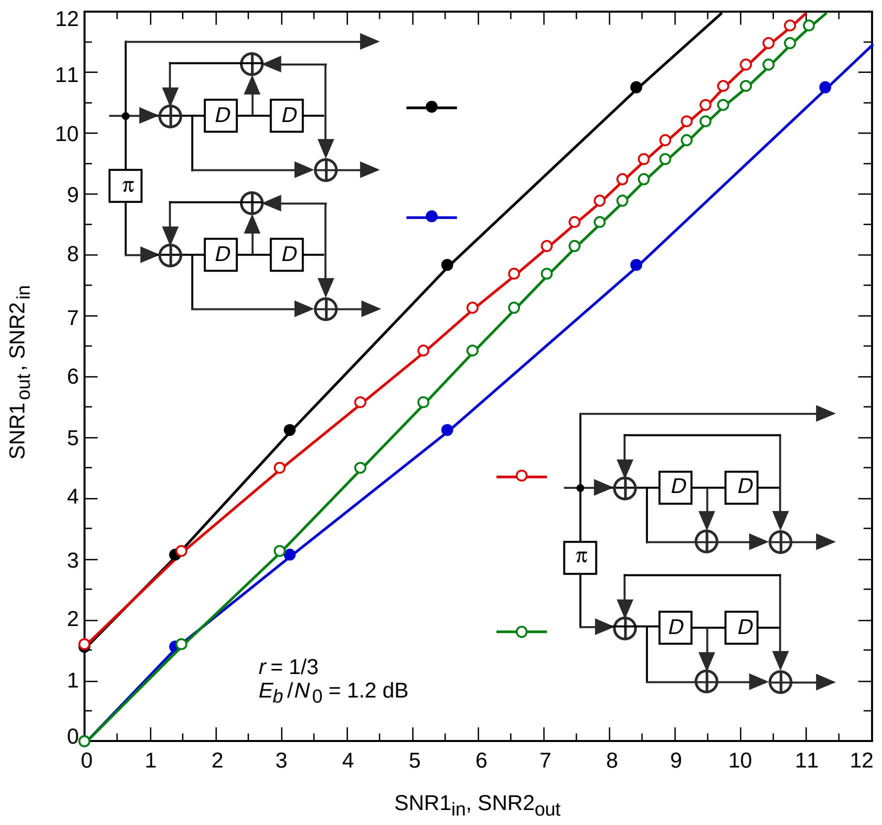

Fig. 12. Rate-1/3 PCCC with two 4-state codes (primitive versus nonprimitive feedback).

V. The Analysis Method for Serial Concatenations

A similar analysis method can be applied to turbo-like codes constructed by serial concatenation, through a large random interleaver, of an outer repetition code with an inner recursive convolutional code.

A. Turbo Codes Viewed as Serially Concatenated Codes

Figure 14 shows how this method can be applied when a parallel concatenated (turbo) code is viewed alternatively as a serial concatenation, using an outer repetition code. The concatenated code in this figure is the same, octal 5/7, rate-1/3 turbo code with 4-state constituents considered above. The outer code produces a set of systematic (uncoded) bits and two repetitions of these bits, which are then encoded by a rate-1 recursive convolutional inner code. The $\mathrm{SNR}_{\mathrm{out}}$ versus $\mathrm{SNR}_{\mathrm{in}}$ characteristic of the outer repetition-2 code is a simple straight line with slope 1, starting at a point determined by the channel SNR of the systematic bits. The SNR characteristic of the 4-state, rate-1 inner code (octal 5/7) is the same as that shown in Fig. 8, and it suffers from the same problem of starting with an output SNR of 0. However, in this case, the nonzero x-axis intercept of the outer code's SNR characteristic (due to the effect of information from the channel on the systematic bits) is sufficient to allow an iterative staircase to be followed in the direction of the iterative decoding tunnel.

Another example is the code introduced in Figs. 3 and 4, which is similar to a turbo code except that the permutation π acts on a double-length block of information bits repeated twice, rather than on one block of information bits separately from the other. Figure 15 shows the $\mathrm{SNR}_{\mathrm{out}}$ versus $\mathrm{SNR}_{\mathrm{in}}$ characteristic for the inner code, along with the straight-line SNR characteristic for the outer repetition

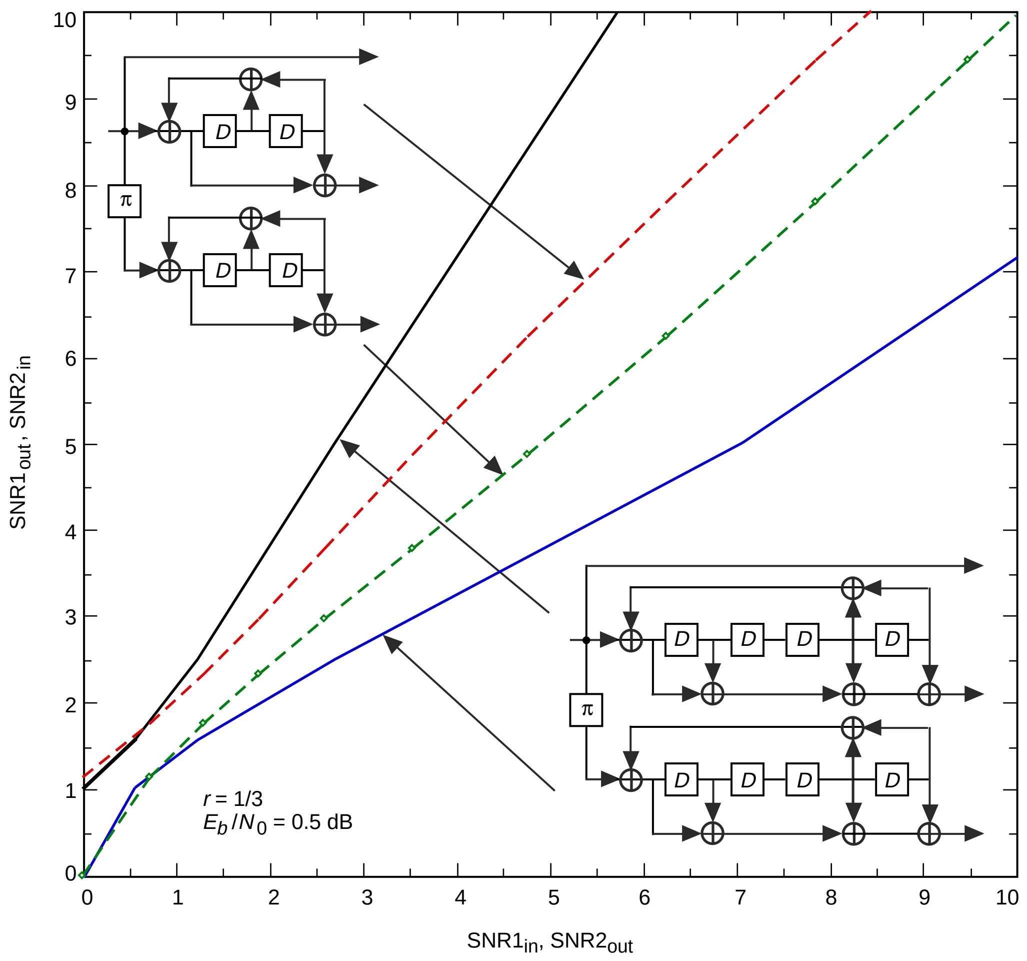

Fig. 13. Comparison of rate-1/3 PCCCs with different state complexities.

code (including the effects of a nonzero offset due to the systematic bits sent directly to the channel). For this graph, the value of E_b/N_0 was reduced to 0.589 dB, the point at which the iterative decoding tunnel closes under the actual density evolution model. Again, the Gaussian approximation gives a slightly optimistic prediction that the decoding tunnel would still be penetrable at this value of E_b/N_0.

B. Serially Concatenated Codes in General

Figure 16 illustrates the analysis method applied to a serially concatenated code for which the outer code is not a simple repetition code. In this example, the outer and inner codes are identical 4-state, rate-1/2 recursive convolutional codes (octal 1,5/7), except that one-fourth of the output symbols of the inner code are punctured to make the overall code rate 1/3. Comparing this figure to the previous figure, we see that the much stronger (1,5/7) outer code has an SNR characteristic with much sharper curvature than the slope-1 straight line for the simple repetition code in Fig. 14. This produces very fast convergence beyond the iterative decoding tunnel, but at the same time the rapid initial rise of the outer code's SNR curve determines the minimum E_b/N_0 (in this case, about 0.4 dB) required to keep the inner code's SNR curve from intersecting it.

Despite its initial sharp curvature, the outer code's SNR curve eventually approaches a straight-line asymptote. The asymptotic slope of the SNR$_{in}$ versus SNR$_{out}$ curve for this code, as plotted in the figure, is 1/4. In general, this asymptotic slope will be no larger than $1/(d_{min} - 1)$, where d_{min} is the minimum distance of the outer code. The argument for this is similar to that used earlier to establish the

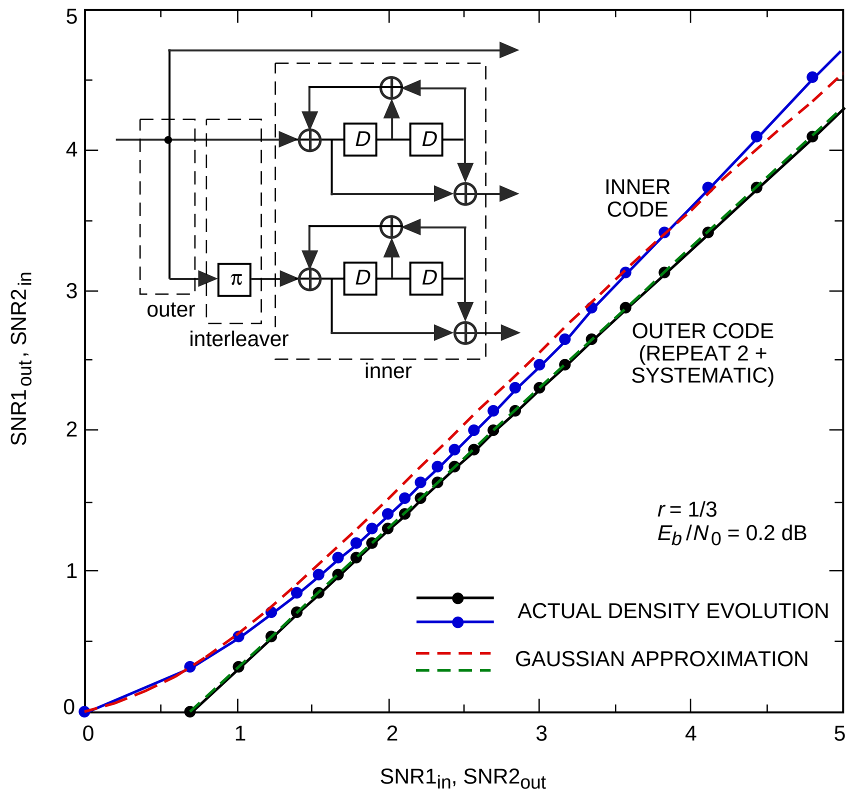

Fig. 14. Turbo code viewed as serial concatenation of outer repetition code with inner code.

1:1 asymptotic slope of the SNR curve for a recursive convolutional inner code. For a given outer code symbol to be incorrect, at least $d_{\min} - 1$ additional code symbols must also be incorrect to satisfy the code constraint. If, from the extrinsic information, all code symbols are independently incorrect with small probability ε, then after applying the code constraint, the probability that a given symbol is incorrect is reduced to $\varepsilon^{d_{\min}}$ or lower. The reduction in this probability from ε to $\varepsilon^{d_{\min}}$ corresponds to output extrinsic information equal to $d_{\min} - 1$ times the input extrinsic information. For some outer codes, the asymptotic slope will be different for the input–output SNR characteristics corresponding to different code symbols. In general, it can be argued that the reciprocal of the asymptotic slope of the $\mathrm{SNR}_{\mathrm{in}}$ versus $\mathrm{SNR}_{\mathrm{out}}$ curve will equal the "minimum extrinsic distance" of the corresponding code symbol. For a linear (n, k) code, the minimum extrinsic distance can be defined as one less than the smallest weight of any codeword containing a 1 in the location of the given code symbol.

VI. Concatenated Codes with Mixed Inner or Outer Codes

The analytical method can also be applied to discover combinations of constituent codes whose individual strengths and weaknesses complement each other. Turbo-like concatenated codes can then be constructed using a mixture of such complementary constituent codes that outperform codes formed from either constituent alone. In this section and for the remainder of the article, all $\mathrm{SNR}_{\mathrm{out}}$ versus $\mathrm{SNR}_{\mathrm{in}}$ curves were computed using Gaussian approximations rather than actual density evolution. However, for the case of code mixtures, we also developed a "mixture of Gaussians" model, as described below.

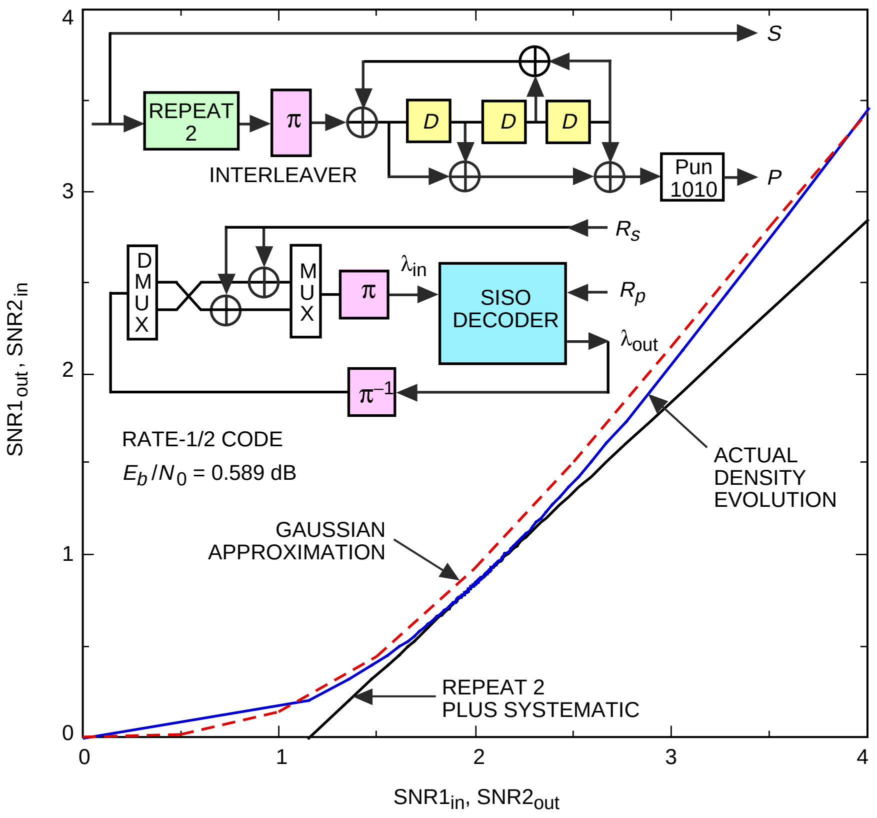

Fig. 15. Iterative decoding threshold for the turbo-like serial concatenation considered in Figs. 3 and 4.

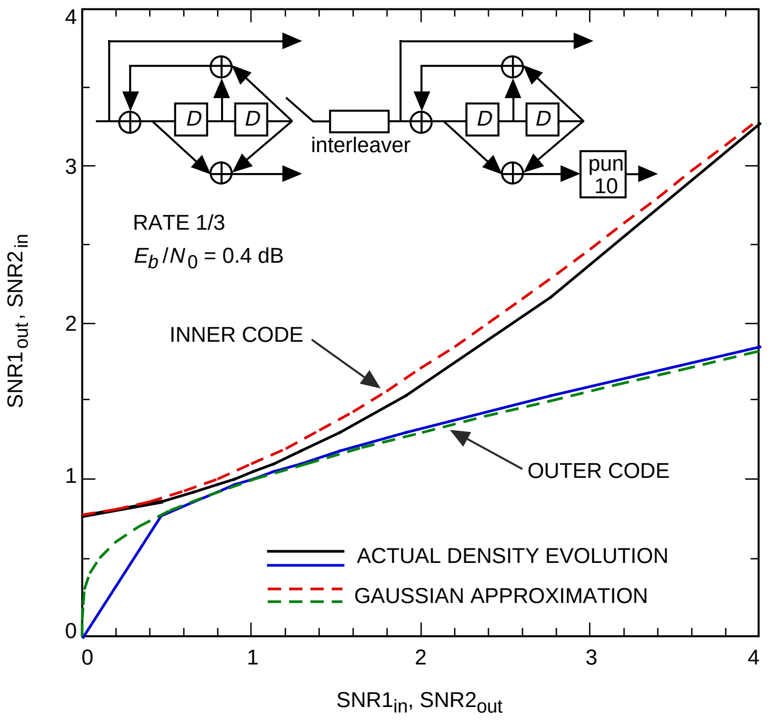

Fig. 16. Example of a rate-1/3 serial turbo code.

A. Concatenated Codes with Mixed Inner Codes

An example of constituent codes suitable for such a construction is the following. A repetition-3 outer code is serially concatenated with two possible rate-1 inner codes, through an infinitely long random interleaver. One code is the 2-state accumulator code (octal 1/3), and the second is a 4-state code (octal 5/7). The overall code rate in each case is 1/3.

When the inner code is the accumulator code, we obtain the code whose overall noise figure was shown in Fig. 6. This code can also be analyzed using the SNR_{out} versus SNR_{in} characteristics of the individual constituent codes, as shown in Fig. 17. The SNR characteristic of the outer repetition-3 code is a straight line with slope 1/2. The SNR characteristic of the accumulator code has slope lower than 1/2 for small values of input SNR, and its slope increases very slowly with increasing input SNR. An E_b/N_0 of 0.49 dB is just sufficient to keep open a long narrow tunnel for successful iterative decoding.[3] Thus, a great many iterations are required when E_b/N_0 is close to this code's iterative decoding threshold.

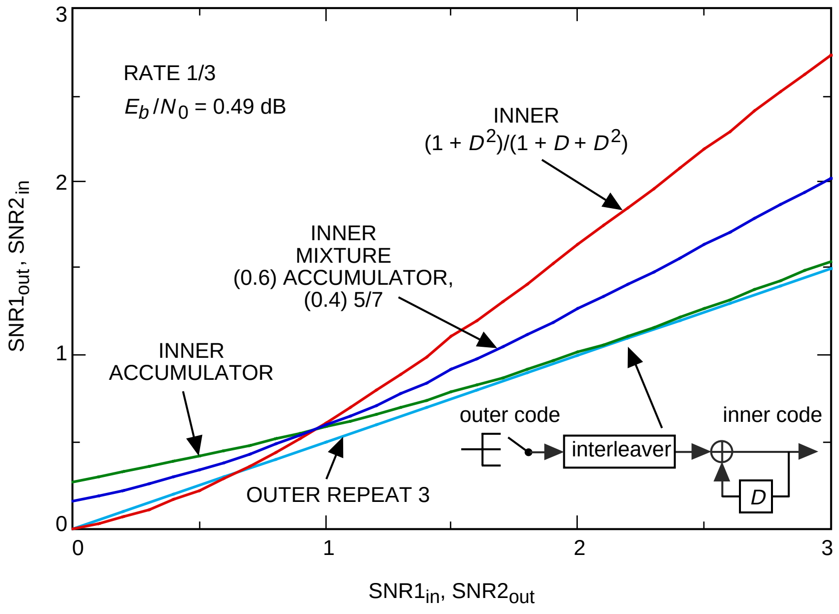

Fig. 17. Example of a concatenated code with mixed inner codes.

In contrast, the slope of the SNR characteristic of the rate-1, four-state code is much higher than 1/2 in the SNR region where the accumulator is having the most difficulty. If this more powerful 4-state code were to replace the 2-state accumulator as the inner code, the iterative decoder would converge much faster and at lower E_b/N_0 once the inner code's SNR increased past a value of roughly 1. Unfortunately, the performance of the 4-state, octal 5/7 code falls apart completely at low values of input SNR. At the initial iteration, when the input SNR of the extrinsics is equal to zero, the output SNR for this code is also zero, because the code is rate-1 and it includes a nontrivial feedforward polynomial. In contrast, a decoder for the lowly accumulator code is able to start its iterations with a modest nonzero output SNR, because in this case there is only one feedforward component.

An improved concatenated code can be formed by using a mixture of the accumulator code and the octal 5/7 code as the inner code. Figure 17 also shows the SNR characteristic of a mixed inner code for which 60 percent of the input bits are encoded by the accumulator and 40 percent are encoded by the octal 5/7 code. This mixed code gets some nonzero initial output SNR from its accumulator component

[3] Note that the analysis in Fig. 17, based on the consistent Gaussian density approximation, gives slightly optimistic predictions compared to that in Fig. 6, based on actual density evolution.

and thus avoids the start-up problem of the pure 5/7 inner code. At the same time, its SNR characteristic curve picks up some of the higher slope and sharper curvature of the 5/7 code's curve, thus shortening and widening the iterative decoding tunnel at $E_b/N_0 = 0.49$ dB, which results in faster decoder convergence and allows the iterative decoding threshold to be reduced further.

Figure 18 shows the SNR characteristic curves for the same codes in Fig. 17 when E_b/N_0 is reduced to 0 dB. From this figure we see that $E_b/N_0 = 0$ dB is just sufficient to decode the concatenated code formed from the 60–40 mixture, but it is clearly below the iterative decoding threshold when either inner constituent code is used without the other. The optimum mixing proportion was obtained by evaluating the SNR characteristic for different mixes. In this case, almost a half-dB of improvement is obtained by mixing the two inner codes in an optimal proportion.

The analysis method for obtaining the SNR characteristic curves in the previous two figures was based on an assumption that the probability density of the output extrinsics from both the inner and outer codes can be approximated as a consistent Gaussian density at each iteration. For the case of mixed codes, the assumption of a pure Gaussian density at each step of the iterations seems to contravene the analysis. As similarly observed in [9] for irregular low-density parity check codes with varying degrees of connectivity between variable nodes and check nodes, what might start out as a pure Gaussian density output from each of the inner decoders becomes a mixture of two Gaussians (at different means) when the outer decoder takes outputs from two different inner decoders after passing through a random infinitely long interleaver. Then a mixture of two Gaussians at the input of the repetition-3 outer decoder becomes a mixture of three Gaussians at the output of the outer decoder or when fed to the input of the inner

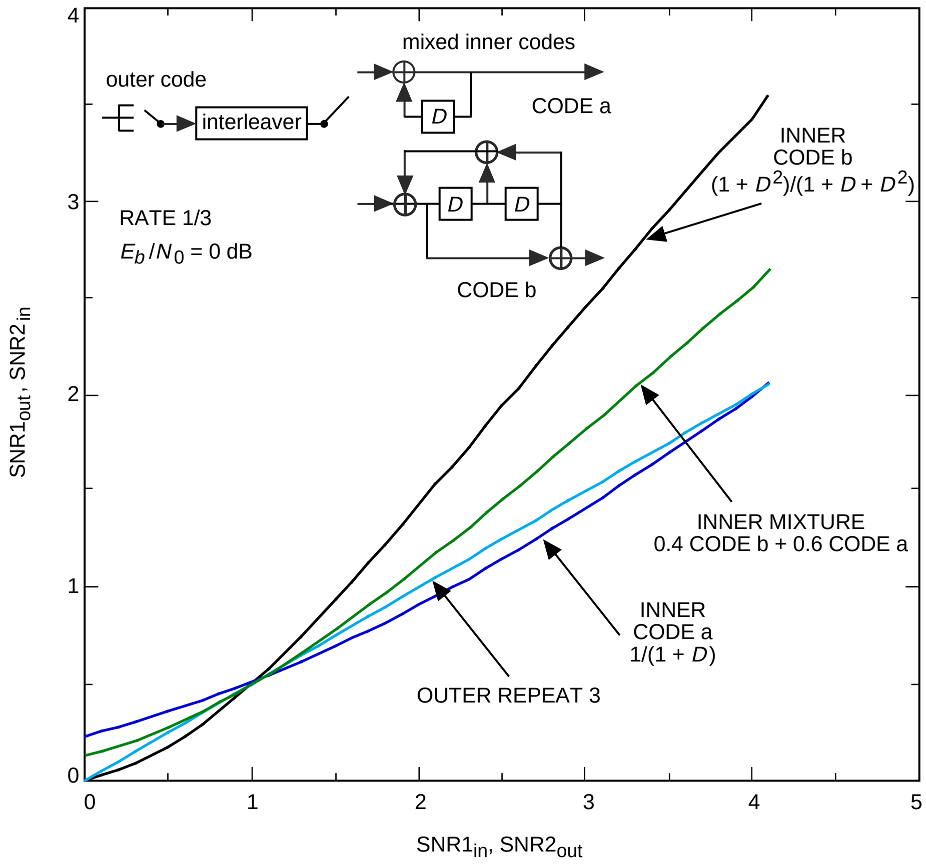

Fig. 18. Reduced decoding threshold for a concatenated code with mixed inner codes.

decoder. Thus, for the case when the outer code is a simple repetition and the inner code is a mixture, it is analytically established that a pure Gaussian density at the output of the inner decoder will imply a non-Gaussian density at the output of the outer decoder, and thus the assumption that both outputs are purely Gaussian cannot be exactly correct. However, it is empirically observed that a pure Gaussian assumption at each point in the iteration is nonetheless a robust model for determining performance thresholds.

Figure 19 shows noise figure calculations for the mixed concatenated code analyzed in the previous two figures. In this figure, the concatenated code's noise figure at $E_b/N_0 = 0.0$ dB is computed in two different ways. The first method enforces the pure Gaussian assumption at the outputs of both the inner and outer decoders, while the second method applies the Gaussian assumption only at the output of the inner decoders and uses the analytically derived mixture-of-Gaussians model at other points during an iteration. Since the input–output SNR characteristics were determined by simulation, this means that Gaussian-mixture random variables were generated for one case, and pure Gaussian random variables for the other case, at the inputs to the inner decoders. For both methods, Gaussian-mixture random variables were generated at the input to the outer decoder. Also for both methods, the pure Gaussian density at the output of the inner decoders was modeled using the consistency condition, i.e., only the mean was needed. We observe from Fig. 19 that the pure Gaussian model yields almost the identical noise figure, and hence decoder convergence properties, as the mixture-of-Gaussians model, except that it is slightly more optimistic about the speed of decoder convergence on the far side of the iterative decoding tunnel.

Figure 20 shows another variation [22] of a mixing scheme for a rate-1/3 code, wherein the output of the repetition-3 outer code is either sent uncoded to the channel or sent through a rate-1 inner recursive convolutional code, octal 7/3. In this case, the optimum mix uses only 1 uncoded bit for each 79 convolutionally encoded bits. To achieve this mix while still maintaining an overall code rate of 1/3, the puncturer keeps 1 out of every 80 uncoded bits and 79 out of every 80 convolutionally encoded bits

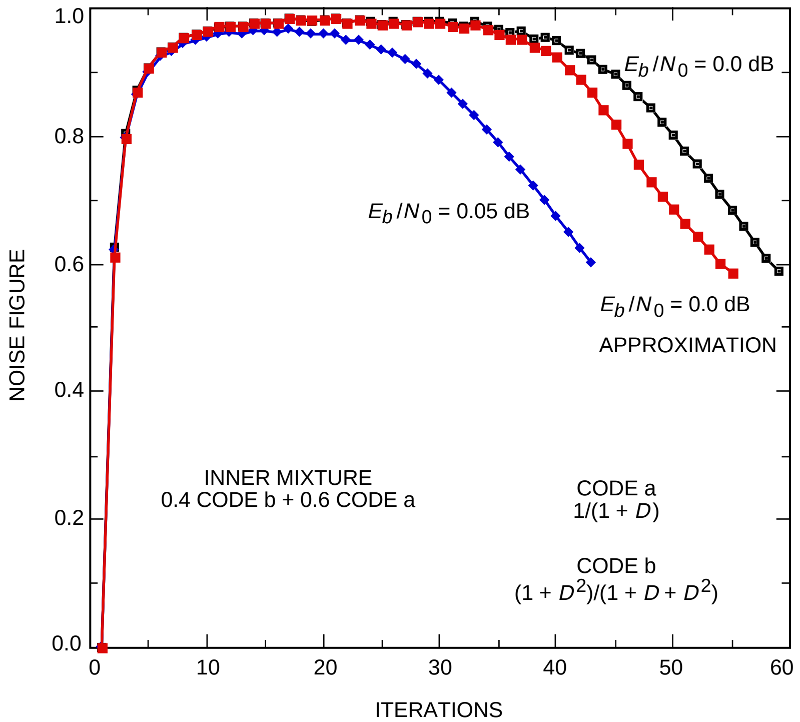

Fig. 19. Comparison of noise figures computed from Gaussian and a mixture of Gaussian models (for the output of the outer repetition code).

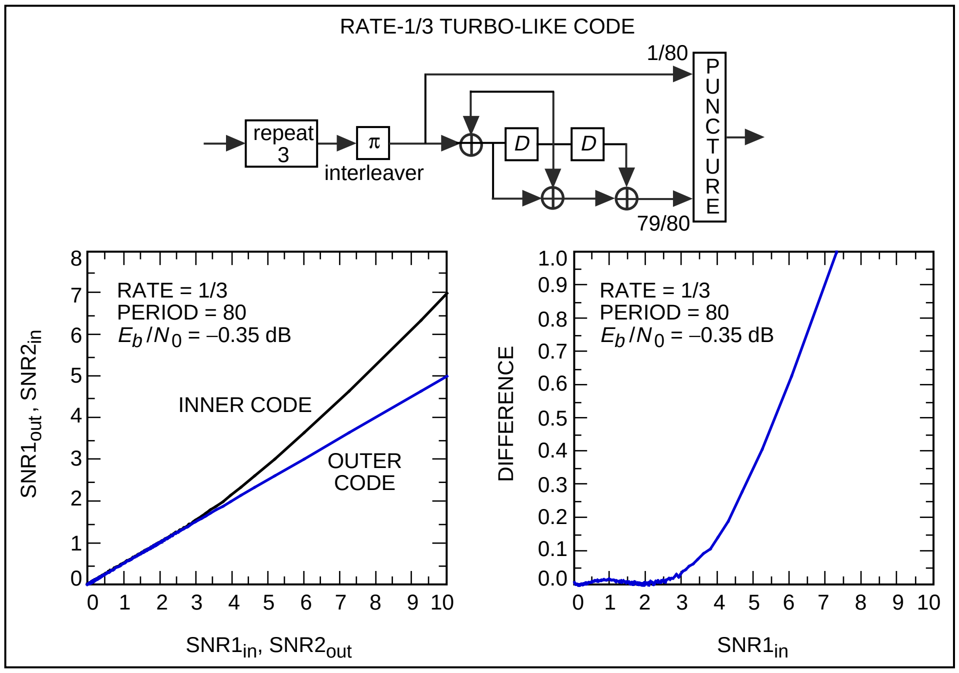

Fig. 20. A rate-1/3 concatenated code with mixed inner codes using doping as a mixing agent.

(denoted by "period = 80" in the figure). This technique of sending an occasional uncoded bit amongst many coded bits has been called "code doping" by ten Brink [23]. Without the code doping, the decoding of the inner code would fail to get started, because the rate-1, octal 7/3 code has a nontrivial feedforward component, and so its $\mathrm{SNR}_{\mathrm{out}}$ versus $\mathrm{SNR}_{\mathrm{in}}$ characteristic starts at the origin. The occasional uncoded bit raises the initial SNR just enough above zero to enable the iterative decoder to get started. Once started, the decoder moves through an extremely narrow iterative decoding tunnel that again constricts almost to the point of closure near SNR = 2. If the doping rate is decreased below 1 bit every 80 bits, the tunnel closes at this point and the SNR will not advance further. This code is an example for which the narrowest constrictions of the tunnel occur at two separated points, and care must be exercised to make sure that the decoder can squeeze through both openings. These two constriction points, near SNR = 0 and SNR = 2, are highlighted in the lower right graph in Fig. 20, which plots the *difference* between the $\mathrm{SNR}_{\mathrm{out}}$ versus $\mathrm{SNR}_{\mathrm{in}}$ characteristic of the mixed inner code and the straight-line SNR characteristic of the repetition-3 outer code.

The doped code in Fig. 20 yields an iterative decoding threshold of −0.35 dB, just 0.15 dB above the capacity limit, but at a price of extremely slow convergence through the long iterative decoding tunnel.

B. Concatenated Codes with Mixed Outer Codes

Next we consider an example showing that it is also possible to achieve a low decoding threshold by using a mixture of outer codes concatenated with a single inner code. The inner code in this example is a 4-state, rate-1 code (octal 1/7). Like the 2-state accumulator code, this code does not suffer from the extreme start-up deficiency of the 4-state, octal 5/7, rate-1 code, because the octal 1/7 code lacks multiple feedforward connections and each of its input bits is affected by only 3 channel symbols. The outer code is a mixture of two rate-1/2 codes, a simple repetition-2 code and a convolutional (1,5/7) code. The optimal mixing proportion in this case is to send about 2/3 of the bits to the repetition code and 1/3 of the bits to the convolutional code. Figure 21 shows the SNR characteristics of the individual codes and the optimally mixed outer code. We see that the SNR characteristics predict an iterative decoding threshold of about 0.5 dB, which is only about 0.3 dB above the capacity limit for rate-1/2 codes.

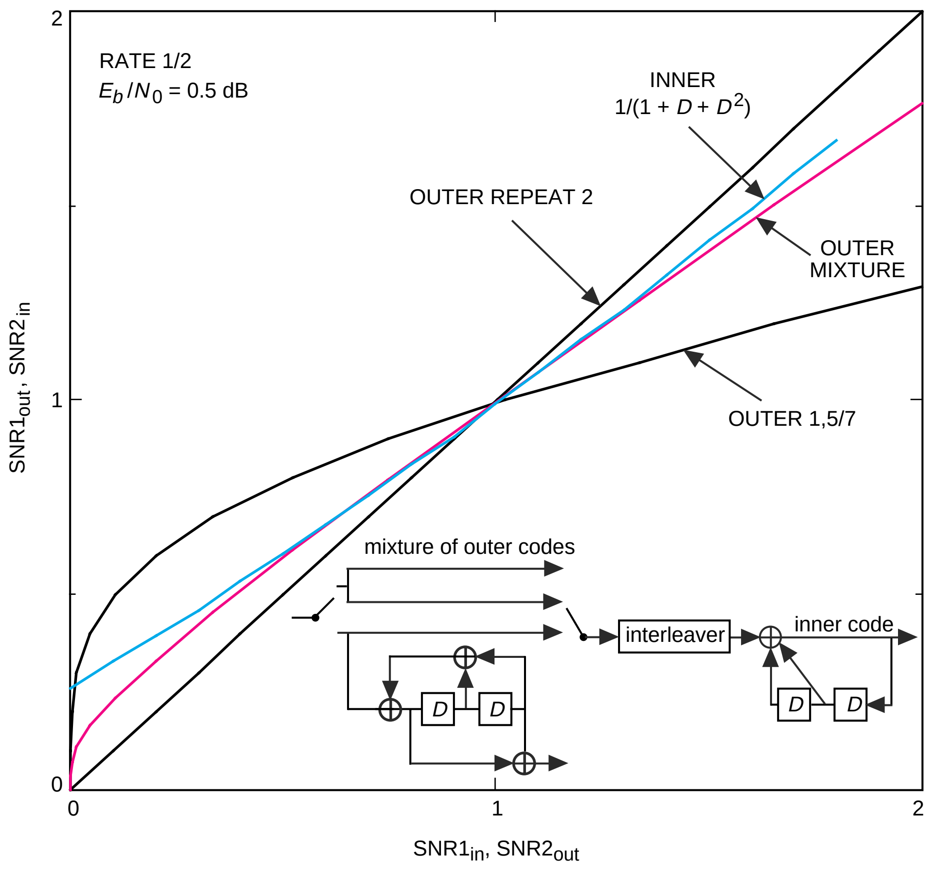

Fig. 21. Decoding threshold near capacity limit for a rate-1/2 concatenated code with mixed outer codes.

An alternative way to produce a mixture code with the same component codes and the same mixture proportions is shown in Fig. 22. This code is punctured deterministically to yield the same 2/3, 1/3 split of outer code symbols coming from the repetition code and the convolutional code, respectively. One of the repeated information blocks is unpunctured, 2/3 of the bits are deleted from the other uncoded outer block (puncturing pattern 001), and 1/3 of the bits are deleted from the convolutionally coded outer block (puncturing pattern 110). This deterministic mixing of the outer code components produces a code with minimum distance 5 instead of 2 and, hence, better asymptotic slope for its $\mathrm{SNR_{out}}$ versus $\mathrm{SNR_{in}}$ curve. The iterative decoding threshold for this code is 0.45 dB, just 0.27 dB above the capacity limit.

A similar rate-1/2 code with even lower complexity and equivalent iterative decoding threshold can be constructed as follows. Start with an outer recursive convolutional code, octal (1,5/7). Send 2/3 of the parity bits directly to the channel. Send the remaining 1/3 of the parity bits and all of the systematic (information) bits through an (infinitely long) interleaver to an inner 2-state rate-1 accumulator code. Figure 23 compares the iterative decoding performance of this code with that of a more straightforward serial concatenation of the octal (1,5/7) code (unpunctured) with the 2-state accumulator code. We see that the straightforward serial concatenation requires a minimum E_b/N_0 of 0.85 dB, because the sharp curvature of the SNR characteristic of the outer (1,5/7) convolutional code requires the inner accumulator code to have a fairly high starting output SNR value. The SNR characteristic of the hybrid outer code, sending 2 of every 6 bits directly to the channel, is not as sharply curved. This hurts the convergence rate at high SNR, but it widens the iterative decoding tunnel at its narrowest constriction and thus allows the accumulator's characteristic curve to drop substantially. Successful iterative decoding for this rate-1/2 code can take place at an E_b/N_0 of about 0.45 dB, the same as for the code in the previous figure.

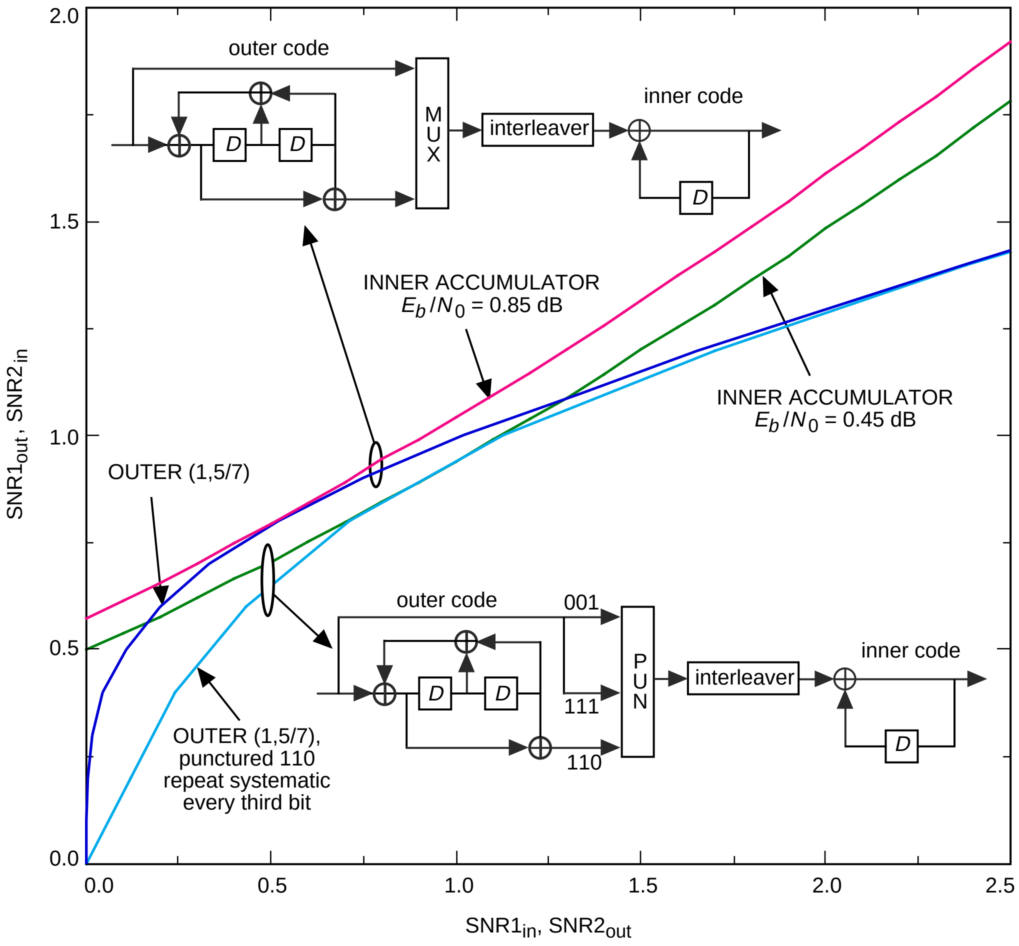

Fig. 22. Decoding threshold for a variation of the code in Fig. 21 based on deterministic mixing by selective puncturing.

VII. Low-Density Parity Check Codes

The graphical analysis method also gives insights into the design of low-density parity-check (LDPC) codes. The method is the same as that of [8,9], except now we interpret the set of variable nodes and the set of check nodes in the LDPC decoder's belief propagation network as two constituents of a turbo-like decoder, with separate SNR_{out} versus SNR_{in} characteristics. Since each variable node simply combines (independent) information about a given bit from several incoming sources, the messages passed to and from the variable nodes are like those passed to and from a decoder for a repetition code. The corresponding SNR_{in} versus SNR_{out} characteristic is a straight line with slope $1/(d_v - 1)$, where d_v is the degree of variable nodes, which is the number of connections to the variable node. The SNR characteristic for the collection of degree-d_c check nodes is obtained by averaging a product of tanh functions, as shown in [9]. Alternatively, this SNR characteristic may be obtained by simulation.

Figure 24 shows the SNR characteristics for the variable nodes and the check nodes of rate-1/2 LDPC codes with degrees (d_v, d_c) = (2,4), (3,6), (4,8), and (5,10). In this figure, the SNR characteristics for the variable nodes are straight lines with slopes 1, 1/2, 1/3, and 1/4, emanating from a nonzero SNR determined by the channel E_b/N_0 of 1.1 dB. The SNR_{out} versus SNR_{in} characteristics of the check nodes start from SNR = 0 and increase more slowly with SNR when the degree of the check nodes increases. In this example, the (2,4), (4,8), and (5,10) codes are not reliably decodable at this E_b/N_0 because their

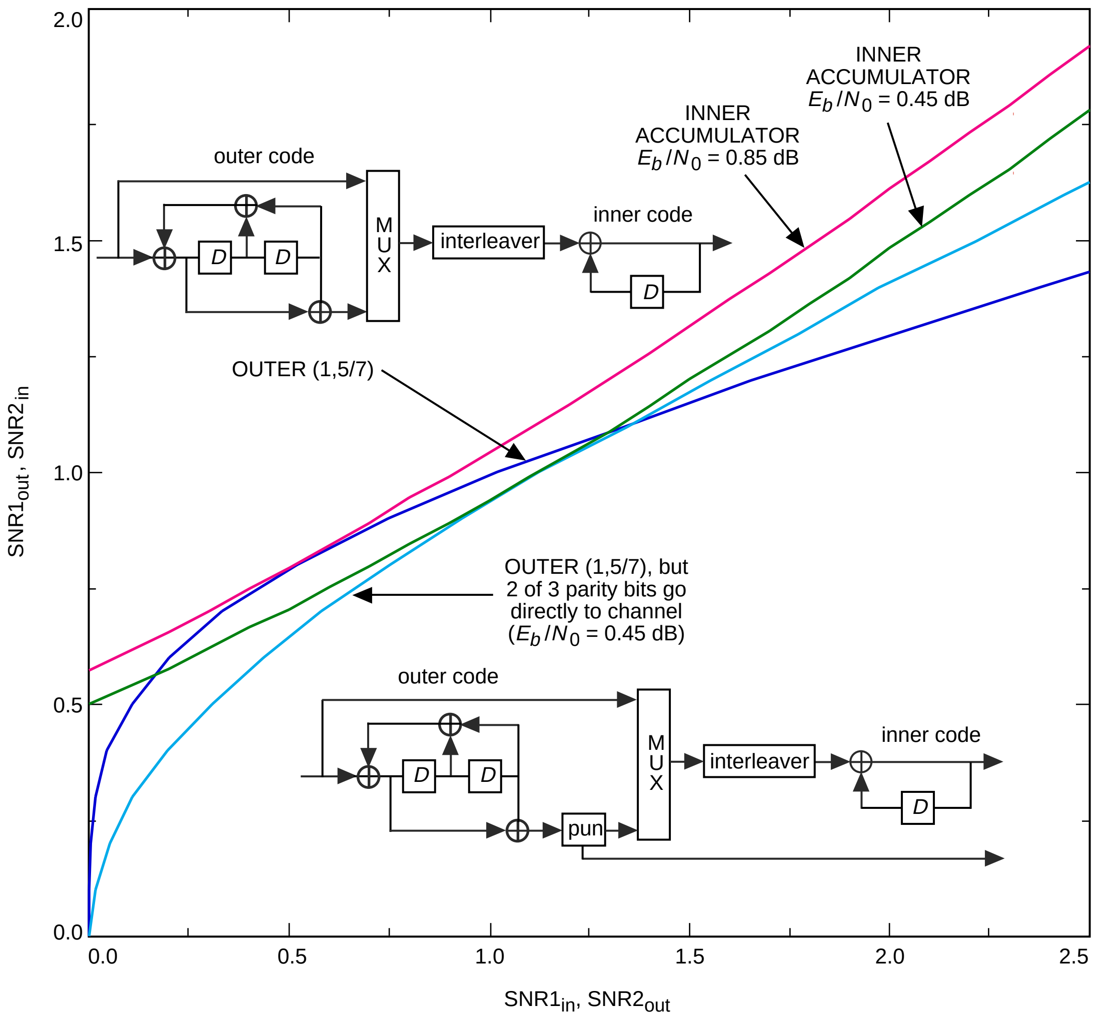

Fig. 23. Decoding threshold near capacity limit for a hybrid concatenated code.

respective SNR characteristics intersect. However, the (3,6) code is just at the limit of preserving a narrow iterative decoding tunnel for reliable convergence.

Now we look at an example showing the improvement obtainable by allowing variable nodes with mixed degrees. First consider a rate-1/2 (4,8) regular LDPC code. The iterative decoding threshold of this code is $E_b/N_0 = 1.6$ dB [9]. Then consider another code with two equal-sized groups of variable nodes of degrees 2 and 6. All of the check nodes are assumed to have degree 8. The resulting iterative decoding threshold for this code is 1.0 dB. This gives 0.6 dB improvement over the (4,8) regular LDPC, and the code rate is still 1/2. Figure 25 shows SNR curves for the check nodes and the variable nodes. As shown in the figure, the $\mathrm{SNR_{out}}$ versus $\mathrm{SNR_{in}}$ characteristic of the check nodes degrades due to the mixture input. However, the slope of the $\mathrm{SNR_{in}}$ versus $\mathrm{SNR_{out}}$ characteristic of the mixture of variable nodes decreases more, such that at $E_b/N_0 = 1.0$ dB the two curves touch each other. At the same E_b/N_0, the two SNR curves for the (4,8) regular LDPC code cross each other.

Returning to Fig. 24, we see that the SNR curves for the check nodes of different degrees all approach parallel straight lines with 1:1 slopes for high SNR. The asymptote for a degree-d_c check node satisfies the equation $\mathrm{SNR_{out}} = \mathrm{SNR_{in}} - 2\ln(d_c - 1)$. The straight-line equation for a variable node of degree d_v is $\mathrm{SNR_{out}} = (d_v - 1)\mathrm{SNR_{in}} + 2RE_b/N_0$. In the special case of variable nodes with degree $d_v = 2$, both slopes are equal. In this case, the $\mathrm{SNR_{out}}$ versus $\mathrm{SNR_{in}}$ asymptote for the check nodes will lie entirely above the $\mathrm{SNR}_{\mathrm{in}}$ versus $\mathrm{SNR}_{\mathrm{out}}$ curve for the variable nodes only if $2RE_b/N_0 > 2\ln(d_c-1)$, or $E_b/N_0 > [d_c/(d_c-2)]\ln(d_c-1)$, since in this case the code rate $R = 1-2/d_c$. This can serve as a lower bound on the E_b/N_0 threshold. This coincides with the result obtained by Wiberg [3] and is a special case of the stability condition obtained by Richardson et al. [8] for LDPC codes.

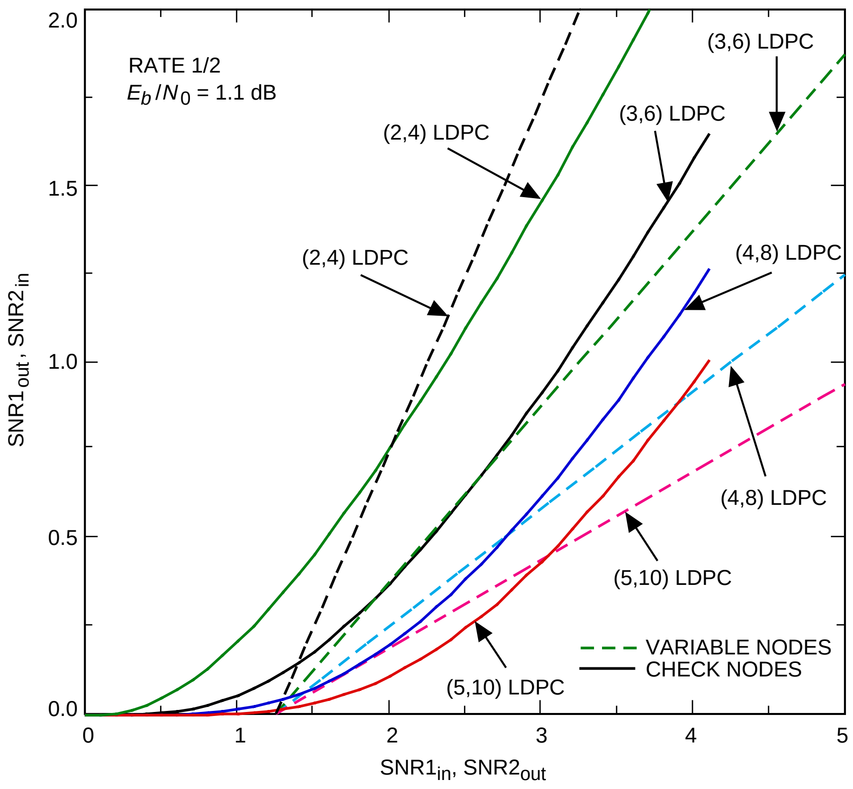

Fig. 24. Iterative decoding threshold analysis for rate-1/2 LDPC codes.

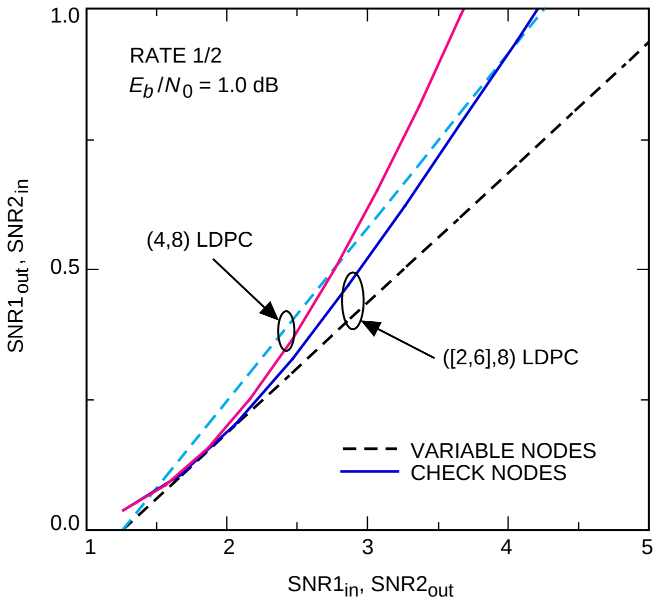

Fig. 25. Iterative decoding threshold analysis for rate-1/2 LDPC codes with mixture variable nodes.

VIII. Analytic Estimates of Gaussian Density Evolution for 2-State Constituent Codes

Analytic computation of the evolving mean of the extrinsic information messages for a concatenated code with convolutional constituent codes is in general a difficult problem. Approximate solutions have not produced accurate computations of iterative decoding thresholds. However, by imitating the analysis method used in [9] for LDPC codes, we are able to derive analytic expressions for general serial and parallel concatenations using 2-state constituent codes.

A. Analysis for LDPC codes

For iterative decoding of LDPC codes using a belief-propagation network, there are two types of nodes and three types of extrinsic information messages. There are variable nodes corresponding to coded symbols and check nodes corresponding to the parity check equations. The variable nodes send messages v to the check nodes and receive messages λ_c from the channel and u from the check nodes. The check nodes send messages u to the variable nodes and receive messages v from the variable nodes.

The message v going from a variable node to a given check node is computed as a linear sum of the incoming channel message and the extrinsic messages from the other check nodes:

$$v = \lambda_c + \sum_{m=1}^{d_v-1} u_m \tag{1}$$

where d_v is the degree of the variable node and $\{u_m, m = 1, \cdots d_v - 1\}$ are the incoming messages from the $d_v - 1$ other check nodes connected to the given variable node. The message going from a check node to a variable node is computed as a nonlinear function of the extrinsic messages from the other variable nodes connected to this check node. For the sum-product algorithm, the result is expressed as

$$\tanh\left(\frac{u}{2}\right) = \prod_{m=1}^{d_c-1} \tanh\left(\frac{v_m}{2}\right) \tag{2}$$

Using the Gaussian approximation and the consistency condition, we only need to compute the mean of the extrinsic messages. The mean $\bar{v}$ of the message from a variable node to a check node is simply the sum of the mean $\bar{\lambda}_c$ of the channel message and the means $\bar{u}$ of the incoming messages $\{u_m\}$ from the other check nodes,

$$\bar{v} = \bar{\lambda}_c + (d_v - 1)\bar{u} \tag{3}$$

For an AWGN channel, $\bar{\lambda}_c = 2/\sigma^2$, where $1/(2\sigma^2) = (k/n)(E_b/N_0)$.

At the next step in the iterative process, the mean $\bar{u}$ of the message from a check node back to a variable node is computed by averaging Eq. (2) over the assumed Gaussian densities for u and $\{v_m\}$. Define $\psi(\mu)$ to be the average of $\tanh(y/2)$ over a Gaussian random variable y with mean μ and variance 2μ:

$$\psi(\mu) = \int_{-\infty}^{\infty} \tanh\left(\frac{y}{2}\right) \frac{1}{\sqrt{4\pi\mu}} \exp\left(-\frac{(y-\mu)^2}{4\mu}\right) dy \tag{4}$$

In terms of $\psi(\cdot)$, the update equation for the mean of a message from a check node to a variable node is

$$\psi(\bar{u}) = [\psi(\bar{v})]^{d_c - 1} \tag{5}$$

Here we have assumed that the variable node messages $\{v_m\}$ and the check node messages $\{u_m\}$ all have the same means, $\bar{v}$ and $\bar{u}$, respectively. These update equations for LDPC codes have been obtained in [9].

B. Analysis for Concatenated Codes with 2-State Inner Codes

We consider two versions of the 2-state convolutional code: the rate-1 accumulator code, octal (1/3); and the rate-1/2 recursive convolutional code, octal (1,1/3). Although convolutional codes are conventionally constructed on a trellis, these 2-state codes have simple Tanner graph representations [13] without loops. This graph representation is shown in Fig. 26. Here the nodes corresponding to the parity bits and the information bits play the role of the variable nodes for LDPC codes, and they are connected through a set of check nodes.

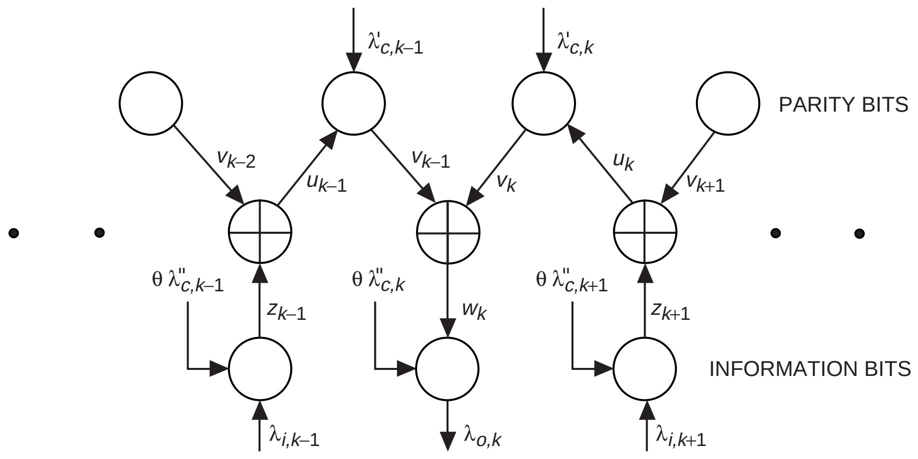

Fig. 26. Tanner graph for a 2-state convolutional inner code, used to calculate extrinsic-information messages corresponding to input information bits.

First we use the 2-state convolutional code, either rate-1 or rate-1/2, as the inner code of a serial concatenation or as one of the constituent codes of a parallel concatenation. The input extrinsic information corresponding to the jth information bit is denoted $\lambda_{i,j}$, and we wish to compute the output extrinsic information $\lambda_{o,k}$ for information bit k.

The message from the channel corresponding to the kth parity bit is denoted $\lambda'_{c,k}$, and the message from the channel corresponding to the (systematic) information bit is denoted by $\theta\lambda''_{c,k}$, where $\theta = 1$ for the case of the rate-1/2 code and $\theta = 0$ for the rate-1 accumulator code. We also define intermediate messages $\{u_j\}$, $\{v_j\}$ between the check nodes and the variable nodes associated with the parity bits, and intermediate messages $\{w_j\}$, $\{z_j\}$ between the check nodes and the variable nodes associated with the information bits, as indicated in the figure.

Two key message-passing equations are evident from the graph. The equation for the messages $\{u_k = v_k - \lambda'_{c,k}\}$ sent from the check nodes to the variable nodes associated with the parity bits is

$$\tanh\left(\frac{v_k - \lambda'_{c,k}}{2}\right) = \tanh\left(\frac{v_{k+1}}{2}\right)\tanh\left(\frac{\lambda_{i,k+1} + \theta\lambda''_{c,k+1}}{2}\right) \tag{6}$$

and the equation for messages $\{w_k = \lambda_{o,k} - \theta\lambda''_{c,k}\}$ sent from the check nodes to the variable nodes associated with the information bits is

$$\tanh\left(\frac{\lambda_{o,k} - \theta\lambda^{''}_{c,k}}{2}\right) = \tanh\left(\frac{v_{k-1}}{2}\right)\tanh\left(\frac{v_k}{2}\right) \tag{7}$$

With large code blocks, we can argue that averages over the various types of messages will be independent of the bit location k. Taking the steady-state averages of these two expressions produces first an equation to be solved for the mean $\bar{v}$ of the messages $\{v_k\}$, in terms of the mean $\bar{\lambda}_i$ of the input extrinsics $\lambda_{i,k}$ and the mean $\bar{\lambda}_c = 2/\sigma^2$ of the channel messages $\lambda^{'}_{c,k}$, $\lambda^{''}_{c,k}$,

$$\psi\left(\bar{v} - \bar{\lambda}_c\right) = \psi\left(\bar{v}\right)\psi\left(\bar{\lambda}_i + \theta\bar{\lambda}_c\right) \tag{8}$$

and second an equation for the mean $\bar{\lambda}_o$ of the output extrinsics $\lambda_{o,k}$,

$$\psi\left(\bar{\lambda}_o - \theta\bar{\lambda}_c\right) = \left[\psi\left(\bar{v}\right)\right]^2 \tag{9}$$

C. Analysis for Doubly Serial Concatenations of 2-State Codes

We can also use the 2-state convolutional code, either rate-1 or rate-1/2, as an outer code or more generally as the middle code of a serial concatenation of three codes [15,16]. Again we use a Tanner graph representation for the 2-state convolutional code. In the general case, there are three types of input and output extrinsic messages that must be analyzed, corresponding to the code's input and output (if rate-1/2) information bits and to its output parity bits.

The graph representation for computing the output extrinsic messages $\lambda_{o,k}$, $\lambda^{''}_{o,k}$, corresponding to the input or output (if rate-1/2) information bits, respectively, is almost identical to that of Fig. 26, except that the input messages from the channel $\lambda^{'}_{c,k}$, $\lambda^{''}_{c,k}$ are replaced by input extrinsic-information messages $\lambda^{'}_{i,k}$, $\theta\lambda^{''}_{i,k}$ from an inner code, corresponding to the parity bits and the (systematic) information bits (if rate-1/2), respectively. There are also input extrinsic-information messages $\lambda_{i,k}$ from an outer code, corresponding to the information bits. Now the output extrinsic messages depend on all three types of input extrinsic messages, and the means $\bar{\lambda}_o$, $\bar{\lambda}^{''}_o$ of the output messages are functions of the corresponding means $\bar{\lambda}_i$, $\bar{\lambda}^{'}_i$, $\bar{\lambda}^{''}_i$ of $\lambda_{i,k}$, $\lambda^{'}_{i,k}$, $\lambda^{''}_{i,k}$, respectively. To determine $\bar{\lambda}_o$ and $\bar{\lambda}^{''}_o$, first solve the following equation for $\bar{v}$:

$$\psi\left(\bar{v} - \bar{\lambda}^{'}_i\right) = \psi(\bar{v})\psi\left(\bar{\lambda}_i + \theta\bar{\lambda}^{''}_i\right) \tag{10}$$

Then the solution $\bar{v}$ can be used to calculate $\bar{\lambda}_o$ as the solution of

$$\psi\left(\bar{\lambda}_o - \theta\bar{\lambda}^{''}_i\right) = \left[\psi\left(\bar{v}\right)\right]^2 \tag{11}$$

and $\bar{\lambda}^{''}_o$ as the solution of

$$\psi\left(\bar{\lambda}^{''}_o - \bar{\lambda}_i\right) = \left[\psi(\bar{v})\right]^2 \tag{12}$$

The graph representation for computing the output extrinsic messages $\lambda^{'}_{o,k}$ corresponding to the parity bits is shown in Fig. 27. The message flow in this graph produces a nonlinear equation,

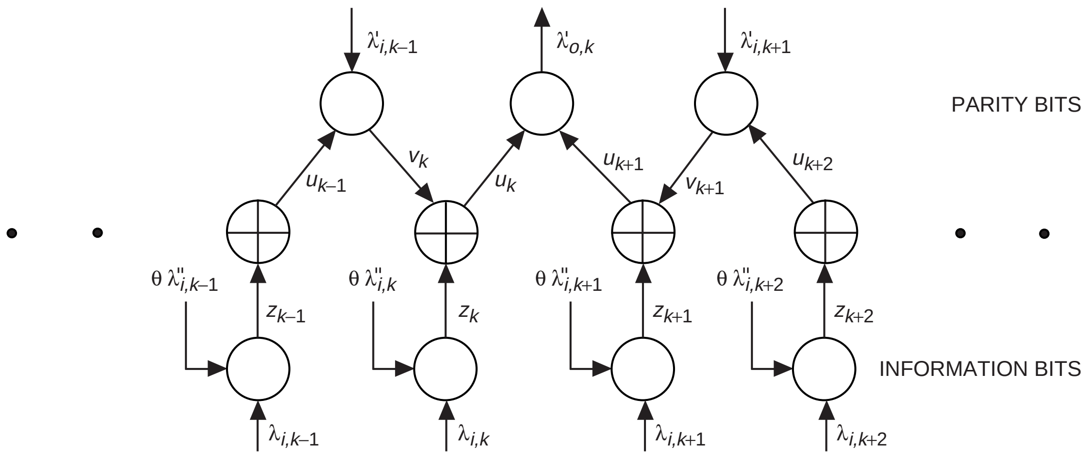

Fig. 27. Tanner graph for a 2-state convolutional middle code, used to calculate output extrinsic-information messages corresponding to output parity bits.

$$\tanh\left(\frac{u_k}{2}\right) = \tanh\left(\frac{u_{k-1} + \lambda'_{i,k-1}}{2}\right)\tanh\left(\frac{\lambda_{i,k} + \theta\lambda''_{i,k}}{2}\right) \tag{13}$$

for the output of the check nodes, and a linear equation,

$$\lambda'_{o,k} = u_k + u_{k+1} \tag{14}$$

for the output of the variable nodes corresponding to the parity.

Again we assume a steady-state condition such that averages are independent of bit location k and obtain the following equation to be solved for the mean $\bar{u}$ of u_k:

$$\psi(\bar{u}) = \psi\left(\bar{u} + \bar{\lambda}'_i\right)\psi\left(\bar{\lambda}_i + \theta\bar{\lambda}''_i\right) \tag{15}$$

Finally, the mean $\bar{\lambda}'_o$ of the output extrinsic-information messages for the parity bits is computed in terms of $\bar{u}$ by averaging the linear equation above to obtain

$$\bar{\lambda}'_o = 2\bar{u} \tag{16}$$

The results in this section are for a general configuration where the 2-state code is situated amidst a series of concatenations. For the rate-1 accumulator code, set $\theta = 0$, or set $\theta = 1$ for the 2-state code with rate-1/2. If the code is used as an outer code, set $\bar{\lambda}_i = 0$. If the entire output of the rate-1/2 code is serially concatenated through an infinitely long random interleaver with an inner code, then we can also set $\bar{\lambda}'_i = \bar{\lambda}''_i$. However, we can also generally allow $\bar{\lambda}'_i \neq \bar{\lambda}''_i$ to model the case when the rate-1/2 code's systematic bits and parity bits are sent through separate interleavers to different inner codes. In particular, this applies to a code concatenation that sends the systematic bits uncoded to the channel, while permuting the parity bits and sending them to another layer of coding.

To extend these analytic results to more complex constituent codes, the evolution of the mean of the extrinsic information can be approximately computed from stage to stage of a general code trellis, but the approximation requires ignoring a normalization factor, and to date we have not found this method to give satisfactory predictions of decoding thresholds.

An example of this analysis method applied to a 2-state code used in a doubly serial configuration is shown in Fig. 28. This is a rate-1/3 systematic code obtained by sending one copy of the information bits to the channel and two copies through a series of two rate-1 accumulators preceded by interleavers. Compared to the rate-1/3 repeat-and-accumulate (RA) code in Fig. 6, the repeat-and-doubly-accumulate (RDA) code in this figure has a slightly lower iterative decoding threshold of about 0.4 dB. The curves in Fig. 28 analyze this code in two pieces, with the innermost rate-1 accumulator as one constituent and the rest of the code as the second. The $\mathrm{SNR}_{\mathrm{out}}$ versus $\mathrm{SNR}_{\mathrm{in}}$ characteristics of these two pieces are shown along with the straight-line characteristic of the repetition-3 outer code used in the construction of the plain RA code [19] in Figs. 6 and 17. We see from the figure that the SNR characteristic of the repetition-3 outer code just barely intersects the SNR characteristic of the rate-1 accumulator code. This produces a slightly higher E_b/N_0 iterative decoding threshold for the plain RA code. More significantly, the SNR characteristic of the stronger outer code defined in Fig. 28 curves sharply away from the straight-line characteristic of the repetition-3 code. This shortens the iterative decoding tunnel and enables the iterative decoder to converge much faster.

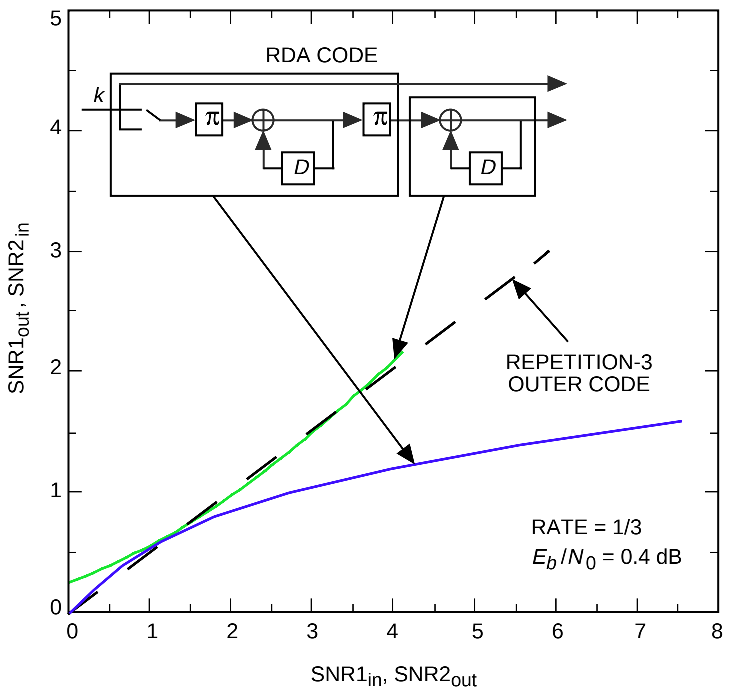

Fig. 28. Iterative decoding threshold analysis for rate-1/3 RDA code.

IX. Conclusion

We tracked the density of extrinsic information in iterative turbo decoders by actual density evolution, and also approximated it by consistent Gaussian density functions. Then we used both models to analyze the convergence of iterative decoding for turbo codes and for serially concatenated codes. The approximate Gaussian method, although not as exact as actual density evolution, nonetheless gives accurate predictions of iterative decoding thresholds compared to simulated decoder performance.

A noise figure was defined for one full loop through the iterative decoder, such that the turbo decoder will converge to the correct codeword if the noise figure is bounded by a number lower than 1. By decomposing the code's noise figure into individual curves of output SNR versus input SNR corresponding to the individual constituent codes, we gained many new insights into the performance of the iterative decoder for different constituents. Many mysteries of turbo codes can be explained based on this analysis. For example, we illustrated why certain codes converge better with iterative decoding than do more powerful codes, which are only suitable for maximum-likelihood decoding. The roles of systematic bits and of recursive convolutional codes as constituents of turbo codes were explained based on this analysis.

Having identified the strengths and weaknesses of particular inner and outer constituent codes through their input–output SNR characteristics, we then generalized the analysis to include serial concatenations of mixtures of different outer and inner constituent codes. Such mixtures allow us to design better constituent codes that exhibit more of the strengths and fewer of the weaknesses of the individual components of the mix. The input–output SNR analysis method provides good graphical insight into understanding how to choose mixture components that complement each other. We gave examples of simple rate-1/2 and rate-1/3 mixture configurations, using component codes with at most four states, that approach their respective capacity limits within 0.3 dB to 0.5 dB.

While the general method for determining the constituent codes' input–output SNR characteristics was by Monte Carlo simulation, we also gave analytic expressions for these curves for the particular case of a 2-state constituent. These expressions cover both the rate-1, octal (1/3) accumulator code and the rate-1/2, octal (1,1/3) recursive convolutional code, used in any concatenation configuration, whether as an inner code, an outer code, or as a middle code in a series of more than two concatenated codes.

References

[1] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon Limit Error-Correcting Coding: Turbo Codes," *Proceedings 1993 IEEE International Conference on Communications*, Geneva, Switzerland, pp. 1064–1070, May 1993.

[2] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara, "Serial Concatenation of Interleaved Codes: Performance Analysis, Design, and Iterative Decoding," *IEEE Transactions on Information Theory*, vol. 44, issue 3, pp. 909–926, May 3, 1998.

[3] N. Wiberg, *Codes and Decoding on General Graphs*, Ph.D. Dissertation, Department of Electrical Engineering, Linkoping University, S-581 83 Linkoping, Sweden, 1996.

[4] H. El Gamal, *On the Theory and Application of Space-Time and Graph Based Codes*, Ph.D. Dissertation, University of Maryland at College Park, 1999.

[5] H. El Gamal and A. R. Hammons, Jr., "Analyzing the Turbo Decoder Using the Gaussian Approximation," ISIT 2000, p. 319; also submitted to *IEEE Transactions on Information Theory*, January 2000.

[6] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal Decoding of Linear Codes for Minimizing Symbol Error Rate," *IEEE Transactions on Information Theory*, vol. IT-20, pp. 284–287, 1974.

[7] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara, "Soft-Input Soft-Output Maximum A Posteriori (MAP) Module to Decode Parallel and Serial Concatenated Codes," *The Telecommunications and Data Acquisition Progress Report 42-127, July–September 1996*, Jet Propulsion Laboratory, Pasadena, California, pp. 1–20, November 15, 1996.
http://tmo.jpl.nasa.gov/tmo/progress_report/42-127/127H.pdf

[8] T. Richardson and R. Urbanke, "The Capacity of Low Density Parity Check Codes Under Message Passing Decoding," submitted to *IEEE Transactions on Information Theory Theory.*
http://cm.bell-labs.com/cm/ms/former/tjr/papers/ldpc.ps

[9] S.-Y. Chung, T. Richardson, and R. Urbanke, "Analysis of Sum-Product Decoding of Low-Density Parity-Check Codes Using Gaussian Approximation," submitted to *IEEE Transactions on Information Theory.*
http://lthcwww.epfl.ch/papers/gaussianapproximation.ps

[10] T. Richardson, A. Shokrollahi, and R. Urbanke, "Design of Provably Good Low-Density Parity Check Codes," submitted to *IEEE Transactions on Information Theory.*
http://cm.bell-labs.com/cm/ms/former/tjr/papers/degree.ps

[11] T. Richardson and R. Urbanke, "Analysis and Design of Iterative Decoding Systems," 1999 IMA Summer Program: Codes Systems and Graphical Models, Minnesota, August 2-6, 1999.

[12] Consultative Committee for Space Data Systems (CCSDS), "Telemetry Channel Coding," vol. 101.0-B-4, Blue Book, issue 4, May 1999.
http://www.ccsds.org/documents/pdf/CCSDS-101.0-B-4.pdf

[13] R. M. Tanner, "A Recursive Approach to Low Complexity Codes," *IEEE Transactions on Information Theory*, vol. IT-27, pp. 533–547, 1981.

[14] S. ten Brink, "Convergence of Iterative Decoding," *Electronics Letters*, vol. 35, no. 13, pp. 806–808, May 24, 1999.

[15] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara, "Analysis, Design, and Iterative Decoding of Double Serially Concatenated Codes with Interleavers," *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 2, pp. 231–244, February 1998.

[16] H. Pfister and P. H. Siegel, "On the Serial Concatenation of Rate-One Codes Through Uniform Random Interleavers," *Proceedings 37th Allerton Conference on Communication, Control and Computing*, Monticello, Illinois, September 22–24, 1999.

[17] J. Hagenauer, E. Offer, and L. Papke, "Iterative Decoding of Binary Block and Convolutional Codes," *IEEE Transactions on Information Theory*, vol. 42, no. 2, pp. 429–431, March 1996.

[18] M. Luby, M. Mitzenmacher, A. Shokrollahi, and D. Spielman, "Analysis of Low Density Codes and Improved Designs Using Irregular Graphs," *Proceedings of the 30th Annual ACM Symposium on Theory of Computing*, New York, pp. 249–258, 1998.

[19] D. Divsalar, H. Jin, and R. J. McEliece, "Coding Theorems for 'Turbo-Like' Codes," 1998 Allerton Conference, Monticello, Illinois, September 23–25, 1998.

[20] J. Hagenauer and P. Hoeher, "Concatenated Viterbi Decoding," International Workshop on Information Theory, Sweden, August 1989.

[21] P. D. Alexander, A. J. Grant, and M. C. Reed, "Iterative Detection in Code-Division Multiple-Access with Error Control Coding," *ETT*, vol. 9, no. 5, pp. 419–425, September–October 1998.

[22] D. Divsalar, S. Dolinar, and F. Pollara, "Low Complexity Turbo-Like Codes," 2nd International Symposium on Turbo Codes, oral presentation, Brest, France, pp. 73–80, September 2000.

[23] S. ten Brink, "Rate One-Half Code for Approaching the Shannon Limit by 0.1 dB," *Electron. Letters*, vol. 36, no. 15, pp. 1293–1294, July 2000.

[24] O. M. Collins, O. Y. Takeshita, and D. J. Costello, Jr., "Iterative Decoding of Nonsystematic Turbo Codes," *Proceedings 2000 IEEE International Symposium on Information Theory*, Sorrento, Italy, p. 172, June 2000.

[25] D. Agrawal and A. Vardy, "The Turbo Decoding Algorithm and Its Phase Trajectories," *Proceedings 2000 IEEE International Symposium on Information Theory*, Sorrento, Italy, p. 316, June 2000.

[26] T. Richardson and R. Urbanke, "Thresholds for Turbo Codes," *Proceedings 2000 IEEE International Symposium on Information Theory*, Sorrento, Italy, p. 317, June 2000.

[27] S. Vialle and J. Boutros, "Performance Limits of Concatenated Codes with Iterative Decoding," *Proceedings 2000 IEEE International Symposium on Information Theory*, Sorrento, Italy, p. 150, June 2000.

[28] 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 Communications Letters*, vol. 5, issue 2, February 2001.