A Distance Spectrum Interpretation of Turbo Codes

Lance C. Perez, *Member, IEEE*, Jan Seghers, and Daniel J. Costello, Jr., *Fellow, IEEE*

***Abstract*— The performance of Turbo codes is addressed by examining the code's distance spectrum. The "error floor" that occurs at moderate signal-to-noise ratios is shown to be a consequence of the relatively low free distance of the code. It is also shown that the "error floor" can be lowered by increasing the size of the interleaver without changing the free distance of the code. Alternatively, the free distance of the code may be increased by using primitive feedback polynomials. The excellent performance of Turbo codes at low signal-to-noise ratios is explained in terms of the distance spectrum. The interleaver in the Turbo encoder is shown to reduce the number of low-weight codewords through a process called "spectral thinning." This thinned distance spectrum results in the free distance asymptote being the dominant performance parameter for low and moderate signal-to-noise ratios.**

***Index Terms*—Turbo codes, convolutional codes, distance spectrum.**

I. Introduction

THE DISCOVERY of Turbo codes and the near-capacity performance reported in [1] has stimulated a flurry of research efforts to fully understand this new coding scheme [2]–[43]. Initially greeted with some skepticism, the original results were independently reproduced by several researchers [6], [7], [10]–[13], and [30]. Subsequently, recent research on Turbo codes has focused on understanding the reasons for their outstanding performance [8], [9], [16] [22]–[24], [28].

At this point, there are two fundamental questions regarding Turbo codes. First, does the iterative decoding scheme presented in [1] always converge to the optimum solution? Second, assuming optimum or near-optimum decoding, why do the Turbo codes perform so well? In this paper, we address the second issue in a semitutorial manner by examining the distance spectrum of Turbo codes. In doing so, we will draw on the work of several research groups involved with Turbo codes [6]–[9], [12]–[14], [18]–[24]. Due to the intense interest in this subject, many results involving Turbo codes have been developed independently by others and the reader is encouraged to peruse the references for an alternative point of view. In particular, the recent papers by Benedetto and Montorsi [8], [9] were the first to offer a comprehensive picture of Turbo codes.

Manuscript received January 18, 1996; revised April 30, 1996. This work was supported by NASA under Grant NAG 5-557, by NSF under Grant NCR95-22939, and by the Lockheed–Martin Corporation.

L. C. Perez was with the Division of Engineering, The University of Texas at San Antonio, San Antonio, TX 78229 USA. He is now with the Department of Electrical Engineering, The University of Nebraska-Lincoln, Lincoln, NB 68588-0511 USA.

J. Seghers is with the Institute for Signal and Information Processing, Swiss Federal Institute of Technology, Zurich, Switzerland.

D. J. Costello, Jr. is with the Department of Electrical Engineering, University of Notre Dame, Notre Dame, IN 46616 USA.

Publisher Item Identifier S 0018-9448(96)07521-9.

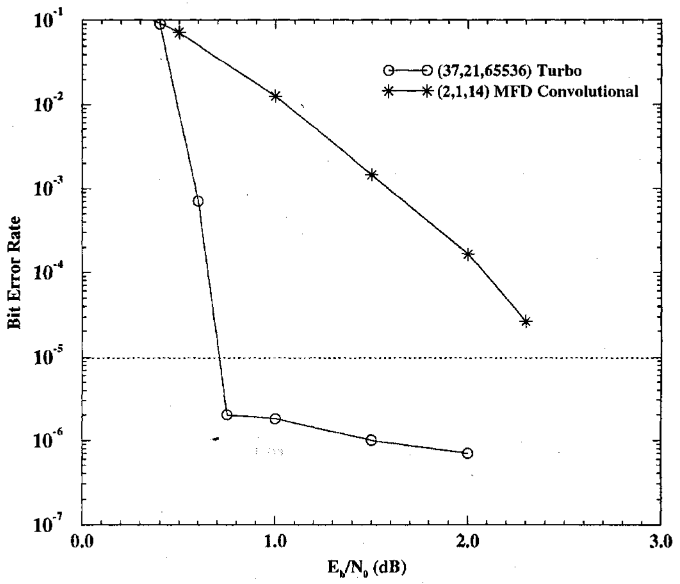

Fig. 1. Simulation results for a$(37, 21, 65536)$ Turbo code and a $(2, 1, 14)$ MFD convolutional code.

The simulated performance of a rate $1/2$ Turbo code with the same parameters as in [1] is shown in Fig. 1 along with simulation results for a rate $R = 1/2$, memory $\nu = 14$, convolutional code. The comparison of these simulation results raises two issues regarding the performance of Turbo codes. First, what is it that allows Turbo codes to achieve a bit error rate (BER) of 10^{-5} at a signal-to-noise ratio (SNR) of $E_b/N_0 = 0.7$ dB, which is only 0.7 dB from the Shannon limit? Second, what causes the "error floor" [10], [13], that is, the flattening of the performance curve, for moderate to high SNR's? Here, we endeavor to explain the performance of Turbo codes, and thus address these two issues, in terms of the code's distance spectrum. We do not attempt to address the many interesting questions concerning the iterative decoding method (see, e.g., [15], [26], [42], and [43]), i.e., we assume that an optimum or near-optimum decoder is available.

In order to explain their performance in terms of the free distance and the distance spectrum, we will examine the codeword structure of Turbo codes in detail. Here, the free distance is defined to be the minimum Hamming weight of all possible codewords and the error coefficient is the total number, or multiplicity, of free-distance codewords. The goal is to use specific examples to elucidate the key structural properties that result in the near-capacity performance of Turbo codes at BER's around 10^{-5}. As will be seen, this effort leads to an explanation that applies to Turbo codes and also lends insight into designing codes in general. Throughout the paper, Turbo codes are compared to a maximum free-distance, rate

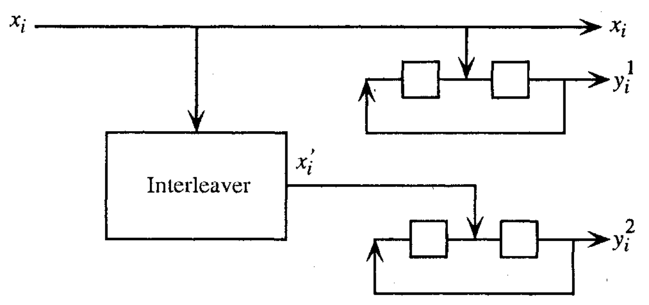

Fig. 2. Block diagram of a Turbo encoder with two identical constituent encoders ($h_0(D) = 1 + D^2, h_1(D) = D$) and without puncturing.

$R = 1/2$, memory $\nu = 14$, i.e., a $(2,1,14)$, convolutional code to emphasize the differences in performance and structure. Techniques for analyzing the performance of Turbo codes using transfer functions and related methods may be found in [6], [9], and [22].

The paper begins with a detailed examination of the structure of codewords in a Turbo code in Section II. This leads to the calculation of the free distance of a particular Turbo code and an explanation for the "error floor" in its performance curve. In Section III, the distance spectrum of "average" Turbo codes is considered and a theory called spectral thinning is introduced and used to explain the performance of Turbo codes at low SNR's. The idea of spectral thinning is then formalized in Section IV through the use of random interleaving. Finally, some conclusions are drawn concerning the distance spectrum of Turbo codes and the consequences of this on the design of codes in general.

II. The Free Distance of Turbo Codes

In order to find the free distance of a Turbo code, it is necessary to understand the basic structure of the encoder and the resulting codewords. A typical Turbo encoder consists of the parallel concatenation of two or more, usually identical, rate $1/2$ encoders, realized in systematic feedback form, and an interleaver. This encoder structure is called a parallel concatenation because the two encoders operate on the same *set* of input bits, rather than one encoding the output of the other. A block diagram of a Turbo encoder with two constituent convolutional encoders is shown in Fig. 2. For the remainder of the paper, only Turbo encoders with two identical constituent convolutional encoders are considered, though the conclusions are easily extended to the general case.

The interleaver is used to permute the input bits such that the two encoders are operating on the same *set* of input bits, but different input *sequences*. Thus the first encoder receives the input bit x_i and produces the output pair (x_i, y_i^1) while the second encoder receives the input bit x_i' and produces the output pair (x_i', y_i^2). The input bits are grouped into finite-length sequences whose length N equals the size of the interleaver. Since both the encoders are systematic and operate on the same set of input bits, it is only necessary to transmit the input bits once and the overall code has rate $1/3$. In order to increase the overall rate of the code to $1/2$, the two parity sequences $\{y^1\}$ and $\{y^2\}$ can be punctured by alternately deleting y^1 and y^2. We will refer to a Turbo code whose constituent encoders have parity-check polynomials h_0 and h_1 and whose interleaver is of length N as an (h_0, h_1, N) Turbo code.

For example, consider the Turbo encoder shown in Fig. 2, where each constituent encoder is a $(2,1,2)$ encoder with parity-check polynomials $h_0(D) = 1 + D^2$ and $h_1(D) = D$, where D is the delay operator [44]. For purposes of illustration, assume a pseudorandom interleaver of size $N = 16$ bits which generates a $(1 + D^2, D, 16)$ Turbo code. The interleaver is realized as a 4×4 matrix which is filled sequentially, row by row, with the input bits x_i. Once the interleaver has been filled, the input sequence to the second encoder, x_i', is obtained by reading the interleaver in a pseudorandom manner until each bit has been read once and only once. The pseudorandom nature of the interleaver in this example is represented by a permutation

$$\Pi_{16} = \{6, 14, 4, 7, 11, 8, 3, 5, 9, 13, 0, 2, 12, 1, 10, 15\}$$

which implies $x_0' = x_{15}, x_1' = x_{10}$, and so on.

If the input sequence is

$$x = \{x_{15} \cdots x_0\} = \{0,1,0,1,0,0,0,0,0,0,0,1,0,0,0,1\}$$

and the interleaver is represented by the permutation Π_{16}, then the input sequence to the second encoder is

$$x' = \Pi_{16}(x) = \{0,1,1,0,0,0,0,0,0,0,1,0,1,0,0,0\}.$$

The trellis diagrams for both constituent encoders with these inputs are shown in Fig. 3. The corresponding unpunctured parity sequences are

$$y^1 = \{0,0,1,0,0,0,0,0,0,0,0,0,1,0,1,0\}$$

and

$$y^2 = \{1,1,0,0,0,0,0,0,0,0,0,1,0,0,0,0\}.$$

The resulting codeword has Hamming weight

$$d = w(x) + w(y^1) + w(y^2) = 4 + 3 + 3 = 10$$

without puncturing, where $w(x)$ is the Hamming weight of the sequence x. If the code is punctured beginning with y_0^1, then the resulting codeword has weight $d = 4 + 3 + 2 = 9$. If, on the other hand, the puncturing begins with y_0^2, then the punctured codeword has Hamming weight $4 + 1 = 5$.

Finding the free distance of a Turbo code is complicated by the fact that Turbo encoders are time-varying due to the interleaver. That is, if $\tilde{x} = Dx$, then $\tilde{y}^1 = Dy^1$, but $\tilde{x}' \neq Dx'$ (with high probability) and $\tilde{y}^2 \neq Dy^2$. (Here, we only consider delays of a finite length sequence x for which no ones are lost.) Continuing the example, if $\tilde{x} = Dx$, then

$$\tilde{x}' = \Pi_{16}(\tilde{x}) = \{0,0,0,0,0,0,0,1,0,1,0,0,0,1,0,1\}$$

and

$$\tilde{y}^2 = \{0,0,0,0,0,0,0,0,1,0,0,0,0,0,1,0\}.$$

Thus time-shifting the input bits results in codewords that differ in both the bit positions of the ones and overall Hamming weight. The variation in the weights of codewords

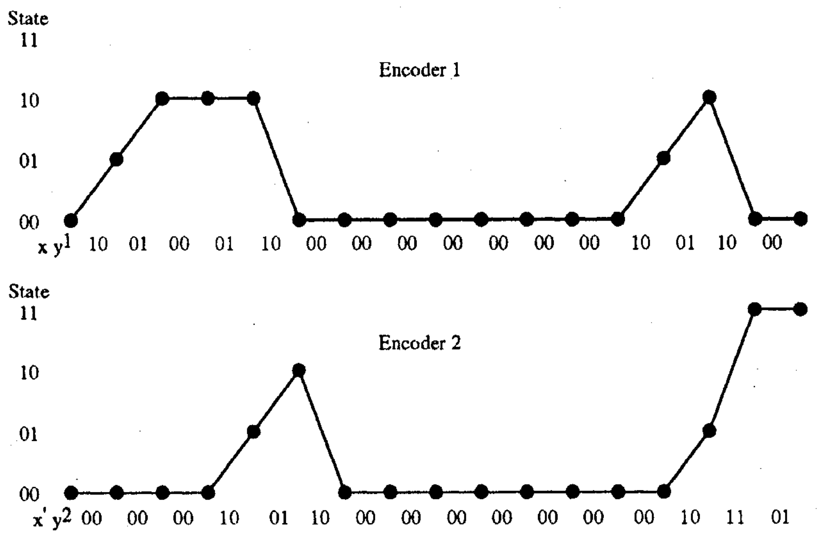
State
11
10
01
00
Encoder 1
x y1 10 01 00 01 10 00 00 00 00 00 00 00 10 01 10 00
State
11
10
01
00
Encoder 2
x' y2 00 00 00 10 01 10 00 00 00 00 00 00 00 10 11 01

Fig. 3. Trellis diagrams for a codeword in the $(1 + D^2, D, 16)$ Turbo code.

corresponding to time-shifted input sequences is magnified by puncturing.

This simple example illustrates several important points concerning the structure of the codewords. First, because the pseudorandom interleaver permutes the input bits, the two input sequences x and x' are almost always different, though of the same weight, and the two encoders will (with high probability) produce parity sequences of different weights. Second, it is easily seen that a codeword may consist of a number of distinct error events in each encoder, where an error event is a path in the trellis that diverges from the all-zero state and then remerges with the all-zero state within a finite number of branches. Note that since the constituent encoders are realized in systematic feedback form a nonzero tail of ν bits is required to return the encoder to the all-zero state. Thus since at least one nonzero bit is required to start an error event, all error events are associated with information sequences of weight 2 or greater [8].

Finally, with a pseudorandom interleaver it is highly unlikely that both encoders will be returned to the all-zero state at the end of the codeword, even when the last ν bits of the input sequence x are used to force the first encoder back to the all-zero state. This is due to the fact the two encoders will with high probability end up in different states and thus require two different ν bit tails. It is highly unlikely that the ν-bit tail for the first encoder will be interleaved by a pseudorandom interleaver to the correct tail for the second encoder.

If neither encoder is forced back to the all-zero state, i.e., no tail is used, then the sequence consisting of $N - 1$ zeroes followed by a one is a valid input sequence x to the first encoder. For some interleavers, this x will be permuted to itself and x' will be the same sequence. In this case, with puncturing, the weight of this codeword and the free distance of the code will be at most two. Note that this codeword is caused by an information sequence of weight one. Thus forcing the first encoder to return to the all-zero state insures that every information sequence has at least weight two and eliminates the possibility of a weight-two codeword. For this reason, it is common to assume that the first encoder is forced to return to the all-zero state *by appending a ν-bit tail.*

The ambiguity of the final state of the second encoder has been shown by simulation to result in negligible performance degradation for large interleavers [12], [30]. For these reasons, it will be assumed in the remainder of the paper that the first encoder is forced to return to the all-zero state and that the final state of the second encoder is unknown. Special interleaver structures that result in both encoders returning to the all-zero state are discussed in [31], [32], [34]–[36].

A. Performance Bounds

In order to make clear the distinction between Turbo codes and convolutional codes, it is useful to consider these codes as block codes. To this end, the input sequences are restricted to length N, where N corresponds to the size of the interleaver in the Turbo encoder. With finite-length input sequences of length N, a $(2, 1, \nu)$ convolutional code may be viewed as a block code with 2^N codewords of length $2(\nu + N)$.

The bit error rate (BER) performance of a convolutional code with maximum-likelihood (ML) decoding on an additive white Gaussian noise (AWGN) channel can be upper-bounded using a union bound technique by [12]

$$P_b \leq \sum_{i=1}^{2^N} \frac{w_i}{N} Q\left(\sqrt{d_i \frac{2RE_b}{N_0}}\right) \tag{1}$$

where w_i and d_i are the information weight and total Hamming weight, respectively, of the ith codeword. Collecting codewords of the same total Hamming weight and defining the average information weight per codeword as

$$\tilde{w}_d = \frac{W_d}{N_d}$$

where W_d is the total information weight of all codewords of weight d and N_d is the total number, or multiplicity, of codewords of weight d, yields

$$P_b \leq \sum_{d=d_{\text{free}}}^{2(\nu+N)} \frac{N_d \tilde{w}_d}{N} Q\left(\sqrt{d \frac{2RE_b}{N_0}}\right) \tag{2}$$

where d_{free} is the free distance of the code. (In this development, the multiplicity N_d includes codewords due to multiple error events for $d \geq 2d_{\text{free}}$).

For large enough N, if a convolutional code has N_d^0 codewords of weight d caused by information sequences x whose first one occurs at time 0, then it also has N_d^0 codewords of weight d caused by the information sequences Dx, N_d^0 codewords of weight d caused by the information sequences D^2x, and so on. Thus as the length of the information sequences increases, we have

$$\lim_{N\to\infty} \frac{N_d}{N} = N_d^0$$

and

$$\lim_{N\to\infty} \tilde{w}_d = \lim_{N\to\infty} \frac{W_d}{N_d} = \frac{W_d^0}{N_d^0} \triangleq \tilde{w}_d^0$$

where W_d^0 is the total information weight of all codewords with weight d which are caused by information sequences whose first one occurs at time 0. Thus the bound on the BER of a convolutional code with ML decoding becomes

$$\begin{aligned} P_b &\leq \sum_{d=d_{\text{free}}}^{2(\nu+N)} N_d^0 \tilde{w}_d^0 Q\left(\sqrt{d \frac{2RE_b}{N_0}}\right) \\ &= \sum_{d=d_{\text{free}}}^{2(\nu+N)} W_d^0 Q\left(\sqrt{d \frac{2RE_b}{N_0}}\right) \end{aligned} \tag{3}$$

which is the standard union bound for ML decoding. For this reason, efforts to find good convolutional codes for use with ML decoders have focused on finding codes that maximize the free distance d_{free} and minimize the number of free-distance paths N_{free}^0 for a given rate and constraint length.

The performance of a Turbo code with maximum-likelihood decoding can also be bounded using the union bound of (2). However, in the Turbo encoder the pseudorandom interleaver maps the input sequence x to x' and the input sequence Dx to a sequence x'' that is different from Dx' with high probability. Thus unlike convolutional codes, the input sequences x and Dx produce different codewords with different Hamming weights. For Turbo codes with pseudorandom interleavers, $N_d\tilde{w}_d$ is much less than N for low-weight codewords. This is due to the pseudorandom interleaver which, with high probability, maps low-weight parity sequences in the first constituent encoder to high-weight parity sequences in the second constituent encoder. Thus for low-weight codewords

$$\frac{N_d \tilde{w}_d}{N} \ll 1$$

where

$$\frac{N_d}{N} \tag{4}$$

is called the *effective multiplicity* of codewords of weight d. The effect of the interleaver size on the multiplicity is also studied in [7]–[9].

B. Asymptotic Performance

For moderate and high signal-to-noise ratios, it is well known that the free-distance term in the union bound on the bit error rate performance dominates the bound [44]. Thus for Turbo codes the asymptotic performance approaches

$$P_b \approx \frac{N_{\text{free}} \tilde{w}_{\text{free}}}{N} Q\left(\sqrt{d_{\text{free}} \frac{2RE_b}{N_0}}\right) \tag{5}$$

where N_{free} is the multiplicity of free-distance codewords and $\tilde{w}_{\text{free}}$ is the average weight of the information sequences causing free-distance codewords. The expression on the right-hand side of (5) and its associated graph is called the *free-distance asymptote*, P_{free}, of a Turbo code.

An algorithm for finding the free distance of Turbo codes is described in [28]. This algorithm was applied to a Turbo code with the same constituent encoders, puncturing pattern, and interleaver size N as in [1] and a *particular* pseudorandom interleaving pattern. The parity check polynomials for this code are $h_0 = D^4+D^3+D^2+D+1$ and $h_1 = D^4+1$, or $h_0 = 37$ and $h_1 = 21$ using octal notation. This $(37, 21, 65536)$ code was found to have $N_{\text{free}} = 3$ paths with weight $d_{\text{free}} = 6$. Each of these paths was caused by an input sequence of weight 2 and thus $\tilde{w}_{\text{free}} = 2$. Though this result was for a particular pseudorandom interleaver, it is true for most pseudorandom interleavers with $N = 65536$. This is consistent with the conclusions in [6] and [8] in which the performance of Turbo codes is averaged over all possible pseudorandom interleavers.

For this particular Turbo code, the free distance asymptote is given by

$$P_{\text{free}} = \frac{3 \times 2}{65536} Q\left(\sqrt{6 \frac{E_b}{N_0}}\right)$$

where the rate loss due to the addition of a 4-bit tail is ignored and

$$\frac{N_{\text{free}}}{N} = \frac{3}{65536}$$

is the effective multiplicity. The free-distance asymptote is shown plotted in Fig. 4 along with simulation results for this code using the iterative decoding algorithm of [1] with 18 iterations. From Fig. 4, it can clearly be seen that the simulation results do in fact approach the free-distance asymptote for moderate and high SNR's. Since the slope of the asymptote is essentially determined by the free distance of the code, it can be concluded that the "error floor" observed with Turbo codes is due to the fact that they have a relatively small free distance and consequently a relatively flat free-distance asymptote.

Further examination of (5) reveals that the manifestation of the "error floor" can be manipulated in two ways. First, increasing the length of the interleaver while preserving the free distance and the multiplicity will lower the asymptote without changing its slope by reducing the effective multiplicity. In this case, the performance curve of Turbo codes does not flatten out until higher SNR's and lower BER's are reached. Conversely, decreasing the interleaver size while maintaining the free distance and multiplicity results in the error floor being raised and the performance curve flattens at lower SNR's and

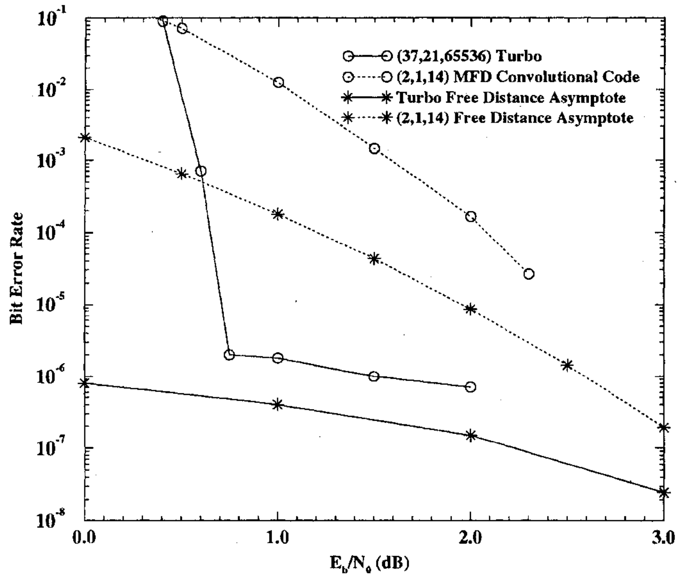

Fig. 4. Simulation results for a $(37, 21, 65536)$ Turbo code and a $(2,1,14)$ MFD convolutional code along wiht their free-distance asymptotes.

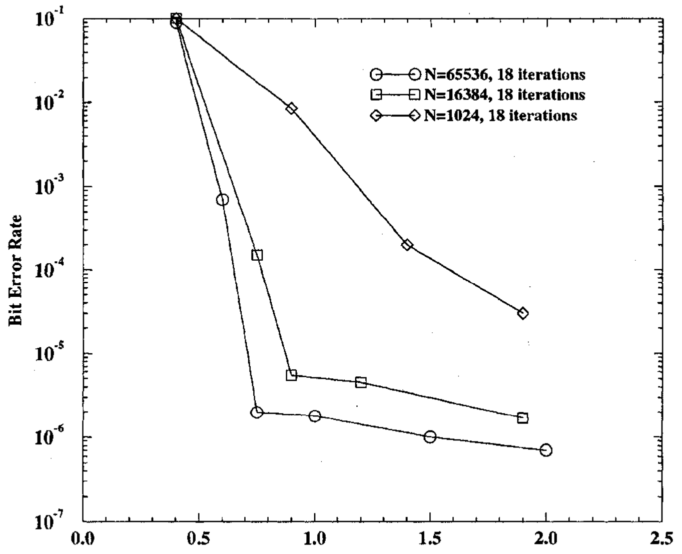

Fig. 5. Simulation results for a $(37, 21, N)$ Turbo code with varying interleaver size N.

higher BER's. This can be seen in the simulation results shown in Fig. 5 for the $(37, 21, N)$ Turbo code with varying N. If the first constituent encoder is not forced to return to the all-zero state and the weight 2 codewords mentioned earlier are allowed, then the error floor is raised to the extent that the code performs poorly even for large interleavers. Thus one cannot completely disregard free distance when constructing Turbo codes.

If the size of the interleaver is fixed, then the "error floor" can be modified by increasing the free distance of the code while preserving the multiplicity. This has the effect of changing the slope of the free-distance asymptote. That is, increasing the free distance increases the slope of the asymptote and decreasing the free distance decreases the slope of the asymptote. It has been shown in [8], [24], [28], and [29] that for a fixed interleaver size, choosing the feedback polynomial to be a primitive polynomial results in an increased free distance and thus a steeper asymptote. An argument to support the use of primitive feedback polynomials in Turbo codes is presented in Section IV.

C. Comparison to the $(2,1,14)$ *Code*

The role that the free distance and effective multiplicity play in determining the asymptotic performance of a Turbo code is further clarified by examining the asymptotic performance of a convolutional code. The free-distance asymptote of a convolutional code is given by the first term in the union bound of (3). The maximum free distance (MFD) $(2,1,14)$ code whose performance is shown in Fig. 4 has $d_{\text{free}} = 18, N_{\text{free}}^0 = 18$ and $W_{\text{free}}^0 = 137$ [45]. Thus the free-distance asymptote for this code is

$$P_{\text{free}} = 137 Q\left(\sqrt{18 \frac{E_b}{N_0}}\right)$$

which is also shown in Fig. 4.

As expected, the free-distance asymptote of the $(2,1,14)$ code is much steeper than the free-distance asymptote of the Turbo code due to the increased free distance. However, because the effective multiplicity of the free-distance codewords, given by (4), of the Turbo code is much smaller than the multiplicity of the $(2,1,14)$ code, the two asymptotes do not cross until $E_b/N_0 = 3.5$ dB. At this E_b/N_0, the BER of both codes is less than 10^{-7}, which is lower than the targeted BER of many practical systems. Thus even though the $(2,1,14)$ convolutional code is asymptotically better than the $(37, 21, 65536)$ Turbo code, the Turbo code is better for the error rates at which many systems operate.

D. A Turbo Code with a Rectangular Interleaver

To emphasize the importance of using a pseudorandom interleaver with Turbo codes, we now consider a Turbo code with a rectangular interleaver. Turbo codes with rectangular interleavers have also been considered in [2], where the effect of the interleaver on the free distance of the code is discussed. The same constituent encoders and puncturing pattern as in [1] are used in conjunction with a 120×120 rectangular interleaver. This rectangular interleaver is realized as a 120×120 matrix into which the information sequence x is written row by row. The input sequence to the second encoder x' is then obtained by reading the matrix column by column. A 120×120 rectangular interleaver implies an interleaver size of $N = 14400$ and thus this is a $(37, 21, 14400)$ Turbo code.

Using the algorithm described in [28], this code was found to have a free distance of $d_{\text{free}} = 12$ with a multiplicity of $N_{\text{free}} = 28900$. For this code, each of the free-distance paths is caused by an information sequence of weight 4, so $\tilde{w}_{\text{free}} = 4$. The free-distance asymptote for this code is thus given by

$$P_{\text{free}} = \frac{28900 \times 4}{14400} Q\left(\sqrt{12 \frac{E_b}{N_0}}\right).$$

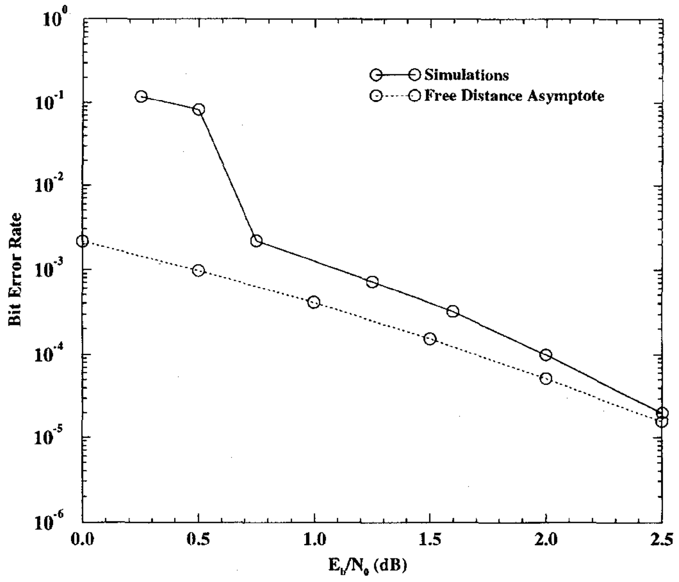

Fig. 6. Simulation results and the free-distance asymptote for the $(37, 21, 14400)$ Turbo code with a 120×120 rectangular interleaver.

The free-distance asymptote is plotted in Fig. 6 along with simulation results using the iterative decoding algorithm of [1] with 18 iterations. This figure clearly shows that the free-distance asymptote accurately estimates the performance of the code for moderate and high E_b/N_0's.

This code achieves a bit error rate of 10^{-5} at an E_b/N_0 of 2.7 dB and thus performs 2 dB worse than the $(37, 21, 65536)$ Turbo code with a pseudorandom interleaver even though it has a much larger free distance. The relatively poor performance of the $(37, 21, 14400)$ Turbo code with a rectangular interleaver is due to the large multiplicity of d_{free} paths. This results in an effective multiplicity of

$$\frac{N_{\text{free}}}{N} = \frac{28900}{14400} \approx 2$$

which is much larger than the effective multiplicity of the $(37, 21, 65536)$ Turbo code. We now show that the large multiplicity is a direct consequence of the use of the rectangular interleaver and that, furthermore, increasing the size of the interleaver does not result in a significant reduction in the effective multiplicity of the free-distance codewords.

The free-distance paths in the Turbo code with the rectangular interleaver are due to four basic information sequences of weight 4. These information sequences are depicted in Fig. 7 as they would appear in the rectangular interleaver. The "square" sequence in Fig. 7a) depicts the sequence

$$x = 1, 0, 0, 0, 0, 1, 0_{594}, 1, 0, 0, 0, 0, 1, 0_{\infty}$$

where 0_{594} denotes a sequence of 594 consecutive zeroes and 0_{∞} represents a sequence of zeroes that continues to the end of the information sequence. In this case, the rectangular interleaver maps the sequence x to itself and therefore $x' = x$. The sequence x results in a parity sequence y^1 from the first constituent encoder which, after puncturing, has weight 4. Similarly, the input sequence $x' = x$ results in a parity sequence y^2 from the second constituent encoder which, after puncturing, also has weight 4. The weight of the codeword is then $d_{\text{free}} = 4+4+4 = 12$. By counting the number of distinct positions in which these "square" sequences can appear in the rectangular interleaver, we can find the multiplicity of the free-distance codewords. Since the "square" sequence in Fig. 7a) can appear in $(\sqrt{N}-5) \times (\sqrt{N}-5) = 13225$ distinct positions in the rectangular interleaver, and in each case $x' = x$ and a codeword of weight $d_{\text{free}} = 12$ results, this results in 13225 free-distance codewords. Note that for every occurrence of the "square" sequence to result in a codeword of weight 12 the weight of both parity sequences must be invariant to which is punctured first.

The "rectangular" sequences in Fig. 7b) and c) also result in weight 12 codewords. For these two sequences, the weight of one of the parity sequences is affected by whether or not it is punctured first and only every other position in which the "rectangular" sequences appear in the interleaver results in a codeword of weight $d_{\text{free}} = 12$. Thus the sequences in Fig. 7b) and c) each result in $0.5(\sqrt{N} - 10) \times (\sqrt{N} - 5) = 6325$ free-distance codewords. For the "square" sequence in Fig. 7d), the weight of both parity sequences is affected by which is punctured first and only one out of four positions in which this "square" sequence appears in the interleaver results in a codeword of weight $d_{\text{free}} = 12$. Consequently, this "square" sequence results in $0.25(\sqrt{N} - 10) \times (\sqrt{N} - 10) = 3025$ free-distance codewords. Summing the contributions of each type of sequence results in a total of $N_{\text{free}} = 28900$ codewords of weight $d_{\text{free}} = 12$.

It is tempting to try to improve the performance of a Turbo code with a rectangular interleaver by increasing the size of the interleaver. However, all of the information sequences shown in Fig. 7 would still occur in a larger rectangular interleaver, so the free distance cannot be increased by increasing N. Also, since the number of free-distance codewords is on the order of N, increasing the size of the interleaver results in a corresponding increase in N_{free} such that the effective multiplicity N_{free}/N does not change significantly. Without the benefit of a reduced effective multiplicity, the free-distance asymptote, and thus the "error floor," of Turbo codes with rectangular interleavers is not lowered enough for them to manifest the excellent performance of Turbo codes with pseudorandom interleavers for moderate BER's. Attempts to design interleavers for Turbo codes generally introduce structure to the interleaver and thus destroy the very randomness that results in such excellent performance at low SNR's.

III. The Distance Spectrum of Turbo Codes

In the previous section, it was shown that the "error floor" observed in the performance of Turbo codes is due to their relatively low free distance. It is now shown that the outstanding performance of Turbo codes at low SNR's is a manifestation of the sparse distance spectrum that results when a pseudorandom interleaver is used in a parallel concatenation scheme. To illustrate this the distance spectrum of an "average" $(37, 21, 65536)$ Turbo code is found and its relationship to the performance of the code is discussed. The distance spectrum of the "average" Turbo code is then compared to the distance

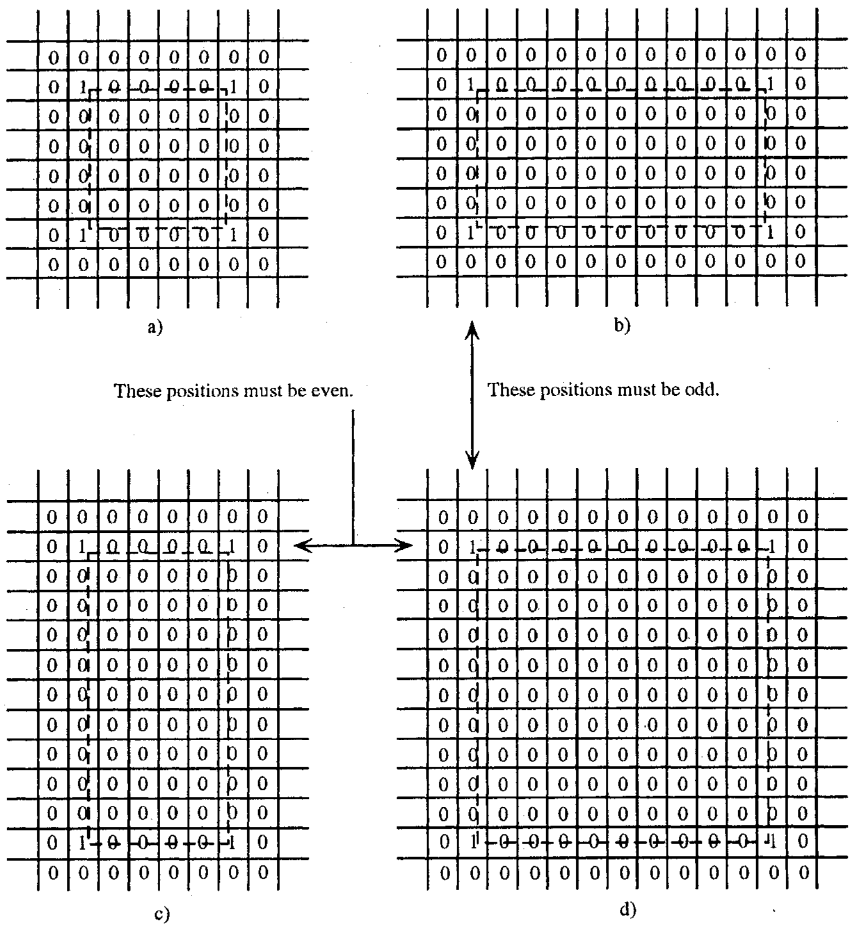

Fig. 7. Portions of the interleaver and information sequences resulting in free-distance codewords for a $(37, 21, 14400)$ Turbo code with a 120×120 rectangular interleaver.

spectrum of the $(2, 1, 14)$ code. An "average" Turbo code is one whose properties have been averaged over all possible pseudorandom interleavers as in [8]. By doing this, the analysis of Turbo codes given in [8] and the algorithm for finding the distance spectrum given in [28] are simplified.

Using the algorithm described in [28], an "average" $(37, 21, 65536)$ Turbo code was found to have the following distance spectrum

d	N_d	W_d
6	4.5	9
8	11	22
10	20.5	41
12	75	150

where N_d is the total number of codewords of weight d and $W_d = N_d \tilde{w}_d$ is the total information weight of all codewords of weight d. The distance spectrum information for a particular distance d is referred to as a spectral line. This data can be used in conjunction with the bound of (2) to estimate the performance of the code. In addition, by plotting each term of (2) the contribution of each spectral line to the overall performance of the code can be estimated.

The performance of a particular $(37, 21, 65536)$ Turbo code is shown in Fig. 8 along with curves showing the contribution of each spectral line for an "average" Turbo code with the same interleaver length. This clearly shows that the contribution to the code's BER by the higher distance spectral lines is less than the contribution of the free-distance term for E_b/N_0's greater than 0.5 dB. (The higher distance spectral lines become important, though, as capacity is approached.) Thus the free-distance asymptote dominates the performance of the code not only for moderate and high E_b/N_0's, but also for low E_b/N_0's. We characterize distance spectra for which this is true as sparse or spectrally thin.

A. Comparison to the $(2, 1, 14)$ *Code*

The ramifications of a sparse-distance spectrum are made evident by examining the distance spectrum of convolutional codes. The $(2, 1, 14)$ convolutional code introduced in Section II has the following distance spectrum:

d	N_d^0	W_d^0
18	33	187
20	136	1034
22	835	7857
24	4787	53994
26	27941	361762
28	162513	2374453
30	945570	15452996
32	5523544	99659236

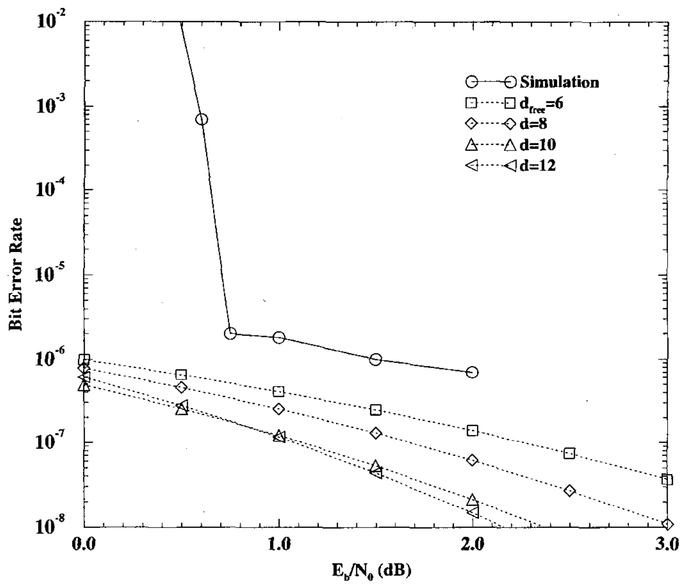

Fig. 8. Performance of a $(37, 21, 65536)$ Turbo code decomposed by spectral line.

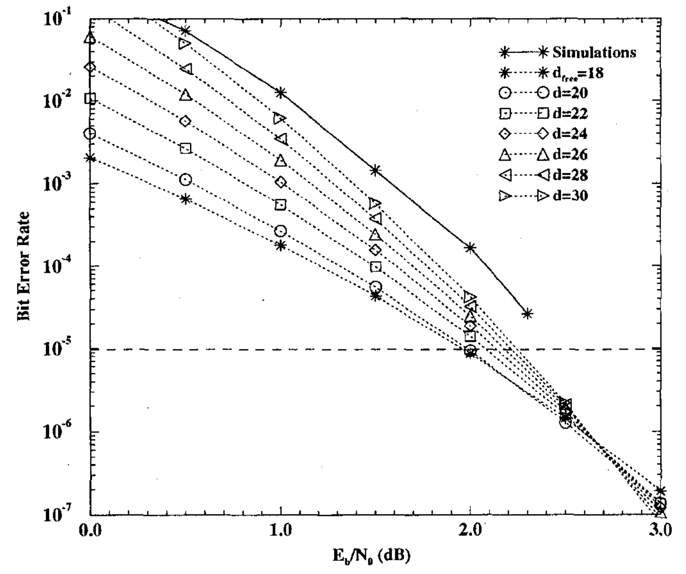

Fig. 9. Performance of the $(2, 1, 14)$ MFD convolutional code decomposed by spectral line.

as reported in [45]. When comparing the distance spectrum of a convolutional code and a Turbo code, it is important to remember that for a convolutional code $N_d \approx N \times N_d^0$ for the low-weight codewords. Fig. 9 shows the performance of this code and the contribution of each spectral line from the bound of (3).

In this case, the contribution of the higher distance spectral lines to the overall BER is greater than the contribution of the free-distance term for E_b/N_0's less than 2.7 dB, which corresponds to BER's of less than 10^{-6}! The large SNR required for the free-distance asymptote to dominate the performance of the $(2, 1, 14)$ code is due to the rapid increase in the path multiplicity for increasing d. We characterize distance spectra for which this is true as spectrally dense. The dense distance spectrum of convolutional codes also accounts for the discrepancy between the real coding gain at a particular SNR and the asymptotic coding gain calculated using just the free distance [44].

Thus it can be concluded that the outstanding performance of Turbo codes at low signal-to-noise ratios is a result of the dominance of the free-distance asymptote, which in turn is a consequence of the sparse-distance spectrum of Turbo codes, as opposed to spectrally dense convolutional codes. Finally, the sparse-distance spectrum of Turbo codes is due to the structure of the codewords in a parallel concatenation and the use of pseudorandom interleaving.

IV. Spectral Thinning

In this section, the observations made concerning the distance spectrum and spectral thinning of Turbo codes is formalized from the point of view of random interleaving. Random interleaving was introduced in [6]–[9] to develop transfer function bounds on the average performance of Turbo codes and to explore issues of code design. Here, random interleaving is used to explore the effect of the interleaver on the distance spectrum of the code. In order to simplify the notation and discussion, only nonpunctured Turbo codes are considered explicitly. The extension to punctured codes is straightforward and may be found in [28].

The fundamental idea of random interleaving is to consider the performance of a Turbo code averaged over all possible pseudorandom interleavers of a given length. For a given N, there are $N!$ possible pseudorandom interleavers and, assuming a uniform distribution, each occurs with probability $1/N!$. Let a particular interleaver map an information sequence x of weight w to an information sequence x', also of weight w. Then there are a total of $w!(N-w)!$ interleavers in the ensemble of $N!$ interleavers that perform this same mapping. Thus the probability that such a mapping occurs and hence that the codeword that results from the input sequences x and x' occurs, is

$$\frac{w!(N-w)!}{N!} = \frac{1}{\binom{N}{w}}.$$

Following [8], define the input redundancy weight enumerating function (IRWEF) of a systematic code as

$$A(W, Z) = \sum_{w} \sum_{z} A_{w,z} W^w Z^z \tag{6}$$

where $A_{w,z}$ is the number of codewords of weight $d = w + z$ generated by input sequences of weight w and parity sequences of weight z. The goal is now to develop a relationship between the codewords in the constituent encoders and $A_{w,z}$ for the Turbo code and to see how that relationship changes with the size of the interleaver.

Recall from Section II that a Turbo codeword is essentially the combination of a codeword from the first constituent encoder plus a codeword from the second constituent encoder. A codeword of weight $d_1 = w + z_1$ from the first constituent encoder caused by an information sequence x of weight w

is composed of n_1 error events of total length l_1. To avoid difficulties in counting codewords when the second constituent encoder is left unterminated, we consider different orderings of the same error events as distinct cases. The ordered set of n_1 error events in the first encoder is denoted by S_1. The information sequence x that results in the set S_1 is mapped by a particular interleaver to the information sequence x', also of weight w, which is then encoded by the second constituent encoder. This results in a codeword of weight $d_2 = w + z_2$, with the ordered set S_2 consisting of n_2 error events of total length l_2.

For example, Fig. 3 depicts a codeword of weight $d_1 = 4 + 3 = 7$ in the first constituent encoder caused by an information sequence of weight $w = 4$ and composed of $n_1 = 2$ error events of total length $l_1 = 5 + 3 = 8$. For the interleaver described in Section II, x is mapped to an x' that results in a codeword of weight $d_2 = 4 + 3 = 7$ in the second constituent encoder consisting of $n_2 = 2$ error events of total length $l_2 = 3 + 3 = 6$. Thus this S_1 and S_2 result in a codeword of weight 10 and a single contribution to $A_{4,6}$ of the Turbo code for this particular interleaver. Averaged over the ensemble of interleavers of length N, a set S_1 with information weight w and parity weight z_1 and a set S_2 with information weight w and parity weight z_2 will contribute a fraction

$$\frac{1}{\binom{N}{w}} \tag{7}$$

to the enumerating function coefficients A_{w,z_1+z_2} of an "average" Turbo code.

Because the sequence of zeros connecting any two distinct error events has no effect on the weight of the information sequence or the parity sequence, there are

$$\binom{N - l_1 + n_1}{n_1} \tag{8}$$

ways that the ordered set S_1 of n_1 error events can be arranged such that their contribution to A_{w,z_1+z_2} is not changed. This is simply the number of ways in which n_1 distinct error events can be arranged in a sequence of length N while maintaining the order in which they appear. Similarly, if the codeword in the second constituent encoder ends in the all-zero state, then the ordered set S_2 will make

$$\binom{N - l_2 + n_2}{n_2} \tag{9}$$

contributions to A_{w,z_1+z_2}. However, because the second encoder is not guaranteed to return to the all-zero state, it is possible that the last of the n_2 error events is not actually an error event, but instead ends in a nonzero state. In this case, the last error event cannot be moved and the set S_2 makes

$$\binom{N - l_2 + (n_2 - 1)}{(n_2 - 1)} \tag{10}$$

contributions to A_{w,z_1+z_2}.

The contribution to the distance spectrum of a Turbo code due to any pair of ordered sets S_1 and S_2, averaged over the ensemble of pseudorandom interleavers, can now be computed using (7)–(10). If the codeword in the second constituent encoder happens to end in the all-zero state, then the contribution of an S_1 and S_2 to A_{w,z_1+z_2} is given by

$$\frac{\binom{N - l_1 + n_1}{n_1}\binom{N - l_2 + n_2}{n_2}}{\binom{N}{w}}. \tag{11}$$

If the codeword in the second constituent encoder does not end in the all-zero state, then the contribution to A_{w,z_1+z_2} is given by

$$\frac{\binom{N - l_1 + n_1}{n_1}\binom{N - l_2 + (n_2 - 1)}{(n_2 - 1)}}{\binom{N}{w}}. \tag{12}$$

The expressions in (11) and (12) can now be used to explore the effect of changing the interleaver size on the distance spectrum of an "average" Turbo code.

Since we are primarily concerned with low-weight codewords in the distance spectrum, we assume that $N \gg n_1, n_2, l_1$, and l_2. If this is not true, then S_1 and S_2 either contain a large number of short error events or a few long error events. In both cases, it is very unlikely that the result is a codeword of low weight. With this assumption, (11) can be approximated by

$$\frac{w!}{n_1! n_2!} N^{n_1 + n_2 - w} \tag{13}$$

where, without loss of generality, it is assumed that $n_1 \geq n_2$. Since each error event is caused by an information sequence of weight at least two, $w \geq 2n_1$. The behavior of (13) for increasing N can be broken down into three cases:

1) $n_1 > n_2$:
   The exponent of N is strictly negative and the contribution to A_{w,z_1+z_2} decreases as N increases.
2) $n_1 = n_2$ and $w > 2n_1$:
   The exponent of N is strictly negative and the contribution to A_{w,z_1+z_2} decreases as N increases.
3) $n_1 = n_2$ and $w = 2n_1$:
   The exponent of N is exactly zero and the contribution to A_{w,z_1+z_2} converges to a finite value as N increases.

For $N \gg n_1, n_2, l_1$, and l_2, (12) can be approximated by

$$\frac{w!}{n_1!(n_2 - 1)!} N^{n_1 + n_2 - w - 1} \tag{14}$$

where $w \geq 2n_1$. However, since the tail in the second encoder may be caused by an information sequence of weight 1, we also have $w \geq 2n_2 - 1$. The behavior of (14) for increasing N can be broken down into three cases:

1) $n_1 > n_2$:
   Since $w \geq 2n_1$, the exponent of N is strictly negative and the contribution to A_{w,z_1+z_2} decreases as N increases.

2) $n_1 = n_2$:
Again, since $w \geq 2n_1$, the exponent of N is strictly negative and the contribution to A_{w,z_1+z_2} decreases as N increases.
3) $n_1 < n_2$:
Since $w \geq 2n_2 - 1$, the exponent of N is strictly negative and the contribution to A_{w,z_1+z_2} decreases as N increases.

The following lemma can now be stated. A similar result was proven in [9].

Lemma 1: Given a Turbo code based on two systematic feedback encoders and a pseudorandom interleaver of length N in which the first encoder is assumed to be forced back to the all-zero state, the contribution of two ordered sets of error events S_1 and S_2 with the same information weight to the distance spectrum of the Turbo code, averaged over all pseudorandom interleavers of length N, converges to a nonzero constant as $N \to \infty$, if and only if

1) S_2 leaves the second encoder in the all-zero state;
2) S_1 and S_2 contain the same number of error events;
3) each error event in S_1 and S_2 is caused by a weight two information sequence.

In all other cases, the contribution goes to zero as $N \to \infty$.

For each term $A_{w,z}$ in the input redundancy weight enumerating function of a Turbo code there is a finite number of pairs of sets S_1 and S_2 that contribute to it. If either set contains a long error event, then it is possible that that pair will be excluded for small interleavers. As the interleaver size increases, eventually all pairs of sets will be allowed and any further increase in N will not result in additional pairs of sets contributing to $A_{w,z}$. Also, as $N \to \infty$, $A_{w,z}$ will be determined by pairs of S_1 and S_2 that satisfy the three conditions of Lemma 1, and thus each $A_{w,z}$ will converge to a finite value. Since each spectral line is a finite sum of $A_{w,z}$ terms, each spectral line converges to a finite value as the interleaver size increases.

The convergence of each spectral line to a finite value as the size of the interleaver increases results in spectral thinning. That is, for small interleavers there may be pairs of sets S_1 and S_2 that do not satisfy the conditions of Lemma 1, but which contribute to the multiplicity of a low-weight spectral line. As the size of the interleaver increases the number of these aberrational sets decreases until the spectral line reaches its final value as determined by Lemma 1. This process of spectral thinning is represented graphically in Fig. 10, which depicts the thinning of low-weight codewords as the size of the interleaver increases for hypothetical distance spectra. It is this thinning of the distance spectrum that enables the free-distance asymptote of a Turbo code to dominate the performance for low SNR and thus to achieve near-capacity performance.

A. Primitive Polynomials and Free Distance

We now consider the ramifications of Lemma 1 with respect to the free-distance codewords of a Turbo code. That is, what does Lemma 1 imply about the information sequences that generate the free-distance codewords in an "average" Turbo code?

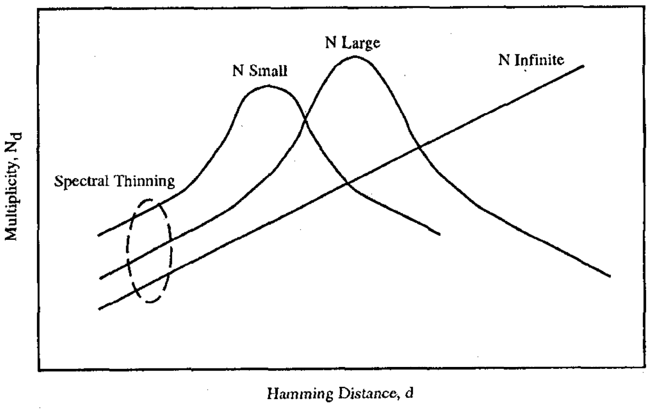

Fig. 10. Graphical representation of spectral thinning for increasing interleaver size.

For an "average" Turbo code, Lemma 1 states that as the size of the interleaver increases each spectral line is the result of contributions only from pairs of ordered sets of error events in which each error event is caused by a weight-two information sequence. It is reasonable to expect that the free-distance spectral line will be among the first spectral lines to converge to its final value. Thus for reasonably large interleavers, the free distance will be determined by the sets S_1 and S_2 satisfying the conditions of Lemma 1. Let s_1 and s_2 be any pair of error events caused by a weight-two information sequence that results in a minimum-weight parity sequence in the first and second constituent encoders, respectively. Note that there may be more than one such pair of minimum-weight error events for the constituent encoders.

A free-distance codeword in an "average" Turbo code must be the result of sets S_1 and S_2 that consist of only those minimum-weight error events s_1 and s_2, respectively. Furthermore, since each additional error event in either S_1 and S_2 adds weight to the codeword, S_1 and S_2 must each contain only one minimum-weight error event. (If each error event does not add weight, then a weight-two information sequence exists that generates a zero weight parity sequence and the free distance of the code would be 2.) Thus the free-distance codewords of an "average" Turbo code are caused by weight-two information sequences, provided the interleaver is large enough.

Therefore we have the following Lemma.

Lemma 2: For an "average" Turbo code, as the size of the interleaver N approaches ∞:

1) the free distance codewords are caused by information sequences of weight 2;
2) the free distance of an "average" Turbo code is maximized by choosing constituent encoders that have the largest output weight for weight-two information sequences.

Based on this lemma, we now present an intuitive argument showing that choosing the feedback polynomial $h_0(D)$ in a $(2, 1, \nu)$ systematic feedback encoder to be a primitive polynomial maximizes the output weight for weight-two information sequences. It follows that the free distance of an "average" Turbo code is maximized by using a primitive polynomial as the feedback polynomial in the constituent encoders. This result was independently derived in [9] using the transfer function of an "average" Turbo code.

The generator matrix of a $(2, 1, \nu)$ systematic feedback encoder is given by

$$G_{\text{fb}}(D) = \begin{bmatrix} 1 & \dfrac{h_1(D)}{h_0(D)} \end{bmatrix}$$

where $h_1(D)$ and $h_0(D)$ are referred to as the feedforward and feedback polynomials, respectively, and $h_0(D)$ is of degree ν. Since only information sequences of weight 2 are being considered, the systematic output contributes weight 2 to the overall codeword weight for all the encoders being considered. Therefore, only the weight contributed by the parity sequence, that is,

$$y(D) = x(D)\,\frac{h_1(D)}{h_0(D)},$$

needs to be maximized. Furthermore, since we are concerned only with the choice of $h_0(D)$, $h_1(D)$ is assumed to be a polynomial such that $h_0(D)$ and $h_1(D)$ are relatively prime. (There is empirical evidence that the choice of both polynomials can affect the performance of the code [28], [30], but we will not address that issue in this paper.)

Let $1 + D^K$, for some finite K, be the shortest input sequence of weight 2 that generates a finite-length codeword. The resultant parity sequence is

$$y(D) = (1 + D^K)\,\frac{h_1(D)}{h_0(D)} = \frac{h_1(D)}{h_0(D)} + D^K\,\frac{h_1(D)}{h_0(D)}.$$

Since $y(D)$ is of finite length, $h_1(D)/h_0(D)$ must be periodic with period K. Increasing the period K increases the length of the shortest weight 2 input sequence that generates a finite-length codeword and therefore increases the length of that codeword. Intuitively, one would expect that increasing its length would result in the codeword gaining weight. That is, on average, half of the added bits would be ones.

A strictly proper rational function of two polynomials, like $h_1(D)/h_0(D)$, is periodic with period $K \leq 2^\nu - 1$. The period is maximized, that is, $K = 2^\nu - 1$, when $h_0(D)$ is a primitive polynomial. Since the free distance of an "average" Turbo code is determined by information sequences of weight 2, for sufficiently large interleavers the free distance will be maximized by maximizing K. Therefore, choosing $h_0(D)$ to be a primitive polynomial will result in a larger free distance for an "average" Turbo code.

To test this, we compare a $(37, 21, 400)$ Turbo code which has $K = 5$ and free distance $d_{\text{free}} = 6$ to a $(23, 35, 400)$ Turbo code. Both codes are punctured as in [1]. The feedback polynomial $h_0 = 23$ in the second Turbo code is a primitive polynomial of degree $\nu = 4$ and thus $1/h_0$ has a period of $K = 2^\nu - 1 = 15$. The free distance of this Turbo code was found to be $d_{\text{free}} = 10$ [12], [28]. Fig. 11 shows simulation results for these two codes using the iterative decoding algorithm of [1] with 18 iterations. As expected, the second Turbo code performs better at moderate and high SNR's because its free distance asymptote is steeper due to the increased free distance.

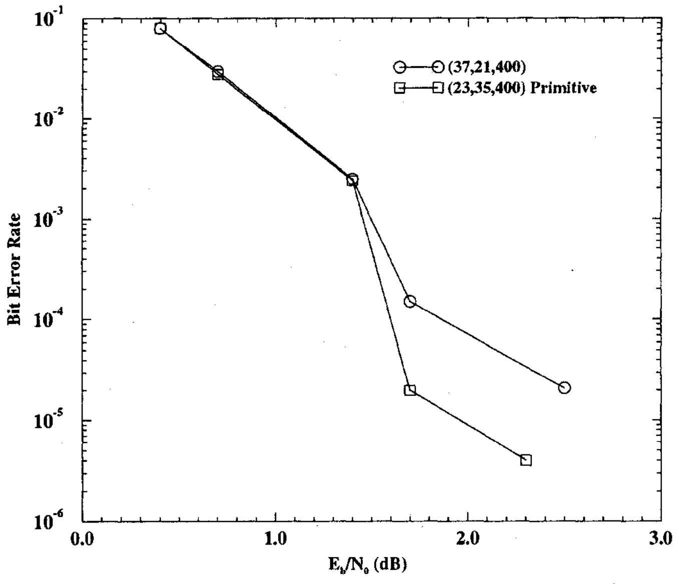

Fig. 11. Simulation results for a $(37, 21, 400)$ Turbo code and a $(23, 35, 400)$ primitive Turbo code.

V. Conclusions

The excellent performance of Turbo codes may be explained in terms of the distance spectrum of the code. The "error floor" observed in simulations of Turbo codes is a manifestation of the free-distance asymptote. Since Turbo codes have relatively low free distances, the free-distance asymptote has a shallow slope, and thus the performance curves flatten out at moderate to high SNR's. The "error floor" may be lowered by increasing the size of the interleaver for a fixed free distance, that is, by reducing the effective multiplicity of the code. Alternately, for fixed interleaver lengths, the performance may be improved for moderate and high SNR's by increasing the free distance. Choosing primitive polynomials as the feedback polynomials in the constituent encoders usually results in an increased free distance.

The exceptional performance of Turbo codes at low SNR's is due to the sparse distance spectrum and the resultant ability of the code to follow the free-distance asymptote at moderate to low SNR's. The use of systematic feedback encoders and pseudorandom interleavers results in spectral thinning, in which information sequences which generate low-weight parity sequences from the first constituent encoder are interleaved with high probability to information sequences that generate high-weight parity sequences in the second constituent encoder. Spectral thinning is enhanced by increasing interleaver lengths. For very large interleavers, spectral thinning results in a sparse distance spectrum in which the first several spectral lines are determined solely by input sequences of weight two. Thus spectral thinning results in few low-weight codewords and a large number of codewords of "average" weight. This is very similar to the type of distance spectrum achieved by "random-like" codes [4].

In a more philosophical light, Turbo codes remind us that information-theoretical arguments imply that long block lengths, but not necessarily large free distances, are required to achieve capacity at moderate BER's. Thus like convolutional codes, Turbo codes are a class of codes that achieve long block lengths, but without the corresponding increased density of the distance spectrum common to convolutional codes, and for which a practical, albeit nontrivial, decoding algorithm exists. In addition, Turbo codes are time-varying due to the pseudorandom interleaver, and the time-varying structure is essential in achieving the thin distance spectrum that results in near-capacity performance at moderate BER's. This suggests that some effort should be made to find other classes of time-varying codes, and decoding algorithms, that have thin distance spectra, rather than just large free distances. Finally, since, in fact, long block lengths are required to achieve near-capacity performance at moderate BER's, only modest coding gains will be achievable in systems that use relatively short block lengths.

References

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

[2] C. Berrou and A. Glavieux, "Turbo-codes: General principles and applications," in *Proc 6th Tirrenia Int. Workshop on Digital Communications* (Tirrenia, Italy, Sept. 1993), pp. 215–226.

[3] G. Battail, C. Berrou, and A. Glavieux, "Pseudo-random recursive convolutional coding for near-capacity performance," in *Proc. Communication Theory Mini-Conf., Globecom '93* (Houston, TX, Dec. 1993), pp. 23–27.

[4] G. Battail, "On random-like codes," preprint.

[5] ——, "Pseudo-random Turbo-codes," in *Proc. Int. Symp. on Signals, Systems and Electronics* (San Francisco, CA, Oct. 25–27, 1995), pp. 419–422.

[6] S. Benedetto and G. Montorsi, "Average performance of parallel concatenated block codes," *Electron. Lett.*, vol. 31, no. 3, pp. 156–158, Feb. 2, 1995.

[7] ——, "Performance evaluation of TURBO-codes," *Electron. Lett.*, vol. 31, no. 3, pp. 163–165, Feb. 2, 1995.

[8] ——, "Unveiling turbo codes: some results on parallel concatenated coding schemes," *IEEE Trans. Inform. Theory*, vol. 42, no. 2, pp. 409–428, Mar. 1996.

[9] ——, "Design of parallel concatenated convolutional codes," *IEEE Trans. Commun.*, vol. 44, no. 5, pp. 591–600, May 1996.

[10] J. D. Andersen, "The TURBO coding scheme," Rep. IT-146, Tech. Univ. of Denmark, Lyngby, June 1994.

[11] J. D. Andersen, "TURBO coding for deep space applications," in *Proc 1995 IEEE Int. Symp. on Information Theory* (Whistler, BC, Canada, Sept. 1995), p. 36.

[12] P. Robertson, "Illuminating the structure of parallel concatenated recursive systematic (TURBO) codes," in *Proc. GLOBECOM '94* (San Francisco, CA, Nov. 1994), vol. 3, pp. 1298–1303.

[13] J. Hagenauer and L. Papke, "Decoding 'Turbo'-codes with the soft output Viterbi algorithm," in *Proc. 1994 IEEE Int.. Symp. on Information Theory* (Trondheim, Norway, June 1994), p. 164.

[14] J. Hagenauer, P. Robertson, and L. Papke, "Iterative (TURBO) decoding of systematic convolutional codes with MAP and SOVA algorithms," in *Proc. ITG Conf. on "Source and Channel Coding"* (Frankfurt, Germany, Oct. 1994), pp. 1–9.

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

[16] Y. V. Svirid, "Weight distributions and bounds for turbo codes," *European Trans. Telecommun.*, vol. 6, no. 5, pp. 543–556, Sept.–Oct. 1995.

[17] Y. V. Svirid and S. Riedel, "Threshold decoding of turbo codes," in *Proc. 1995 IEEE Int. Symp. on Information Theory* (Whistler, BC, Canada, Sept. 1995), p. 39.

[18] D. Divsalar and F. Pollara, "Low-rate turbo codes for deep-space communications," in *Proc. 1995 IEEE Int. Symp. on Information Theory* (Whistler, BC, Canada, Sept. 1995), p. 35.

[19] D. Divsalar and F. Pollara, "Turbo codes for PCS applications," in *Proc. 1995 IEEE Int. Conf. on Communications* (Seattle, WA, June 1995).

[20] ——, "Turbo codes for deep-space communications," TDA Progr. Rep. 42–120, Jet Propulsion Lab., Pasadena, CA, pp. 29-39, Feb. 15, 1995.

[21] ——, "Multiple turbo codes for deep-space communications," TDA Progr. Rep. 42–121, Jet Propulsion Lab., Pasadena, CA, pp. 66–77, May 15, 1995.

[22] D. Divsalar, S. Dolinar, F. Pollara, and R. J. McEliece, "Transfer function bounds on the performance of turbo codes," TDA Progr. Rep. 42-122, Jet Propulsion Lab., Pasadena, CA, pp. 44–55, Aug. 15, 1995.

[23] S. Dolinar and D. Divsalar, "Weight distributions for turbo codes using random and nonrandom permutations," TDA Progr. Rep. 42-122, Jet Propulsion Lab., Pasadena, CA, pp. 56–65, Aug. 15, 1995.

[24] D. Divsalar and F. Pollara, "On the design of turbo codes," TDA Progr. Rep. 42-123, Jet Propulsion Lab., Pasadena, CA, pp. 99–121, Nov. 15, 1995.

[25] J.-F. Cheng and R. J. McEliece, "Unit-memory Hamming turbo codes," in *Proc. 1995 IEEE Int. Symp. on Information Theory* (Whistler, BC, Canada, Sept. 1995), p. 33.

[26] R. J. McEliece, E. R. Rodemich, and J.-F. Cheng, "The turbo decision algorithm," in *Proc. 33rd Annu. Allerton Conf. on Communication, Control and Computing* (Monticello, IL, Oct. 1995), p. 366.

[27] D. Divsalar and R. J. McEliece, "The effective free distance of turbo codes," *Electron. Lett.*, submitted for publication.

[28] J. Seghers, "On the free distance of TURBO codes and related product codes," Final Rep., Diploma Project SS 1995, no. 6613, Swiss Federal Institute of Technology, Zurich, Switzerland, Aug. 1995.

[29] J. Seghers, L. C. Perez, and D. J. Costello, Jr., "On selecting code generators for turbo codes," in *Proc. 33rd Annual Allerton Conf. on Communication, Control and Computing* (Monticello, IL, Oct. 1995), pp. 357–361.

[30] D. Arnold and G. Meyerhans, "The realization of the the turbo-coding system," Semester Project Rep., Swiss Federal Institute of Technology, Zurich, Switzerland, July 1995.

[31] A. S. Barbulescu and S. S. Pietrobon, "Interleaver design for turbo codes," *Electron. Lett.*, vol. 30, no. 25, p. 2107, Dec. 8, 1994.

[32] ——, "Terminating the trellis of turbo-codes in the same state," *Electron. Lett.*, vol. 31, no. 1, pp. 22–23, Jan. 5, 1995.

[33] ——, "Rate compatible turbo codes," *Electron. Lett.*, vol. 31, no. 7, pp. 535–536, Mar. 30, 1995.

[34] ——, "Interleaver design for three dimensional turbo codes," in *Proc. 1995 IEEE Int. Symp. on Information Theory* (Whistler, BC, Canada, Sept. 1995), pg. 37.

[35] M. C. Reed and S. S. Pietrobon, "Turbo-code termination for short frames," *Electron. Lett.*, submitted for publication.

[36] O. Joerssen and H. Meyr, "Terminating the trellis of turbo-codes," *Electron. Lett.*, vol. 30, no. 16, pp. 1285–1286, Aug. 4, 1994.

[37] P. Jung and M. Naßhan, "Performance evaluation of turbo codes for short frame transmission systems," *Electron. Lett.*, vol. 30, no. 2, pp. 111–113, Jan. 20, 1994.

[38] ——, "Dependence of the error performance of turbo-codes on the interleaver structure in short frame transmission systems," *Electron. Lett.*, vol. 30, no. 4, pp. 285–288, Feb. 17, 1994.

[39] P. Jung, "Novel low complexity decoder for turbo-codes," *Electron. Lett.*, vol. 31, no. 2, pp. 86–87, Jan. 19, 1995.

[40] R. Podemski, W. Holubowicz, C. Berrou, and A. Glavieux, "Distance spectrum of the turbo-codes," in *Proc. 1995 IEEE Int. Symp. on Information Theory* (Whistler, BC, Canada, Sept. 1995), p. 34.

[41] R. Podemski, W. Holubowicz, C. Berrou, and G. Battail, "Hamming distance spectra of turbo-codes," *Ann. Télécommun.*, vol. 50, no. 9–10, pp. 790–797, Sept.–Oct. 1995.

[42] G. Caire, G. Taricco, and E. Biglieri, "On the convergence of the iterated decoding algorithm," in *Proc. 1995 IEEE Int. Symp. on Information Theory* (Whistler, BC, Canada, Sept. 1995), p. 472.

[43] N. Wiberg, H.-A. Loeliger, and R. Kötter, "Codes and iterative decoding on general graphs," *European Trans. Telecommun.*, vol. 6, no. 5, pp. 513–525, Sept.–Oct. 1995.

[44] S. Lin and D. J. Costello, Jr., *Error Control Coding: Fundamentals and Applications*. Englewood Cliffs, NJ: Prentice-Hall, 1983.

[45] M. Cedervall and R. Johannesson, "A fast algorithm for computing distance spectrum of convolutional codes," *IEEE Trans. Inform. Theory*, vol. 35, no. 6, pp. 1146–1159, Nov. 1989.