Iterative Turbo Decoder Analysis Based on Gaussian Density Evolution

Dariush Divsalar, Sam Dolinar, and Fabrizio Pollara
Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA
e-mail: {dariush,sam,fabrizio}@shannon.jpl.nasa.gov

ABSTRACT

We model the density of extrinsic information in iterative turbo decoders by Gaussian density functions. This model is verified by experimental measurements. We consider evolution of these density functions through the iterative turbo decoder as a nonlinear dynamical system with feedback. Iterative decoding of turbo codes and of serially concatenated codes are analyzed based on this method. Many mysteries of turbo codes can be explained based on this analysis. For example we can explain why certain codes converge better with iterative decoding than 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 are explained based on this analysis.

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 [8].

The analysis of iterative decoders for concatenated codes with short blocks is an unsolved problem. However for very large block sizes this analysis is possible under certain assumptions. The asymptotic (as block size goes to infinity) iterative decoding analysis can be based on the method of density evolution proposed by Richardson and Urbanke [4], see also [6], [9]. Using this method, the capacity threshold for low density parity check (LDPC) codes over binary input AWGN channels was computed. Wiberg [3] in his dissertation has shown that the extrinsic information in iterative decoding can be approximated by a Gaussian density function. El Gamal [5] in his dissertation considered the soft-input soft-output APP module in turbo decoders as a signal-to-noise ratio (SNR) transformer, and also suggested a method for analyzing the overall turbo decoder. A method for analyzing the convergence of the decoder similar to the one developed here, but based on mutual information, was discussed in [13].

In this paper, we analyze turbo codes and serially concatenated codes by approximating the density functions for the extrinsics as Gaussian densities, and then computing the mean and variance in the Gaussian density evolution. This approximation was used to obtain a threshold on minimum bit signal-to-noise ratio E_b/N_0 for LDPC codes [6], based on using only the means of Gaussian densities. First we determine the input and output Gaussian means and variances of the individual SISO modules by simulation. Then, in Sections III and IV, we use these results to explain conditions of turbo decoder convergence and to resolve other mysteries associated with turbo codes. At each iteration, we compute input and output SNR's for the two component decoders. We also define, as in low noise amplifiers, a "noise figure" for turbo decoders. We argue that if the noise figure of the iterative turbo decoder is below 0 dB then the iterative decoder converges to the correct codeword. Another method, for concatenated codes with two component codes such as parallel and serial turbo codes, 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 cross, then the iterative decoder converges. We used all the assumptions made by Richardson and Urbanke 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).

Our method of modeling the density of the extrinsics

This work was funded by the TMOD Technology Program and performed at the Jet Propulsion Laboratory, California Institute of Technology under contract with the National Aeronautics and Space Administration.

using independent measurements of mean and variance will result in a slightly pessimistic threshold since the Gaussian density has the highest entropy for a given variance. Slightly optimistic threshold results are obtained if we impose density consistency as proposed by Richardson et al [6]. Consistency under the Gaussian assumption implies that the variance of the extrinsic density function is twice its mean, and therefore SNR=mean/2, given that all +1s are transmitted. This makes the analysis easier since we only need to determine the mean of the extrinsic information. The mean may be determined by simulation for any convolutional code, or analytically for certain simple codes such as 2-state recursive convolutional codes.

A concentration theorem [12], [9] can be used to make these results more rigorous. The concentration theorem says that the average bit error probability 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. Such 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 [9], 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 traceback in Viterbi decoding).

II. The Gaussian Density Evolution Model

Consider a parallel turbo code as shown in Fig. 1. The turbo decoder is based on two SISO modules as shown in Fig. 2 and described in detail in [8].

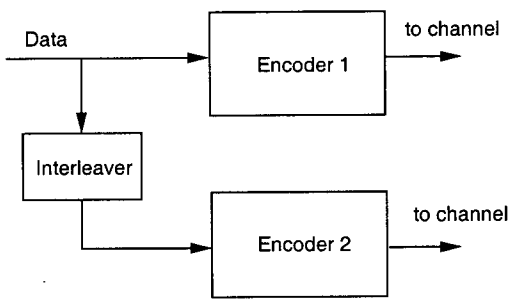

Fig. 1. The structure of a turbo code.

The iterative decoder can be viewed as a nonlinear dynamical feedback system. Extrinsic information messages are passed from one decoder to the other. We computed the histogram of the extrinsic information at the output of a SISO module. As shown in Fig. 3, a Gaussian assumption is a good approximation for the

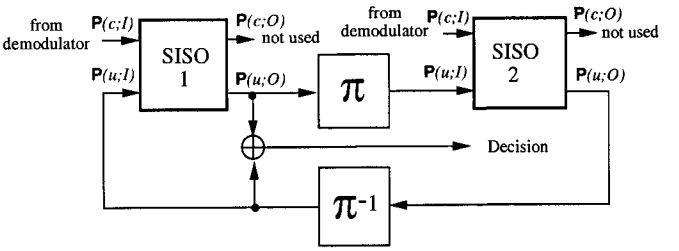

Fig. 2. Iterative turbo decoder for turbo codes.

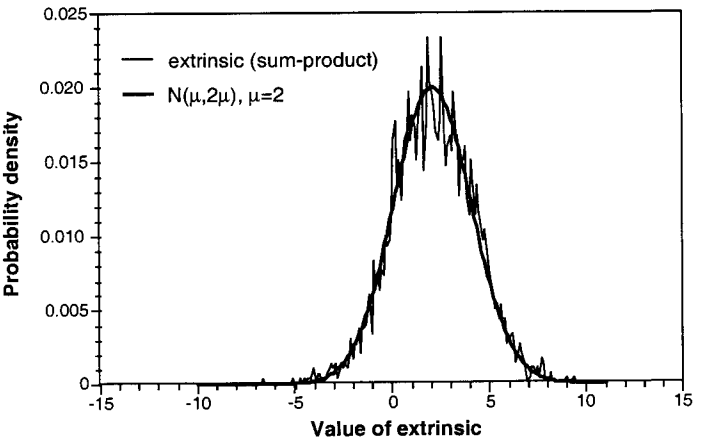

Fig. 3. The density function of extrinsic and its Gaussian approximation

probability density function of the extrinsic information.

With large interleavers, the extrinsic information messages are independent and identically distributed, given, say that the all-zero codeword is transmitted (corresponding to, say, transmission of +1's on the channel). Each message is modeled by a Gaussian random variable with mean μ_i and variance σ_i^2 at the ith iteration, and the signal-to-noise ratio (SNR) of this random variable is defined as SNR $\triangleq \mu_i^2/\sigma_i^2$. If the consistency assumption is used, then $\sigma_i^2 = 2\mu_i$.

Consider the input and output SNR's for each decoder at each iteration as shown in Fig. 4. These are denoted SNR1_{in}, SNR1_{out}, SNR2_{in}, SNR2_{out}, and they represent the SNR's associated with the extrinsic information messages, not the SNR associated with the channel observations. A nonzero E_b/N_0 from the channel enables decoder 1 to produce a nonzero SNR1_{out} for the output extrinsic information despite starting with $\text{SNR1}_{\text{in}} = 0$. 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. 4. We have $\text{SNR1}_{\text{out}} = G_1(\text{SNR1}_{\text{in}}, E_b/N_0)$ and $\text{SNR2}_{\text{out}} =$

$G_2(\mathrm{SNR2}_{\mathrm{in}}, E_b/N_0)$. Also, $\mathrm{SNR2}_{\mathrm{in}} = \mathrm{SNR1}_{\mathrm{out}}$, and thus $\mathrm{SNR2}_{\mathrm{out}} = G_2(G_1(\mathrm{SNR1}_{\mathrm{in}}, E_b/N_0), E_b/N_0)$.

III. A Model for Decoder Convergence

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 is bounded lower than 1 for the entire range of input SNR to decoder 1, then the SNR's 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. These claims can be justified by the results in [4], [6], [12], [9] on iterative decoding decoding thresholds for LDPC codes.

Equivalently we can test the decoder convergence by plotting 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. 5. In this figure we considered a rate 1/3 CCSDS turbo code [10] 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 curve corresponds to the input-output function G_1 for decoder 1, and the lower curve corresponds to G_2^{-1} for decoder 2.

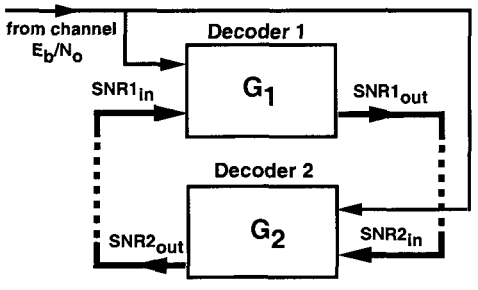

Fig. 4. Analysis of turbo decoding as a nonlinear dynamical system with feedback using Gaussian density evolution.

Figure 5 also 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. 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 and 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 $\mathrm{SNR1}_{\mathrm{in}} = 0$ is dependent on the E_b/N_0 due to the channel observations. If we reduce E_b/N_0 from the value of 0.8 dB used in Fig. 5, 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.

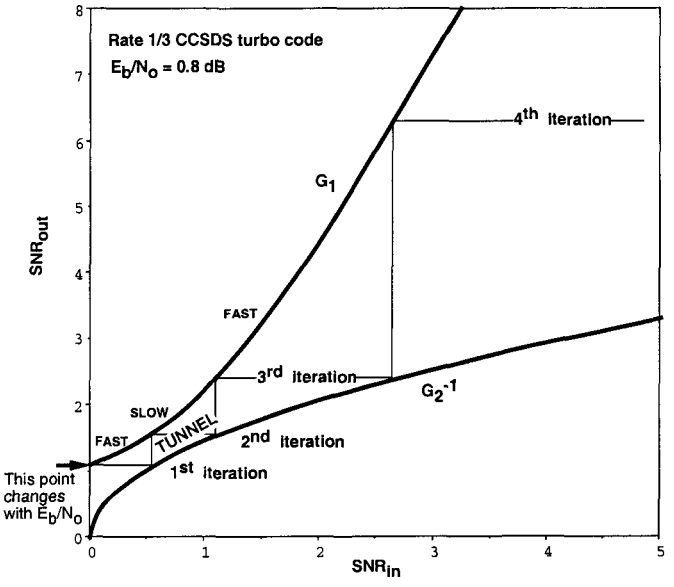

Fig. 5. Iterations and convergence of a turbo decoder.

IV. Some Turbo Code Mysteries Explained

The "noise figure" analysis is more accurate, but the graphical method using separate $\mathrm{SNR}_{\mathrm{out}}$ versus $\mathrm{SNR}_{\mathrm{in}}$ for the constituent codes can help to explain many mysteries of turbo codes and provide more insight into the design of good concatenated codes. For example, 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 with-

out transmission of the systematic bits. An explanation for the *role of the systematic bits* can be surmised from Fig. 6. Although the received bits corresponding to

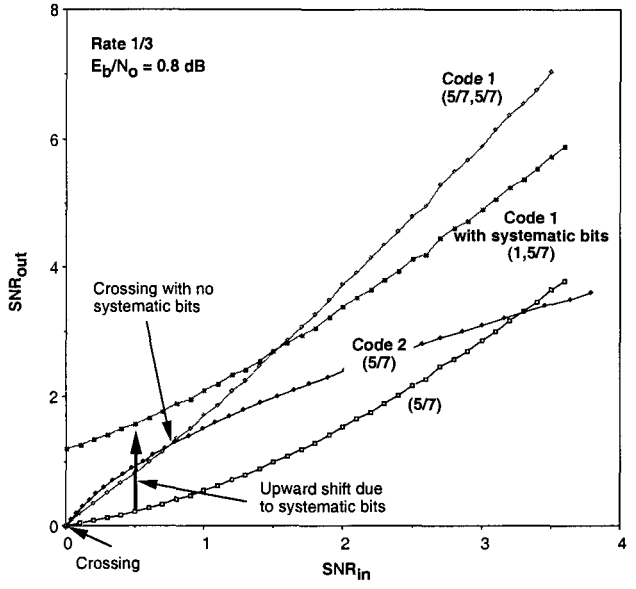

Fig. 6. The role of systematic bits in turbo codes

parity bits have nonzero SNR, if we don't send the systematic bits the SNR of the extrinsic information at the first iteration will be almost zero. This causes the curve for Code 2 in Fig. 6 to intersect, near SNR = 0, the two curves for Code 1 in the same figure that correspond to nonsystematic constituent codes, octal (5/7) and octal (5/7,5/7). In these cases, there will be no convergence 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 a rate-1 recursive convolutional code (obtained by puncturing the systematic bits of a rate-1/2 systematic recursive convolutional code). If the highest degree of the feed-forward polynomial is greater than or equal to 1, then the 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 highest degree of the feed-forward polynomial is zero, then the input equals the modulo-2 sum of only a few output parity bits, depending on the highest 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 feed-forward polynomial is 1 (highest degree is zero). Another example is the rate-1 $\frac{1}{1+D+D^2}$ convolutional code. For example, consider a parallel turbo code constructed from a differential encoder and a rate-1 recursive 16-state convolutional code, as shown in Fig. 7. For this code, there is a substantial nonzero

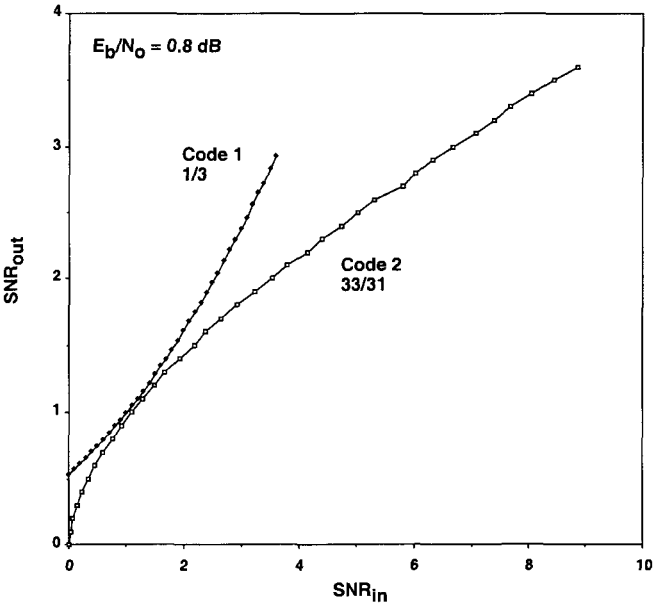

Fig. 7. Example of not sending the systematic bits

$\mathrm{SNR1}_{\mathrm{out}}$ for decoder 1 at the end of the first iteration. In such cases, the 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 feed-forward 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 feed-forward polynomial coefficients $\{p_i\}$ and feed-back polynomial coefficients $\{q_i\}$. Assume without loss of generality that $p_j = 1$ for some j. Then, using the algebra introduced by Hagenauer [14], 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 $\{y_k\}$:

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

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 feed-forward component, the iterations will start with a nonzero SNR, because in that case there

are no tanh(·) 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 [7], [8], the distribution of α and β remains uniform. Due to the symmetry of 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 is 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.

Next we explain the *role of recursive convolutional codes* by considering what happens when the component codes are nonrecursive. As shown in Fig. 8, 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 $\mathrm{SNR1}_{\mathrm{out}}$ 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.

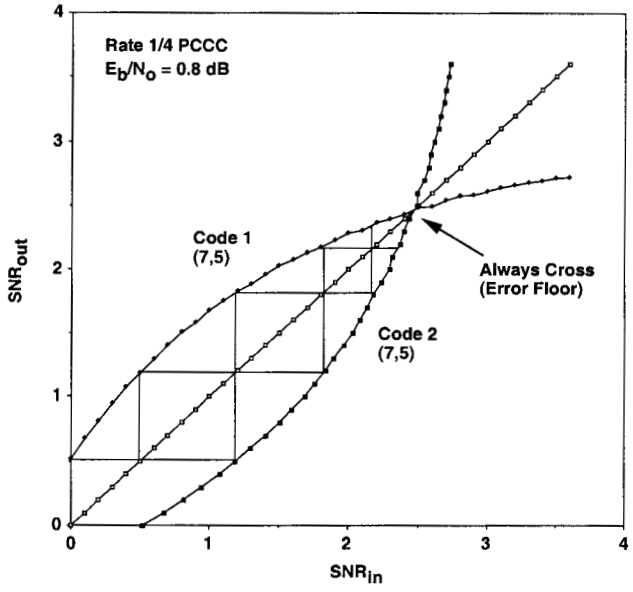

Fig. 8. Convergence of non-recursive encoders

For 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 [1].

Figure 9 elucidates the *role of a primitive feedback polynomial.* In this figure we show the $\mathrm{SNR_{out}}$ versus $\mathrm{SNR_{in}}$ curves for two different rate-1/3 four-state turbo codes. One code is the same code considered in Fig. 6, for which the feedback polynomial is primitive (octal 7). The other code uses a nonprimitive feedback polynomial (octal 5). We see that for high SNR's the two curves for the code using primitive feedback diverge from each other much more rapidly than the curves for the code with nonprimitive feedback. This produces faster convergence when the iterations reach this region. On the other hand, in the low SNR region the iterative decoding tunnel is slightly narrower for the code with primitive feedback than for the one with nonprimitive feedback. This can cause the iterative decoding threshold for the code with nonprimitive feedback to be lower than that for the code with primitive feedback, even though the opposite conclusion would be true if the decoders were maximum likelihood.

In Fig. 9, a difference in slopes corresponds to the rate of convergence of the iterative decoder. A large slope difference implies faster convergence. Primitive feedbacks, e.g., octal (5/7) instead of octal (7/5) may result in faster convergence above the decoding threshold.

Figure 10 shows the *role of different state complexities.* The $\mathrm{SNR_{out}}$ versus $\mathrm{SNR_{in}}$ curves are plotted for rate-1/3 turbo codes with 4-state and 16-state constituents. 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 higher decoding threshold.

Figure 11 shows how a similar analysis method can be applied when turbo codes are viewed as a serial concatenation of an outer repetition code with an inner recursive convolutional code. The two concatenated codes in this figure are rate-1/3 turbo codes with 4-state and

[1]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/No.

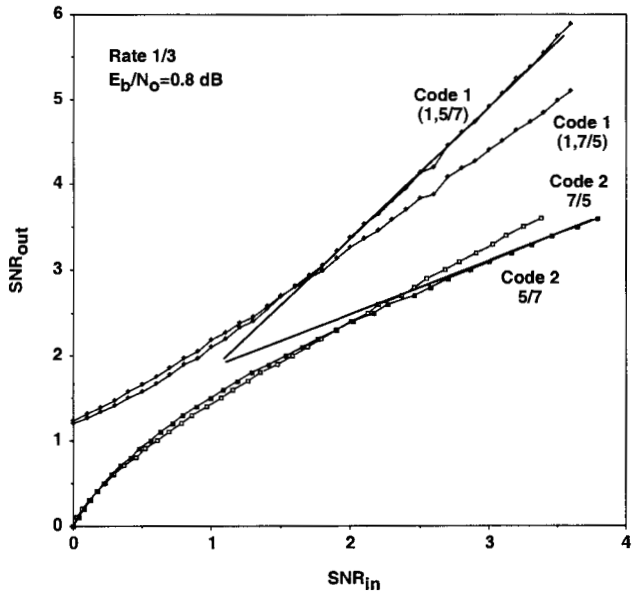

Fig. 9. Rate 1/3 PCCC with two 4-state Codes (primitive vs nonprimitive feedback)

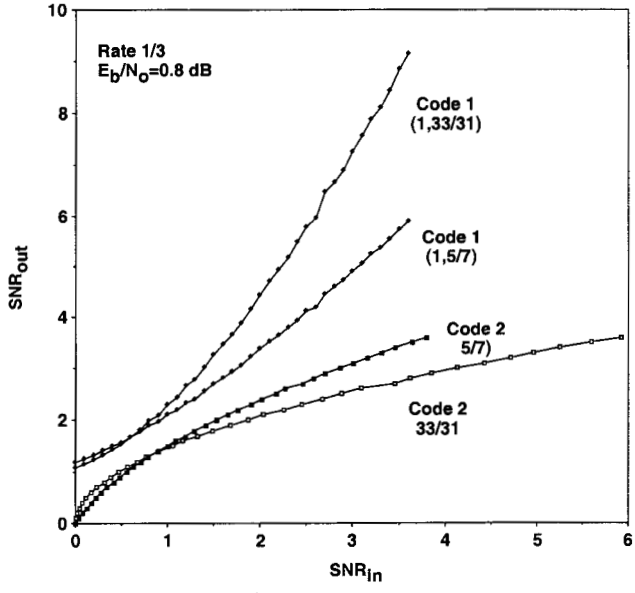

Fig. 10. Comparison of rate 1/3 PCCCs with different state complexities

8-state constituents. In each case, 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 $\mathrm{SNR_{out}}$ versus $\mathrm{SNR_{in}}$ 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. 6, and it suffers from the same problem of starting with an

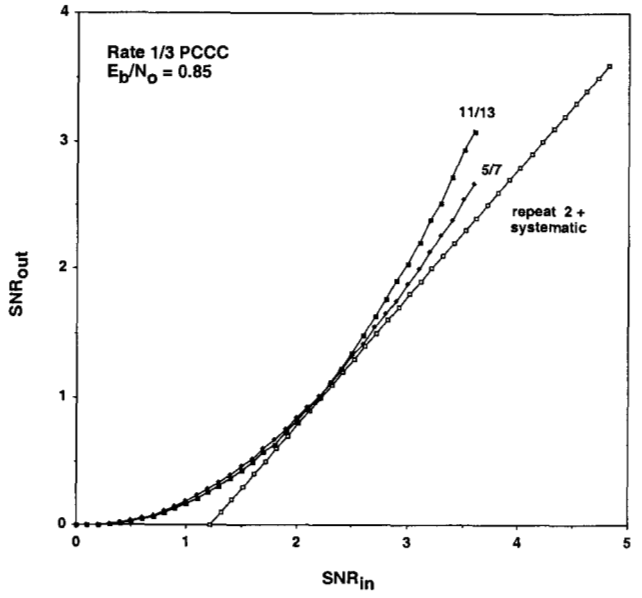

Fig. 11. Turbo codes viewed as serial concatenation of outer repetition code with inner code

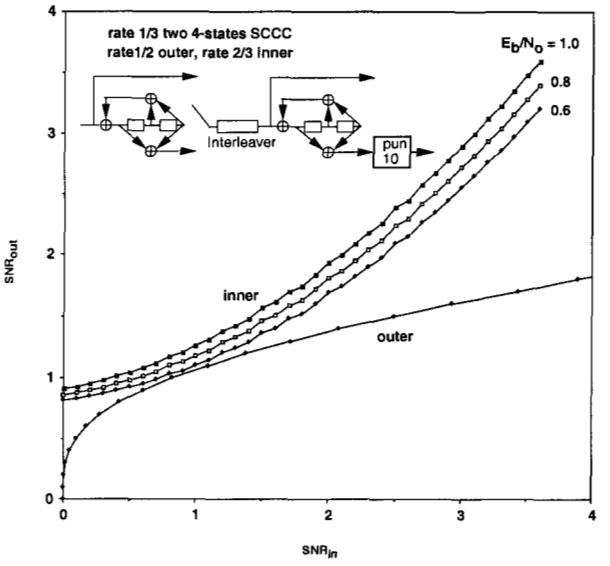

Fig. 12. Example of a rate 1/3 serial turbo code

output SNR near 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. Decoder convergence for the 8-state code (octal 11/13) is similar, though small performance differences can be deduced due to the different curvatures, as in Fig. 10.

Figure 12 illustrates the analysis method applied to a serially concatenated 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. 11. 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 0.6 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 $\mathrm{SNR_{in}}$ versus $\mathrm{SNR_{out}}$ 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 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 asymptotic slope of the $\mathrm{SNR_{in}}$ versus $\mathrm{SNR_{out}}$ 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.

Finally, we show in Fig. 13 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 [11], tested at two values of E_b/N_0 just above the iterative decoding threshold. The noise figure in this case rises (with each iteration i) from an initial value of 0 to a maximum just below 1, indicating that decoder convergence is achieved.

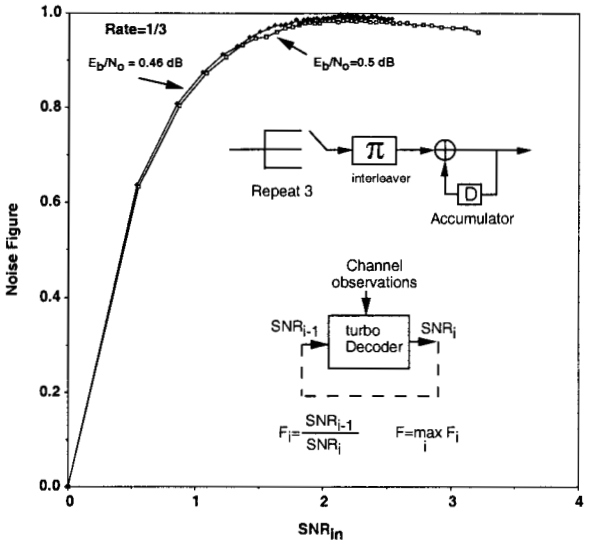

Fig. 13. Noise figure for a rate 1/3 repeat-and-accumulate (RA) code.

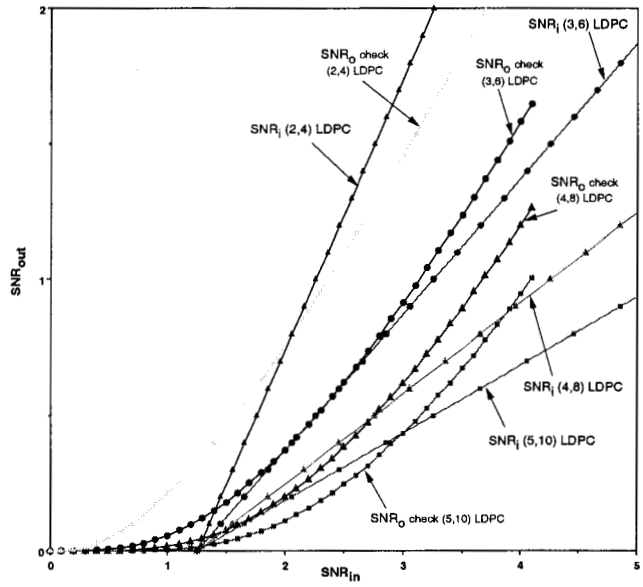

Fig. 14. Iterative decoding threshold analysis for rate-1/2 LDPC codes.

V. 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 [4], 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 $\mathrm{SNR}_{\mathrm{out}}$ versus $\mathrm{SNR}_{\mathrm{in}}$ 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 $\mathrm{SNR}_{\mathrm{in}}$ versus $\mathrm{SNR}_{\mathrm{out}}$ 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 check nodes is obtained by averaging a product of tanh functions, as shown in [4]. Alternatively, this SNR characteristic may be obtained by simulation.

Figure 14 shows the SNR characteristics for the variable nodes and the check nodes of rate-1/2 LDPC codes with degrees (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, 1/4, emanating from a nonzero SNR determined by the channel E_b/N_0; in this case, $E_b/N_0 = 1.1$ dB. The $\mathrm{SNR}_{\mathrm{out}}$ versus $\mathrm{SNR}_{\mathrm{in}}$ 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 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 1.6 dB [6]. Now consider another code with two equal-size groups of variable nodes of degrees 2 and 6. All of the check nodes are assumed to have degree 8. The resulting E_b/N_0 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 15 shows SNR curves for the check nodes and the variable nodes. As shown in the figure, the $\mathrm{SNR}_{\mathrm{out}}$ versus $\mathrm{SNR}_{\mathrm{in}}$ characteristic of the check nodes degrades due to the mixture input. However, the slope of the $\mathrm{SNR}_{\mathrm{in}}$ versus $\mathrm{SNR}_{\mathrm{out}}$ 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. 14, 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 $\mathrm{SNR}_{\mathrm{out}} = \mathrm{SNR}_{\mathrm{in}} - 2\ln(d_c - 1)$. The straight-line equation for a variable node of degree d_v is $\mathrm{SNR}_{\mathrm{out}} = (d_v - 1)\mathrm{SNR}_{\mathrm{in}} + 2RE_b/N_0$. In the special case of variable nodes with degree $d_v = 2$, both

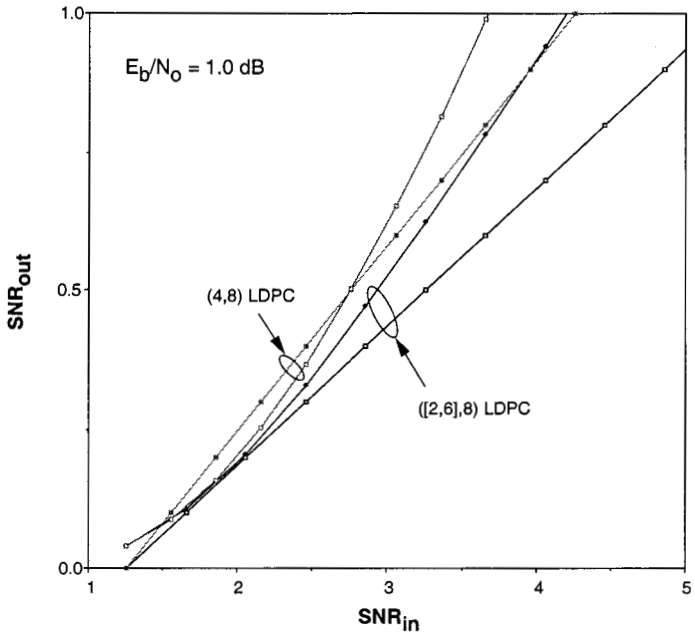

Fig. 15. Iterative decoding threshold analysis for rate-1/2 LDPC codes with mixture variable nodes.

slopes are equal. In this case the $\mathrm{SNR}_{\mathrm{out}}$ versus $\mathrm{SNR}_{\mathrm{in}}$ 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 $E_b/N_0 > \frac{d_c}{d_c-2}\ln(d_c - 1)$, 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 [4] for LDPC codes.

References

[1] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon Limit Error-Correcting Coding: Turbo Codes", *Proc. 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*, May 3, 1998.

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

[4] T. Richardson and R. Urbanke, The capacity of low density parity check codes under message passing decoding, submitted to IEEE trans. on Information Theory.

[5] H. El Gamal, On the theory and application of space-time and graph based codes," Ph.D. dissertation, 1999, University of Maryland at College Park.

[6] 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.

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

[8] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara, "Soft input soft output MAP module to decode parallel and serial concatenated codes", *The Telecommunications and Data Acquisition Progress Report 42-127, July–September 1996*, JPL, Pasadena, California, November 15, 1996.

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

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

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

[12] 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, 1998. pp. 249-258.

[13] S. ten Brink, "Convergence of iterative decoding", Electronics Letters, vol. 35, no. 13, 24th June 1999.

[14] J. Hagenauer, E. Offer, and L. Papke, "Iterative decoding of binary block and convolutional codes", IEEE Trans. on Information Theory, Vol. 42, No. 2, March 1996, pp. 429-431.