Fundamentals of Turbo Codes

E Sowmya Sudha (03010215), Y. Praveen Kumar (03010240), Kanchan Mishra (03010242)
Department of Electronics and Communication Engineering
Indian Institute of Technology, Guwahati (India)

{esudha, ypkumar, kanchan}@iitg.ernet.in

Abstract — **This paper presents fundamentals of a family of convolutional codes, nicknamed turbo-codes, built from a particular concatenation of two recursive systematic codes, linked together by non uniform interleaving. Decoding calls on iterative processing in which each component decoder takes advantage of the work of the other at the previous step, with the aid of the original concept of extrinsic information. For sufficiently large interleaving sizes, the correcting performance of turbo-codes,investigated by simulation, appears to be close to the theoretical limit predicted by Shannon.**

I. Introduction

Convolutional error correcting or channel coding has become widespread in the design of digital transmission systems. One major reason for this is the possibility of achieving real-time decoding without noticeable information losses thanks to the well-known soft-input Viterbi algorithm [I].Moreover, the same decoder may serve for various coding rates by means of puncturing [2]. Two kinds of convolutional codes are of practical interest: nonsystematic convolutional (NSC) and recursive systematic convolutional (RSC) codes. Though RSC codes have the same free distance d_f as NSC codes and exhibit better performance at low signal to noise ratios (SNR's) and/or when punctured, only NSC codes have actually been considered for channel coding, except in Trellis-coded modulation (TCM) [3]. For a given rate, the error-correcting power of convolutional codes, measured as the coding gain at a certain binary error rate (BER) in comparison with the uncoded transmission, grows more or less linearly with code memory ν**.** Unfortunately, the complexity of the decoder is not a linear function of ν and it grows exponentially as $\nu.2^{\nu}$.

In order to obtain high coding gains with moderate decoding complexity, concatenation has proved to be an attractive scheme. Concatenated coding schemes were first proposed by Forney [4] as a method for achieving large coding gains by combining two or more relatively simple building blocks or component codes (sometimes called constituent codes). The resulting codes had the error-correction capability of much longer codes, and they were endowed with a structure that permitted relatively easy to moderately complex decoding. A serial concatenation of codes is most often used for power-limited systems such as transmitters on deep-space probes. The most popular of these schemes consists of a Reed-Solomon outer (applied first, removed last) code followed by a convolutional inner (applied last, removed first) code [5].

A turbo code can be thought of as a refinement of the concatenated encoding structure plus an iterative algorithm for decoding the associated code sequence. Turbo codes were first introduced in 1993 by Berrou, Glavieux, and Thitimajshima, and reported in [6, 7], where a scheme is described that achieves a bit-error probability of 10-5 using a rate 1/2 code over an additive white Gaussian noise (AWGN) channel and BPSK modulation at an E_b/N_0 of 0.7 dB.

Section II presents the principle of RSC codes, which are at the root of the study of turbo-codes. Section III describes the construction of turbo-codes. The codes are constructed by using two or more component codes on different interleaved versions of the same information sequence. The decoding of turbo-codes is described in Section IV. Whereas, for conventional codes, the final step at the decoder yields hard-decision decoded bits (or, more generally, decoded symbols), for a concatenated scheme such as a turbo code to work properly, the decoding algorithm should not limit itself to passing hard decisions among the decoders. To best exploit the information learned from each decoder, the decoding algorithm must effect an exchange of soft decisions rather than hard decisions. For a system with two component codes, the concept behind turbo decoding is to pass soft decisions from the output of one decoder to the input of the other decoder, and to iterate this process several times so as to produce more reliable decisions. We thus describe the soft or extrinsic information from the decoder in Section V describes the error performance of turbo-codes. Some basic results are given before concluding in Section VI.

II. Recursive Systematic Convolutional Codes

Consider a binary rate R = 1/2 convolutional encoder with constraint length K and memory ν = K - 1. The input to the encoder at time k is a bit d_k and the corresponding binary couple (X_k , Y_k) is equal to

$$X_k = \sum_{i=0}^{\nu} g_{1i} d_{k-i} \qquad g_{1i} = 0, 1 \qquad \text{(1a)}$$

$$Y_k = \sum_{i=0}^{\nu} g_{2i} d_{k-i} \qquad g_{2i} = 0, 1 \qquad \text{(1b)}$$

The RSC code, presented below, combines the properties of NSC and systematic codes. In particular, it can be better than the equivalent NSC code, at any SNR, for code rates larger than 2/3. **A** binary rate RSC code is obtained from a NSC code by using a feedback loop and setting one of the two outputs X_k or Y_k equal to the input bit d_k . The shift register (memory) input is a new binary

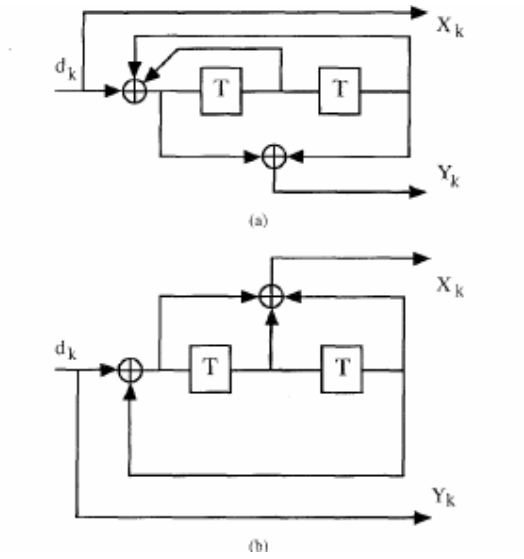

Fig. 1 Two associated Recursive systematic convolutional encoders with memory v=2, rate R=1/2 and generators G1=7, G2=5.

variable a_k .calculated recursively as

$$a_k = d_k + \sum_{i=1}^{\nu} \gamma_i a_{k-i} \tag{2}$$

where γ_i is respectively equal to $g_{1\,i}$ if $X_k = d_k$ and to g_{2i} if $Y_k = d_k$. Equation (2) can be rewritten as

$$d_k = \sum_{i=0}^{\nu} \gamma_i a_{k-i}. \tag{3}$$

Two RSC encoders with memory u = 2 and rate R = 1/2, obtained from a NSC encoder defined by generators G1 =7, G2 = 5 **,** are depicted in Fig. 1.

When puncturing is considered, some output bits X_k, or Y_k, are deleted according to a chosen perforation pattern defined by a matrix P . For instance, starting from a rate R = ½ code, the matrix *P* of rate 2/3 punctured code can be equal to

$$\boldsymbol{P} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}. \tag{4}$$

III. CONSTRUCTION OF TURBO CODES

Turbo-codes are constructed using parallel concatenation of RSC codes with non-uniform interleaving. The use of systematic codes enables the construction of a concatenated encoder in the form given in Fig. 3, called parallel concatenation. The data flow (d_k at time k) goes directly to a first elementary RSC encoder C_1 and after interleaving, it feeds (d_n at time k) a second elementary RSC encoder C,. These two

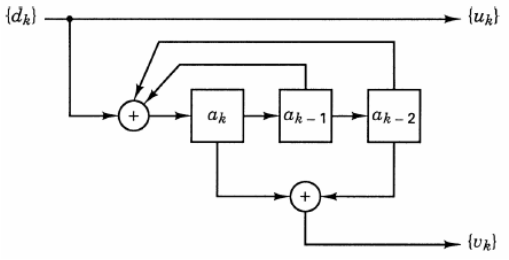

Fig. 2 A Simple Recursive systematic convolutional Encoder

encoders are not necessarily identical. Data d_k is systematically transmitted as symbol X_k: and redundancies Y_{1k} and Y_{2k} produced by C_1 and C_2 may be completely transmitted for an R = 1/3 encoding or punctured for higher rates. The two elementary coding rates R_1 and R_2 associated with C_1 and C_2, after puncturing, may be different, but for the best decoding performance, they will satisfy $R_1 \leq R_2$. The global rate R of the composite code, R_1 and R_2 are linked by (5).

$$\frac{1}{R} = \frac{1}{R_1} + \frac{1}{R_2} - 1. \tag{5}$$

Thus, the two rate 1/3 RSC encoders in Fig. 3 have been punctured to give rate ½ turbo-codes. Unlike the classical (serial) concatenation, parallel concatenation enables the elementary encoders, and therefore the associated elementary decoders, to run with the same clock. This point constitutes an important simplification for the design of the associated circuits, in a concatenated scheme. Good turbo codes have been constructed from component codes having short constraint lengths (K = 3 to 5). There is no limit to the number of encoders that may be concatenated, and in general the component codes need not be identical with regard to constraint length and rate.

The goal in designing turbo codes is to choose the best component codes by maximizing the effective free distance of the code [8]. At large values of E_b/N_0, this is tantamount to maximizing the minimum-weight codeword. However, at low values of E_b/N_0 (the region of greatest interest), optimizing the weight distribution of the code words is more important than maximizing the minimum-weight codeword [9]. The turbo encoder in Fig. 3 produces code words from each of two component encoders. The weight distribution for the code words out of this parallel concatenation depends on how the code words from one of the component encoders are combined with code words from the other encoder. Intuitively, we should avoid pairing low-weight code words from one encoder with low-weight code words from the other encoder. Many such pairings can be avoided by proper design of the interleaver. An interleaver that permutes

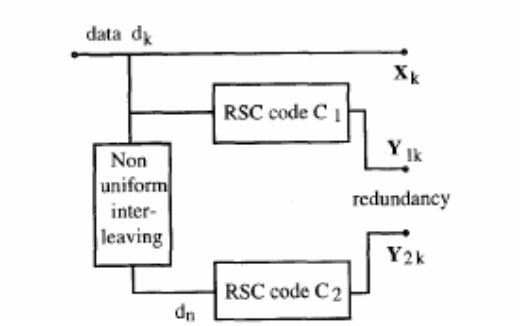

Fig. 3 Basic turbo-encoder (rate 1/3)

the data in a random fashion provides better performance than the familiar block interleaver . Such an interleaver is called the Non-uniform interleaver.

Non-uniform interleaving must satisfy two main conditions: the maximum scattering of data, as in usual interleaving, and the maximum disorder in the interleaved data sequence. The latter, which may be in conflict with the former, is to make redundancy generation by the two encoders as diverse as possible. In this case, if the decision by the decoder associated with C_1 about particular data implies a few items of redundancy Y_1, then the corresponding decision by the decoder associated with C_2 will rely on a large number of values Y_2, and vice-versa. Then, the minimum distance of the turbo-code may be increased to much larger values than that given by uniform interleaving.

If the component encoders are not recursive, the unit weight input sequence 0 0 … 0 0 1 0 0 … 0 0 will always generate a low-weight codeword at the input of a second encoder for any interleaver design. In other words, the interleaver would not influence the output-codeword weight distribution if the component codes were not recursive. However, if the component codes are recursive, a weight-1 input sequence generates an infinite impulse response (infinite-weight output). Therefore, for the case of recursive codes, the weight-1 input sequence does not yield the minimum-weight codeword out of the encoder. The encoded output weight is kept finite only by trellis termination, a process that forces the coded sequence to terminate in such a way that the encoder returns to the zero state. In effect, the convolutional code is converted to a block code. The important aspect of the building blocks used in turbo codes is that they are recursive (the systematic aspect is merely incidental). It is the RSC code's IIR property that protects against the generation of low-weight code words that cannot be remedied by an interleaver. One can argue that turbo code performance is largely influenced by minimum-weight code words that result from the weight-2 input sequence. The argument is that weight-1 inputs can be ignored, since they yield large codeword weights due to the IIR encoder structure. For input sequences having weight-3 and larger, a properly designed interleaver makes the occurrence of low-weight output code words relatively rare [8].

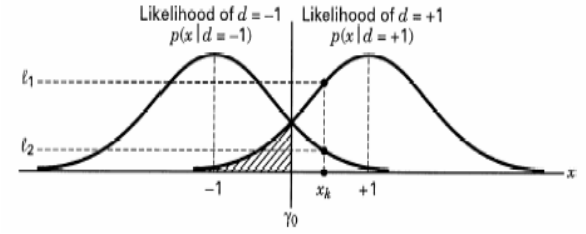

Fig. 4 Likelihood functions

Hence, one important property of the turbo-code is that its minimum distance d_m is not fixed, chiefly, by the constituent RSC codes but by the interleaving function and finding out the optimum interleaver for turbo-codes remains a real challenge.

IV. Decoding Turbo-Codes

A. Log-likelihood ratio

Let the binary logical elements 1 and 0 be represented electronically by voltages +1 and -1, respectively. The variable d is used to represent the transmitted data bit, whether it appears as a voltage or as a logical element. Sometimes one format is more convenient than the other; the reader should be able to recognize the difference from the context. Let the binary 0 (or the voltage value -1) be the null element under addition. For signal transmission over an AWGN channel, Fig. 4 shows the conditional pdfs referred to as likelihood functions. In Fig. 4, one such arbitrary value x_k is shown, where the index denotes an observation in the kth time interval. A line subtended from x_k intercepts the two likelihood functions, yielding two likelihood values $\ell_1 = p(x_k|d_k = +1)$ and $\ell_2 = p(x_k|d_k = -1)$. A well-known hard-decision rule, known as maximum likelihood, is to choose the data $d_k = +1$ or $d_k = -1$ associated with the larger of the two intercept values, ℓ_1 or ℓ_2, respectively. For each data bit at time k, this is tantamount to deciding that $d_k = +1$ if x_k falls on the right side of the decision line labeled γ_0, otherwise deciding that $d_k = -1$.

A similar decision rule, known as maximum a posteriori (MAP), which can be shown to be a minimum probability of error rule, takes into account the a priori probabilities of the data. The general expression for the MAP rule in terms of APPs is as follows:

$$P(d = +1|x) \underset{H_2}{\overset{H_1}{\gtrless}} P(d = -1|x) \tag{6}$$

Equation (6) states that you should choose the hypothesis H_1, (d = +1), if the APP P(d = +1|x), is greater than the APP P(d = -1|x). Otherwise, you should choose hypothesis H_2, (d= -1). Using the Bayes' theorem, the APPs in Equation (6) can be replaced by their equivalent expressions, yielding the following:

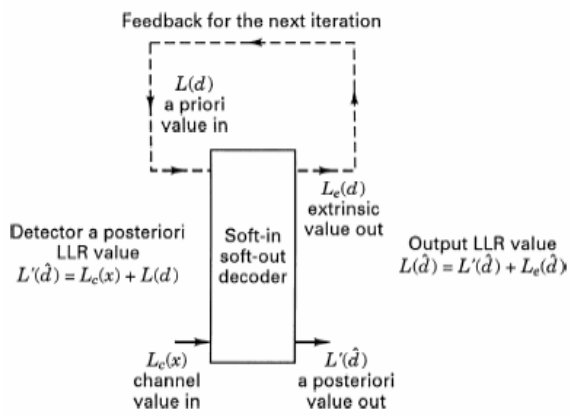

Fig. 5 Soft input/soft output decoder

$$p(x|d=+1)\ P(d=+1) \underset{H_2}{\overset{H_1}{\gtrless}} p(x|d=-1)\ P(d=-1) \tag{7}$$

Equation (7) is generally expressed in terms of a ratio, yielding the so-called likelihood ratio test, as follows:

$$\frac{p(x|d=+1)}{p(x|d=-1)} \underset{H_2}{\overset{H_1}{\gtrless}} \frac{P(d=-1)}{P(d=+1)} \quad \text{or} \quad \frac{p(x|d=+1)\ P(d=+1)}{p(x|d=-1)\ P(d=-1)} \underset{H_2}{\overset{H_1}{\gtrless}} 1 \tag{8}$$

By taking the logarithm of the likelihood ratio, we obtain a useful metric called the log-likelihood ratio (LLR). It is a real number representing a soft decision output of a detector, designated as follows:

$$L(d|x) = \log\left[\frac{P(d=+1|x)}{P(d=-1|x)}\right] = \log\left[\frac{p(x|d=+1)\ P(d=+1)}{p(x|d=-1)\ P(d=-1)}\right] \tag{9}$$

$$L(d|x) = \log\left[\frac{p(x|d=+1)}{p(x|d=-1)}\right] + \log\left[\frac{P(d=+1)}{P(d=-1)}\right] \tag{10}$$

$$L(d|x) = L(x|d) + L(d) \tag{11}$$

To simplify the notation, Equation (11) is rewritten as follows:

$$L'(\hat{d}) = L_c(x) + L(d) \tag{12}$$

where the notation $L_c(x)$ emphasizes that this LLR term is the result of a channel measurement made at the receiver. The equations above were developed with only a data detector in mind. Next, the introduction of a decoder will typically yield decision-making benefits. For a systematic code, it can be shown that the LLR (soft output) $L(\hat{d})$ out of the decoder is equal to Equation (13):

$$L(\hat{d}) = L'(\hat{d}) + L_e(\hat{d}) \tag{13}$$

where $L'(\hat{d})$ is the LLR of a data bit out of the demodulator (input to the decoder), and $L_e(\hat{d})$, called the extrinsic LLR, represents extra knowledge gleaned

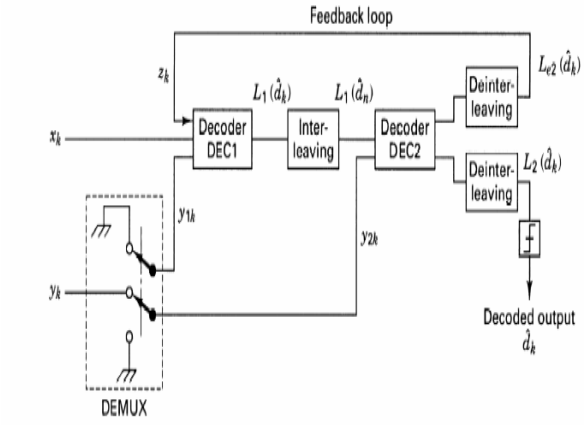

Fig. 6 Feedback decoder

from the decoding process. The output sequence of a systematic decoder is made up of values representing data bits and parity bits. From Equations (12) and (13), the output LLR $L(\hat{d})$ of the decoder is now written as follows:

$$L(\hat{d}) = L_c(x) + L(d) + L_e(\hat{d}) \tag{14}$$

Equation (14) shows that the output LLR of a systematic decoder can be represented as having three LLR elements—a channel measurement, a priori knowledge of the data, and an extrinsic LLR stemming solely from the decoder. To yield the final $L(\hat{d})$, each of the individual LLRs can be added as shown in Equation (14), because the three terms are statistically independent. This soft decoder output $L(\hat{d})$ is a real number that provides a hard decision as well as the reliability of that decision. The sign of $L(\hat{d})$ denotes the hard decision; that is, for positive values of $L(\hat{d})$ decide that d = +1, and for negative values decide that d =-1. The magnitude of $L(\hat{d})$ denotes the reliability of that decision. Often, the value of $L_e(\hat{d})$ due to the decoding has the same sign as $L_c(x) + L(d)$, and therefore acts to improve the reliability of $L(\hat{d})$.

B. Principles of Iterative (Turbo) Decoding

In a typical communications receiver, a demodulator is often designed to produce soft decisions, which are then transferred to a decoder. The improvement in error-performance of systems utilizing such soft decisions is typically approximated as 2 dB, as compared to hard decisions in AWGN. Such a decoder could be called a soft input/ hard output decoder, because the final decoding process out of the decoder must terminate in bits (hard decisions). With turbo codes, where two or more component codes are used, and decoding involves feeding outputs from one decoder to the inputs of other decoders in an iterative fashion, a hard-output decoder would not be suitable. That is because hard decisions into a decoder degrades system performance (compared to soft decisions). Hence, what is needed for the decoding of turbo codes is a soft input/ soft output decoder. For the first decoding iteration of such a soft input/soft output

decoder, illustrated in Figure 5, we generally assume the binary data to be equally likely, yielding an initial a priori LLR value of L(d)=0. The channel LLR value, $L_c(x)$, is measured by forming the logarithm of the ratio of the values of ℓ_1 and ℓ_2 for a particular observation of x (see Fig. 4), which appears as the second term in Equation (10). The output $L(\hat{d})$ of the decoder in Fig. 5 is made up of the LLR from the detector, $L'(\hat{d})$, and the extrinsic LLR output, $L_e(\hat{d})$, representing knowledge gleaned from the decoding process. As illustrated in Fig. 5, for iterative decoding, the extrinsic likelihood is fed back to the decoder input, to serve as a refinement of the a priori probability of the data for the next iteration.

C. Feedback Decoders

The Viterbi algorithm (VA) is an optimal decoding method for minimizing the probability of sequence error. Unfortunately, the VA is not suited to generate the a posteriori probability (APP) or soft-decision output for each decoded bit. A relevant algorithm for doing this has been proposed by Bahl et al. [10]. The Bahl algorithm was modified by Berrou, et al. [6] for use in decoding RSC codes. The Bahl algorithm can be used for decoding of turbo-codes using the feedback decoder shown in Fig. 6. The fundamental principle for feeding back information to another decoder is that a decoder should never be supplied with information that stems from itself (because the input and output corruption will be highly correlated).

V. ERROR PERFORMANCE OF TURBO-CODES

Performance results using Monte Carlo simulations have been presented in [3] for a rate 1/2, K = 5 encoder implemented with generators G1 = {1 1 1 1 1} and G2 = {1 0 0 0 1}, using parallel concatenation and a 256 × 256 array interleaver. The modified Bahl algorithm was used with a data block length of 65,536 bits. After 18 decoder iterations, the bit-error probability *PB* was less than 10-5 at Eb/N0 = 0.7 dB. The error-performance improvement as a function of the number of decoder iterations is seen in Fig 7. For binary modulation, several authors use *PB* = 10-5 and *Eb*/*N*0 = 0.2 dB as a *pragmatic* Shannon limit reference for a rate ½ code. Thus, with parallel concatenation of RSC convolutional codes and feedback decoding, the error performance of a turbo code at *PB* = 10-5 is within 0.5 dB of the (pragmatic) Shannon limit.

VI. CONCLUSIONS

This article described the concept of turbo coding, whose basic configuration depends on the concatenation of two or more component codes.Basic statistical measures such as a posteriori probability and likelihood were reviewed, and these measures were used to describe the error performance of a soft input/soft output decoder We showed how performance is improved when soft outputs from concatenated decoders are used in an iterative decoding process. We applied these concepts to the parallel concatenation of recursive systematic convolutional (RSC) codes, and explained why such codes are the preferred building blocks in turbo codes. A feedback decoder was described in general ways, and its remarkable performance was presented.

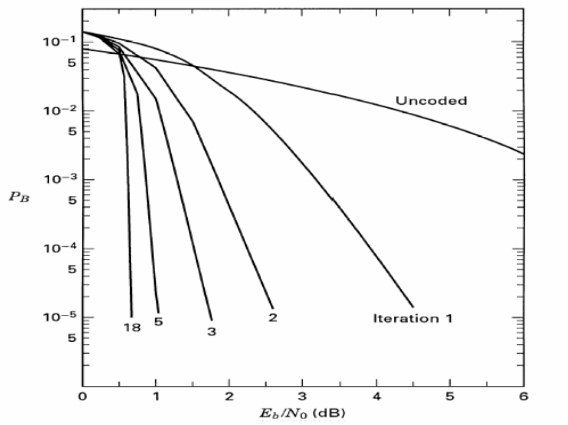

Fig. 7 Bit-error probability as a function of Eb/N0 and multiple iterations

REFERENCES

[1] G. D. Forney, "The Viterbi algorithm," *Proc. IEEE,* vol. 61, no. 3, pp.268-278, Mar. 1973.

[2] J. B. Cain, G. C. Clark, and J. M. Geist, "Punctured convolutional codes of rate (n-l)/n and simplified maximum likelihood decoding,"*IEEE Trans. Inform. Theory,* vol. IT-25, pp. 97-100, Jan 1979.

[3] G. Ungerboeck, "Channel coding with multilevel/phase signals," *IEEE Trans. Inform. Theory,* vol. IT-28, no. 1, pp. 55-67, Jan. 1982.

[4] Forney, G. D., Jr., *Concatenated Codes* (Cambridge, MA: MIT Press, 1966).

[5] Yuen, J. H., et al., "Modulation and Coding for Satellite and Space Communications," *Proc. IEEE*, vol. 78, no. 7, July 1990, pp. 1250- 1265.

[6] Berrou, C., Glavieux, A., and Thitimajshima, P., "Near Shannon Limit Error-Correcting Coding and Decoding: Turbo Codes," *IEEE Proceedings of the Int. Conf. on Communications*, Geneva, Switzerland, May 1993 (ICC '93), pp. 1064-1070. ones".

[7] Berrou, C. and Glavieux, A., "Near Optimum Error Correcting Coding and Decoding: Turbo-Codes," *IEEE Trans. on Communications*, vol. 44, no. 10, October 1996, pp. 1261-1271.

[8] Divsalar, D. and McEliece, R. J., "Effective Free Distance of Turbo Codes," *Electronic Letters*, vol. 32, no. 5, Feb. 29, 1996, pp. 445-446.

[9] Dolinar, S. and Divsalar, D., "Weight Distributions for Turbo Codes Using Random and Nonrandom Permutations," *TDA Progress Report 42-122*, Jet Propulsion Laboratory, Pasadena, California, August 15, 1995, pp. 56-65.

[10] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Truns.Inform.Theory,* vol. IT-20, pp. 248-287, Mar. 1974.