Stochastic Analysis of Turbo Decoding

Minyue Fu, *Fellow, IEEE*

***Abstract*—This paper proposes a stochastic framework for dynamic modeling and analysis of turbo decoding. By modeling the input and output signals of a turbo decoder as random processes, we prove that these signals become ergodic when the block size of the code becomes very large. This basic result allows us to easily model and compute the statistics of the signals in a turbo decoder. Using the ergodicity result and the fact that a sum of lognormal distributions is well approximated using a lognormal distribution, we show that the input–output signals in a turbo decoder, when expressed using log-likelihood ratios (LLRs), are well approximated using Gaussian distributions. Combining the two results above, we can model a turbo decoder using two input parameters and two output parameters (corresponding to the means and variances of the input and output signals). Using this model, we are able to reveal the whole dynamics of a decoding process. We have discovered that a typical decoding process is much more intricate than previously known, involving two regions of attraction, several fixed points, and a stable equilibrium manifold at which all decoding trajectories converge. Some applications of the stochastic framework are also discussed, including a fast decoding scheme.**

***Index Terms*—Iterative decoding, maximum *a posteriori* probability (MAP) decoding, soft decoding, turbo codes, turbo decoding.**

I. Introduction

THIS paper addresses the following question: How does a turbo decoder behave when the code block size becomes very large? It turns out that many (known and new) properties of turbo decoding can be understood by studying this question.

Many papers can be found which attempt to uncover the mystery of the turbo decoding method invented in [1], see, e.g., [2]–[14]. There are two approaches: deterministic and stochastic. The deterministic approach treats decoding as a deterministic process. That is, the aim is to characterize the behavior of the decoder for each instance of the input signal (the received signal). The work of Richardson [4] exemplifies this approach. Typical questions this approach attempts to answer are concerned with the existence, uniqueness, and stability of equilibrium (or fixed) points. Using geometrical analysis, [4] is able to reveal a number of dynamic behaviors of turbo decoding, including the existence of fixed points and some conditions for the uniqueness and stability of fixed points. The existence of fixed points is also proved in [7] using a somewhat simpler but also deterministic approach. However, the existence of a fixed point does not imply anything about the convergence of turbo decoding, as the system may exhibit multiple fixed points or even limit cycles. Moreover, the conditions for the uniqueness and stability are conservative, specific to each decoding block, and difficult to compute, meaning that they are not useful in predicting the performance of a given turbo decoder. The drawbacks of the deterministic approach are largely due to the fact that a turbo decoder is a high-dimensional nonlinear mapping. Nevertheless, the deterministic approach provides good insights into the dynamics of turbo decoding.

The stochastic approach, on the other hand, views the input and output of a decoder as random processes and tries to characterize their statistics. The most prominent feature of the stochastic approach is that these statistics are easily computable using realizations (or instances) of the random processes. This approach is naturally motivated by the fact that the end result we want is always some sort of statistical measure of the decoder. Notable examples of the stochastic approach include the methods given in [9]–[12]. These methods all use a single statistical parameter to characterize the input or output signal in each decoding phase or iteration. More precisely, a log-maximum *a posteriori* (Log-MAP) decoder is modeled as a single-input single-output (SISO)[1] model. This parameter is either the so-called mutual information [9], [10], [12] or the signal-to-noise ratio (SNR) [10], [11]. The SISO model stems from the motivation that a Log-MAP decoder, or any decoder for that matter, can be viewed as a mapping for transforming the *a priori* mutual information or SNR into the extrinsic mutual information or SNR. Aiding to this motivation is the following observation initially made in [13]: The log-likelihood ratios (LLRs) of the *a priori* signal and extrinsic signal in a turbo decoder can be approximated using Gaussian distributions. Using SISO models, [9]–[12] are able to explain a number of important features of turbo decoding, including the well-known step-like (or waterfall) performance of turbo decoding and fixed points at a low SNR. In particular, the so-called extrinsic information transfer (EXIT) chart of [9] and similar charts in [10], [11] are found particularly useful in understanding and quantifying the dynamics of turbo decoding.

It is also possible to combine the deterministic and stochastic approaches. This is done in [6] by extending the approach of [4] to a stochastic framework. By doing so, it is shown that a unique fixed point exists with an arbitrarily high probability when the received signal has an asymptotically low SNR. It is also shown that when the received signal has an asymptotically high SNR, a locally stable fixed point exists with an arbitrarily high probability. These results are in agreement with the typical step-like performance of turbo decoding, but only so in a qualitative sense. A SISO stochastic model is also given in [6].

This paper also embarks on the stochastic approach. Our study, however, aims to answer a number of important questions which are not addressed by the existing work on the stochastic approach. Namely, we want to know the following.

Manuscript received January 12, 2004; revised July 28,2004.

The author is with School of Electrical Engineering and Computer Science, University of Newcastle, Callaghan, N.S.W. 2308 Australia (e-mail: eemf@ee.newcastle.edu.au).

Communicated by R. J. McEliece, Associate Editor for Coding Theory.
Digital Object Identifier 10.1109/TIT.2004.839494

[1]Please do not confuse this acronym with *soft-input soft-output*.

1) When is the stochastic approach meaningful?
2) What statistical parameters are needed to model a turbo decoder?
3) How do we compute these statistics?
4) How does the dynamics of a turbo decoder (or turbo dynamics for short) behave?
5) How do we explain these behaviors?

The first question is answered by our first main result which roughly says that the stochastic approach is indeed valid when the block size is very large. More precisely, we prove the following.

When the received signal is subject to additive white Gaussian noise (AWGN) and the interleaver is chosen randomly, the turbo decoding output for each iteration approaches an ergodic random process as the block size approaches infinity.

The implications of the ergodicity are significant. First of all, an ergodic random process, by definition, is stationary. This implies that all the samples have the same distribution and thus the same statistics. Second, all the commonly used statistics (such as mean, variance, cumulative density function (CDF), SNR) can be computed by averaging over a *single* realization of the random process, justifying the meaningfulness of the stochastic approach. This fact also answers the third question above, i.e., only one (but long) decoding simulation needs to be performed and the statistics computed from different realizations are identical with probability 1 (or with high probability when the block size is large but finite). In other words, the ergodic property allows us to analyze the performance of turbo decoding (and also the performance of the associated turbo code) without being boggled down by the nonlinearities of the decoding function.

Based on the ergodic property above, we proceed to answer the second question listed by studying the distributions of the decoding output. We consider Log-MAP decoding for turbo codes with AWGN. We first show that the input and output of Log-MAP decoding, when expressed using a scaled LLR (SLLR), are related in the following way: The output is a log-sum of exponentials of the input. In other words, the exponential of the output is the sum of exponentials of the input. Then, using the well-known fact that a sum of lognormal distributions can be well approximated using a lognormal distribution, we establish our second important result.

When the received signal is subject to AWGN and the block size approaches infinity, the decoding output of each turbo decoding iteration (based on Log-MAP decoding) is well approximated using a Gaussian distribution.

The significance of this result is that a Log-MAP decoder can now be viewed as a two-input two-output (TITO) model with input–output parameters corresponding to the means and variances of the SLLR of the *a priori* signal and the SLLR of the extrinsic signal. This greatly simplifies the characterization and estimation of the decoding process.

We noted earlier that the Gaussian approximability has been observed in [9]–[12] and used to build SISO models. However, our work differs from these papers in the following ways. First, we provide theoretical justifications for the Gaussian approximation. That is, Gaussian approximations are valid because 1) the decoding output, when the block size becomes very large, becomes an ergodic random process, and 2) sums of lognormal distributions can be approximated using lognormal distributions. Second, we point out that the SISO models in [9]–[12] are inadequate for explaining the evolution of the mean and variance of the LLRs in the decoding process. To make this point clear, we note that the received signal, when subject to AWGN, has the property that its LLR also has a Gaussian distribution. Moreover, we have $\sigma^2 = 2m$, where m and σ^2 are the mean and variance of the LLR, respectively (see Section III-A). That is, the mean and variance are not two independent parameters. It turns out that although the input–output signals (also expressed using LLR) can be approximated using Gaussian distributions, their means and variances do not obey the same relationship as mentioned above. We thus need two parameters (mean and variance) to characterize each signal. The inadequacy in [9]–[12] is that the mean-to-variance relationship above is maintained throughout the decoding process. This is done to satisfy the so-called *symmetry condition* of [34] (see Remark 2 for details). We will show that this may lead to somewhat unsatisfactory approximations. Third, we show that an inadequate use of a SISO model may also lead to very erroneous implications. For example, based on a SISO model, [11] concludes that a Log-MAP decoder is a nondecreasing mapping from the input SNR to the output SNR ([11, Proposition 1]). This property is then used to prove the superb performance of a typical turbo decoder and the existence of a possible fixed point associated with a high bit-error rate (BER) ([11, Propositions 2 and 3]). We will show that this nondecreasing property is false. Another example is that the SISO models using mutual information in [9], [10], [12] are unable to explain a "strange" phenomenon which we will demonstrate that a Log-MAP decoder can take an input with very "good" mutual information and produce an output with much "poorer" mutual information.

Before addressing questions 4 and 5, we need to make a small detour to discuss SNRs. The goal of a turbo decoder is to convert a received signal with a low SNR to a decoded signal with a high SNR. To differentiate these two types of SNR, we will call the former the *channel SNR* and the latter the *decoder SNR*. For AWGN channels, the channel SNR is characterized by E_b/N_0 (the bit-energy-to-noise-density ratio). The decoder SNR determines the BER of the decoding.

Now we can return to questions 4 and 5. One of the applications of the TITO model is the analysis of fixed points. When the block size becomes very large, it turns out a turbo decoder may have two stable fixed points: one at a low decoder SNR and one at a high decoder SNR. For simplicity, we will call them low- and high-SNR fixed points, respectively. A low-SNR fixed point occurs when the channel SNR, E_b/N_0, is too low (typically, a fraction of a decibel away from the Shannon limit). A low-SNR fixed point results in a high BER and cannot be improved by increasing the block size. However, they can be avoided by increasing E_b/N_0. Using ergodicity, we are able to estimate the E_b/N_0 thresholds for different turbo codes. In contrast, the high-SNR fixed point is the result of a finite block size. We develop a simplified model for Log-MAP decoding at a high input SNR. By using this model, we are able to show that for all commonly used $1/3$-rate turbo codes, the decoder SNR can grow indefinitely as the block size increases. That is, the high-SNR fixed point can be pushed up arbitrarily by increasing the block size. This implies that, when the turbo decoder is void of a low-SNR fixed point, its BER can be made arbitrarily small by increasing the block size.

Moreover, by using the TITO model, we reveal that turbo dynamics is much more intricate than previously known. Indeed, when the block size is very large and E_b/N_0 is low, the state space of turbo decoding contains two regions of attraction separated by a stability boundary: one leading to a low-SNR fixed point and another to a high-SNR fixed point. There is also a stable equilibrium manifold in the state space at which all decoding trajectories converge. These features are illustrated in Fig. 15. When E_b/N_0 exceeds a certain threshold, the two regions of attraction merge and the low-SNR fixed point disappears, resulting in the familiar superb performance of turbo decoding.

In addition to developing and analyzing the TITO model, we demonstrate some simple applications, including a fast turbo decoding algorithm. This algorithm can outperform Log-MAP-based turbo decoding as it is faster without degrading the BER.

The rest of this paper is divided into three parts. Part 1 (Sections II–IV) studies ergodicity. Part 2 (Sections V–VIII) is devoted to Gaussian approximations. Part 3 (Sections IX–XIII) investigates turbo dynamics and discusses applications. Final conclusions are reached in Section XIV. For easier reading, all the proofs are contained in the Appendix.

Part 1: Ergodicity

The next three sections are concerned with the ergodicity of the input–output signals in a turbo decoder. More specifically, Section II introduces some basic concepts and results for random processes. Section III studies stochastic modeling for Log-MAP decoding. Section IV studies stochastic modeling for turbo decoding.

II. Random Processes

In this section, we introduce some necessary background information on random processes. The material is extracted from [15] but simplified to avoid "heavy" mathematics. Throughout this paper, we consider random variables and random processes defined in the probability space $(\mathbb{R}, \boldsymbol{B}, P)$, where $\mathbb{R} = (-\infty, \infty)$, $\boldsymbol{B}$ is the Borel σ-algebra, and P is the probability measure. For multivariate random processes, the associated probability space becomes $(\mathbb{R}^m, \boldsymbol{B}, P)$.

Definition 1: A (discrete-time) random process $x = \{x_t : t = 0, \pm 1, \pm 2, \ldots\}$ is said to be *independent* if for any n-tuple $(t_1, t_2, \ldots, t_n)$

$$P(x_{t_1}, x_{t_2}, \ldots, x_{t_n}) = \prod_{i=1}^{n} P(x_{t_i}).$$

Two random processes x and y are said to be *independent* if for any n-tuple $(t_1, t_2, \ldots, t_n)$ and m-tuple $(k_1, k_2, \ldots, k_m)$

$$\begin{aligned} &P(x_{t_1}, x_{t_2}, \ldots, x_{t_n}, y_{k_1}, y_{k_2}, \ldots, y_{k_m}) \\ &\qquad = P(x_{t_1}, x_{t_2}, \ldots, x_{t_n}) P(y_{k_1}, y_{k_2}, \ldots, y_{k_m}). \end{aligned}$$

Definition 2: A random process x is said to be *stationary* (in the strict sense) if for any n-tuple $(t_1, t_2, \ldots, t_n)$, $P(x_{\tau+t_1}, x_{\tau+t_2}, \ldots, x_{\tau+t_n})$ is independent of τ.

Note that a stationary random process has the same distribution (and thus the same probability measure) for all t.

Definition 3: A random process x is said to be *ergodic* if it is stationary and for any measurable set $\Delta \subset \mathbb{R}$ and any infinite sequence $(\ldots t_{-1} < t_0 < t_1, \ldots)$

$$\frac{1}{2N} \sum_{i=-N}^{N} I_\Delta(x_{t_i}) \to P(\Delta), \qquad \text{as } N \to \infty$$

with probability 1, where $I_\Delta(\xi)$ is the so-called *indicator function* which equals 1 when $\xi \in \Delta$ or zero otherwise. Similarly, a set of random processes $x^{(1)}, x^{(2)}, \ldots, x^{(n)}$ (including $n = \infty$) are called *jointly ergodic* if they are stationary and for any measurable set $\Delta \in \mathbb{R}^n$ and any infinite sequence $(\ldots t_{-1} < t_0 < t_1, \ldots)$

$$\frac{1}{2N} \sum_{i=-N}^{N} I_\Delta([x_{t_i}^{(1)}, x_{t_i}^{(2)}, \ldots, x_{t_i}^{(n)}]) \to P(\Delta), \quad \text{as } N \to \infty$$

with probability 1.

Lemma 1: If a set of random processes $x^{(1)}, x^{(2)}, \ldots, x^{(n)}$ (including $n = \infty$) are ergodic (individually) and independent of each other, then they are jointly ergodic.

Lemma 2: Suppose a set of random processes $x^{(1)}$, $x^{(2)}, \ldots, x^{(n)}$ (including $n = \infty$) are jointly ergodic. Then, given any measurable function $f : \mathbb{R}^n \to \mathbb{R}$, the random process y defined by

$$y_t = f(x_t^{(1)}, x_t^{(2)}, \ldots, x_t^{(n)})$$

is ergodic.

Corollary 1: Suppose x is an ergodic random process with probability measure P_x and $f : \mathbb{R} \to \mathbb{R}$ is a measurable "statistical function." Then, for any infinite sequence $(\ldots, t_{-1}, t_0, t_1, \ldots)$, we have

$$\frac{1}{2N} \sum_{i=-N}^{N} f(x_{t_i}) \to \mathcal{E}(f), \quad N \to \infty$$

where $\mathcal{E}(f)$, the mean of f, is the corresponding "statistics."

Note that usual statistics include mean, variance, correlation, and higher order moments. Also note that Gaussian white noises are ergodic. Hence, any random processes generated by a stationary measurable function of Gaussian white noises are ergodic, and their usual statistics all enjoy the ergodic properties as stated in Corollary 1.

III. Stochastic Modeling of MAP Decoding

For noniterative decoding, the decoder is simply a mapping from a received signal to a decoded signal. When the received signal is subject to AWGN, the performance of the decoder, as measured by an averaged decoding BER, is fully determined by the SNR of the received signal. For this reason, a single curve of SNR versus BER is typically used to characterize the performance of the decoder.

For iterative decoding, each constituent decoder also takes an *a priori* signal as an input. The output of the decoder becomes the so-called extrinsic information which, after an appropriate transformation such as interleaving, becomes the *a priori* signal for the next constituent decoder. The behavior of the (constituent) decoder turns out to be much more complicated. It is generally insufficient to characterize the performance of an constituent decoder by the SNRs of the input signals.

In this section, we analyze the asymptotic behavior of Log-MAP decoding with a very large code block size. By modeling the input and output signals in a Log-MAP decoder as random processes, we show that the asymptotic behavior of Log-MAP decoding can be described by a simple stochastic model.

A. Scaled LLRs

To help understand the distributions of LLR signals, we introduce the notion of *scaled log-likelihood ratio* (SLLR). Given any signal s which is a noisy version of a binary signal x with elements $x_i \in \{-1, 1\}$, recall that its LLR, denoted by L_s, is defined as

$$L_{s,i} = \log \frac{P(s_i|x_i = 1)}{P(s_i|x_i = -1)}.$$

Its SLLR, denoted by $\boldsymbol{S}$, is defined as

$$\boldsymbol{S}_i = \frac{x_i}{2} L_{s,i}.$$

The SLLR indicates how well s is "correlated" with x. Note that for the analysis of a linear code, one may assume that the transmitted signal is an all-one sequence, which corresponds to binary zeros. In this case, SLLR = LLR/2.

When a (received) signal r is subject to AWGN, its SLLR has a Gaussian distribution. Indeed, if $r = x + n$, where x is a transmitted binary signal and n is a zero-mean Gaussian white noise with variance σ_0^2, then the LLR of r is given by

$$\begin{aligned} L_r &= \ln \left\{ \frac{\text{Prob}(r|x = 1)}{\text{Prob}(r|x = -1)} \right\} \\ &= \ln \left\{ \frac{\exp(-(r-1)^2/(2\sigma_0^2))}{\exp(-(r+1)^2/(2\sigma_0^2))} \right\} = \frac{2(x+n)}{\sigma_0^2}. \end{aligned}$$

Therefore, the SLLR of r is a Gaussian distribution given by

$$\boldsymbol{R} = (x^2 + xn)\sigma_0^{-2} = (1 + xn)\sigma_0^{-2}$$

with the following mean and variance:

$$\mu_r = \sigma_0^{-2} \,(= \text{SNR})\,; \quad \sigma_r^2 = \sigma_0^{-2}.$$

We see from above the SLLR of r meets the unique condition

$$\mu_r = \sigma_r^2. \tag{1}$$

This condition is known ([9], [13]) but we have scaled the LLR purely for convenience.

However, as we will see later, that the preceding relationship no longer holds for extrinsic signals (and the *a priori* signals in later iterations). Because of this, we introduce the notion of *mean-to-variance ratio* (MVR)

$$d = \frac{\mu}{\sigma^2}.$$

In particular, the MVR for r is $d_r = 1$.

B. Log-MAP Decoding

The well-known Log-MAP decoding algorithm [1], [16] takes an *a priori* signal a (representing the *a priori* probability of the information signal u) and a received signal r, and produces an extrinsic signal e and an *a posteriori* signal d (representing the extrinsic and *a posteriori* probabilities of the information signal). Their LLR expressions are denoted by L_a, L_r, L_e, and L_d, respectively. Their SLLR expressions are denoted by $\boldsymbol{A}$, $\boldsymbol{R}$, $\boldsymbol{E}$, and $\boldsymbol{D}$, respectively, with means and variances denoted by μ_a and σ_a^2, etc.

When applied to a linear, binary block code, Log-MAP decoding can be interpreted as follows. Let

$$u = \{u_i : i = 1, 2, \ldots, n, u_i \in \{-1, 1\}\}$$

and

$$x = \{x_{ij} : i = 1, 2, \ldots, n; j = 1, 2, \ldots, m, x_{ij} \in \{-1, 1\}\}$$

represent the information and code signals, respectively. The coding rate is $1/m$ in this case. Let $r = x + w$ be a received signal, where w is a zero-mean AWGN with variance σ_0^2. Let $p_{a,i}(\pm 1)$ be the *a priori* probability for $u_i = \pm 1$, and let $p_{r,i,j}(\pm 1)$ be the conditional probability for r_{ij} when $x_{ij} = \pm 1$. Then, by Bayes' rule, the *a posteriori* probability for the information sequence to be equal to a given candidate information sequence v is given by

$$p(v) = \prod_{i=1}^{n} p_{a,i}(v_i) \prod_{j=1}^{m} p_{r,i,j}(y_{ij})$$

where y is the code signal for v. The Log-MAP decoder computes the following LLR:

$$L_{d,k} = \ln \frac{\sum_{v:v_k=1} p(v)}{\sum_{v:v_k=-1} p(v)}, \qquad k = 1, 2, \ldots, n.$$

Dividing both the numerator and denominator above by

$$p(u) = \prod_{i=1}^{n} p_{a,i}(u_i) \prod_{j=1}^{m} p_{r,i,j}(x_{ij})$$

and using the following LLR expressions:

$$L_{a,i} = \ln \frac{p_{u,i}(1)}{p_{u,i}(-1)}; \qquad L_{r,i,j} = \ln \frac{p_{r,i,j}(1)}{p_{r,i,j}(-1)}$$

we can rewrite L_d as

$$L_{d,k} = L_{d,k}^{+} - L_{d,k}^{-}$$

where

$$L_{d,k}^{\pm} = \ln \sum_{v:v_k=\pm 1} \exp \left(\sum_{i:v_i \neq u_i} v_i L_{a,i} + \sum_{\substack{i,j: \\ y_{i,j} \neq x_{i,j}}} y_{i,j} L_{r,i,j} \right).$$

Using the SLLR expressions, the above becomes

$$\boldsymbol{D} = \boldsymbol{D}^{+} - \boldsymbol{D}^{-}$$

where

$$\boldsymbol{D}_k^{\pm} = u_k \ln \sum_{v:v_k=\pm 1} \exp\left(-\sum_{i:v_i\neq u_i} \boldsymbol{A}_i - \sum_{\substack{i,j:\\ y_{i,j}\neq x_{i,j}}} \boldsymbol{R}_{i,j} \right).$$

To simplify our analysis, we assume, without loss of generality, that the information signal $u = \{\ldots, 1, 1, 1, \ldots\}$. It follows that

$$\boldsymbol{D}_k^{+} = \ln \left\{ 1 + \sum_{\substack{v:v\neq u,\\ v_k=1}} \exp\left(-\sum_{i:v_i\neq u_i} \boldsymbol{A}_i - \sum_{\substack{i,j:\\ y_{i,j}\neq x_{i,j}}} \boldsymbol{R}_{i,j} \right) \right\}$$

$$\boldsymbol{D}_k^{-} = \ln \sum_{v:v_k=-1} \exp\left(-\sum_{i:v_i\neq u_i} \boldsymbol{A}_i - \sum_{\substack{i,j:\\ y_{i,j}\neq x_{i,j}}} \boldsymbol{R}_{i,j} \right).$$

Note that each exponential function in $\boldsymbol{D}_k^{\pm}$ involves two sums. The number of terms in the first (respectively, second) sum equals the weight difference between the u and v (respectively, x and y).

Typically, the following extrinsic signal needs to be computed:

$$L_{e,k} = L_{d,k} - L_{a,k} - \sum_{j\in J_k} L_{r,k,j}$$

where J_k is the set of code bits shared by other constituent encoders. In the case of a parallel concatenated turbo code, J_k corresponds to the systematic bit. Using the SLLR expressions, we have

$$\boldsymbol{E}_k = \ln \left\{ 1 + \sum_{\substack{v:v\neq u,\\ v_k=1}} \exp\left(-\sum_{i:v_i\neq u_i} \boldsymbol{A}_i - \sum_{\substack{i,j:\\ y_{i,j}\neq x_{i,j}}} \boldsymbol{R}_{i,j} \right) \right\} - \ln \sum_{v:v_k=-1} \exp\left(-\sum_{\substack{i:i\neq k,\\ v_i\neq u_i}} \boldsymbol{A}_i - \sum_{\substack{i,j:\\ j\notin J_k \text{ if } i=k,\\ y_{i,j}\neq x_{i,j}}} \boldsymbol{R}_{i,j} \right) \quad (2)$$

or simply

$$\boldsymbol{E} = \ln \left(1 + \sum_{t=1}^{T_1} \exp(-C_t^{(1)}\boldsymbol{A} - C_t^{(2)}\boldsymbol{R}) \right) - \ln \left(\sum_{t=1}^{T_2} \exp(-C_t^{(3)}\boldsymbol{A} - C_t^{(4)}\boldsymbol{R}) \right) \quad (3)$$

where $C_t^{(i)}$ are row vectors consisting of zeros and ones and $T_1, T_2 \geq 0$.

Remark 1: Although in (2) the whole sequences of $\boldsymbol{A}$ and $\boldsymbol{R}$ are used to compute each $\boldsymbol{E}_k$, it is well known that the contributions of the terms $\boldsymbol{A}_{k+\tau}$ and $\boldsymbol{R}_{k+\tau,j}$ diminish as $|\tau|$ becomes large; see, e.g., [13], [14]. More precisely, if we denote by $\tilde{\boldsymbol{A}}$ and $\tilde{\boldsymbol{R}}$ the approximation of $\boldsymbol{A}$ and $\boldsymbol{R}$ without the terms $\boldsymbol{A}_{k+\tau}$ and $\boldsymbol{R}_{k+\tau,j}$ for $|\tau| > N$ for some given N, then the approximation error becomes negligible when N becomes sufficiently large. □

C. Asymptotic Behavior of Log-MAP Decoding

From the preceding analysis, it is clear that the output signal of MAP decoding can be modeled as a random process. The key question we now ask is how to model this random process. We are particularly interested in the behavior of the output signal when the code block size is very large. Our main result is given as follows.

Theorem 1: Given a convolutional code with an infinite block size, suppose the SLLR of the received signal $\boldsymbol{R}$ and the SLLR of the *a priori* signal $\boldsymbol{A}$ are ergodic random processes, and $\boldsymbol{R}$ and $\boldsymbol{A}$ are both independent by themselves and independent of each other. Then, the outputs of the Log-MAP decoder (i.e., $\boldsymbol{D}$ and $\boldsymbol{E}$) are both ergodic random processes.

The implications of Theorem 1 are important. When the received signal is subject to AWGN (which is ergodic) and $\boldsymbol{A}$ is ergodic, the result above says that the statistics of $\boldsymbol{E}_k$ are independent of k and can be computed using a *single* realization of $\boldsymbol{E}$, i.e., solving only a single (but long) Log-MAP decoding.

IV. Stochastic Modeling of Turbo Decoding

We now want to generalize the ergodicity result in Theorem 1 to turbo decoding. Again, we are interested in the case when the block size approaches infinity.

So far, from the analysis of Log-MAP decoding, we understand that if the received signal is subject to AWGN and the SLLR of the *a priori* signal, $\boldsymbol{A}$, is an independent ergodic random process, then the SLLR of the extrinsic signal $\boldsymbol{E}$ is also ergodic. In turbo decoding, we start with $\boldsymbol{A} = 0$, which is a Gaussian white noise with zero mean and zero variance. Therefore, it is natural to conjecture that the SLLR of the extrinsic signal in every iteration is an ergodic random process. It turns out that this is generally incorrect. The reason is that the extrinsic signal is "locally" correlated. More specifically, $\boldsymbol{E}_k$ and $\boldsymbol{E}_{k+t}$ can be strongly correlated if $|t|$ is small. It follows that the stochastic properties of $\boldsymbol{A}$ for the next iteration are largely influenced by the interleaver. It is easy to imagine that a nonstationary $\boldsymbol{A}$ is possible if a "bad" interleaver is used.

Fortunately, the correlation in $\boldsymbol{E}$ decays. Therefore, if the interleaver has a "good" spreading property, the interleaved extrinsic signal, which becomes the *a priori* signal, should be no longer correlated "locally." Since $\boldsymbol{E}_k$ depends only on those $\boldsymbol{A}_i$ which are "local" to k, the interleaved extrinsic signal is effectively an uncorrelated signal.

To understand how well an interleaver works, we introduce the notion of a *spreading factor*. Given an interleaver T of size n, its spreading factor S_T is given by

$$S_T = \min_S \{S : 1 \leq i,j \leq n; |i-j| \leq S \Rightarrow |T(i) - T(j)| > S\}.$$

The following result reveals the spreading property of random interleavers.

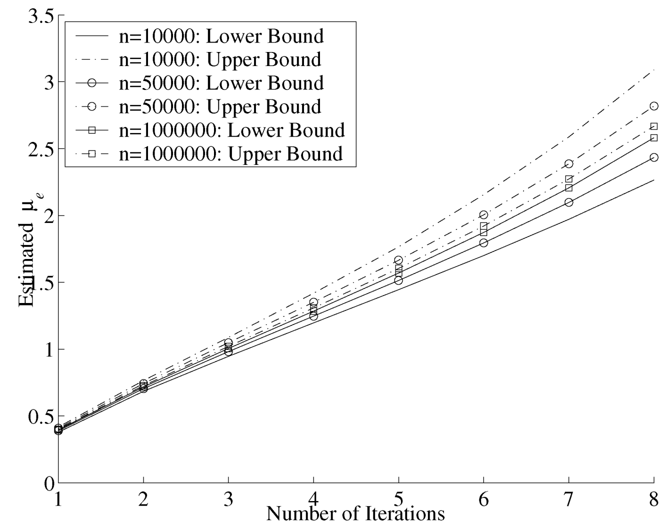

Fig. 1. Convergence of the means of $\boldsymbol{E}(\ell, n)$.

Lemma 3: Given any $S > 0$, if an interleaver T of size n is chosen randomly, then

$$P(S_T \geq S) \rightarrow 1, \qquad \text{as } n \rightarrow \infty. \tag{4}$$

Lemma 3 leads us to the main result for Part 1.

Theorem 2: Given a turbo code with block size n, suppose a random interleaver T is used and the received signal is subject to AWGN. Denote by $\boldsymbol{E}(\ell, n)$ the SLLR of the extrinsic signal from the ℓth iteration of Log-MAP decoding. Then, for any $\ell \geq 1$, $\boldsymbol{E}(\ell, n)$ approaches an ergodic random process as $n \rightarrow \infty$.

In Theorem 2 and the rest of the paper, the number of iterations refers to the number of Log-MAP decoding processes. This differs from what is commonly understood in the literature where the number of iterations refers to the number of turbo cycles.

To demonstrate the ergodicity of $\mu_e(\ell, n)$, we simulate a 1/3-rate turbo code with $G(D) = (1,\ 5/7)$, $E_b/N_0 = 0.5$ dB, and pseudorandom interleaver. For each n and ℓ, many runs of $\mu_e(\ell, n)$ are simulated. These values are used to compute a lower bound and an upper bound for $\mu_e(\ell, n)$. The lower bound is the average of these $\mu_e(\ell, n)$ values minus their standard deviation, whereas the upper bound is the average of these $\mu_e(\ell, n)$ values plus their standard deviation. The size of the gap between the lower and upper bounds shows how well $\mu_e(\ell, n)$ converges as $n \rightarrow \infty$.

The simulation results are shown in Fig. 1. Three observations are made. First, the gap between the lower and upper bound curves becomes smaller as n becomes larger. Second, the gaps are nested in the sense that the curves for a larger n are within those for a smaller n. Finally, the gap becomes larger as ℓ becomes larger. The first two observations confirm the convergence of $\mu_e(\ell, n)$. The third observation means that a larger n is required for a larger ℓ when the same accuracy of estimation is required.

Although not shown here, the convergence properties of $\sigma_e(\ell, n)$ are also simulated and they are checked to be similar to those of $\mu_e(\ell, n)$.

Part 2: Gaussian Approximations

In the following four sections, we study Gaussian approximations for Log-MAP decoding and turbo decoding. Section V introduces some known results for sums and log-sums of lognormal distributions. Section VI demonstrates Gaussian approximations for Log-MAP decoding in various cases. Section VII analyzes Gaussian approximations for Log-MAP decoding. Section VIII extends Gaussian approximations to turbo decoding.

V. Log-Sum of Lognormal Distributions

Given a set of Gaussian-distributed random variables X_i with means μ_i and variances σ_i^2, $i = 1, 2, \ldots, n$, we define

$$Z = \ln \sum_{i=1}^{n} \exp(X_i).$$

Then, each $\exp(X_i)$ is a lognormal distribution and

$$\exp(Z) = \sum_{i=1}^{n} \exp(X_i)$$

is a sum of lognormal distributions (SLND). We will call Z a log-sum of lognormal distributions (LSLND).

The motivation for studying LSLND will be given in the next section where we will see that a Log-MAP decoder can be viewed as a function with inputs being Gaussian distributions and outputs being LSLND.

The statistical properties of SLND have been well studied. It is well known that the distribution of a SLND can be closely approximated using a lognormal distribution when X_i are independent with the same mean and variance. In this case, Z is well approximated by a Gaussian distribution. Although no closed-form description is given on the distribution of an SLND or an LSLND, a number of methods are available for computing the mean and variance (or equivalently, the first and second moments) of Z. All the available methods rely on the assumption that the sum of two independent lognormal distributions is still a lognormal distribution. It turns out that when X_i are correlated and/or the statistics of X_i are different, Z is also well approximated using a Gaussian distribution, provided that the correlation is not very strong and the statistical differences are not significant. We discuss four popular methods below.

The first method is the so-called *Fenton–Wilkinson* (FW) method given by [17]. This is based on the simple fact that

$$\mathcal{E}[\exp(X_i)] = \exp\left(\frac{2\mu_i + \sigma_i^2}{2}\right).$$

It follows that the first and second moments of $\exp(Z)$ can be computed as follows:

$$m_1 = \mathcal{E}[\exp(Z)] = \sum_{i=1}^{n} \exp\left(\frac{2\mu_i + \sigma_i^2}{2}\right)$$

and

$$\begin{aligned} m_2 = \mathcal{E}[(\exp(Z))^2] &= \sum_{i=1}^{n} \exp(2\mu_i + 2\sigma_i^2) \\ &+ 2\sum_{i=1}^{n} \sum_{j=i+1}^{n} \exp(\mu_i + \mu_j + (\sigma_i^2 + \sigma_j^2 + 2r_{ij}\sigma_i\sigma_j)/2) \end{aligned}$$

where r_{ij} is the correlation coefficient of X_i and X_j defined as

$$r_{ij} = \frac{\mathcal{E}[(X_i - \mu_i)(X_j - \mu_j)]}{\sigma_i \sigma_j}.$$

The mean and variance for Z is approximated using

$$\mu_z = 2\ln m_1 - \frac{1}{2}\ln m_2;\ \sigma_z^2 = \ln m_2 - 2\ln m_1.$$

The FW method works well only when X_i are weakly correlated with similar means and variances and that the variances are small.

The second method is the Ho–Schwartz–Yeh method, originally given in [18] and later modified in [19], gives a more accurate approximation for the mean and variance of Z at the expense of more computations. The basic idea here is to approximate the sum of two log-normal distributions at a time using more accurate approximations for the mean and variance and do it recursively until all the terms are exhausted. The algorithm is quite involved; see [19] for details.

The third method, called cumulant matching approximation, is a modification of the FW method with correction terms called the Gram–Charlier series [20]. Since this method works poorly when X_i are correlated (see [19]), we will not discuss it further.

The fourth method is the simple Monte Carlo simulation method. Despite its simplicity, Monte Carlo simulations seem to give the most reliable estimates for the mean and variance; see [22]. The main disadvantage of the Monte Carlo method seems to be the lack of insight into the relationships between the parameters of X_i and those of Z.

Various approximation methods have been extensively studied; see details in [19]–[25].

To demonstrate the Gaussian approximation, we consider the simple case where only two lognormal distributions are added. The purpose of this example is twofold: First, we want to examine how well Gaussian approximations work. Second, we want to get some idea about the relationships between the distributions of X_i and the distribution of Z.

Fig. 2 shows the normalized CDF of

$$Z = -\ln(\exp(-X) + \exp(-Y)) \tag{5}$$

for different distributions of X and Y. The normalization is done by removing the mean and scaling the variance to 1, so that we can easily see the accuracy of Gaussian approximations. The MVRs for X and Y are chosen to be equal to 1. It is shown in the figure that Gaussian approximations are generally very good even when X and Y have quite different distributions and/or quite strong correlations. The two curves which do show some noticeable approximation errors are the curve 1 ($\mathcal{E}[X] = 0$, $\mathcal{E}[Y] = -0.25$, $r_{xy} = 0$) and curve 3 ($\mathcal{E}[X] = 2$, $\mathcal{E}[Y] = 3$, $r_{xy} = -0.5$). The former is caused by the significantly different mean values (in a relative sense), whereas the latter is caused by the strong correlation.

Fig. 3 shows how the MVR changes for Z in (5). In the figure, X and Y are independent with the same mean and MVR equal to 1. We see that the MVR for Z varies. When X and Y have small means, d_z is slightly less than 1. When their means become large, $d_z > 1$.

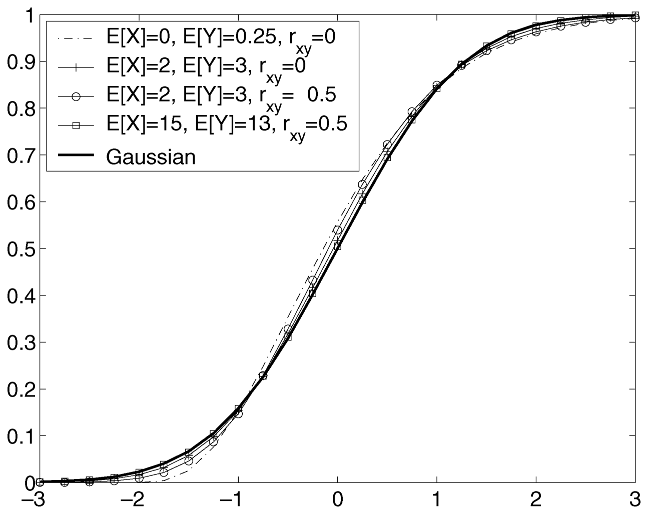

Fig. 2. Normalized CDF of $Z = -\ln(\exp(-X) + \exp(-Y))$.

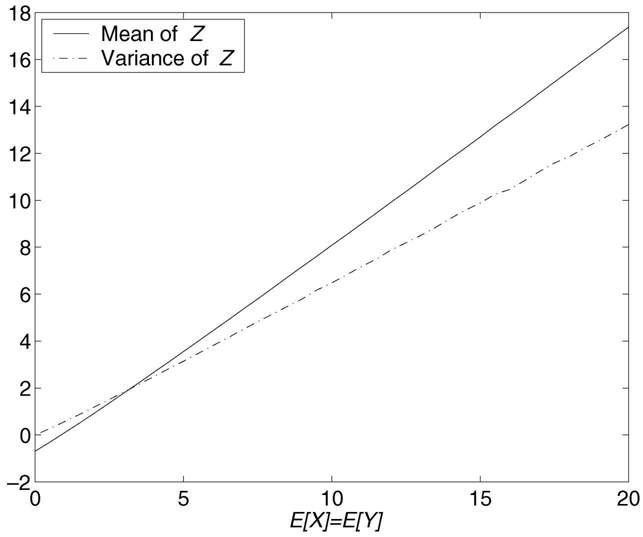

Fig. 3. Mean and variance of $Z = -\ln(\exp(-X) + \exp(-Y))$.

In both Figs. 2 and 3, 10 000 samples were used to generate each distribution.

VI. Gaussian Approximations for Log-MAP Decoding

We now want to analyze the distribution of $\boldsymbol{E}$ (the SLLR of the extrinsic signal) for Log-MAP decoding. Our purpose is twofold. First, we want to show that Gaussian approximations work well. Second, we want to determine the key input and output parameters in a Log-MAP decoder and understand their relationships.

Consider the expression $\boldsymbol{E}_k$ from (2). Recall that $\boldsymbol{R}$ is a vector of (independent) Gaussian distributions when the received signal r is subject to AWGN. Suppose $\boldsymbol{A}$ is also Gaussian distributed. Then, the terms $C_t^{(1)}\boldsymbol{A} + C_t^{(2)}\boldsymbol{R}$ and $C_t^{(3)}\boldsymbol{A} + C_t^{(4)}\boldsymbol{R}$ are also Gaussian distributed. It follows that $\exp(-C_t^{(1)}\boldsymbol{A} - C_t^{(2)}\boldsymbol{R})$ and $\exp(-C_t^{(1)}\boldsymbol{A} - C_t^{(2)}\boldsymbol{R})$ are log-normally distributed and $\boldsymbol{E}$ is the difference between the two LSLNDs. This observation is summarized as follows.

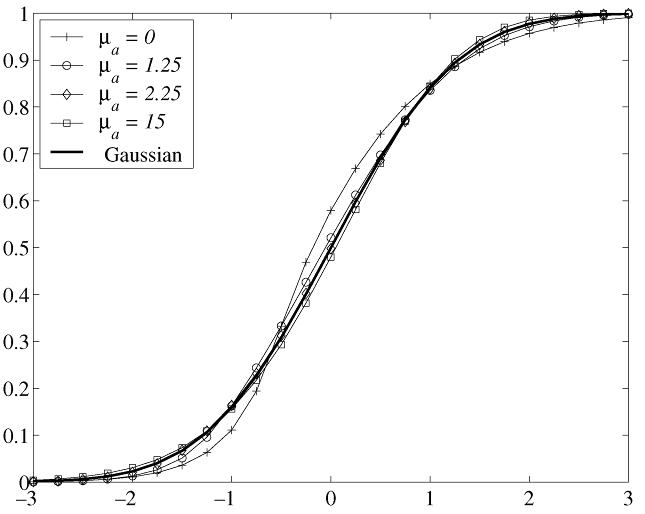

Fig. 4. Normalized CDF of extrinsic signal (SLLR).

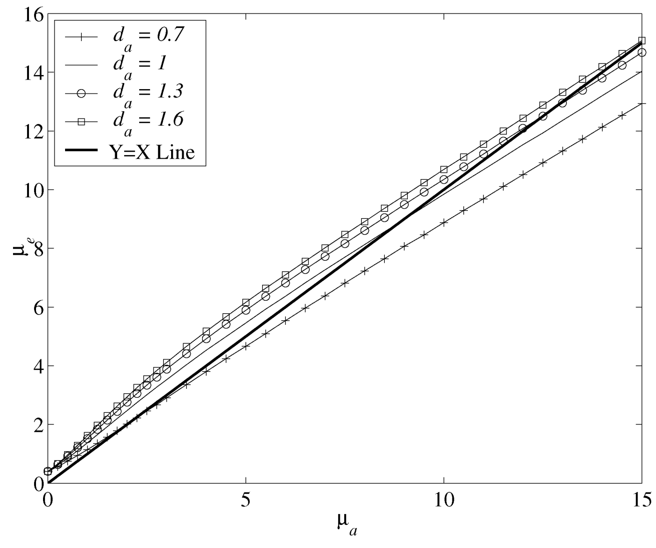

Fig. 5. Mean of extrinsic signal (SLLR).

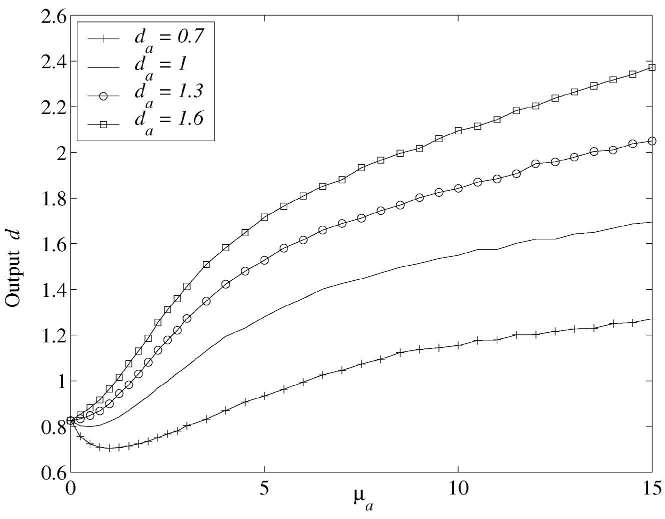

Fig. 6. MVR of extrinsic signal.

For a convolutional binary code with an infinite block size, if the received signal is subject to AWGN and the SLLR of the *a priori* signal is a Gaussian distribution, then the SLLR of the extrinsic signal can be well approximated using a Gaussian distribution.

Although the preceding result says that the SLLR of the extrinsic signal can be approximated using a Gaussian distribution, its MVR is no longer equal to 1 in general. This occurs even when the *a priori* signal has zero information (i.e., $\boldsymbol{A} = 0$). Therefore, it is insufficient to characterize the output signal by its SNR. Instead, two parameters, the mean and variance of the SLLR need to be used. Since the output signal is fed back to the next constituent decoder, the *a priori* signal to a constituent decoder also needs to be characterized by the mean and variance of its SLLR.

In view of the analysis above, we conclude the following.

A Log-MAP decoder can be approximated as a mapping $\mathcal{M}$ from (μ_r, μ_a, σ_a) to (μ_e, σ_e). If μ_r is suppressed, the decoder is simply a mapping from (μ_a, σ_a) to (μ_e, σ_e).

To illustrate the behavior of Log-MAP decoding, we consider the simple $1/2$-rate, 4-state, systematic convolutional code $G(D) = (1, 5/7)$ (in octal). The received signal is subject to AWGN with $E_b/N_0 = -1.2609$ dB (which corresponds to $E_b/N_0 = 0.5$ dB when the code is used as a constituent code of a $1/3$-rate turbo code). A block size of 500 000 is used in the simulation.

Fig. 4 shows the normalized CDFs of $\boldsymbol{E}$ for different values of μ_a but with $d_a = 1$. It is observed that for very low values of μ_a, $\boldsymbol{E}$ is only roughly approximated using Gaussian distributions. As μ_a increases, the approximation becomes very accurate. When μ_a is very high, the approximation becomes slightly off again.

Fig. 5 plots μ_e versus μ_a. It is observed that μ_e exceeds μ_a for low values of μ_a. However, for high values of μ_a, the converse is true. The crossover point is critical in determining the convergence of turbo decoding. A more important feature is that the crossover point is seriously affected by the MVR. The lower (or higher) the value of d_a, the lower (or higher) the crossover point is.

Fig. 6 plots d_e (the MVR of $\boldsymbol{E}$) versus μ_a. It is observed that when μ_a is low, $d_e < 1$. As μ_a increases, d_e may go down a bit before it quickly picks up. Also, d_a has a significant influence on the MVR of the extrinsic signal. The higher the former, the higher the latter.

We emphasize that the phenomena observed in Figs. 4–6 are common to all convolutional codes.

The influence of the MVR on the performance of the Log-MAP decoding complicates the characterization of the decoding behavior. A natural question to ask is whether it is possible to simplify this characterization. Since the MVR effectively alters the BER of the SLLR, one possible idea is to adjust the SLLR so that the MVR becomes 1 but the BER remains the same. More precisely, given an SLLR $\boldsymbol{L}$, which is Gaussian distribution with mean $\mu \geq 0$ and variance σ^2, we define the adjusted SLLR $\tilde{\boldsymbol{L}}$ such that it is a Gaussian distribution with mean $\tilde{\mu}$ and variance $\tilde{\sigma}^2$ related by $\tilde{\mu} = \tilde{\sigma}^2$ (thus, the MVR $= 1$) and that

$$\mathrm{Prob}\{\boldsymbol{L} < 0\} = \mathrm{Prob}\{\tilde{\boldsymbol{L}} < 0\}.$$

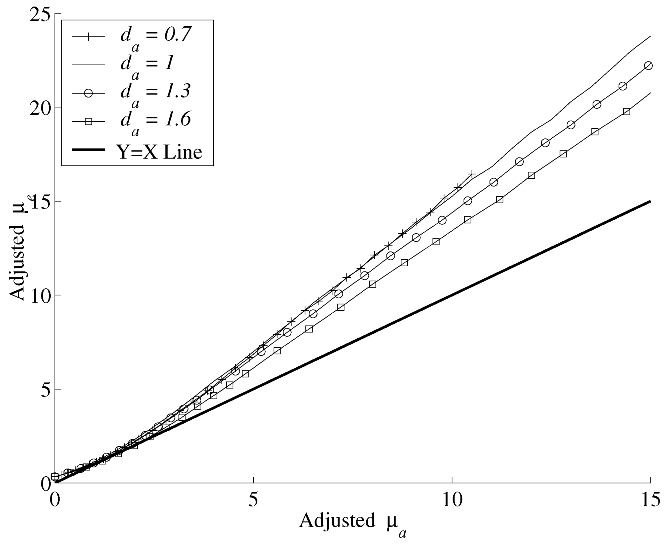

Fig. 7. Adjusted means of extrinsic signals (SLLR): global view.

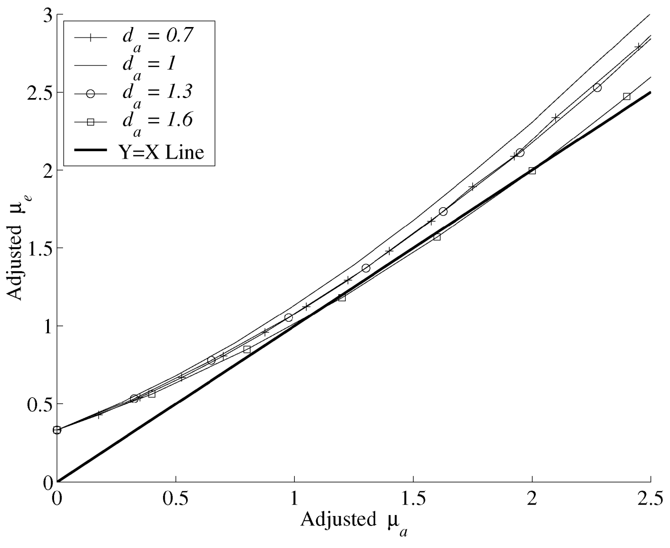

Fig. 8. Adjusted means of extrinsic signals (SLLR): local view.

The parameter $\tilde{\mu}$ (and thus $\tilde{\sigma}$) can be found out as follows: Since both $\boldsymbol{L}$ and $\tilde{\boldsymbol{L}}$ are Gaussian distributed, we have

$$\begin{aligned}\mathrm{Prob}\{\boldsymbol{L} < 0\} &= Q(\mu/\sigma)\\ \mathrm{Prob}\{\tilde{\boldsymbol{L}} < 0\} &= Q(\tilde{\mu}/\tilde{\sigma}).\end{aligned}$$

It follows that

$$\mu/\sigma = \tilde{\mu}/\tilde{\sigma}.$$

Using $\tilde{\mu} = \tilde{\sigma}^2$ and $d = \mu/\sigma^2$ (the MVR of $\boldsymbol{L}$), we obtain the following relationship:

$$\tilde{\mu} = d\mu.$$

Figs. 7 and 8 plot the adjusted μ_e versus μ_a. They are the same plots but with different input ranges. It can be seen that the curves for different MVRs have similar trends but they differ noticeably. This suggests it is possible to use a SISO model (i.e., input SNR versus output SNR) to describe the qualitative behavior of a Log-MAP decoder. But for more accurate analysis of Log-MAP decoding, it is necessary to use a model with two parameters (i.e., the mean and variance, or equivalently, the MVR of the SLLR) for both the input and output.

Remark 2: It is known [32] that turbo decoding is a particular instance of a more general decoding algorithm called *belief propagation*. In [33], it is proved that the LLRs (also called *messages*) in a belief propagation algorithm obey the so-called symmetry (or consistency) condition when the block size is very large. This condition states that the density function f of a message obeys

$$f(x) = e^x f(-x), \qquad \forall x \in (-\infty,\ \infty).$$

If f is a Gaussian distribution with mean m and variance σ^2, the above condition is the same as requiring $\sigma^2 = 2m$. When applied to a scaled LLR with a Gaussian distribution, the symmetry condition requires $\sigma^2 = m$ (or MVR $= 1$).

Throughout this paper, we will argue for the need of using two separate parameters (both mean and variance) to model the (scaled) LLR. This seems to contradict the symmetry condition. However, the answer lies in the fact that Gaussian distributions are only approximations. That is, the Gaussian approximation aims to give a good approximate model by scarifying the symmetry condition. We argue that the violation of the symmetry condition does not create a serious problem. To see this, we note that the distribution of a (scaled) LLR is mostly one sided and decays exponentially fast as $|x| \to \infty$. Thus, it is not important to enforce $f(x) = e^x f(-x)$ when either $f(x)$ or $f(-x)$ is very small. What is much more important is how to capture the portion of the density with a significant mass distribution using a simple model, which is achieved by Gaussian approximation. This claim is also supported by the analysis in Section VI where the use of adjusted SLLRs (which obeys the symmetry condition) fails to adequately model the turbo dynamics. $\square$

VII. Analysis of Log-MAP Decoding

We now give some explanations to the Log-MAP decoding behavior we observed earlier. More specifically, we try to answer the following questions:

- What causes the errors in Gaussian approximations?
- What causes the MVR to change?

A. Simplified Model for Log-MAP Decoding at High SNRs

To help understand the behavior of Log-MAP decoders, we derive a simplified model at high SNRs. Here, by a high SNR, we mean that μ_a is large and $d_a \geq 1$. The simplified model is given as follows.

Lemma 4: Suppose $\boldsymbol{A}$ is approximately Gaussian distributed with $d_a \geq 1$, the received signal r is subject to AWGN, and

$$\mu_a \gg \mu_r.$$

Also, suppose that the convolutional code is recursive, the code block is infinitely long, and the information sequence u for transmission is an all-one sequence. For each k, denote by $V_{k,2}$ the set of weight-2 information sequences v with $v_k = -1$

which are terminating. Then, the following approximation holds:

$$\boldsymbol{E}_k \approx -\ln \sum_{v: v\in V_{k,2}} \exp\left(-\boldsymbol{A}_t - \sum_{\substack{i,j:\\ j\notin J_k \text{ if } i=k,\\ y_{i,j}=-1}} \boldsymbol{R}_{i,j} \right) \tag{6}$$

where t is the index for the other information bit with $v_t = -1$ and J_k is as defined earlier.

We note that the number of weight-2 information sequences in $V_{k,2}$ can usually be significantly reduced. This is because those which generate a high weight code can be eliminated without affecting the computation. In practice, only a few weight-2 information sequences need to be included in $V_{k,2}$.

B. Gaussian Approximation Errors

To have a better view of the Gaussian approximation errors in Fig. 4, we show two magnified local plots of it in Figs. 9 and 10. Combined with Fig. 4, we can make the following observations (which are also noted in [10]).

- For low values of $\boldsymbol{A}$, relatively large approximation errors occur. The two tails are skewed toward lower probabilities and the center is skewed toward higher probabilities.
- For high values of $\boldsymbol{A}$, the opposite occurs, although the approximation errors are significantly smaller.
- For some mid values of $\boldsymbol{A}$, Gaussian approximations are most accurate.

It turns out that the observations above can be explained using the expression of $\boldsymbol{E}_k$. Recall from (3) that $\boldsymbol{E}_k$ is the difference between the two logarithmic terms. Therefore, the accuracy of the Gaussian approximation for $\boldsymbol{E}_k$ depends on the accuracy of the Gaussian approximations for the two logarithmic terms. Recall that a sum of lognormal distributions can be approximated using a lognormal distribution with the accuracy of approximation depending on two factors: the correlations of the individual lognormal distributions and variations in their means and variances. Therefore, to have very accurate approximations, the individual lognormal distributions are required to have independent and identical distributions.

When μ_a is low, each of the two logarithmic terms in (3) contains the sum of many exponential terms. Take the first logarithmic term for example. Although $C_t^{(1)}\boldsymbol{A} + C_t^{(2)}\boldsymbol{R}$ are correlated and have different means and variances for different values of t, the influence of the correlations and the variations in the mean and variance do not have significant contributions to the Gaussian approximation. There are two reasons for this. First, the sum $\sum_t \exp(-C_t^{(1)}\boldsymbol{A} - C_t^{(2)}\boldsymbol{R})$ is dominated by the terms of $C_t^{(1)}\boldsymbol{A} + C_t^{(2)}\boldsymbol{R}$ which have the smallest means and that for these terms, their means and variances are similar. Second, because a large number of terms are involved in the sum, the correlations between different terms are relatively weak. Therefore, both

$$Y_1 = \ln \sum_{t=1}^{T_1} \exp(-C_t^{(1)}\boldsymbol{A} - C_t^{(2)}\boldsymbol{R})$$

and

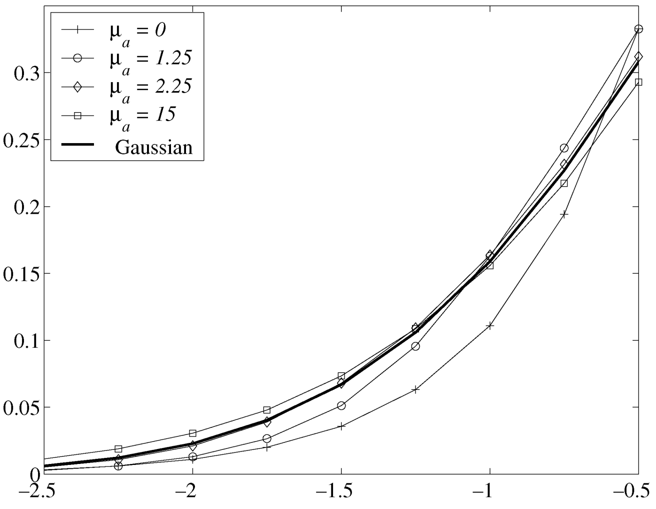

Fig. 9. Normalized CDFs of extrinsic signals (local view 1).

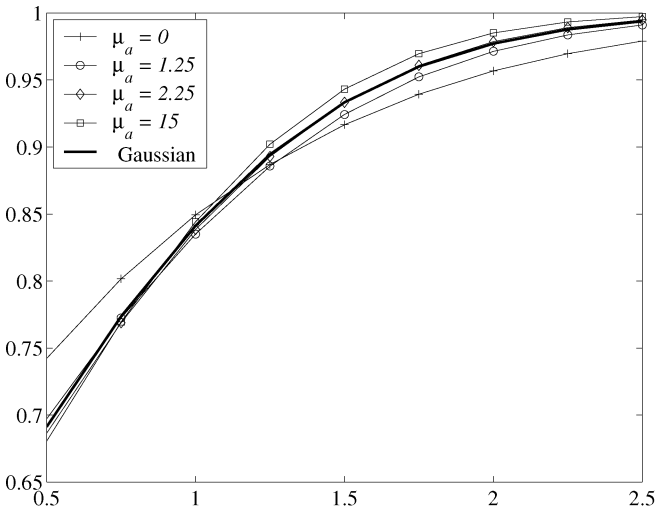

Fig. 10. Normalized CDFs of extrinsic signals (local view 2).

$$Y_2 = \ln \sum_{t=1}^{T_2} \exp(-C_t^{(3)}\boldsymbol{A} - C_t^{(4)}\boldsymbol{R})$$

are well approximated using Gaussian distributions and their correlations are again weak because of the large number of terms involved in the two sums.

The main source of approximation errors comes from $\boldsymbol{D}_k^+ = \ln(1+\exp(Y_1))$. In fact, Y_1 has a negative mean and its variance is not negligible. If we rewrite the term "1" as $\exp(Y_0)$, then $Y_0 = 0$ and it can be viewed as a Gaussian distribution with zero mean and zero variance. It follows that Y_0 and Y_1 have significantly different distributions, which implies that $\ln(1+\exp(Y_1))$ is only poorly approximated using a Gaussian distribution. If we approximate $\ln(1+\exp(Y_1))$ using first-order approximation, i.e.,

$$\ln(1+\exp(Y_1)) \approx \exp(Y_1)$$

then we see that $\ln(1+\exp(Y_1))$ resembles a lognormal distribution. Fortunately, $\ln(1+\exp(Y_1))$ is usually much smaller in magnitude compared to Y_2, its contribution to $\boldsymbol{E}_k$ is relatively small. Nevertheless, the lognormal resemblance of $\ln(1+\exp(Y_1))$ with a negative mean for Y_1 gives a small degree of skewing to the distribution of $\boldsymbol{E}_k$ that we observe in Figs. 4, 9, and 10.

When μ_a is high, the simplified model (6) becomes valid, i.e., the term $\ln(1+\exp(Y_1))$ disappears, leading to a much better Gaussian distribution for $\boldsymbol{E}$. The reason for a small degree of skewing is that the weight-2 sequences in (6) on the same side of k (either left or right) share some common bits (which imply correlations) and have different means and variances. The reason for the skewing in the direction opposite to that in the case of a low $\boldsymbol{A}$ is that the skewing now happens in the second logarithmic term.

Because the directions of skewing are different for low $\boldsymbol{A}$ and high $\boldsymbol{A}$, there exists a mid range for μ_a for which the skewing has roughly disappeared and the Gaussian approximation is most accurate.

C. MVR Analysis

Fig. 6 shows that the MVR of $\boldsymbol{E}$ is small (below 1) when μ_a is low and becomes large (above 1) when μ_a is high. This behavior is easily explained using the result in Fig. 3.

We now show that the simplified model (6) can be used to accurately predict the MVR of $\boldsymbol{E}$ at a high SNR. To this end, we first note that a terminating weight-2 sequence v for $G(D) = (1, 5/7)$ must be

$$v = [\ldots\ 1\,1\,-1\ \underbrace{1\,1\ \ldots\ 1}_{2+3m \text{ ones}}\,-1\,1\,1\ \ldots], \quad m = 0, 1, 2, \ldots.$$

It follows that

$$V_{k,2} = \{v(k, k+m) : m = \pm 1, \pm 2, \ldots\}$$

where $v(k, k+m)$ is an information sequence with

$$v_i = \begin{cases} -1, & i = k,\ k+m \\ 1, & \text{otherwise.} \end{cases}$$

In Figs. 11 and 12, $m = \pm 1, \pm 2, \ldots, \pm 5$ are used in $V_{k,2}$ to simulate the simplified model (6). It is clear that these plots closely match Figs. 5 and 6 in the high-SNR region.

VIII. Gaussian Approximations for Turbo Decoding

From the analysis of Log-MAP decoding, we understand that if $\boldsymbol{A}$ and $\boldsymbol{R}$ are independent Gaussian white noises, then $\boldsymbol{E}$ is well approximated using a stationary process with a Gaussian distribution. In turbo decoding, we start with $\boldsymbol{A} = 0$, which is a Gaussian white noise with zero mean and zero variance. Therefore, $\boldsymbol{E}$ from the first iteration is well approximated using a Gaussian distribution. We note that $\boldsymbol{E}$ is not independent. However, recalling the spreading property of random interleavers in Lemma 3, we understand that if a random interleaver is used, $\boldsymbol{A}$ for the next iteration will become effectively independent when the block size is large. Hence, Gaussian approximations can continue, i.e., $\boldsymbol{E}$ in every iteration is well approximated using a Gaussian distribution.

To formalize our analysis, we define the *Gaussian approximation model* for a turbo decoder as follows:

> For each decoding iteration, the SLLR of the *a priori* signal is well approximated using an uncorrelated Gaussian distribution and the SLLR of the extrinsic signal is well approximated using a (possibly locally correlated) Gaussian distribution.

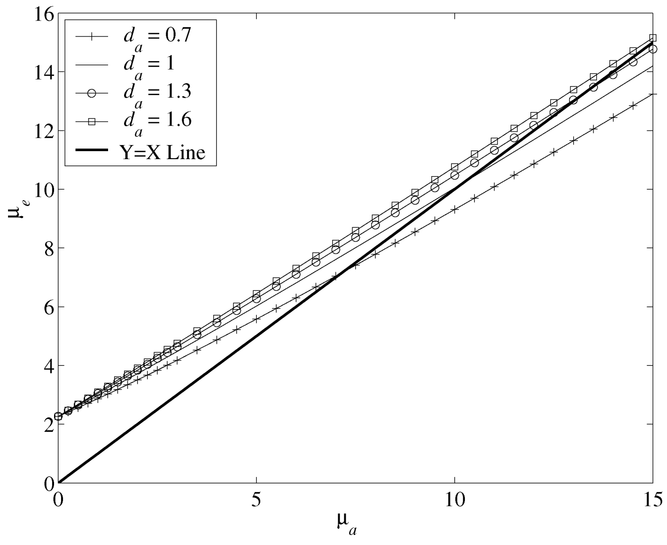

Fig. 11. Means of extrinsic signals (SLLR): simplified model.

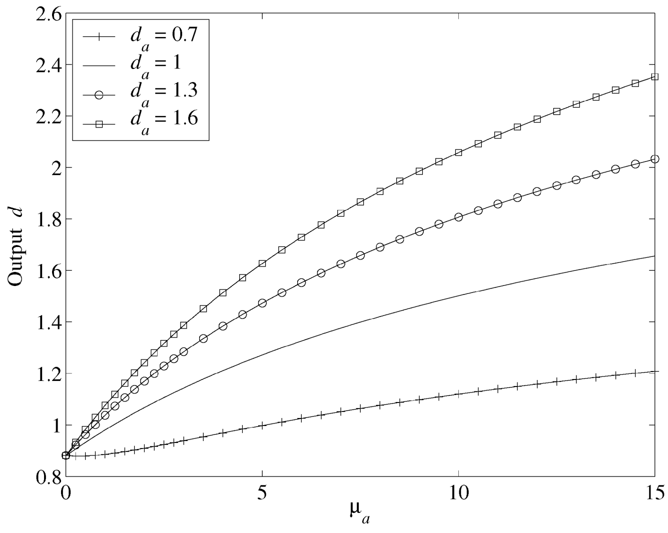

Fig. 12. MVRs of extrinsic signals: simplified model.

To check the validity of the Gaussian approximation model, we compare it to turbo decoding using an example. The turbo code in this example is a $1/3$-rate one with $G(D) = (1, 15/13)$, $n = 500\,000$, and a pseudorandom interleaver. We take $E_b/N_0 = 0.3$ dB which is about 0.8 dB away from the Shannon limit. Twelve iterations are used. For the method using the Gaussian approximation model, $\boldsymbol{A}$ in each iteration is chosen to be a Gaussian distribution with the same mean and variance as those for the $\boldsymbol{A}$ fed into the corresponding iteration of the turbo decoding. Fig. 13 compares the means and variances of $\boldsymbol{E}$ in the two cases. It is clear from the figure that the Gaussian approximation model gives a very good match to turbo decoding. However, there are some small but noticeable errors. These are caused by the fact that Gaussian approximations are slightly skewed when μ_a is either small or very large, as we pointed out earlier. More precisely, the Gaussian approximation model tends to slightly overestimate the μ_e and underestimate σ_a at a high SNR. At a low SNR, the opposite is true although the estimation errors are not as serious.

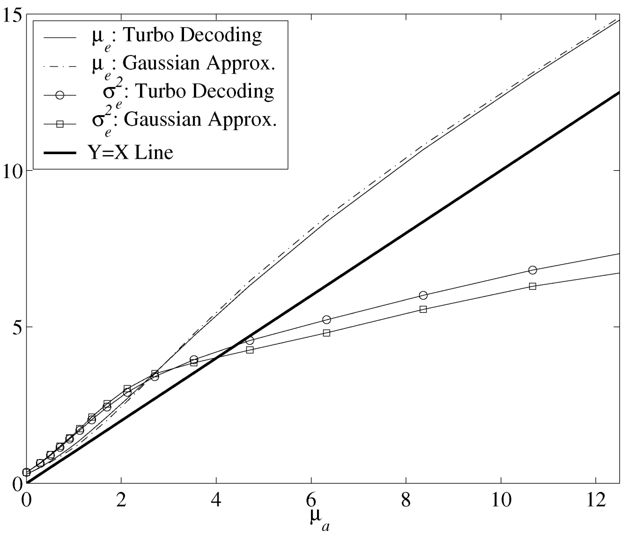

Fig. 13. Validity of Gaussian approximation.

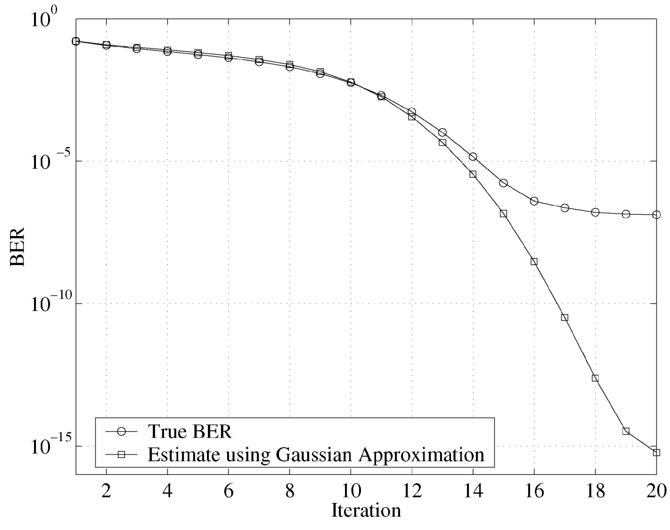

Fig. 14. BER estimate using Gaussian approximation.

If the turbo decoding is terminated after iteration ℓ, the BER can be estimated by

$$\text{BER} \approx Q(\mu_d/\sigma_d) = Q\left(\frac{\mu_e(\ell)+\mu_e(\ell-1)+\mu_r}{\sqrt{\sigma_e^2(\ell)+\sigma_e^2(\ell-1)+\sigma_r^2}}\right) \quad (7)$$

where μ_r and σ_r are the mean and variance of the systematic part of $\boldsymbol{R}$, respectively, $\mu_e(k)$ and $\sigma_e(k)$ denote, respectively, the μ_e and σ_e from iteration k, and $Q(\cdot)$ is the CDF for the Gaussian distribution with zero mean and unity variance.

Fig. 14 shows how well this BER estimate is. The turbo code used here is a $1/3$-rate one with $G(D) = (1,\ 15/13)$, pseudorandom interleaver, $n = 50\,000$, and $E_b/N_0 = 0.3$ dB. It is observed that the Gaussian approximation tends to overestimate the BER slightly when it is large ($> 3\times 10^{-3}$). This is consistent with Fig. 9. When the BER is small, the Gaussian approximation tends to underestimate it. This is also consistent with Fig. 9. We also note that the BER estimation error by the Gaussian approximation becomes very significant when the BER is very small ($< 10^{-6}$). A part of the reason for the Gaussian approximation to fail at very low BER is that there are simply not enough error bits in a block to "influence" the estimates of the mean and variance. Simulations suggest that the Gaussian approximation predicts the BER reasonably well provided that the BER is higher than the inverse of the block size. In our example, this BER value is 2×10^{-5}.

Part 3: Dynamics of Turbo Decoding

In this part, we analyze the dynamic behavior and convergence properties of turbo decoding (Sections IX–XII). We will also study some simple applications (Section XIII).

It is known that the outputs of turbo decoding typically "converge" at either some constant values or a quasi-periodic trajectory as the number of iterations increases. Occasionally, a seemingly convergent decoding trajectory may suddenly "diverge" and move into a different trajectory. A detailed account of these behaviors can be found in [5] and the references thereof.

These behaviors, however, are the consequence of having a finite code block size n. Recall that when $n \to \infty$, the decoding output for each iteration becomes an ergodic random process. Each decoding instance is a realization of the random process and the decoded signal has the same statistics (with probability 1). Therefore, we conclude that, as $n \to \infty$, the state of the decoded signal either converges at a finite stable fixed point or diverges. As we will see later, the former scenario occurs only at a low-SNR value, leading to a large BER. In contrast, the latter scenario leads to an ever-increasing SNR and thus arbitrarily low BER. However, when the block size is finite, this trend cannot be sustained indefinitely. As the number of iterations increases, the spreading property of the (de)interleaver becomes less effective, causing the decoding process to converge at a high (but finite) SNR point. Thus, we have two stable fixed points, one with a low SNR and one with a high SNR. This conclusion is consistent with [34] which is obtained by studying low-density parity-check (LDPC) codes and the belief propagation algorithm.

One key difference between these two types of fixed points is that the low–SNR ones cannot be improved by increasing the block size, whereas the high-SNR ones have their SNR values improved indefinitely as the block size increases. Another key difference is that the low-SNR fixed points can be avoided when the received signal has sufficient E_b/N_0. The required threshold of E_b/N_0 depends on the constituent encoder but is typically a fraction of a decibel away from the Shannon limit when the block size is large. It is these low-SNR fixed points which prevent the corresponding turbo codes to reach the Shannon limit when the block size approaches infinity. In contrast, the high-SNR fixed points cannot be avoided, implying that a small BER is never avoidable, although it can be made arbitrarily small by increasing the block size.

Turbo dynamics is in fact much more complex than indicated by the two stable fixed points. The complete picture is illustrated in Fig. 15. The turbo code used here is a $1/3$-rate code with $G(D) = (1,\ 5/7)$, $E_b/N_0 = -0.1$ dB, $n = 10^6$, and a pseudorandom interleaver. We see from the figure that there is a stable

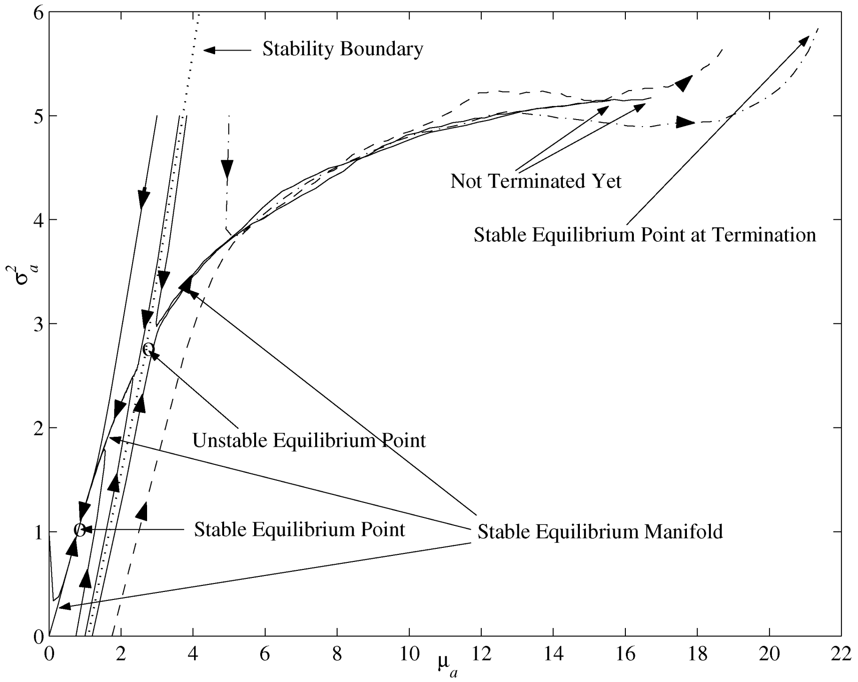

Fig. 15. Dynamics of turbo decoding.

equilibrium manifold at which every turbo decoding trajectory converges,regardless of what the initial point is. On the stable manifold, there is a stable fixed point with a low SNR which is paired with an unstable fixed point above it. The whole state space (i.e., the space of $(\mu_a,\ \sigma_a^2)$) is divided into two regions by the so-called *stability boundary* which intersects the unstable fixed point. When the initial state is to the left of the stability boundary, the decoding trajectory quickly moves to the stable equilibrium manifold and then converges at the stable fixed point with a low SNR. When the initial state is to the right of the stability boundary, the decoding trajectory again approaches the stable equilibrium manifold very quickly, then moves to the right along the manifold for a while but eventually converges at a stable fixed point or region with a high SNR.

The scenario in Fig. 15 corresponds to the case where E_b/N_0 is below a certain threshold. If E_b/N_0 exceeds this threshold, the stable and unstable equilibrium points coalesce and disappear. In this case, there is no stability boundary. That is, no matter where the initial point is, the decoding trajectory always converges at a stable fixed point or region with a high SNR, and the final SNR is only limited by the block size.

When E_b/N_0 is slightly above the threshold mentioned above (e.g., by a small fraction of a decibel) and the block size is finite but not too small, the dynamics in Fig. 15 is still roughly valid. The main difference is that the equilibrium manifold, the fixed points, and stability boundary all become somewhat fuzzy. This leads to the typical decoding phenomena mentioned earlier. Indeed, the fuzziness of the upper part of the stable equilibrium manifold is clearly visible in Fig. 15. Following this observation, it is clear that the high-SNR fixed point varies from one block to another considerably.

The detailed analysis of the turbo decoding dynamics are given in the following sections.

TABLE I
E_b/N_0 THRESHOLDS FOR AVOIDING LOW-SNR EQUILIBRIUM POINTS

$G(D)$ in octal	E_b/N_0 Threshold
(1, 5/7) (4-state)	0.04 dB
(1, 15/13) (8-state)	-0.04 dB
(1, 33/23) (16-state)	-0.01 dB
(1, 35/23) (16-state)	-0.01 dB
(1, 37/25) (16-state)	-0.15 dB
(1, 41/77) (32-state)	-0.21 dB
(1, 113/111) (64-state)	-0.22 dB

IX. LOW-SNR ANALYSIS

The low-SNR stable equilibrium point is invariant when the block size becomes very large due to the ergodicity of $\boldsymbol{E}$. Thus, they can be easily found by simulating a single but long block code. By adjusting the value of E_b/N_0, we can easily search for the threshold at which the low-SNR equilibrium point vanishes. Table I gives a list of SNR thresholds for different turbo codes. In these examples, the coding rate is $1/3$.

It is interesting to note from these examples that the thresholds are all a fraction of a decibel away from the Shannon limit $(-0.4906\ \text{dB})$. This means that the Shannon limit cannot be approached by merely increasing the block size. Table I also shows that little gain is made by increasing the number of states beyond 16. We stress that these thresholds do not rely on Gaussian approximations.

We note that the threshold values in Table I are similar but somewhat different from those given in [11]. We suspect that the differences are caused by the possibility that, in [11], the *a priori* signal (not its LLR) in each iteration is replaced by a Gaussian signal with an equivalent SNR. In our analysis, no such approximation is used. Our values are more accurate than those given in [11].

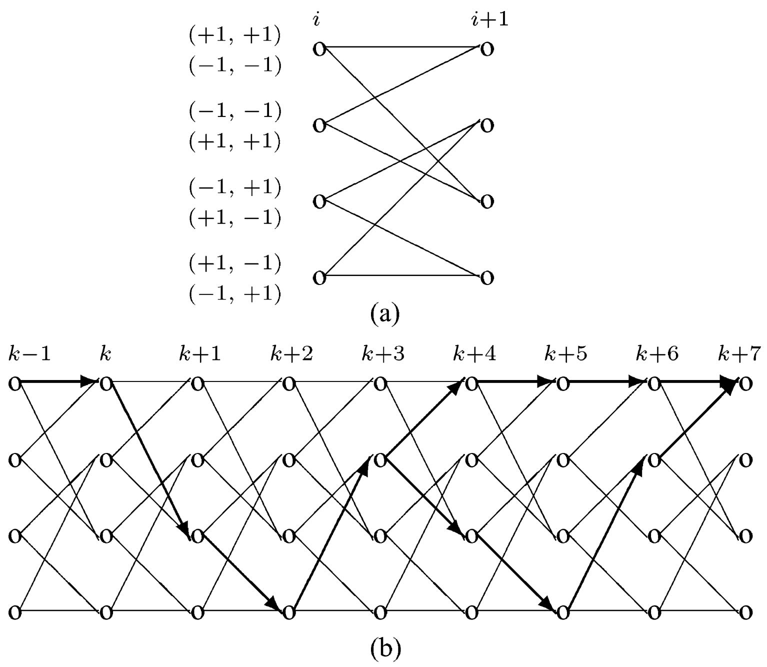

Fig. 16. Trellis diagram for $G(D) = (1,\ 5/7)$. (a) From i to $i+1$. (b) From $k-1$ to $k+7$.

If a low-SNR fixed point exists, the BER of turbo decoding can be easily calculated. Indeed, when (μ_e, σ_e^2) reaches a stable fixed point $(\bar{\mu}_e, \bar{\sigma}_e^2)$, so does (μ_d, σ_d^2), and its fixed point $(\bar{\mu}_d, \bar{\sigma}_d^2)$ is given by

$$\bar{\mu}_d = 2\bar{\mu}_e + \mu_r; \;\; \bar{\sigma}_d^2 = 2\bar{\sigma}_e^2 + \sigma_r^2. \tag{8}$$

Consequently, the BER of turbo decoding is given by

$$\text{BER} \approx Q(\bar{\mu}_d/\bar{\sigma}_d) = Q\left(\frac{2\bar{\mu}_e + \mu_r}{\sqrt{2\bar{\sigma}_e^2 + \sigma_r^2}}\right). \tag{9}$$

We point out that the idea of using simulations to search for E_b/N_0 thresholds has also been used in [14] where thresholds are estimated for a number of rate-1/2 codes. What is new in this paper is that we justify the simulation method using the ergodicity theory.

X. High-SNR Analysis

In order to understand the behavior of turbo decoding at a high SNR, it is necessary to study the simplified model (6). To help this study, we use a simple example of a $1/2$-rate convolutional code with $G(D) = (1,\ 5/7)$. We denote the systematic sequence and parity sequence by u and y, respectively. Its trellis diagram is shown in Fig. 16(a). The brackets on the left of the trellis indicate the systematic and parity bits corresponding to each branch of the trellis.

For a given k, the computation of $\boldsymbol{E}_k$ involves only weight-2 sequences, denoted by v, which takes 1 for each element except $v_k = -1$ and $v_{k+t} = -1$, $t = \pm 3, \pm 6, \ldots$. Two of these weight-2 sequences (corresponding to $t = 3$ and $t = 6$) are shown in Fig. 16(b) (thick arrows). For the first (shorter) weight-2 sequence, the systematic and parity bits are given by

$$\begin{aligned}(u_k, \ldots, u_{k+3}) &= (-1, +1, +1, -1)\\ (y_k, \ldots, y_{k+3}) &= (-1, -1, -1, -1).\end{aligned}$$

Similarly, for the second weight-2 sequence

$$\begin{aligned}(u_k, \ldots, u_{k+6}) &= (-1, +1, +1, +1, +1, +1, -1)\\ (y_k, \ldots, y_{k+6}) &= (-1, -1, -1, +1, -1, -1, -1).\end{aligned}$$

The third weight-2 sequence (not shown) is given by

$$\begin{aligned}(u_k, \ldots, u_{k+9}) &= (-1, +1, +1, +1, +1, +1, +1, +1, +1, -1)\\ (y_k, \ldots, y_{k+9}) &= (-1, -1, -1, +1, -1, -1, +1, -1, -1, -1).\end{aligned}$$

For notational simplicity, we denote the systematic part and parity part of $\boldsymbol{R}$ by α and β, respectively, denote $\boldsymbol{A}$ by γ, and drop the index k in α, β, and γ. Then, the contributions of the three weight-2 sequences above to $\boldsymbol{E}_k$ are given by

$$\begin{aligned}L_1 &= \alpha_3 + \gamma_3 + \beta_0 + \beta_1 + \beta_2 + \beta_3\\ L_2 &= \alpha_6 + \gamma_6 + \beta_0 + \beta_1 + \beta_2 + \beta_4 + \beta_5 + \beta_6\\ L_3 &= \alpha_9 + \gamma_9 + \beta_0 + \beta_1 + \beta_2 + \beta_4 + \beta_5 + \beta_7 + \beta_8 + \beta_9.\end{aligned}$$

respectively. If we use $L_{-1}, L_{-2}, \ldots$ to denote the contributions of the weight-2 sequences "traveling" to the left (i.e., with $t < 0$), we can rewrite (6) as

$$\boldsymbol{E}_k = -\ln(\exp(-L_+) + \exp(-L_-)) \tag{10}$$

where

$$L_+ = -\ln(\exp(-L_1) + \exp(-L_2) + \exp(-L_3) + \cdots)$$

and L_- is similarly defined. In particular, note that L_+ and L_- are independent and have the same statistics.

Also, from the expressions of L_1, L_2, L_3, we note that they share common terms. By re-arranging the terms, L_+ can be alternatively expressed using the following recursion:

$$\begin{aligned}L_+ &= \beta_0 + L_+^{(1)}\\ L_+^{(1)} &= \beta_1 + \beta_2 - \ln(\exp(-(\alpha_3 + \gamma_3 + \beta_3)) + \exp(-L_+^{(2)}))\\ L_+^{(2)} &= \beta_4 + \beta_5 - \ln(\exp(-(\alpha_6 + \gamma_6 + \beta_6)) + \exp(-L_+^{(3)}))\\ &\;\;\cdots\cdots\end{aligned}$$

The preceding observations also apply to a general convolutional code. We have the following result.

Lemma 5: Given a $1/2$-rate systematic, recursive, binary convolutional code $G(D) = (1,\ P(D)/Q(D))$ with an infinitely long block size, suppose $P(D)$ and $Q(D)$ are coprime and monic polynomials with the same degree. Denote by w_0 the weight of the parity sequence corresponding to the shortest weight-2 information sequence. Suppose the SLLR of the received signal $\boldsymbol{R}$ and SLLR of the *a priori* signal $\boldsymbol{A}$ are independent Gaussian white noises with means μ_r and μ_a and variances σ_r^2 and σ_a^2, respectively. Then, the simplified model (6) of the SLLR of the extrinsic signal $\boldsymbol{E}$, when only the first N weight-2 sequences on each side of the trellis are included, can be rewritten as (10) with

$$\begin{aligned} L_\pm &= \rho_\pm + L_\pm^{(1)} \\ L_\pm^{(i)} &= \delta_\pm^{(i)} - \ln(\exp(-\eta_\pm^{(i)}) + \exp(-L_\pm^{(i+1)})), \\ &\qquad\qquad i = 1, 2, \ldots, N-1 \\ L_\pm^{(N)} &= \delta_\pm^{(N)} + \eta_\pm^{(N)} \end{aligned} \tag{11}$$

where $\rho_\pm, \delta_\pm^{(i)}, \eta_\pm^{(i)}$ are independent Gaussian variables with the following distributions:

$$\begin{aligned} \rho_\pm &\sim \mathcal{N}(\mu_r,\ \sigma_r^2) \\ \delta_\pm^{(i)} &\sim \mathcal{N}((w_0 - 2)\mu_r,\ (w_0 - 2)\sigma_r^2) \\ \eta_\pm^{(i)} &\sim \mathcal{N}(2\mu_r + \mu_a,\ 2\sigma_r^2 + \sigma_a^2) \end{aligned} \tag{12}$$

where $\mathcal{N}(\mu, \sigma^2)$ denotes a Gaussian distribution with mean μ and variance σ^2.

Using the model in (10) and (11), we can analyze the evolution of μ_e and σ_e. Each iteration can be viewed as a mapping $\mathcal{M}$ from $(\mu_a,\ \sigma_a)$ to $(\mu_e,\ \sigma_e)$, i.e.,

$$(\mu_e,\ \sigma_e) = \mathcal{M}(\mu_a,\ \sigma_a).$$

A careful examination of (10) and (11) shows that the mapping $\mathcal{M}$ is *affine* with respect to μ_a, i.e.,

$$(\mu_e - \mu_a,\ \sigma_e) = \mathcal{M}(0,\ \sigma_a).$$

Hence, we only need to examine the decoding behavior for $\mu_a = 0$. Given μ_r (noting that $\sigma_r^2 = \mu_r$), and w_0 and σ_a, the corresponding $\mu_e - \mu_a$ and σ_e can be computed numerically.

Fig. 17 demonstrates a typical decoding behavior at a high SNR. This example uses $G(D) = (1,\ 5/7)$. Two values of E_b/N_0 are used. We consider the case of $E_b/N_0 = 0.5$ first. We see that when σ_a is small, $\sigma_e > \sigma_a$. As σ_a becomes large, $\sigma_e < \sigma_a$. This implies that there is a crossover point which is a stable fixed point for σ_a where $\sigma_e = \sigma_a$. This point occurs at about $\sigma_a = 2$. At this point, $\mu_e - \mu_a \approx 1.1$. Although the plot shows up to $\sigma_a = 2.5$, there are no other crossover points for σ_e. Similar observations apply to the case of $E_b/N_0 = -1.5$. In this case, the stable fixed point occurs at $\sigma_a \approx 1.6$ and at this point, $\mu_e - \mu_a \approx 0$. If we reduce E_b/N_0 further, $\mu_e - \mu_a$ will become negative.

From the preceding discussion, we reach the following conclusion.

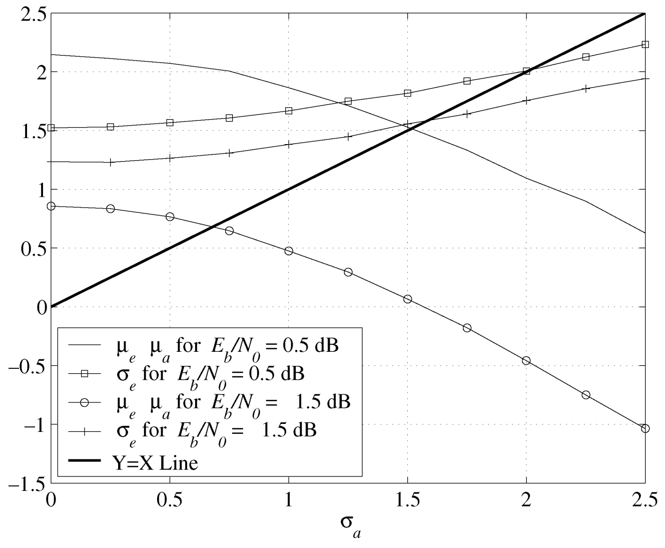

Fig. 17. $\boldsymbol{E}$ versus $\boldsymbol{A}$ at a high SNR.

TABLE II
E_b/N_0 THRESHOLDS: FOR DECODED SNR $\rightarrow \infty$

w_0	Examples of $G(D)$	E_b/N_0 Threshold
4	(1, 5/7) (4-state)	-1.475 dB
6	(1, 15/13) (8-state)	-3.45 dB
10	(1, 33/23), (1, 35/23) (16-state)	-5.8 dB

As the iteration number increases, no matter what the initial values of σ_a and μ_a are (provided that the simplified model is valid), σ_e and $\mu_e - \mu_a$ will approach constant values. Moreover, if $\mu_e - \mu_a$ approaches a positive constant, the SNR of the extrinsic signal (and thus the *a posteriori* signal) will increase indefinitely as the iteration number increases. Conversely, if $\mu_e - \mu_a$ approaches a negative constant, μ_e (and thus μ_a) will decrease indefinitely until the simplified model is no longer valid.

We remark that the conclusion above holds for every turbo code. Table II lists, for different turbo codes, the threshold values of E_b/N_0 for the asymptotic value of $\mu_e - \mu_a$ to be positive. Since these thresholds are far below the Shannon limit, the corresponding turbo decoding always diverges. In fact, the only parameter that determines whether the asymptotic value of $\mu_e - \mu_a$ is positive or not is w_0. Table II shows that for any turbo code where $w_0 \geq 4$, $\mu_e - \mu_a$ approaches a positive value. The following lemma shows that all commonly used turbo codes possess this property.

Lemma 6: Suppose the constituent encoder $G(D) = (1,\ P(D)/Q(D))$ is such that $P(D)$ and $Q(D)$ are coprime, monic, with the same degree, and $Q(D)$ has an odd weight. Then, $w_0 \geq 4$.

Note that all commonly used $G(D)$ have an odd weight for their $Q(D)$ (both primitive or nonprimitive polynomials).

XI. Fixed Point at Termination

For a turbo code with a finite (but large) block size, the effect of correlations in $\boldsymbol{A}$ can no longer be ignored when the number

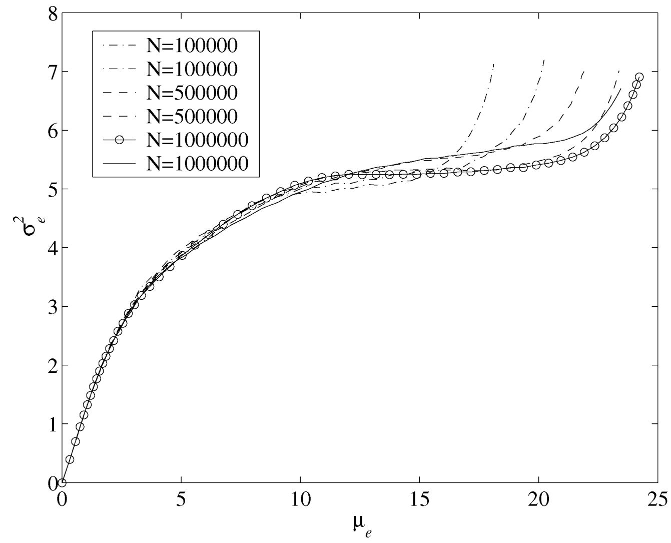

Fig. 18. Three stages of turbo decoding.

of iterations increases. Although the exact effect is difficult to analyze, simulations show that the variance σ_e will increase dramatically as the number of iterations exceeds a certain value. As σ_e (and thus σ_a) increases, the increment $\mu_e - \mu_a$ can no longer be sustained (see Fig. 17). As $\mu_e - \mu_a$ falls to zero, the decoding converges. Due to the finite block size, only quasi-convergence is achieved, i.e., μ_e or σ_a converges at a small neighborhood.

Also, due to the finite block size, which breaks down the ergodicity for $\boldsymbol{E}$ when the number of iterations is large, the fixed point at the termination varies from one block to another considerably. For each block, the decoded BER can still be estimated using (9). However, the resulting BER is only indicative because there are typically too few error bits to make the estimate accurate (This is a common problem when estimating the tail distribution using a finite number of samples.) Also, the BER varies significantly from one block to another and thus needs to be averaged over many blocks.

Fig. 18 illustrates the convergence at termination. The example used is $G(D) = (1,\ 5/7)$ with $E_b/N_0 = 0.25$ dB. The three stages of decoding behavior are clearly seen in Fig. 18. The circles on one of the curves correspond to the values of (μ_e, σ_e^2) for different iterations. In the first several iterations when μ_a is very small, good increments of $\mu_e - \mu_a$ are seen. In the following several iterations when μ_a is close to a "potential" low-SNR fixed point, increments of $\mu_e - \mu_a$ become small. Once this region is passed and decoding enters the high-SNR region, the simplified model (10)–(11) takes over. In this region, σ_e starts to converge and good (and roughly) constant increments of $\mu_e - \mu_a$ are seen. Finally, when the number of iterations increases further, σ_e increases very quickly, causing μ_e and σ_e to converge. It is also seen in the figure that the longer the block size, the larger the final value of μ_e tends to be. However, there is a large variation in the final state for different code blocks due to the breakdown of the ergodicity.

XII. The Inadequacy of SISO Models

In this section, we show two SISO models that are inadequate for characterizing the dynamics of a Log-MAP decoder. The corresponding input–output parameters are SNR and mutual information. We point out the "strange" phenomenon that a Log-MAP decoder can take an input with a very high SNR or very good mutual information and produce an output with a much lower SNR or poorer mutual information. To this end, we first need to explain how mutual information is defined.

Recall that $\boldsymbol{A}$ can be approximated using $\mathcal{N}(\mu_a, \sigma_a^2)$. We need to consider $\boldsymbol{A}u = L_a/2$ since this is the true input signal. Its probability density function is given by

$$p_a(\xi|u_i = \pm 1) = \frac{1}{\sqrt{2\pi}\sigma_a} \exp(-(\xi \mp \mu_a)^2/2\sigma_a^2).$$

The mutual information between u and $\boldsymbol{A}u$ (or L_a) is defined as in [26]

$$I_a = \frac{1}{2} \sum_{u_i \in \{1,-1\}} \int_{-\infty}^{\infty} p_a(\xi|u_i) \cdot \log_2 \frac{2p_a(\xi|u_i)}{p_a(\xi|u_i = 1) + p_a(\xi|u_i = -1)} d\xi.$$

Simplifying the above leads to

$$I_a = 1 - \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \exp\left(-\left(t - \frac{\mu_a}{\sigma_a}\right)^2/2\right) \cdot \log_2\left(1 + \exp\left(-\frac{\mu_a}{\sigma_a}2t\right)\right) dt.$$

This implies that I_a is a function of μ_a/σ_a (or the SNR of $\boldsymbol{A}$). For this reason, we will denote I_a as $I_a(\mu_a/\sigma_a)$. It can be shown that $I_a(\cdot)$ is a monotonically increasing function with $I_a(0) = 0$ and $I_a(\infty) = 1$.

We note that our definition of mutual information is consistent with the definition in [9], [10] but without enforcing the mean-to-variance relationship $\mu_a = \sigma_a^2$.

Fig. 19 offers an alternative view of the turbo dynamics for the same turbo code as in Fig. 15. This is a flowchart for Log-MAP decoding. On the chart, each circle represents an input to the Log-MAP decoder and the attached stem represents the difference between the output and input. The stable equilibrium manifold is clearly visible in the flowchart. We observe that when the input is far away from the stable equilibrium manifold, Log-MAP decoding makes big steps toward the manifold. In contrast, only small steps are made when the input is on the manifold.

To demonstrate the strange phenomenon mentioned above, we consider the input $(\mu_a, \sigma_a^2) = (1, 0)$. Obviously, the input SNR is infinity and the input mutual information equals 1 (perfect). However, the output of Log-MAP decoding has $\mu_e < 2$ but $\sigma_e^2 > 1$, i.e., the output SNR or mutual information is much worse than the input. If we continue with more iterations, the decoding trajectory will eventually lead to the low-SNR fixed point, as shown in Fig. 15. It is also interesting to note that this strange phenomenon also occurs when (μ_a, σ_a^2) is to the right of the stability boundary in Fig. 15 or even when the stability boundary is nonexistent. Indeed, given any input (μ_a, σ_a^2) with $\mu_a > 0$ and sufficiently small σ_a, the output (μ_e, σ_e^2) has a worse SNR or mutual information than the input.

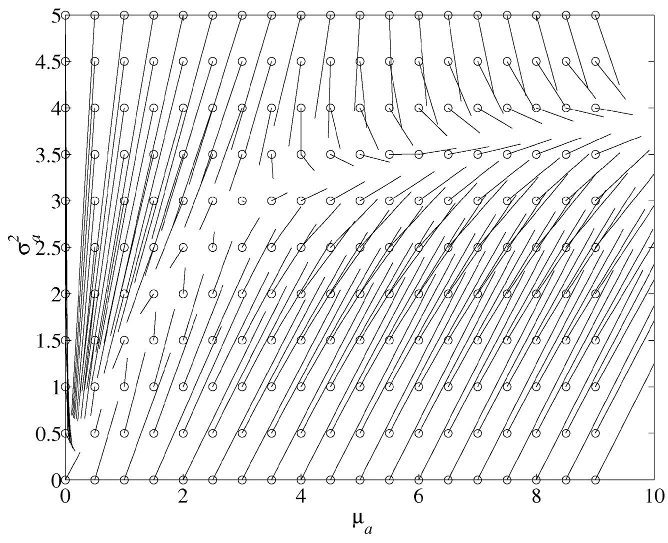

Fig. 19. Flowchart of turbo decoding.

XIII. Applications

Understanding of turbo dynamics is useful in aiding the decoding process. Here we study two issues: decoding termination and fast decoding.

A. Decoding Termination

Many termination algorithms are available. Notable examples include the cross-entropy method [3], hard-decision-aided criteria [27], and the average log-likelihood threshold method [5]. Their common aim is to try to terminate the decoding as early as possible but with a minimal effect on decoding errors. The basic idea in all these algorithms is to check how well the decoded output signals converge and declare termination when certain convergence criteria are met.

From our analysis of turbo dynamics earlier, we understand that the state of the decoder is represented by (μ_e, σ_e^2). Therefore, it is natural to establish termination criteria using (μ_e, σ_e^2). For example, termination can occur if (μ_e, σ_e^2) is such that a sufficiently high SNR is reached or if (μ_e, σ_e^2) starts to converge or oscillate.

One question to ask when using (μ_e, σ_e^2) is how to compute it (noting that the information signal u is not known). To answer this question, we recall that the SLLR signal $\boldsymbol{E}$ is related to the LLR signal L_e (which is available) as follows:

$$\boldsymbol{E}_k = L_{e,k} u_k / 2.$$

This implies

$$\mathcal{E}[\boldsymbol{E}^2] = \mathcal{E}[L_e^2]/4.$$

Using the assumption that $\boldsymbol{E}$ has a Gaussian distribution, there exists a one-to-one (nonlinear) mapping between (μ_e, σ_e^2) and $(\mathcal{E}[|L_e|], \mathcal{E}[L_e^2])$ which can be precomputed and stored in a lookup table. Note that $(\mathcal{E}[|L_e|], \mathcal{E}[L_e^2])$ can be easily computed on-line.

Alternatively, we may simply use the approximation that

$$\mathcal{E}[\boldsymbol{E}] \approx \mathcal{E}[|L_e|]/2$$

which is very accurate when μ_e is large and is reasonably accurate in most cases when decoding is close to termination. It follows that

$$(\mu_e, \sigma_e^2) \approx \left(\frac{\mathcal{E}[|L_e|]}{2}, \frac{\mathcal{E}[L_e^2] - (\mathcal{E}[|L_e|])^2}{4} \right).$$

B. Fast Decoding

Many suboptimal decoding algorithms exist which are faster than the Log-MAP algorithm. Examples include the soft-output Viterbi algorithm (SOVA) [28] and the Max-Log-MAP algorithm [3]. However, suboptimal decoding algorithms yield a larger BER, or, equivalently, requiring a higher E_b/N_0.

The question we ask is: Can we speed up the decoding process without sacrificing the BER?

It turns out that this is quite possible. The key lies in two observations of turbo dynamics. Returning to Fig. 18, we first observe that the improvement of (μ_e, σ_e^2) is very uneven for different iterations. More specifically, small improvements are made when (μ_e, σ_e^2) is close to a potential low-SNR fixed point but large improvements are made elsewhere. This suggests that a suboptimal decoder can be used when (μ_e, σ_e^2) is not close to a potential low-SNR fixed point. The only criterion for the suboptimal decoder is that it should also maintain a flow of improvement for (μ_e, σ_e^2). The second observation is that the decoding trajectory always converges at a stable equilibrium manifold. This suggests that if a suboptimal decoder is found to be inadequate, we can always switch back to Log-MAP decoding without causing decoding failure. The only drawback of this switching is the potential computing cost "wasted" by the suboptimal decoder. However, provided that an appropriate switching strategy is used, an overall time saving is possible.

Fig. 20 compares three decoding methods. The turbo code has 1/3-rate, $G(D) = (1, 5/7)$, $n = 10\,000$, and a pseudorandom interleaver. The three methods are: Log-MAP, MAX-Log-MAP, and a Hybrid method which will be explained later. It is assumed that a MAX-Log-MAP iteration is twice as fast as a Log-MAP iteration, and thus counted as half an (equivalent) iteration. This assumption is based on the facts that a MAX-Log-MAP decoder can be implemented as a modified SOVA decoder [29] and that SOVA decoding is half as complex as Log-MAP decoding [3]. Thirty equivalent iterations are shown in the figure for $E_b/N_0 =$ 0.2, 0.3, 0.4, and 0.6 dB, respectively.

For low values of E_b/N_0 (0.2 and 0.3 dB), we see that MAX-Log-MAP performs slightly better than Log-MAP for a small number of equivalent iterations, but as the number of iterations increases, Log-MAP becomes better. There is a significant margin of BER asymptotically. However, at higher values of E_b/N_0 (0.4 and 0.6 dB), MAX-Log-MAP outperforms Log-MAP. What is more interesting (and somewhat surprising) is that as the number of iterations becomes very large, MAX-Log-MAP decoding has a slightly lower BER floor than Log-MAP decoding. This phenomenon is more evident at $E_b/N_0 = 0.6$ dB. Although only 30 equivalent iterations are shown, both decoding methods have roughly converged by then.

The hybrid method uses a very simple heuristic rule: it applies MAX-Log-MAP for the first iteration and whenever the average

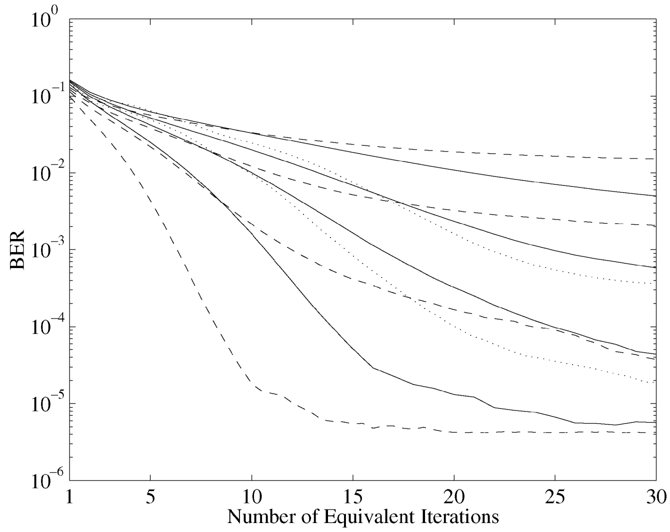

Fig. 20. Comparisons of decoding methods. (Legend: dashed curves for MAX-Log-MAP, solid curves for Log-MAP, and dotted curves for Hybrid. Top two curves: E_b/N_0 =0.2 dB; next three curves: $E_b/N_0 = 0.3$ dB; next three curves: $E_b/N_0 = 0.4$ dB. Bottom two curves: $E_b/N_0 = 0.6$ dB.)

of $|\boldsymbol{E}|$ in the previous iteration is increased by 0.5 when compared with the iteration before; otherwise, Log-MAP is used. The purpose is to demonstrate that it is possible to take the advantages of both Log-MAP and MAX-Log-MAP methods to give a decoding method which outperforms the both of them. This is achieved for $E_b/N_0 = 0.3$ and 0.4 dB when the number of equivalent iterations exceeds 16 and 17, respectively.

We remark that the rule used in this hybrid method is not optimized. More improvement may be possible if the rule is properly tuned. However, we do caution that the hybrid method requires both Log-MAP and MAX-Log-MAP algorithms which may complicate the hardware/software implementation. Also, the assumption that MAX-Log-MAP is twice as fast as Log-MAP may not be realistic for a specific implementation. Nevertheless, the idea behind the hybrid method is hoped to provide a new direction for improving the decoding speed.

XIV. Conclusion

In this paper, we have proposed a stochastic approach to the modeling and analysis of turbo decoding. Two key results, ergodicity and Gaussian approximations, have been established which lead to some new understanding of turbo decoding. In particular, we are able to build a simple dynamic model for turbo decoding and reveal the intricate behavior of turbo decoding unknown previously.

Some concluding remarks are in order.

First, we note that we have restricted our study to turbo codes without puncturing. It is straightforward to generalize the analysis to turbo codes with puncturing. The only difference is that if a turbo code is punctured, the input–output signals are no longer stationary. Instead, these signals become (quasi-)cyclostationary if a (quasi-)periodic puncturer is used. However, a slightly weaker notion of stationarity called *asymptotic mean stationarity* [31] can be used to study the ergodicity. This allows us to maintain the essential property that the statistics of the extrinsic signals can be computed from a single realization, ensuring that Gaussian approximations and modeling of turbo dynamics work as in the case without puncturing.

Second, the stochastic approach generalizes naturally to other types of turbo codes, including turbo codes with more than two constituent encoders and those with serial concatenations. It would be interesting to conduct a comparative study of different types of turbo codes using the stochastic approach.

Third, the stochastic approach generalizes to other iterative decoding schemes, including the well-known sum-of-product based iterative schemes [30] for LDPC codes. The ergodicity result in this paper can be easily extended to regular LDPC codes. For irregular LDPC codes, it seems that the weaker notion of asymptotic mean stationarity [31] can be used to establish a similar ergodicity result. Gaussian approximations should still be valid for any sum-of-product decoding schemes. More precisely, if the input variables have lognormal distributions with similar statistics, then the output variables of a sum-of-product decoder are well approximated using lognormal distributions. We note that there is a lot of work done on stochastic analysis of LDPC codes, see, e.g., [33], [34]. We hope the approach used in this paper can complement and expand the work in this area.

Finally, we expect that the better understanding of turbo decoding we have gained from the stochastic approach is useful in searching for more powerful turbo codes and more efficient decoding algorithms. The fast decoding example in this paper demonstrates this point.

Appendix
Proofs

A. *Proof of Lemma 1*

For simplicity, we prove the result for $n = 2$. The proof for a general case is similar except that the notation is more involved. Let Δ be a given region in $\mathbb{R}^2$. Suppose Δ is a rectangle, i.e., $\Delta = \Delta_1 \oplus \Delta_2$. Then, for any infinite sequence $(\ldots t_{-1} < t_0 < t_1, \ldots)$ and $N > 0$

$$\frac{1}{2N}\sum_{i=-N}^{N} I_\Delta([x_{t_i}^{(1)}, x_{t_i}^{(2)}]) = \frac{1}{2N}\sum_{i=-N}^{N} I_{\Delta_1}(x_{t_i}^{(1)}) I_{\Delta_2}(x_{t_i}^{(2)}) = \frac{1}{2N}\sum_{t_i : x_{t_i}^{(2)} \in \Delta_2} I_{\Delta_1}(x_{t_i}^{(1)}).$$

Denote by N_2 the number of $t_i, -N \le i \le N$ for which $x_{t_i}^{(2)} \in \Delta_2$. It follows from the ergodicity of $x^{(1)}$ that

$$\sum_{t_i : x_{t_i}^{(2)} \in \Delta_2} I_{\Delta_1}(x_{t_i}^{(1)}) \to N_2 P_1(\Delta_1), \qquad \text{as } N_2 \to \infty$$

with probability 1. From the ergodicity of $x^{(2)}$, we have, with probability 1

$$\frac{N_2}{2N} \to P_2(\Delta_2).$$

In the above, P_j is the probability measure for $x^{(j)}$, $j = 1, 2$. Hence,

$$\frac{1}{2N}\sum_{i=-N}^{N} I_\Delta([x_{t_i}^{(1)}, x_{t_i}^{(2)}]) \to P_1(\Delta_1)P_2(\Delta_2) \qquad (13)$$

with probability 1, as $N, N_2 \to \infty$. Indeed, this holds even when $N_2 \not\to \infty$ because this corresponds to the case where $P_2(\Delta_2) = 0$ and in this case, the limit above is still true.

Now, suppose a given measurable set Δ is nonrectangular. In this case, we may approximate it to an arbitrary precision using the union of a finite set of disjoint rectangular sets $\Delta^{(k)}$ and apply the ergodicity result above to each of these rectangular regions. Hence, the ergodicity result still holds for any region Δ. □

B. Proof of Lemma 2

Since the function f is time invariant, the random process y is stationary. Given any interval $\Delta_y \subset \mathbb{R}$, then the set $\Delta = f^{-1}(\Delta_y)$ is a measurable set in $\mathbb{R}^n$ because f is a measurable function. Hence, for any infinite sequence $(\ldots t_{-1} < t_0 < t_1, \ldots)$

$$\frac{1}{2N}\sum_{i=-N}^{N} I_{\Delta_y}(y_{t_i}) = \frac{1}{2N}\sum_{i=-N}^{N} I_{\Delta}(f(x_{t_i}^{(1)}, x_{t_i}^{(2)}, \ldots, x_{t_i}^{(n)})) \to P_x(\Delta) = P_y(\Delta_y)$$

with probability 1. □

C. Proof of Theorem 1

We prove Theorem 1 for $\boldsymbol{E}$ only since the proof for $\boldsymbol{D}$ is similar. Note from (2) that the function for computing $\boldsymbol{E}_k$ is stationary (with respect to k) and measurable. We define random processes $x^{(n)}$ and $y^{(n,m)}$ as

$$x_i^{(n)} = \boldsymbol{A}_{i+n}; \quad y_{i,j}^{(n,m)} = \boldsymbol{R}_{i+n,j+m};$$

Hence, the set of these new random processes are both independent and independent of each other. From Lemma 1, they are jointly ergodic. Returning to (2), $\boldsymbol{E}_k$ can now be viewed as a function of $x_k^{(n)}$ and $y_k^{(n,m)}$. Obviously, this function is measurable. It follows from Lemma 2 that $\boldsymbol{E}$ is ergodic. □

D. Proof of Lemma 3

Given any $1 \le i \le n$, there are at most $2S$ number of j such that $|i-j| \le S$. For each of these j, since $T(j)$ is chosen randomly, the probability that $|T(i) - T(j)| \le S$ is at most $2S/N$. It follows that

$$P(|T(i) - T(j)| > S, \qquad \forall\, |i-j| \le S) \ge (1 - 2S/N)^{2S}$$

which approaches 1 as $N \to \infty$. This is the same as (4). □

E. Proof of Theorem 2

We start with $\ell = 1$. Note that $\boldsymbol{A}(1,n) = 0$ is an ergodic random process when $n \to \infty$. It follows from Theorem 1 that $\boldsymbol{E}(1,n)$ approaches an ergodic random process as $n \to \infty$. This random process is not necessarily independent. However, the dependence is only "local" in the sense that the dependence between $\boldsymbol{E}_k(1,n)$ and $\boldsymbol{E}_{k+\tau}(1,n)$ vanishes as $|\tau|$ becomes sufficiently large. This property follows from Remark 1. Hence, the spreading property of T given in Lemma 3 implies that after interleaving, the resulting $\boldsymbol{A}(2,n)$ for the next iteration becomes independent as $n \to \infty$. Also, $\boldsymbol{A}(2,n)$ is independent of $\boldsymbol{R}(2,n)$ due to the random interleaving. Hence, by invoking Theorem 1 again, it follows that $\boldsymbol{E}(2,n)$ approaches an ergodic random process as $n \to \infty$. Similarly, after de-interleaving, the resulting $\boldsymbol{A}(3,n)$ for the next iteration becomes independent as $n \to \infty$. The arguments above can continue for other iterations. Hence, $\boldsymbol{E}(\ell,n)$ approaches an ergodic random process for each ℓ when $n \to \infty$. □

F. Proof of Lemma 4

Recall the expression (2) for $\boldsymbol{E}_k$. Consider any information sequence v with $v_k = 1$ in the first logarithmic sum. In order for v to have a noticeable contribution in the sum, v must be terminating. Since the convolutional code is recursive, such a v must be at least weight-2, i.e., v_i must equal -1 for at least two $i \neq k$. When μ_a is large and the MVR is ≥ 1, the contributions of these sequences to the logarithmic sum are negligible. It follows that the first logarithm sum is negligible. In contrast, the second logarithmic sum dominates. For this sum, again, we only need to consider terminating weight-2 information sequences because μ_a is large. For each such sequence, there is only one $\boldsymbol{A}_t$ term contributing to the logarithmic sum. □

G. Proof of Lemma 5

The expressions for $L_\pm$ are derived similarly to the example prior to Lemma 5. The terms $\rho_\pm, \delta_\pm^{(i)}, \eta_\pm^{(i)}$ have independent Gaussian distributions because $\boldsymbol{R}$ and $\boldsymbol{A}$ are independent Gaussian white noises. Due to the properties of $P(D)$ and $Q(D)$, the parity sequence corresponding to the first (shortest) weight-2 information sequence has the patten $(-1, d_1, d_2, \ldots, d_m, -1)$, where $d_i = \pm 1$ with the weight sum equal to $w_0 - 2 \ge 0$. Similarly, the parity sequence corresponding to the second weight-2 information sequence has the pattern $(-1, d_1, d_2, \ldots, d_m, +1, d_1, d_2, \ldots, d_m, -1)$. In particular, the two sequences above share $(w_0 - 2)$ parity bits. Similarly, if we continue this exercise, we will find that the ith and $(i+1)$th parity sequences share $(w_0 - 2)$ parity bits. Using this fact, the means and variances in (12) are easily verified. □

H. Proof of Lemma 6

Let $1 + D^k$ be the weight-2 information sequence which yields the minimum weight for the parity sequence. Then, $1 + D^k = Q(D)V(D)$ for some polynomial $V(D)$. Since $1 + D^k$ has an even weight and $Q(D)$ has an odd weight, $V(D)$ must have an even weight. This means that the parity sequence, which equals $V(D)P(D)$, must have an even weight. Therefore, we only need to eliminate the possibility that the weight of $V(D)P(D)$ equals 2. Since $P(D)$ and $Q(D)$ are monic and with the same degree, $V(D)P(D)$ having weight equal 2 would mean that $V(D)P(D) = 1 + D^k$, violating the assumption that $P(D)$ and $Q(D)$ are coprime. □

ACKNOWLEDGMENT

The author wishes to thank the anonymous reviewers for providing constructive comments and additional references relevant to this work.

References

[1] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo-codes," in *Proc. Int. Conf. Communications*, Geneva, Switzerland, May 1993, pp. 1064–1070.

[2] S. Benedetto and G. Montorsi, "Unveiling turbo codes: Some results on parallel concatenated coding schemes," *IEEE Trans. Inf. Theory*, vol. 42, no. 2, pp. 409–429, Mar. 1996.

[3] J. Hagenauer, E. Offer, and L. Papke, "Iterative decoding of binary block and convolutional codes," *IEEE Trans. Inf. Theory*, vol. 42, no. 2, pp. 429–445, Mar. 1996.

[4] T. Richardson, "The geometry of turbo-decoding dynamics," *IEEE Trans. Inf. Theory*, vol. 46, no. 1, pp. 9–23, Jan. 2000.

[5] A. C. Reid, T. A. Gulliver, and D. P. Taylor, "Convergence and errors in turbo-decoding," *IEEE Trans. Commun.*, vol. 49, no. 12, pp. 2045–2051, Dec. 2001.

[6] D. Agrawal and A. Vardy, "The turbo decoding algorithm and its phase trajectories," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 699–722, Feb. 2001.

[7] L. Duan and B. Rimoldi, "The iterative turbo decoding algorithm has fixed points," *IEEE Trans. Inf. Theory*, vol. 47, no. 7, pp. 2993–2995, Nov. 2001.

[8] A. Sella and Y. Be'ery, "Convergence analysis of turbo decoding of product codes," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 723–735, Feb. 2001.

[9] S. ten Brink, "Convergence behavior of iteratively decoded parallel concatenated codes," *IEEE Trans. Commun.*, vol. 49, no. 10, pp. 1727–1737, Oct. 2001.

[10] D. Divsalar, S. Dolinar, and F. Pollara, "Iterative turbo decoder analysis based on density evolution," *IEEE J. Sel. Areas Commun.*, vol. 19, no. 5, pp. 891–907, May 2001.

[11] H. El Gamal and A. R. Hammons, "Analyzing the turbo decoder using the Gaussian approximation," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 671–686, Feb. 2001.

[12] J. W. Lee and R. E. Blahut, "Generalized EXIT chart and BER analysis of finite-length turbo codes," in *Proc. GLOBECOM 2003*, vol. 4, San Fransisco, CA, Dec. 2003, pp. 2067–2072.

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

[14] T. Richardson and R. Urbanke, "Thresholds for turbo codes," in *Proc. IEEE Int. Symp. Information Thoery*, Sorrento, Italy, 2000, p. 317.

[15] J. L. Doob, *Stochastic Processes*. New York: Wiley, 1953.

[16] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Trans. Inf. Theory*, vol. IT-20, no. 3, pp. 284–287, May 1974.

[17] L. F. Fenton, "The sum of log-normal probability distributions in scatter transmission systems," *IEE Trans. Commun. Syst.*, vol. CS-8, no. 1, pp. 57–67, Mar. 1960.

[18] S. Schwartz and Y. S. Yeh, "On the distribution function and moments of power sums with log-normal components," *Bell Syst. Tech. J.*, vol. 61, pp. 1441–1462, Sep. 1982.

[19] C.-L. Ho, "Calculating the mean and variance of power sums with two log-normal components," *IEEE Trans. Veh. Technol.*, vol. 44, no. 4, pp. 756–762, Nov. 1995.

[20] D. C. Schleher, "Generalized Gram-Charlier series with application to the sum of log-normal variates," *IEEE Trans. Inf. Theory*, vol. IT-23, no. 2, pp. 275–280, Mar. 1977.

[21] A. A. Abu-Dayya and N. C. Beaulieu, "Outage probabilities in the presence of correlated lognormal interferers," *IEEE Trans. Veh. Technol.*, vol. 43, no. 1, pp. 164–173, Feb. 1994.

[22] P. Pirinen, "Statistical power sum analysis for nonidentically distributed correlated lognormal signals," in *Proc. 2003 Finnish Signal Processing Symp.*, Tampere, Finland, May 2003, pp. 254–258.

[23] N. C. Beaulieu, A. Abu-Dayya, and P. J. McLane, "Estimating the distribution of independent lognormal random variables," *IEEE Trans. Commun.*, vol. 43, pp. 2869–2873, Dec. 1995.

[24] A. Safak, "Statistical analysis of the power sum of multiple correlated log-normal components," *IEEE Trans. Veh. Technol.*, vol. 42, no. 1, pp. 58–61, Feb. 1993.

[25] S. B. Slimane, "Bounds on the distribution of a sum of independent lognormal random variables," *IEEE Trans. Commun.*, vol. 49, no. 6, pp. 975–978, Jun. 2001.

[26] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. New York: Wiley, 1991.

[27] R. Shao, S. Lin, and M. Fossorier, "Two simple stopping criteria for turbo decoding," *IEEE Trans. Commun.*, vol. 47, no. 8, pp. 1117–1120, Aug. 1999.

[28] J. Hagenauer and P. Hoeher, "A Viterbi algorithm with soft-decision outputs and its applications," in *Proc. GLOBECOM*, Dallas, TX, Nov. 1989, pp. 1680–1686.

[29] M. P. C. Fossorier, F. Burkert, S. Lin, and J. Hagenauer, "On the equivalence between SOVA and MAX-Log-MAP decodings," *IEEE Commun. Lett.*, vol. 2, no. 5, pp. 137–139, May 1998.

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

[31] R. M. Gary, *Probability, Random Processes, and Ergodic Properties*. New York: Springer-Verlag, 1988.

[32] R. J. McEliece, D. J. C. MacKay, and J.-F. Cheng, "Turbo decoding as an instance of Pearl's 'belief propagation' algorithm," *IEEE J. Sel. Areas Commun.*, vol. 16, no. 2, pp. 140–152, Feb. 1998.

[33] T. J. Richardson, M. A. Shokrollahi, and R. L. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," *IEEE Trans. Inf.. Theory*, vol. 47, no. 2, pp. 619–637, Feb. 2001.

[34] T. J. Richardson and R. L. Urbanke, "The capacity of low-density parity-check codes under message-passing decoding," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 599–618, Feb. 2001.