2005 International Conference on Control and Automation (ICCA2005)
June 27-29, 2005, Budapest, Hungary

Dynamic Modeling and Analysis of Iterative Decoding for Turbo Codes

Minyue Fu *Fellow, IEEE*

***Abstract*— Turbo codes and low density parity check codes are two classes of most powerful error correcting codes. What makes these codes so powerful is the use of the so-called iterative decoding or turbo decoding. Roughly speaking, an iterative decoding process is an iterative learning process for a complex system where the objective is to provide a good suboptimal estimate of a desired signal. Iterative decoding is used when the true optimal estimation is impossible due to prohibitive computational complexities. Despite that iterative decoding algorithms are known to be very successful, there is no satisfactory understanding of their "magical" power. In fact, the behavior of iterative decoding is a big mystery in the coding theory. The aim of this presentation is to show how to model and analyze an iterative decoding process using a system-theory based approach. More specifically, we can view the iterative decoding process as a feedback system. With this view, we propose a stochastic framework for dynamic modeling and analysis of iterative decoding. By using appropriate statistical parameters to describe the signals in an iterative decoding process, we show that the process can be adequately approximated by a two-input, two-output nonlinear dynamic model. We have discovered that a typical decoding process is much more intricate than previously known, involving two regions of attractions, several fixed points, and a stable equilibrium manifold at which all decoding trajectories converge. This new modeling approach is useful in gaining new knowledge on iterative decoding and devising better decoding algorithms.**

I. Introduction

Control and digital communications are two important disciplines in electrical engineering. Control theory studies modeling, analysis, design, and control of dynamical systems, and digital communications theory studies reliable transmission of digital information. The two disciplines seem to employ different mathematical tools because control theory deals with continuous (or analog) signals whereas digital communications theory deals with finite-alphabet signals. Nevertheless, there is a longstanding history of interdisciplinary research between the two. Notable examples include channel equalization, system modeling and identification, signal analysis (temporal, spectral, and spatial), and signal detection.

Recently, the interplay between control and digital communications has been further strengthened by important developments in at least two specific areas. The first area is the control of dynamical systems over communication links [1]-[11]. Traditional control theory assumes that the feedback channel is analog and solely dedicated to control purposes. However, more and more industrial systems are controlled via digital communication links such as Fieldbus, local area networks and even the internet. These communication links are shared with other network functions. Moreover, feedback signals must be sampled and quantized. This scenario is depicted in Figure 1.

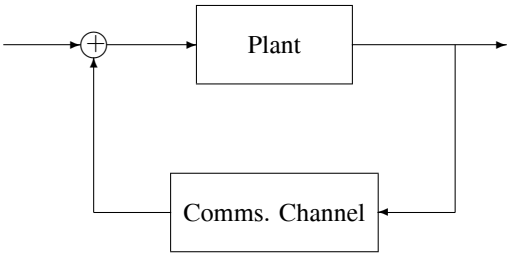

Fig. 1. Feedback Control over Communications Links

The second area is digital signal decoding and detection. Traditional maximum likelihood (ML) decoding and detection algorithms aim to find an optimal solution from a given finite set of possible solutions. In more complicated detection scenarios, such as multi-user and space-time communications, ML is no longer a viable detection strategy because it becomes computationally too expensive. A tremendous amount of research has been conducted on iterative algorithms, which can offer a performance close to that of ML at a significantly reduced computational cost [12]-[16]. When applied to turbo codes and low-density-parity-check (LDPC) codes, iterative algorithms are capable of approaching Shannon's channel capacity bound within a very small margin [12], [14]. The key to the design of iterative algorithms is that every iteration produces probabilistic measures for all possible solutions. These measures are then fed back and used as input to the next iteration. Because every iteration accepts continuous-valued input and produces continuous-valued output, these algorithms are often referred to as soft decision algorithms.

Despite that iterative decoding algorithms are known to be very successful, there is no satisfactory understanding of their "magical" power. In fact, the behavior of iterative decoding is a big mystery in the coding theory.

This presentation is concerned with modeling and analysis of iterative decoding for turbo codes. Our aim is to show how to model and analyze an iterative decoding process using a system theory based approach. More specifically, we can view the iterative decoding process as a feedback system. With this view, we propose a stochastic framework for dynamic modeling and analysis of iterative decoding. By using appropriate statistical parameters to describe the

Minyue Fu is with School of Electrical Engineering and Computer Science, University of Newcastle, Callaghan, N.S.W. 2308 Australia. (email: minyue.fu@newcastle.edu.au)

signals in an iterative decoding process, we show that the process can be modeled by a two-input, two-output nonlinear dynamic system. We have discovered that a typical decoding process is much more intricate than previously known, involving two regions of attractions, several fixed points, and a stable equilibrium manifold at which all decoding trajectories converge. This new modeling approach enables us to gain new understanding of iterative decoding processes and to devise better decoding algorithms. The material for this presentation is mostly based on [26], [27].

The rest of the presentation is organized as follows. In Section 2, we give a brief introduction to the turbo codes. In Section 3, we introduce iterative decoding algorithms. Section 3 is the main body of the presentation where stochastic modeling is introduced. In Section 4, we discuss some possible applications. Concluding remarks are given in Section 5.

II. Turbo Codes

Ever since Shannon published his famous channel coding theorem [28] in 1948, the advances in the field of communications theory can in one way be viewed as a painstaking pursue for discovering practical coding and decoding algorithms which enable us to approach the Shannon capacity limit. A major breakthrough in communications theory since the pioneering work of Shannon has been the invention of parallel concatenated convolutional codes, also known as turbo codes [12]. The bit error performance of these codes is only a fraction of dB away from Shannon's capacity limit, yet the decoding complexity is well within today's computing power. The invention of turbo codes also inspires the rediscovery of the so-called low-density parity check (LDPC) codes [14] which are even more powerful (0.01dB off the Shannon limit) and more suitable for applications such as digital storage.

In this section, we introduce the turbo codes. To help understand these wonderful codes, we first need to review some basics of channel coding. Figure 2 represents a typical digital communications system. The source signal u is a sequence of finite alphabet data, typically binary. Error correcting coding (or encoding) is applied to u to introduce adequate redundancies so that possible transmission errors can be corrected at the receiver end. The coded signal c is modulated before being transmitted to the channel. The received signal r is first demodulated and its output y is then decoded to give $\hat{u}$, an estimate of u. The source signal u can be either a finite sequence or an infinite sequence. In the former case, the error correcting code (or code for short) is called a block code. Both turbo codes and LDPC codes are block codes, although the block size is typically very large (e.g., thousands to hundreds of millions bits) in order to achieve a good performance.

A turbo code is illustrated in Figure 3. It is a binary code, consisting of two constituent encoders $G_1(z)$ and $G_2(z)$, an interleaver (or called permutator), and an optional punctuator. The two constituent encoders are typically the same and will be denoted by $G(z)$. In a turbo code, $G(z)$ is a recursive

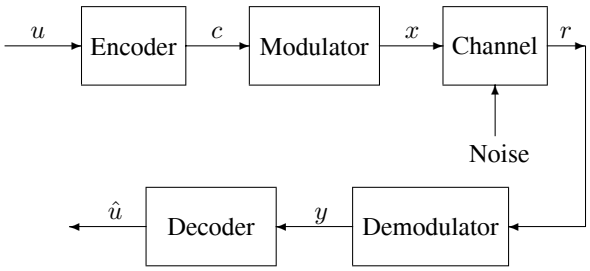

Fig. 2. Typical Digital Communications System

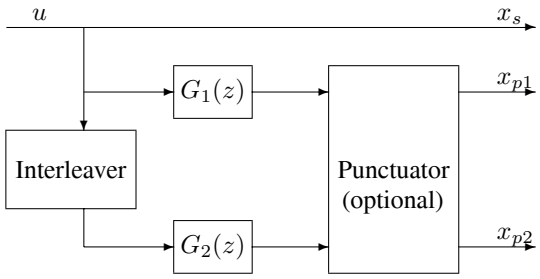

Fig. 3. Turbo Code

convolutional code, i.e., $G(z)$ is a rational function of a given degree. The so-called systematic codes are used in a constituent encoder. This means that the source signal u is a part of the coding output denoted by x_s and that the extra data from the encoder form the so-called a parity sequence, which is denoted by x_{p1} and x_{p2} for the two constituent encoders, respectively. The two constituent encoders share the same systematic sequence x_s. The primary role of the interleaver is the randomize the source sequence so that the two parity sequences x_{p1} and x_{p2} have as little correlation as possible. The punctuator is used to reduce the data size of the parity sequences. Without the punctuator, the coding rate is 1/3 because of the two parity sequences. If this rate is too low, the parity sequences can be punctured to give a higher rate. For example, by puncturing out the even symbols of x_{p1} and odd symbols of x_{p2}, the coding rate is increased to 1/2. By convention in digital communications, we assume that the binary 0 and 1 are mapped to physical values of $+1$ and -1 by the modulator.

Figure 4 shows the schematic diagram for an constituent encoder. In this example, $G(z) = N(z)/D(z)$ with $D(z) = 1 + z^{-1} + z^{-3}$ and $N(z) = 1 + z^{-2} + z^{-3}$. Since only binary coefficients are used, it is common to use the octal representation of the coefficients of $G(z)$. For this example, the binary coefficient sequences for $D(z)$ and $N(z)$ are 1101 and 1011, respectively. Their octal representations are 15 and 13. Therefore, it is common to denote $G(z) = (15, 13)$.

Turbo code design is a big research area by itself. For our purposes, it suffices to understand some basic rules and features of turbo code design. A turbo code is a linear code in the sense that the sum (i.e., exclusive-or) of any two codes is still a codeword. For a linear code, analysis can be done by assuming that the all-zero codeword is transmitted. Then,

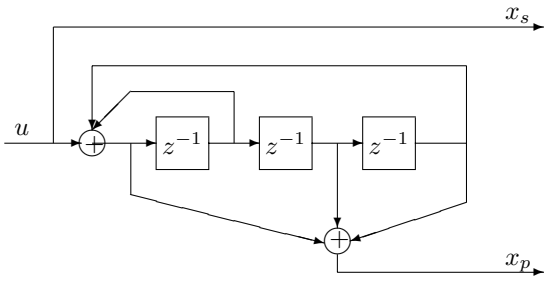

Fig. 4. Systematic Recursive Convolutional Encoder with $G(z) = (15, 13)$

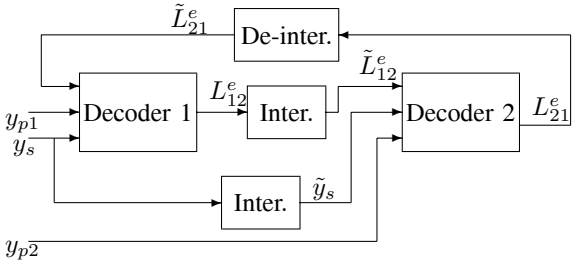

Fig. 5. Turbo Decoding

the probability that this codeword will be mistaken by the receiver as another codeword is related to the weight of the later. The weight of a codeword is the number of 1's in the codeword. A low weight code is "bad" because it can be easily mistaken as the all-zero code (the transmitted one). The quality of a code is measured by the distribution of the weights of the codewords. A good linear code should have very few low weight codewords. Turbo codes achieve this property by using recursive constituent encoders and an interleaver [12], [14].

We now try to understand the design of constituent encoders. A codeword consists of three sequences, x_s, x_{p1} and x_{p2}. Recall that a "bad" codeword has a low weight, i.e. it is such that x_s, x_{p1} and x_{p2} all must have low weights. We first consider the case x_s has weight equal to 1. We note that x_{p1} cannot have a low weight unless the non-zero element in x_s is close to the tail. This is because $G(z)$ is recursive. More specifically, if x_s is weight-1, then x_{p1} is a non-terminating sequence. So it must have a high weight, unless the non-zero element in x_s starts very late. If the non-zero element in x_s is close to the tail, it is mostly likely not close to the tail after interleaving. This implies x_{p2} is mostly likely a high weight sequence. The upshot of the above analysis is that a weight-1 x_s is unlikely to produce a low-weight codeword. Secondly, we consider the case of weight-2 x_s. If a weight-2 x_s is such that $G(z)x_s$ has low weight, then the interleaved version, $\tilde{x}_s$, has a very small probability to give $x_{p2} = G(z)\tilde{x}_s$ a low weight. Finally, it can be argued that for higher weight x_s, the probability that both x_{p1} and x_{p2} have a low weight is much smaller than the weight-2 case. Furthermore, the probabilities for a weight-1 and weight-2 x_s to give a low weight code diminishes to zero as the block size increases to infinity. This is a powerful observation and it also suggests that turbo codes are most effective when the block size is large.

We see from the above that the interleaver plays a crucial role in turbo code design. For a small to median block size, the interleaver needs to be carefully designed to ensure that the minimum weight of the code is maximized. For a large block size, the choice of the interleaver is not crucial, as long as it provides sufficient randomization. Consequently, it usually suffices to use a pseudo-random interleaver.

III. Iterative Decoding

A key for turbo codes to achieving excellent performances is the use of iterative decoding algorithms. In this section, we introduce the iterative decoding algorithm used in turbo codes. This algorithm is also referred to as the turbo decoding algorithm.

The turbo decoding algorithm is depicted in Figure 5. The input data to the turbo decoder are y_s, y_{p1} and y_{p2}, which are the the noisy version of x_s and x_{p1} or x_{p2}, coming from demodulation. Two constituent decoders, Decoder 1 and Decoder 2, are at work. In each iteration of turbo decoding, either Decoder 1 or Decoder 2 is employed and they take turns. Decoder 1 takes the relevant parts of the demodulated signal, i.e., y_s and y_{p1} as inputs. In addition, some prior information regarding u, denoted by $\tilde{L}^e_{21}$, is also taken as an input. This prior information is called the *extrinsic information* and it is void for the first iteration or comes from Decoder 2 through de-interleaving for other iterations. Decoder 2 works the same as as Decoder 1, except that y_s and L^e_{12} (the extrinsic information about u from Decoder 1) are interleaved before being used.

Each constituent decoder produces an produces a *soft estimate* of u based on the given inputs. Traditional decoding processes typically apply a maximum likelihood algorithm such as the Viterbi algorithm to produce an optimal estimate of u which has the highest probability to generate the given inputs. This decoding process is referred to as *hard decoding*. Such an approach is inappropriate for turbo decoding because it does not yield sufficient information for the proceeding iterations. Instead of using hard decoding, the so-called *soft decoding* is used. The result is not an optimal estimate of u. Instead, it gives a *soft estimate*, which is nothing but the *a posteriori* probability (APP) value in $[0,\ 1]$, for each bit of u to be 1 or 0. If the current iteration is the final one, a *hard estimate* of u can be obtained by rounding the APP value. If more iterations are required, the extrinsic information is generated using the soft estimate. This process continues until some stopping criterion is met.

The output extrinsic information generated by each constituent decoder is simply obtained from the APP value by deducting the input extrinsic information. In other words, the output extrinsic information represents the additional information given by the associated constituent decoder.

The soft decoding algorithm used in turbo decoding is

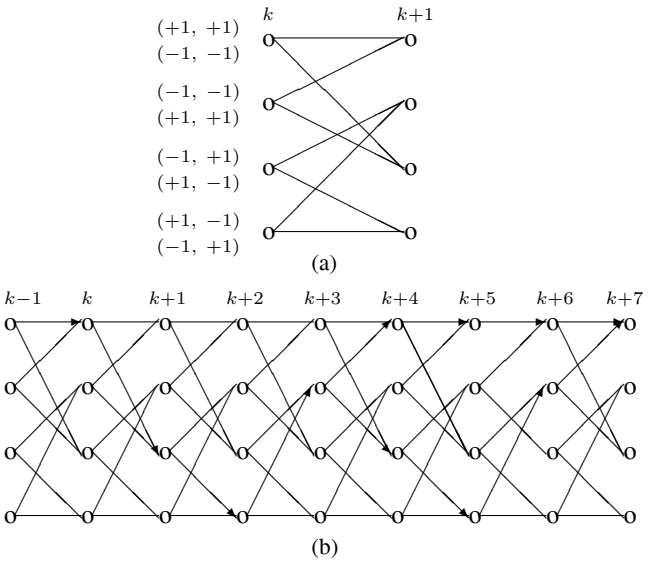

Fig. 6. Trellis Diagram for $G(z) = (7, 5)$: (a) From k to $k + 1$; (b) From $k - 1$ to $k + 7$

the so-called *maximum a posterior* (MAP) algorithm. This algorithm is also called BCJR algorithm, named after the four authors who first gave the algorithm in early 1970s [29]. In comparison with the well-known Viterbi algorithm, the BCJR algorithm is twice as complex computationally, but it outputs APP values (soft estimates) instead of an optimal estimate (a hard estimate). Therefore, it suits well for iterative decoding purposes.

The MAP algorithm uses the state transition diagram, also known as the trellis diagram, of the constituent encoder. Figure 6 illustrates the trellis diagram for $G(z) = (7, 5)$. In this example, $D(z) = 1 + z^{-1} + z^{-2}$ and $N(z) = 1 + z^{-2}$. Therefore, the order of $G(z)$ is 2 and there are 4 possible states at each symbol time, as shown in Figure 6(a). For each possible state, there are two possible branches depending on the input symbol. The brackets on the left of the trellis indicate the systematic and parity bits corresponding to each branch of the trellis.

We now describe the MAP algorithm [29], following the tutorial paper [30]. Note that this algorithm is applied to each constituent decoder. Without loss of generality, we consider Decoder 1 in Figure 5. From Figure 5, we see that the inputs include y_s, y_{p1} and the extrinsic information. To aid the description of the MAP algorithm, we denote by n the block size of the code and introduce the following notation:

$$y_k = (y_{s,k}, y_{1p,k}); \quad y_i^j = (y_i, y_2, \ldots, y_j); \quad y = y_1^n$$

where $y_{s,k}$ is the k-th element of y_s, and $y_{1p,k}$ is similarly defined.

The MAP algorithm is concerned with computing the so-called *a posteriori* probability (APP) ratio of u, as defined below:

$$R(u_k) = \frac{P(u_k = +1|y)}{P(u_k = -1|y)} \tag{1}$$

where u_k is the k-th symbol of u. Since this ratio ranges from 0 to ∞, it tends to cause numerical underflows and overflows easily. For this reason, it is common to consider the log *a posteriori* probability (LAPP) ratio below, also known as the *log-likelihood ratio* (LLR):

$$L(u_k) = \log\left(\frac{P(u_k = +1|y)}{P(u_k = -1|y)}\right) \tag{2}$$

The associated decoding algorithm is often referred to as the *Log-MAP algorithm*.

It is clear that both of the *a posteriori* probabilities $P(u_k = +1|y)$ and $P(u_k = -1|y)$ can be computed from $L(u_k)$ because of the constraint

$$P(u_k = +1|y) + P(u_k = -1|y) = 1$$

The computation of $L(u_k)$ relies on the trellis diagram of the constituent encoder. We consider the transition of the state of the code from $(k-1)$-th symbol to the k-th symbol. Denoting the set of all possible states by S and the state at the k-th symbol by s_k, we can partition S into two subsets at each k: S^+ (respectively, S^-) is the set of all state transitions from s_{k-1} to s_k caused by $u_k = +1$ (respectively, $u_k = -1$). Now using the Bayes rule, we have

$$L(u_k) = \log\left(\frac{\sum_{S^+} P(s_{k-1} = s', s_k = s, y)/P(y)}{\sum_{S^-} P(s_{k-1} = s', s_k = s, y)/P(y)}\right) \tag{3}$$

It is clear from above that $P(y)$ can be canceled and we only need to find a way for computing $P(s_{k-1} = s', s_k = s, y)$, or $P(s', s, y)$ for short. By breaking y into $(y_1^{k-1}, y_k, y_{k+1}^n)$ and applying the Bayes rule again, we can write

$$P(s', s, y) = \alpha_{k-1}(s')\gamma_k(s', s)\beta_k(s) \tag{4}$$

where

$$\alpha_{k-1}(s') = P(s_{k-1} = s', y_1^{k-1}); \tag{5}$$

$$\gamma_k(s', s) = P(s_k = s, y_k|s_{k-1} = s'); \tag{6}$$

$$\beta_k(s) = P(y_{k+1}^n|s_k = s) \tag{7}$$

These terms can be computed recursively using the Bayes rule again. More precisely,

$$\alpha_k(s) = \sum_{s' \in S} \alpha_{k-1}(s')\gamma_k(s', s) \tag{8}$$

with the initial conditions

$$\alpha_0(s = 0) = 1; \quad \alpha_0(s \neq 0) = 0$$

where $s = 0$ is the known initial state for the code. Similarly,

$$\beta_{k-1}(s) = \sum_{s \in S} \beta_k(s)\gamma_k(s', s) \tag{9}$$

with the terminating conditions

$$\beta_n(s = 0) = 1; \quad \beta_n(s \neq 0) = 0$$

if $s = 0$ is the known terminating state for the code. If the code is not terminated, $\beta_n(s)$ is usually set equally.

It remains to compute $\gamma_k(s', s)$, for which we have

$$\gamma_k(s', s) = P(s|s')P(y_k|s', s) = P_a(u_k)P(y_k|u_k, x_{p1,k}) \tag{10}$$

where the values of u_k and $x_{p1,k}$ correspond to the transition from s' to s. The term $P_a(u_k)$ is the *a priori* probability of u_k which is related to the extrinsic information $\tilde{L}^e_{21}$ as follows:

$$\tilde{L}^e_{21,k} = \log\left(\frac{P_a(u_k=+1)}{P_a(u_k=-1)}\right) \tag{11}$$

i.e., the k-th element of $\tilde{L}^e_{21}$ is the log *a priori* probability ratio for u_k.

The term $P(y_k|u_k, x_{p1,k})$ in (10) is the conditional probability of y_k knowing the state transition. This depends on the channel characteristics and is assumed to be known for decoding. For example, if the channel is an additive Gaussian white noise (AGWN) channel with noise variance σ^2, then $y_{s,k}$ and $y_{p1,k}$ are independent and we have

$$\begin{aligned} & P(y_k|u_k, x_{p1,k}) \\ = \; & P(y_{s,k}|u_k)P(y_{p1,k}|x_{p1,k}) \\ = \; & C\exp\left[-\frac{(y_{s,k}-u_k)^2}{2\sigma^2}\right]\exp\left[-\frac{(y_{p1,k}-x_{p1,k})^2}{2\sigma^2}\right] \end{aligned} \tag{12}$$

with a constant C which does not affect $L(u_k)$.

From the above, we can see that $L(u_k)$ can be computed by first calculating all $\gamma_k(s', s)$ using (10)-(12), then recursively calculating all $\alpha_k(s)$ and $\beta_k(s)$ using (8)-(9) and finally calculating $L(u_k)$ using (3)-(4).

If we want to produce a hard estimate of u_k, we simply take

$$\hat{u}_k = \text{sign}(L(u_k)) \tag{13}$$

If we want to compute the extrinsic information L^e_{12} for further iterations, we simply subtract the input extrinsic information from $L(u_k)$, i.e.,

$$L^e_{12,k} = L(u_k) - \tilde{L}^e_{21,k} \tag{14}$$

We remark that the actual implementation of the MAP algorithm is usually different from what has been described above for numerical reasons; see [30].

IV. Stochastic Modeling

The precise reason for the power of iterative decoding algorithms remains one of the mysteries in communications. Much research has been devoted to this problem [17]-[24]. There are two approaches: deterministic and stochastic. The deterministic approach [17]-[20] treats decoding as a deterministic process and aims to characterize the behavior of the decoder for each instance of the received signal. It can be used to analyze the existence, stability, and uniqueness of fixed points. However, this approach is mostly qualitative and fails to explain the behavior of specific codes and decoders. 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. Notable examples of the stochastic approach include [21]-[24], which model the decoder in every iteration as single-input-single-output (SISO) with just one statistical parameter. This parameter is either mutual information or the signal-to-noise ratio (SNR). Using SISO models, [21]-[24] are able to explain a number of important features of turbo decoding, including the well-known step-like (waterfall/error floor) performance of turbo decoding and fixed points at low SNR. In particular, the so-called Extrinsic Information Transfer (EXIT) chart of [24] is found to be particularly useful in understanding and quantifying the dynamics of turbo decoding. However, the SISO model also suffers from a number of drawbacks. In particular, it fails to model the probability distribution of the input-output signals properly and it does not provide a good prediction of the dynamic behavior of the iterative process [26].

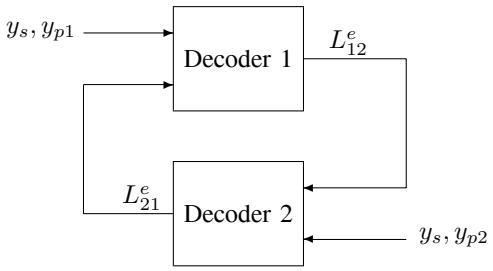

Fig. 7. Feedback View of Turbo Decoding

We present a more rigorous approach to stochastic modeling of turbo decoding based on the systems theory. Indeed, we can actually redraw the turbo decoding process in Figure 5 as a feedback loop in Figure 7 by absorbing the interleavers and de-interleaver into the constituent decoders. This feedback view automatically prompts the need for a suitable simple model for each constituent decoder that can capture the dynamic behavior of the loop.

We first show that, when the received signal is subject to additive Gaussian white noise (AGWN) and the interleaver is chosen randomly, the turbo decoding output for each iteration approaches an ergodic random process when the block size is very large. We then show that decoding output for each iteration, when expressed using a scaled logarithmic likelihood ratio (SLLR), is well approximated using a Gaussian distribution. 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). Using this model, we have discovered that a typical decoding process is much more intricate than previously known, involving two regions of attractions, several fixed points, and a stable equilibrium manifold at which all decoding trajectories converge.

V. Ergodic Properties

A. Scaled Log-likelihood Ratios

We first 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 $\mathbf{S}$, is defined as

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

When a (received) signal r is subject to AGWN, it can be shown that its SLLR has a Gaussian distribution with mean and variance satisfying a unique relationship: $\mu_r = \sigma_r^2$.

However, as we will see later, the relationship above no longer holds for extrinsic signals (and the *a prior* signals in later iterations). Because of this, we introduce the notion of *mean-to-variance ratio* (MVR):$d = \mu/\sigma^2$.

B. Log-MAP Decoding

As we explained earlier, the Log-MAP decoding algorithm [12] takes the extrinsic information (*a priori* information) and a received signal, and produces an extrinsic signal and an *a posteriori* signal. For this section and onwards, we change the notation and denote these four signals by a, r, e and d, respectively. Their SLLR expressions are denoted by $\mathbf{A}$, $\mathbf{R}$, $\mathbf{E}$ and $\mathbf{D}$ respectively. We also denote the mean and variance of $\mathbf{A}$ by μ_a and σ_a^2, etc.

Without loss of generality, we assume that u is an all-one sequence (i.e., all-zero in binary). Using the SLLR expressions, it is shown in [26] that $\mathbf{E}$ can be written in the following form:

$$\begin{aligned} \mathbf{E} &= \ln\left(1 + \sum_{t=1}^{T_1} \exp(-C_t^{(1)}\mathbf{A} - C_t^{(2)}\mathbf{R})\right) \\ &\quad - \ln\left(\sum_{t=1}^{T_2} \exp(-C_t^{(3)}\mathbf{A} - C_t^{(4)}\mathbf{R})\right), \end{aligned} \quad (15)$$

where $C_t^{(i)}$ are row vectors with 0's and 1's and $T_1, T_2 \geq 0$. Following from (14), we also have

$$\mathbf{D} = \mathbf{A} + \mathbf{E}$$

C. Asymptotic Behavior of Log-MAP Decoding

From the analysis above, it is clear that the output signal of MAP decoding can be modelled as a random process. The key question we now ask is how to model this random process when the code block size is very large.

It follows that the input and output signals in each decoding iteration are random processes. In order to be able to model the decoding process using stochastic parameters, it is necessary that these random processes are ergodic. It is somewhat surprising to realize that many stochastic models for decoding processes (including many listed in the reference list) are established without a formal study of ergodicity. To illustrate the danger of not checking the ergodicity, we point out the fact that the ergodicity of a random process can be easily destroyed after common linear operations on the signal. These operations include down-sampling, up-sampling, addition and linear filtering. See details in [27]. Therefore, it is important that ergodicity of signals in a turbo decoder is analyzed.

Our first main result on turbo decoding is given below.

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

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

D. Stochastic Modelling of Turbo Decoding

We now want to generalize the ergodicity result in Theorem 1 to turbo decoding. Again, we assume $n \to \infty$.

From the above analysis of Log-MAP decoding, we understand that if the received signal is subject to AGWN and the SLLR of the *a priori* signal, $\mathbf{A}$, is an independent ergodic random process, then the SLLR of the extrinsic signal, $\mathbf{E}$, is also ergodic. In turbo decoding, we start with $\mathbf{A} = 0$, which is Gaussian with $\mu_a = \sigma_a = 0$. 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 because the extrinsic signal is "locally" correlated. It is easy to imagine that a non-stationary $\mathbf{A}$ is possible if a "bad" interleaver is used.

Fortunately, the correlation in $\mathbf{E}$ 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 $\mathbf{E}_k$ depends only on those $\mathbf{A}_i$ 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:

$$S_T = \min_S \{S : 1 \leq i, j \leq n; |i - j| \leq S \Rightarrow |T_i - T_j| > S\}$$

Lemma 1: Given any $S > 0$, if an interleaver T of size n is chosen randomly, then

$$P(S_T \geq S) \to 1, \quad \text{as } n \to \infty. \quad (16)$$

The lemma above leads us to our second main result.

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 $\mathbf{E}(\ell, n)$ the SLLR of the extrinsic signal from the ℓ-th iteration of Log-MAP decoding. Then, for any $\ell \geq 1$, $\mathbf{E}(\ell, n)$ approaches an ergodic random process as $n \to \infty$.

In the above, the number of iterations refers to the number of Log-MAP decoding processes, rather than the number of turbo cycles.

To demonstrate the ergodicity of $\mu_e(\ell, n)$, we simulate a $1/3$-rate turbo code with $G(z) = (7, 5)$, $E_b/N_0 = 0.5$dB and pseudo-random 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 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 the average of

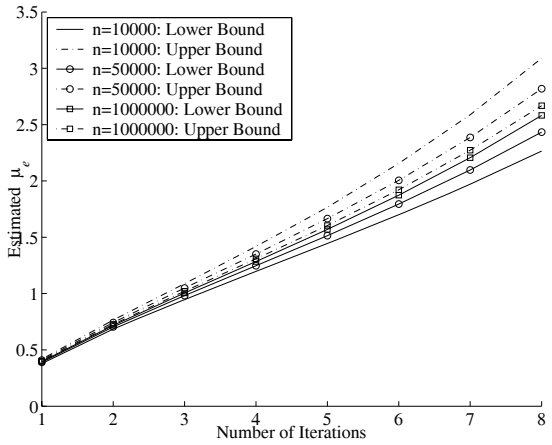

Fig. 8. Convergence of the means of $\mathbf{E}(\ell, n)$

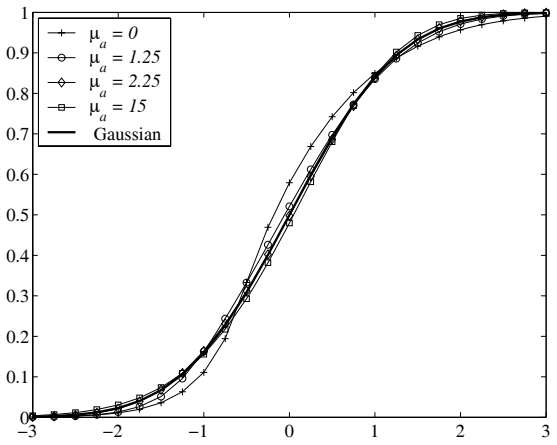

Fig. 9. Normalized CDF of Extrinsic Signal (SLLR)

these $\mu_e(\ell, n)$ values plus their standard deviation. The size of the gap between the lower and upper bound shows how well $\mu_e(\ell, n)$ converges as $n \to \infty$. The simulation results are shown in Fig. 8. It is clear that the gap between the lower and upper bound curves converge as n becomes larger.

We note that a somewhat different version of the ergodicity result can be found in [20].

VI. Gaussian Approximations

In this section, we study Gaussian approximations for Log-MAP decoding and turbo decoding.

A. 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 $\exp(Z)$ is a sum of lognormal distributions (SLND). We will call Z a log-sum of lognormal distributions (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. Correspondingly, Z is well approximated by a Gaussian distribution. Although no closed-form description is given on the distribution of a SLND or LSLND, a number of methods are available for computing the mean and variance (or equivalently the first and second moments) of Z; see, e.g., [25] for a summary. The Gaussian approximation works well when X_i are weakly correlated and their statistical parameters are not significantly different.

B. Gaussian Approximations for Log-MAP Decoding

We now analyze the distribution of $\mathbf{E}$ (the SLLR of the extrinsic signal) for Log-Map decoding. Consider the expression $\mathbf{E}_k$ from (15). Recall that $\mathbf{R}$ is a vector of (independent) Gaussian distributions when the received signal r is subject to AGWN. Suppose $\mathbf{A}$ is also Gaussian distributed. Then, $\mathbf{E}$ is the difference between the two LSLNDs. This observation is summarized below:

For a convolutional binary code with an infinite block size, if the received signal is subject to AGWN 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 result above 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. 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. 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 $G(z) = (7, 5)$ (in octal). The received signal is subject to AWGN with $E_b/N_0 = -1.2609$dB.

Fig. 9 shows the normalized CDFs of $\mathbf{E}$ for different values of μ_a but with $d_a = 1$. It is observed that for very low values of μ_a, $\mathbf{E}$ 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. 10 plots μ_e vs. 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, which is seriously affected by the MVR, is critical in determining the convergence of turbo decoding.

C. Gaussian Approximations for Turbo Decoding

From the analysis of Log-MAP decoding, we understand that if $\mathbf{A}$ and $\mathbf{R}$ are independent Gaussian white noises, then $\mathbf{E}$ is well approximated using a stationary process with

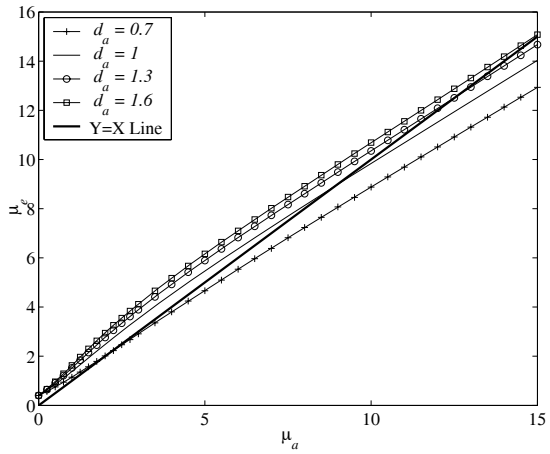

Fig. 10. Mean of Extrinsic Signal (SLLR)

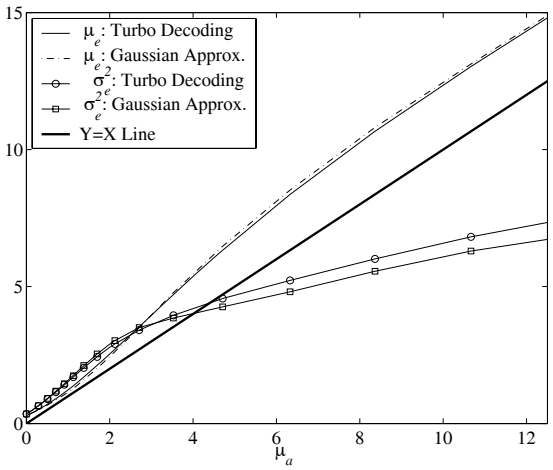

Fig. 11. Validity of Gaussian Approximation

a Gaussian distribution. In turbo decoding, we start with $\mathbf{A} = 0$. Therefore, $\mathbf{E}$ from the first iteration is well approximated using a Gaussian distribution. If a random interleaver is used, $\mathbf{A}$ for the next iteration will become effectively independent when the block size is large. Hence, Gaussian approximations can continue, i.e., $\mathbf{E}$ 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 (locally correlated) Gaussian distribution.*

To check the validity of the Gaussian approximation model, we compare it to turbo decoding using a $1/3$-rate turbo code with $G(z) = (13, 15)$, $n = 500,000$, and a pseudo-random interleaver. We take $E_b/N_0 = 0.3$dB. Twelve iterations are used. For the method using the Gaussian approximation model, $\mathbf{A}$ in each iteration is chosen to be a Gaussian distribution with the same mean and variance as those for the $\mathbf{A}$ fed into the corresponding iteration of the turbo decoding. Fig. 11 compares the means and variances of $\mathbf{E}$ 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 due to the fact that Gaussian approximations are slightly skewed when μ_a is either small or very large, as we pointed out earlier.

VII. Dynamics of Turbo Decoding

It is known [18] 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.

These behaviors, however, are the consequences of having a finite code block size n. Recall that when $n \to \infty$,

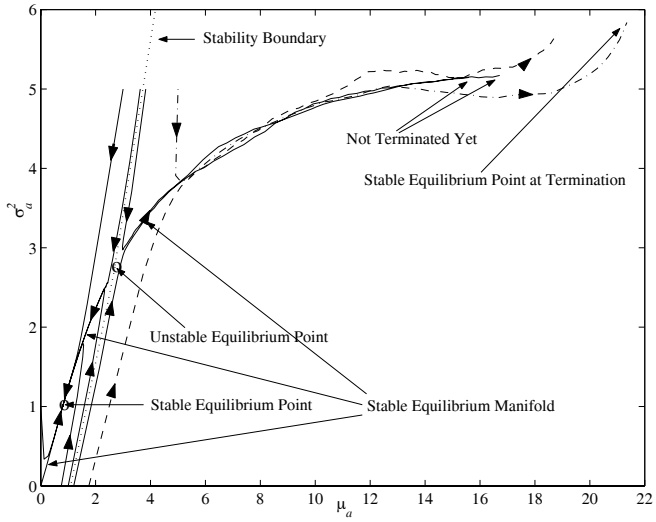

Fig. 12. Dynamics of Turbo Decoding

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). As $n \to \infty$, the state of the decoded signal either converges at a finite *stable fixed point* or diverges. 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 can not 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 SNR point.

Turbo dynamics is in fact much more complex than indicated by the two stable fixed points. The complete picture is illustrated in Fig. 12. The turbo code used here is a 1/3-rate code with $G(z) = (7, 5)$, $E_b/N_0 = -0.1$dB, $n = 10^6$, and a pseudo-random interleaver. We see from the figure that there is a *stable equilibrium manifold* at which every turbo decoding trajectory converges, regardless of the initial

point. On this 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 a *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 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. 12 happens when 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, the decoding trajectory always converges at a stable fixed point or region with a high SNR.

VIII. Conclusions

In this presentation, we have discussed a systems theory approach to stochastic 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. We expect that this model is useful in understanding and improving turbo decoding. Indeed, it has been illustrated in [26] about how to devise decoding algorithm faster than the MAP algorithm.

References

[1] W. S. Wong and R. W. Brockett, "Systems with finite communication bandwidth constraints II: stabilization with limited information feedback," *IEEE Trans. Automatic Control*, vol. 44, no. 5, pp. 1049-1053, May 1999.

[2] J. Baillieul, "Feedback designs in information-based control," *Stochastic Theory and Control Workshop*, Kansas, pp. 35-57, Springer, 2001.

[3] R. W. Brockett and D. Liberzon, "Quantized feedback stabilization of linear systems," *IEEE Trans. Automatic Control*, vol. 45, no. 7, pp. 1279-1289, July 2000.

[4] G. N. Nair and R. J. Evans, "Exponential stabilization of multidimensional linear systems," *Automatica*, vol. 39, pp. 585-593, 2003.

[5] N. Elia and K. Mitter, "Stabilization of linear systems with limited information," *IEEE Trans. Automatic Control*, vol. 46, no. 9, pp. 1384-1400, 2001.

[6] S. Tatikonda and S. Mitter, "Control over noisy channels," *IEEE Trans. Automatic Control*, vol. 49, no. 7, pp. 1196-1201, July 2004.

[7] M. Fu and L. Xie, "The sector bound approach to quantized feedback control," to appear in *IEEE Trans. Automatic Control*; also in *Proc. American Control Conference*, Denver, June 2003.

[8] M. Fu and L. Xie, "Performance Control of Linear Systems using Quantized Feedback," *Proc. 4th International Conference on Control and Automation*, Montreal, June 2003.

[9] M. Fu, "Robust stabilization of linear uncertain systems via quantized feedback," *Proc. IEEE Conf. Decision and Control*, 2003.

[10] A. Sahai, "The necessity and sufficiency of anytime capacity for control over a noisy communication link," *Proc. IEEE Conf. Decision and Control*, Bahamas, 2004.

[11] M. Fu and S. Hara, "Quantized Feedback Control for Sampled-Data Systems," *IFAC World Congress*, Spain, July 2005.

[12] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo-codes," in *Proc. Int. Conf. Commun.*, Geneva, Switzerland, pp. 10641070, May 1993.

[13] R. J. McEliece, D.J. C. MacKay and J.-F. Cheng, "Turbo decoding as an instance of Pearl's 'belief propagation' algorithm," *IEEE J. Selected Areas in Commun.*, vol. 16, no. 2, pp. 148-152, 1998.

[14] D. J. C. MacKay and R. M. Neal, "Near Shannon limit performance of low-density parity-check codes," *Electron. Lett.*, vol. 32, pp. 1645-1646, Aug. 1996.

[15] X. Wang and H. V. Poor, "Iterative (Turbo) soft interference cancellation and decoding for coded CDMA," *IEEE Trans. Commun.*, vol. 47, pp. 1046-1061, July 1999.

[16] A. Grant and C. Schlegel, "Iterative implementations for linear multiuser detectors," *IEEE Trans. Commun.*, vol. 49, no. 10, pp. 1824-1834, Oct. 2001.

[17] T. Richardson, "The geometry of turbo-decoding dynamics," *IEEE Trans. Inform. Theory*, vol. 46, no. 1, pp. 923, 2000.

[18] 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, 2001.

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

[20] T. J. Richardson and R. L. Urbanke, *Modern Coding Theory*, http://lthcwww.epfl.ch/papers/ics.ps.

[21] S. ten Brink, "Convergence behavior of iteratively decoded parallel concatenated codes," *IEEE Trans. Commun.*, vol. 49, no. 10, pp. 17271737, 2001.

[22] D. Divsalar, S. Dolinar, and F. Pollarar, "Iterative turbo decoder analysis based on density evolution," *IEEE J. Selected Areas in Commun.*, vol. 19, no. 5, pp. 891907, 2001.

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

[24] J. W. Lee and R. E. Blahut, "Generalized EXIT chart and BER analysis of finite-length turbo codes," *Proc. GlobeCom 2003*, San Fransisco, Dec. 2003.

[25] N. C. Baulieu, A. Abu-Dayya and P. J. McLane, "Estimating the distribution of independent lognormal random variables," *IEEE Trans. Commun.*, vol. 43, no. 12, pp. 2869-2873, 1995.

[26] M. Fu, "Stochastic analysis of turbo decoding," *IEEE Trans. Info. Theory*, vol. 54, no. 1, pp. 81-100, 2005. Also to appear in *Int. Conf. Commun.*, Seoul, May 2005.

[27] D. Marelli and M. Fu, "Ergodic properties for multirate linear systems," *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Proc.*, Montreal, May 2004.

[28] C. E. Shannon, "A Mathematical Theory of Communication," *Bell Syst. Tech. J.*, vol. 27, pp. 379-423, July and pp. 623-656, October, 1948.

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

[30] W. Ryan, "A turbo code tutorial," available at: `http://www.ece.arizona.edu/ ryan`.