Turbo Codes and Iterative Processing

Matthew C. Valenti

Mobile and Portable Radio Research Group
Virginia Polytechnic Institute and State University
Blacksburg, VA 24061-0350, USA
email: valenti@vt.edu, phone: 540-231-2925, fax: 540-231-2968

Abstract

In 1993, a group of researchers from France presented a new class of error correction codes and an associated iterative decoding technique. These codes, termed "turbo codes", were shown to achieve performance within 0.7 decibels of the Shannon capacity limit. This constituted a significant gain in power efficiency over other coding techniques known at the time. In this paper, an overview of the turbo encoding and decoding process is presented. Implementation issues are discussed, including an overview of the factors that influence turbo code performance. Finally, extensions of the turbo decoding algorithm to other problems in communications are presented.

1 Introduction

Turbo codes are a new class of error correction codes that were introduced along with a practical decoding algorithm in [1]. The importance of turbo codes is that they enable reliable communications with power efficiencies close to the theoretical limit predicted by Claude Shannon in [2]. Since their introduction, turbo codes have been proposed for low-power applications such as deep-space and satellite communications, as well as for interference limited applications such as third generation cellular and personal communication services. Turbo codes have been a "hot topic" in the literature over the past five years, yet there is a relative lack of basic papers serving as a starting point for people interested in this topic[1]. This paper attempts to serve as an informal tutorial introduction for those interested in learning about turbo codes, and as a survey of some of the important issues related to turbo codes.

This paper begins with a review of the concept of channel capacity in order to illustrate the minimum theoretical signal-to-noise ratio required for communications with arbitrary reliability. Next, a review of convolutional codes, which form the foundation of turbo codes, is presented. A description of the turbo-encoding process is presented, followed by a discussion of why turbo codes perform so well. Next, the algorithm used to decode turbo codes is exposed. Finally, a discussion of other applications of the turbo decoding algorithm is presented.

2 Channel Capacity

The capacity of a channel, which was first introduced 50 years ago by Claude Shannon [2], is the theoretical maximum data rate that can be supported by the channel with vanishing error probability. In this discussion, we restrict our attention to the additive white Gaussian noise (AWGN) channel

$$y = x + z. \tag{1}$$

Here, x is a modulated symbol modeled by a random process with zero mean and variance E_s (E_s is the energy per symbol). For the specific case of antipodal signaling[2], $x = \pm\sqrt{E_s}$. z is a sample from an additive white Gaussian noise process with zero mean and variance $N_o/2$. The capacity of the AWGN channel is given by

$$C = \frac{1}{2}\log_2\left(1 + \frac{2E_s}{N_o}\right) \tag{2}$$

bits per channel use.

Signaling at rates close to capacity is achieved in practice by *error correction coding*. An error correction code maps data sequences of k bits to code words of n symbols. Because $n > k$, the code word contains structured redundancy. The *code rate*, $r = k/n$ is a measure of the spectral efficiency of the code. In order to achieve reliable communications, the code rate cannot exceed the channel capacity ($r \leq C$). The minimum theoretical signal to noise ratio[3] E_b/N_o required to achieve arbitrarily reliable

[1] However, good tutorial introductions can be found in [3], [4], and [5].

[2] Binary phase shift keying (BPSK) is a type of antipodal signaling.

[3] $E_b = E_s/r$ is the energy per bit.

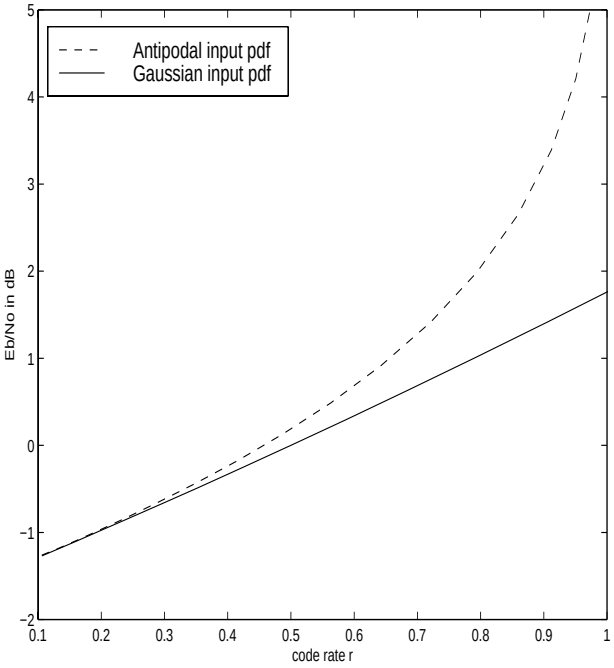

Figure 1: Theoretical minimum E_b/N_o required to achieve channel capacity for code rate r in AWGN.

communications[4] can be found by rearranging equation (2)

$$\frac{E_b}{N_o} \geq \frac{1}{2r}\left(2^{2r}-1\right). \quad (3)$$

This is the minimum E_b/N_o required for any arbitrary distribution for the input x. Equation (3) can be satisfied with equality only if the input is Gaussian distributed. If the input is antipodal instead of Gaussian, slightly higher E_b/N_o is required. Plots of the minimum E_b/N_o required to achieve channel capacity as a function of code rate r are given in Figure 1 for both Gaussian distributed inputs and antipodal inputs.

Shannon's proof of the Channel Coding Theorem[5] used a random coding argument. He showed that if one selects a rate $r < C$ code at random, then the bit error probability approaches zero as the block length n of the code approaches infinity . However, random codes are not practically feasible. In order to be able to encode and decode with reasonable complexity, codes must possess some sort of structure. Unfortunately, structured codes perform considerably worse than random codes. This is the basis of the coding paradox:

[4] Arbitrary reliability means signaling with bit error probability $P_b = \epsilon$, where $\epsilon > 0$ is an arbitrarily small value.

[5] The Channel Coding Theorem states that it is theoretically possible to signal with arbitrary reliability at rates up to channel capacity.

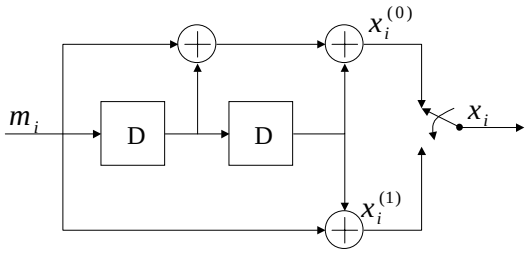

Figure 2: Example rate $r = 1/2$ convolutional encoder with constraint length $K = 3$.

> *Almost all codes are good, except those we can think of.*
>
> J. Wolfowitz

As will become apparent in the following sections, the reason that turbo codes perform so well is that they attack the coding paradox head on. On the one hand, they have structure and therefore can be efficiently encoded and decoded. But on the other hand they appear to be random and thus emulate the good performance of random codes.

3 Convolutional Codes

Before the introduction of turbo codes, power efficient communications was achieved by either a strong convolutional code or the serial concatenation of a convolutional code with a block code[6]. A convolutional code adds redundancy to a continuous stream of input data by using a linear shift register. For a rate $r = k/n$ convolutional encoder, each set of n output symbols is a linear combination of the current set of k input bits and m bits stored in the shift register. The total number of bits that each output depends on is called the *constraint length*, and is denoted by K. An example convolutional encoder is shown in Figure 2. The rate of this code is $r = 1/2$ and the constraint length is $K = 3$.

A convolutional code can be made systematic[7] without affecting its minimum distance properties by feeding back one of the outputs to the input. Such a code is called a Recursive Systematic Convolutional (RSC) code, and is the basic building block for turbo codes. An example RSC encoder derived from the nonsystematic encode of Figure 2 is shown in Figure 3.

[6] For example, the IS-95 CDMA cellular standard uses a constraint length $K = 9$ convolutional code, while the deep space communication standard uses a constraint length $K = 7$ convolutional code concatenated with a Reed Solomon code.

[7] A systematic code is one for which each n symbol code word contains the k data bits. The remaining $n - k$ bits are called *parity bits*.

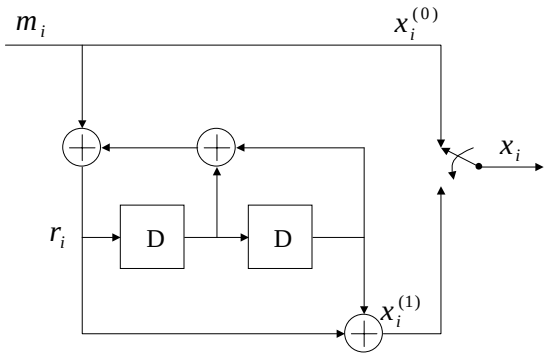

Figure 3: Example recursive systematic convolutional (RSC) encoder.

The decoder for a convolutional code finds the most probable sequence of data bits $\hat{\mathbf{m}}$ given the received sequence $\mathbf{y}$

$$\hat{\mathbf{m}} = \arg\left\{\max_{\mathbf{m}} P[\mathbf{m}|\mathbf{y}]\right\}, \quad (4)$$

where $\mathbf{y}$ is the set of code symbols $\mathbf{x}$ observed through noise. Equation (4) can be solved using the Viterbi algorithm[8], in which case the solution is termed the "maximal likelihood" or ML solution. The complexity of the Viterbi algorithm when used to decode convolutional codes is $O(2^K)$.

4 Turbo Codes: Encoding

A turbo code is the parallel concatenation of two RSC codes separated by an interleaver. An example turbo encoder is shown in Figure 4. Here, the two encoders are identical rate 1/2 RSC encoders. The upper encoder receives the data directly, while the lower encoder receives the data after it has been interleaved by a permutation function α. The interleaver α is in general a *pseudo-random* interleaver — that is it maps bits in position i to position $\alpha(i)$ according to a prescribed, but randomly generated rule. The interleaver operates in a blockwise fashion, interleaving L bits at a time, and thus turbo codes are actually block codes. Since both encoders are systematic and receive the same set of data (although in permuted order), only one of the systematic outputs needs to be sent. By convention, the systematic output of the top encoder is sent while the systematic output of the lower encoder is not transmitted. However, the parity outputs of both encoders are transmitted. The overall code rate of a turbo code composed from the parallel concatenation of two rate 1/2 systematic codes is $r = 1/3$. This code rate can be made higher

[8] The Viterbi algorithm goes beyond the scope of this paper, but an excellent tutorial can be found in [6].

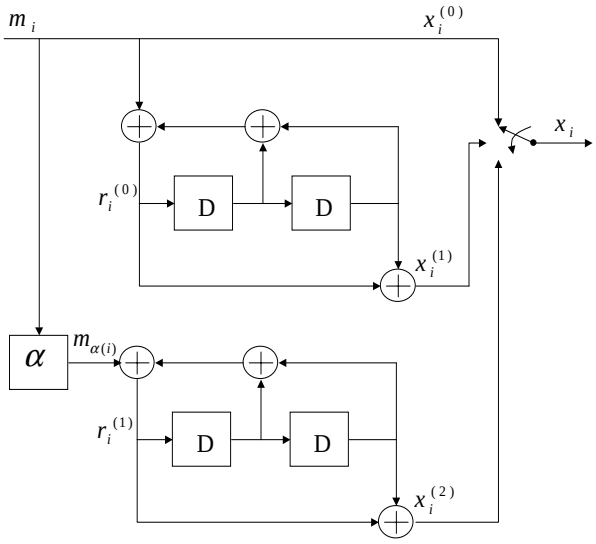

Figure 4: Example turbo encoder.

by *puncturing*[9]. The code rate of a turbo code is typically increased to $r = 1/2$ by only transmitting the odd indexed parity bits from the upper encoder and the even indexed parity bits from the lower encoder (along with all the systematic bits from the top encoder).

5 Turbo Codes: Decoding

As with convolutional codes, an ML solution can be obtained by using Equation (4) and the Viterbi algorithm. However, due to the presence of the interleaver, the complexity of Viterbi algorithm when used to decode turbo codes is $O(2^L)$, where L is the size of the data frame (and interleaver). Due to the prohibitive complexity of the ML solution to decoding turbo codes, we seek a reduced complexity, albeit suboptimal, solution. In particular, a good estimate of the data can be found by solving the following *system* of equations:

$$\Lambda_i^{(1)} = \log \frac{P[m_i = 1|\mathbf{y}^{(0)}, \mathbf{y}^{(1)}, \mathbf{z}^{(2)}]}{P[m_i = 0|\mathbf{y}^{(0)}, \mathbf{y}^{(1)}, \mathbf{z}^{(2)}]} \quad (5)$$

$$\tilde{\Lambda}_i^{(2)} = \log \frac{P[\tilde{m}_i = 1|\tilde{\mathbf{y}}^{(0)}, \mathbf{y}^{(2)}, \tilde{\mathbf{z}}^{(1)}]}{P[\tilde{m}_i = 0|\tilde{\mathbf{y}}^{(0)}, \mathbf{y}^{(2)}, \tilde{\mathbf{z}}^{(1)}]}, \quad (6)$$

where $\mathbf{y}^{(0)}$ is the observed systematic bits, $\mathbf{y}^{(1)}$ is the observed parity bits from encoder one, and $\mathbf{y}^{(2)}$ is the observed parity bits from encoder two. A tilde over a variable represents its interleaved value, i.e. $\tilde{\mathbf{y}}$ is the interleaved version of $\mathbf{y}$. $\mathbf{\Lambda}$ is the a posteriori Log-likelihood ratio (LLR), and $\mathbf{z}$ is the so-called *extrinsic*

[9] Puncturing is the process of deleting (i.e. not transmitting) a subset of the parity bits.

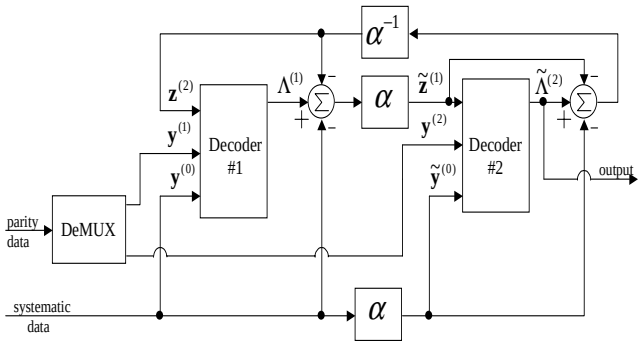

Figure 5: Turbo decoder

information which is related to the LLR by

$$z_i^{(1)} = \Lambda_i^{(1)} - y_i^{(0)} - z_i^{(2)} \quad (7)$$

$$\tilde{z}_i^{(2)} = \tilde{\Lambda}_i^{(2)} - \tilde{y}_i^{(0)} - \tilde{z}_i^{(1)}. \quad (8)$$

The system of Equations (5) to (8) can be solved iteratively using the structure shown in Figure 5. Here Decoder One determines the solution to (5) and Decoder Two determines the solution to (6). Each decoder passes information to the other decoder, which in turn refines the estimated a posteriori probabilities using information derived by the other decoder. The final estimate of the data is obtained by hard limiting the output of one of the decoders (by convention the second decoder's output is used)

$$m_i = \begin{cases} 1 & \text{if } \Lambda_i^{(2)} > 0 \\ 0 & \text{if } \Lambda_i^{(2)} < 1 \end{cases} \quad (9)$$

Turbo codes get their name from the feedback structure of Figure 5 and its analogy to a turbo engine. In fact, there is nothing "turbo" about turbo codes, rather the turbo effect comes from the decoder implementation.

The a posteriori LLR's of (5) and (6) are computed using a derivation of the symbol-by-symbol maximum a posteriori (MAP) algorithm of [7]. Although the algorithm of [7] can be used directly to compute the LLR's, the algorithm is computationally complex and sensitive to finite-precision numerical representations. These problems are alleviated by performing the algorithm in the log-arithmetic domain, as presented in [8] and [9]. The resulting algorithm is termed the *Log-MAP* algorithm. The algorithm consists of two instances of the Viterbi algorithm — one performing a forward recursion and the other performing a backwards recursion. Thus the complexity of the Log-MAP algorithm is twice that of the Viterbi algorithm. The details of the log-MAP algorithm go beyond the scope of this paper, although a thorough explanation can be found in [10].

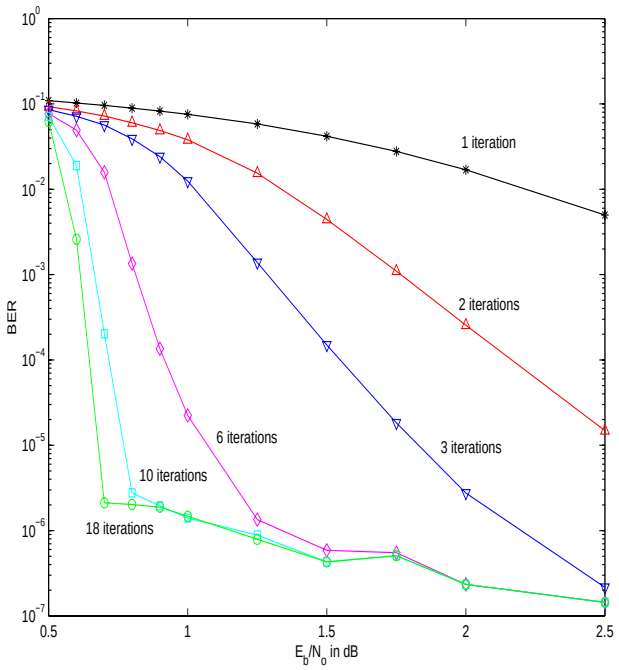

Figure 6: Performance of turbo code with $r = 1/2$, $K = 5$, $L = 65,536$ for various numbers of decoder iterations.

The simulated performance of a turbo code with BPSK modulation is shown in Figure 6. This is the rate $r = 1/2$ turbo code created by the parallel concatenation of two $K = 5$ RSC encoders that was originally reported in [1]. The interleaver size of this code is $L = 65,536$ bits. The performance after various numbers of decoder iterations is shown — note that performance improves as the number of decoder iteration increases. After 18 iterations a bit error rate of 10^{-5} is reached at just slightly below 0.7 dB. Comparing this to Figure 1, we see that the performance is within 0.7 dB of the unconstrained-input channel capacity limit, or within 0.5 dB of capacity when the input is constrained to be antipodal[10]. Also note the presence of an *error floor*[11] as the signal-to-noise ratio increases. This phenomenon will be discussed more in the next section.

6 Turbo Codes: Performance Factors

There are many factors that affect the performance of turbo codes. The most influential parameter is the interleaver size. As the frame/interleaver size increases, performance improves. In Figure 7, the

[10]Since BPSK modulation is assumed, the input is indeed antipodal.

[11]This is not a true error floor, rather it is a severe shallowing of the BER curve.

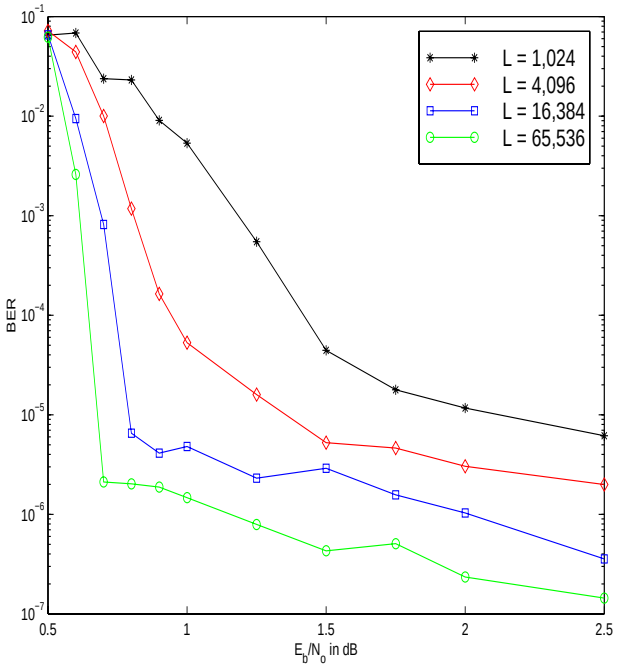

Figure 7: Performance of turbo code with $r = 1/2$, $K = 5$, and 18 decoder iterations for various interleaver sizes.

performance of the $r = 1/2$, $K = 5$ turbo code is shown for various interleaver sizes. Note that as the interleaver gets smaller, performance degrades. This would imply that one would select the largest possible interleaver size. However, as the interleaver size increases so does decoder latency, since the entire code word must be received before decoding can be completed. Thus turbo codes possess an inherent tradeoff between performance and latency. In [11], it is shown that this tradeoff can be exploited for wireless multimedia communication systems.

Another parameter affecting performance is the overall code rate. Just like other codes, performance improves as the code rate becomes lower. When code rates higher than 1/3 are used, then the particular puncturing pattern that is used impacts the performance. Like convolutional codes, the constraint length also influences performance. However, the impact that constraint length has on performance is weak, and thus only the short constraint lengths of K=3, 4, or 5 are considered for practical turbo codes. The interleaver design plays a significant role in the performance of a turbo code, particularly for higher signal-to-noise ratios. In general, a randomly chosen interleaver design will give good performance, while highly structured interleavers such as the "block interleaver" should be avoided.

The choice of decoding algorithm and number of decoder iterations also influences performance. As can be seen in Figure 5, performance improves as the number of iterations increases. This improvement follows a law of diminishing returns. Also, the number of iterations required is a function of the interleavers size — bigger interleavers require more iterations. For example, a turbo code with an interleaver size of 16,384 bits only needs about 9 iterations of decoding in practice. The question of whether or not the iterative decoding technique used by turbo codes converges to the ML solution is still an open ended question.

While turbo codes offer extraordinary performance for bit error rates down to about 10^{-5}, the performance for very small bit error rates is not very impressive. Note the bit error floors that appear for low bit error rates in Figures 6 and 7. For high signal-to-noise ratios, it may in fact be better to use a convolutional code. This phenomenon can be explained in terms of the Hamming distance spectrum of turbo codes. For high signal-to-noise ratios, the performance of a code is approximated by its *free distance asymptote* [4]

$$P_b = w_{eff} Q\left(\sqrt{d_{free}\frac{2rE_b}{N_o}}\right), \qquad (10)$$

where d_{free} is the *free distance*[12] of the code, and w_{eff} is the effective multiplicity of code words of weight d_{free}. The slope of the free distance asymptote is determined by the argument of the Q function, and thus by the free distance. Bit error performance can be improved in two ways: (1) By increasing the free distance, and thus making the asymptote steeper, or (2) Decreasing the effective multiplicity, thereby lowering the entire BER curve. The design of convolutional codes focuses on maximizing free distance, but the design of turbo codes focuses on minimizing the effective multiplicity.

The constraint length $K = 15$ convolutional code[13] has d_{free} of 18, while for the $K = 5$ turbo code, d_{free} is only 6. Thus the free distance asymptote of the convolutional code is much steeper than the free distance asymptote of the turbo code. However, w_{eff} of the convolutional code is 187 while w_{eff} for the turbo code is only $6/N$. For $N = 65,536$ this translates to $w_{eff} \approx .0001$, and thus the coefficient of the turbo code's free distance asymptote is much lower than the convolutional code's.

The number of free distance code words in a turbo code is very small due to the combination of pseudo-random interleaving, recursive encoding, and parallel code concatenation. Due to their recursive nature,

[12]The free distance is the minimum Hamming weight of all non-zero code words. The Hamming weight is found by counting the number of ones in the code word.

[13]The $K = 15$ was chosen for comparison purposes because its decoder has roughly the same complexity as the $K = 5$ turbo code with 18 iterations.

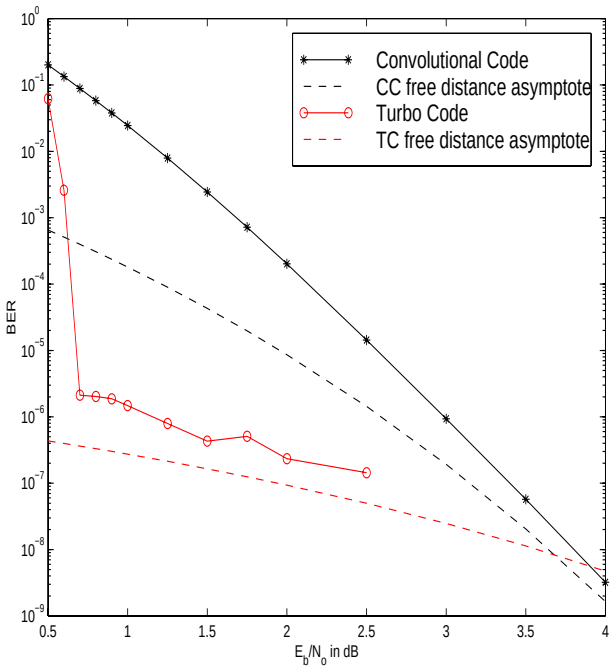

Figure 8: Performance two $r = 1/2$ codes of comparable complexity: (1) Turbo code with $K = 5$ and $L = 65,536$; (2) Convolutional code with $K = 15$.

the impulse response of RSC encoders is infinite. Thus, most input sequences cause large weight output sequences. However there are a few input sequences that cause small weight outputs. But because of interleaving, the two RSC encoders in a turbo code do not receive their inputs in the same order. Thus the probability that both encoders produce small weight outputs is very small, accounting for the small effective multiplicity. For more information about the performance of turbo codes, please review the work of Benedetto [12], [13].

The free distance asymptotes are shown in Figure 8. Note that for BER of 10^{-5}, the turbo code is about 2 decibels more power efficient than the convolutional code. However, at a BER of 10^{-8} the asymptotes of the turbo code and convolutional code intersect, implying that for signal-to-noise ratios greater than about 4 dB the turbo code would actually perform worse than the convolutional code. Recent work has shown that with serially concatenated turbo codes, the bit error floor disappears [14].

7 Other Applications of Iterative Processing

Modern communication receivers typically consist of a cascade of signal processing intensive subsystems, each optimized to perform a single task. Examples of subsystems include adaptive antennas, equalizers, multiuser detectors, and channel decoders. In a "conventional" receiver, the interface between subsystems involves the passing of bits, or hard-decisions, down the stages of the chain. Whenever hard-decisions are made, information is lost and becomes unavailable to subsequent stages. Additionally, stages at the beginning of the processing chain do not benefit from information derived by stages further down the chain. The interface between stages can be greatly improved by using the same strategy used to decode turbo codes. In [15] the term "turbo processing" was coined to describe the general strategy of iterative feedback decoding or detection.

With turbo processing, each subsystem is implemented with a Soft-Input, Soft-Output (SISO) algorithm, such as log-MAP. Soft-decision values, typically in the form of log-likelihood ratios, are passed down the chain and refined by subsequent stages. The soft-output of the final stage is then fed back to the first stage and a second iteration of processing is initiated. Several iterations of turbo processing can be executed, although as with turbo codes, a law of diminishing returns limits the maximum processing gain. Turbo processing can be used to combine channel decoding with source decoding [16], symbol detection [15], equalization [17], or multiuser detection [18].

An interesting example of turbo processing is "turbo equalization", which is a method of combining equalization with channel decoding [17]. An *equalizer* is a signal processing subsystem that compensates for the intersymbol interference (ISI) created by frequency selective channels. A frequency selective channel can be described as a rate $r = 1$ convolutional code defined over the field of real numbers. Following this analogy, the optimal decoding algorithm is a form of the Viterbi algorithm known as the Maximum Likelihood Sequence Estimator (MLSE) [19]. Thus the combination of a convolutional channel code with a frequency selective channel can be viewed as a serial concatenation of two convolutional codes, which can be decoded with a turbo decoder.

The notion of "turbo equalization" can be modified for use with coded multiple access channels. In a multiple access channel, several users transmit at the same time and frequency. The multiplicity of users gives rise to multiple access interference (MAI), which can be described as a form of time varying intersymbol interference. Thus the multiple access channel can be viewed as a rate $r = 1$ convolutional code with time varying coefficients taken over the field of real numbers. In keeping with the analogy, the optimal decoder is implemented using the Viterbi algorithm and is called the optimal multiuser detector (MUD) [20]. The combination of convolutional channel coding and multiple access channel is analogous to the serial concatenation of two convolutional codes,

and is suitable for turbo decoding. For examples of turbo multiuser detection, see [21], [18].

8 Summary

Turbo codes represent an important advancement in the area of power efficient communications. The extraordinary performance of turbo codes is due to the combination of parallel concatenated coding, recursive encoders, pseudo-random interleaving, and an iterative decoder structure. However, the small free distance of turbo codes limits the usefulness of turbo codes to bit error probabilities in the range of 10^{-4} to 10^{-6}. Also, turbo coded systems typically experience significant latency due to interleaving and interative decoding. The iterative decoding technique used by turbo codes can also be applied to other problems in communications such as joint equalization/decoding and joint multiuser-detection/decoding.

For more information about turbo codes and iterative processing, see the following web page:

`http://www.ee.vt.edu/valenti/turbo.html`

References

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

[2] C. E. Shannon, "A mathematical theory of communication," *Bell Sys. Tech. J.*, vol. 27, pp. 379–423 and 623–656, 1948.

[3] J. Hagenauer, "The turbo principle: Tutorial introduction and state of the art," in *Proc., Int. Symp. on Turbo Codes and Related Topics*, (Brest, France), pp. 1–11, Sept. 1997.

[4] L. C. Perez, "Turbo codes," in *Trellis Coding* (C. Schlegel, ed.), ch. 8, pp. 233–262, New York, NY: IEEE Press, 1997.

[5] B. Sklar, "A primer on turbo code concepts," *IEEE Commun. Magazine*, vol. 35, pp. 94–102, Dec. 1997.

[6] G. D. Forney, "The Viterbi algorithm," *Proc. IEEE*, vol. 61, pp. 268–278, Mar. 1973.

[7] L. R. Bahl, J. Cocke, F. Jeinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Trans. Inform. Theory*, vol. 20, pp. 284–287, Mar. 1974.

[8] P. Robertson, P. Hoeher, and E. Villebrun, "Optimal and sub-optimal maximum a posteriori algorithms suitable for turbo decoding," *European Trans. on Telecommun.*, vol. 8, pp. 119–125, Mar./Apr. 1997.

[9] A. J. Viterbi, "An intuitive justification and a simplified implemetation of the MAP decoder for convolutional codes," *IEEE J. Select. Areas Commun.*, vol. 16, pp. 260–264, Feb. 1998.

[10] M. C. Valenti, "Iterative detection and decoding for wireless communications," prelim report, Virginia Tech, Blacksburg, Virginia, September 1998.

[11] M. C. Valenti and B. D. Woerner, "Variable latency turbo codes for wireless multimedia applications," in *Proc., Int. Symp. on Turbo Codes and Related Topics*, (Brest, France), pp. 216–219, Sept. 1997.

[12] S. Benedetto and G. Montorsi, "Unveiling turbo codes: Some results on parallel concatenated coding schemes," *IEEE Trans. Inform. Theory*, vol. 42, pp. 409–428, Mar. 1996.

[13] S. Benedetto and G. Montorsi, "Design of parallel concatenated convolutional codes," *IEEE Trans. Commun.*, vol. 44, pp. 591–600, May 1996.

[14] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara, "Serial concatenation of interleaved codes: Performance analysis, design and iterative decoding," in *Proc., IEEE Int. Symp. on Inform. Theory*, (Ulm Germany), p. 106, June 1997.

[15] J. Lodge and M. Gertsman, "Joint detection and decoding by turbo processing for fading channel communications," in *Proc., Int. Symp. on Turbo Codes and Related Topics*, (Brest, France), pp. 88–95, Sept. 1997.

[16] J. Hagenauer, "Source-controlled channel decoding," *IEEE Trans. Commun.*, vol. 43, pp. 2449–2457, Sep. 1995.

[17] C. Douillard, M. Jezequel, C. Berrou, A. Picart, P. Didier, and A. Glavieux, "Iterative correction of intersymbol interference: Turbo-equalization," *European Trans. on Telecommun.*, vol. 6, pp. 507–511, Sept./Oct. 1995.

[18] M. C. Valenti and B. D. Woerner, "Iterative multiuser detection and convolutionally coded asynchronous DS-CDMA," in *Proc., IEEE PIMRC*, (Boston MA), pp. 213–217, Sept. 1998.

[19] G. D. Forney, "Maximum-likelihood sequence estimation of digital sequences in the presence of intersymbol interference," *IEEE Trans. Inform. Theory*, vol. 18, pp. 368–378, May 1972.

[20] S. Verdú, "Minimum probability of error for asynchronous Gaussian multiple-access channels," *IEEE Trans. Inform. Theory*, vol. 32, pp. 85–96, January 1986.

[21] M. Moher, "An iterative multiuser decoder for near-capacity communications," *IEEE Trans. Commun.*, vol. 46, pp. 870–880, July 1998.