

Multiple Turbo Codes

D. Divsalar and F. Pollara¹

Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA 91109

ABSTRACT: In this paper, we introduce multiple turbo codes and a suitable decoder structure derived from an approximation to the maximum a posteriori probability (MAP) decision rule, which is substantially different from the decoder for two-code-based encoders. We developed new rate 1/3 and 2/3 constituent codes to be used in the turbo encoder structure. These codes, for 2 to 32 states, are designed by using primitive polynomials. The resulting turbo codes have rates b/n , $b = 1, 2$ and $n = 3, 4$, and include random interleavers for better asymptotic performance. A rate 2/4 code with 16QAM modulation was used to realize a turbo trellis coded modulation (TTCM) scheme at 2 bit/sec/Hz throughput, whose performance is within 1 dB from the Shannon limit at BER= 10^{-5} .

I. INTRODUCTION

Coding theorists have traditionally attacked the problem of designing good codes by developing codes with a lot of structure, which lends itself to feasible decoders, although coding theory suggests that codes chosen “at random” should perform well if their block size is large enough. The challenge to find practical decoders for “almost” random, large codes has not been seriously considered until recently. Perhaps the most exciting and potentially important development in coding theory in recent years has been the dramatic announcement of “turbo codes” by Berrou et al. in 1993 [7]. The announced performance of these codes was so good that the initial reaction of the coding establishment was deep skepticism, but recently researchers around the world have been able to reproduce those results [15, 18, 9]. The introduction of turbo codes has opened a whole new way of looking at the problem of constructing good codes [5] and decoding them with low complexity [7, 2].

These codes achieve near-Shannon-limit error correction performance with relatively simple component codes and large interleavers. A required E_b/N_o of 0.7 dB was reported for a bit error rate (BER) of 10^{-5} for a rate 1/2 turbo code [7].

The purpose of this paper is to: (1) Design the best component codes for turbo codes of various rates; (2) Describe a suitable trellis termination rule; (3) Design pseudo-random interleavers; (4) Design turbo codes with multiple component codes; (5) Design an iterative decoding method for multiple turbo codes by approximating the optimum bit decision rule. (6) Design of low-rate turbo codes for power limited channels (deep-space communications) and CDMA; (7) Design of high rate turbo codes for bandwidth limited channels (Turbo trellis coded modulation); (8) Give examples and simulation results.

II. PARALLEL CONCATENATION OF CONVOLUTIONAL CODES

The codes considered in this paper consist of the parallel concatenation of multiple convolutional codes with random interleavers (permutations) at the input of each encoder. This extends the original results on turbo codes reported in [7], which considered turbo codes formed from just two constituent codes and overall rate 1/2.

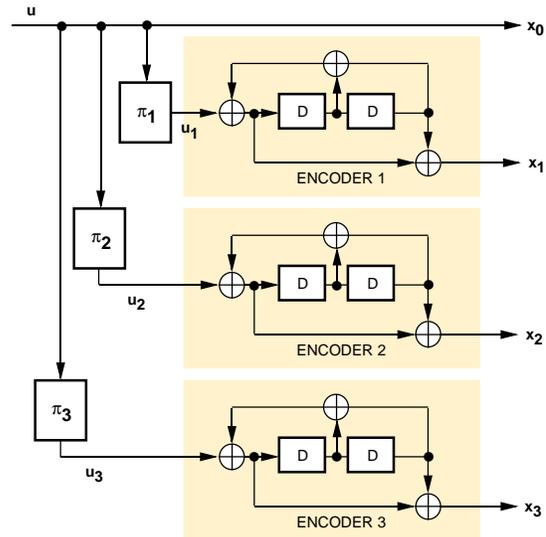


Figure 1: Example of encoder with three codes

Figure 1 illustrates a particular example that will be used in this paper to verify the performance of these codes. The encoder contains three recursive binary convolutional encoders, with m_1 , m_2 and m_3 memory cells, respectively. In general, the three component encoders may not be identical and may not have identical code rates. The first component encoder operates directly (or through π_1) on the information bit sequence $\mathbf{u} = (u_1, \dots, u_N)$ of length N , producing the two output sequences \mathbf{x}_0 and \mathbf{x}_1 . The second component encoder operates on a reordered sequence of information bits, \mathbf{u}_2 , produced by an interleaver, π_2 , of length N , and outputs the sequence \mathbf{x}_2 . Similarly, subsequent component encoders operate on a reordered sequence of information bits, \mathbf{u}_j , produced by interleaver π_j , and output the sequence \mathbf{x}_j . The interleaver is a pseudorandom block scrambler defined by a permutation of N elements with no repetitions: A complete block is read into the interleaver and read out in a specified (fixed) random order. The same interleaver is used repeatedly for all subsequent blocks. Figure 1 shows an example where a rate $r = 1/n = 1/4$ code is generated by three component codes with $m_1 = m_2 = m_3 = m = 2$, producing the outputs $\mathbf{x}_0 = \mathbf{u}$, $\mathbf{x}_1 = \mathbf{u} \cdot g_1/g_0$, $\mathbf{x}_2 = \mathbf{u}_2 \cdot g_1/g_0$, and $\mathbf{x}_3 = \mathbf{u}_3 \cdot g_1/g_0$ (here π_1 is assumed to be an identity, i.e., no permutation), where the generator polynomials g_0 and g_1 have octal representation $(7)_{octal}$ and $(5)_{octal}$, respectively. Note that various code rates can be obtained by proper puncturing of \mathbf{x}_1 , \mathbf{x}_2 , \mathbf{x}_3 , and even \mathbf{x}_0 if the decoder works (for an example, see Section VIII).

We use the encoder in Fig. 1 to generate an $(n(N+m), N)$ block code, where the m tail bits of code 2 and code 3 are not transmitted. Since the component encoders are recursive, it is not sufficient to set the last m information bits to zero in order to drive the encoder to the all-zero state, i.e., to terminate the trellis. The termination (tail) sequence depends on the state of each component encoder after N bits, which makes it impossible to terminate all component encoders with m predetermined tail

¹The research described in this paper was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration.

bits. This issue, which had not been resolved in the original turbo code implementation, can be dealt with by applying the simple method described in [9], which is valid for any number of component codes. A more complicated method is described in [17].

The design of the constituent convolutional codes, which are not necessarily optimum convolutional codes, was originally reported in [5] for rate $1/n$ codes. In this paper we extend the results to rate b/n codes. It was suggested in [2] that good random codes are obtained if g_a is a primitive polynomial (without proof). This suggestion was used in [5] to obtain “good” rate $1/2$ constituent codes, and, in this paper, to obtain “good” rate $1/3$ and $2/3$ constituent codes. A more precise definition of “good” codes is given in Sections III and V.

III. RANDOM INTERLEAVERS

The challenge in designing good turbo codes is to find the pairing of codewords from each individual encoder, induced by a particular set of interleavers. Intuitively, we would like to avoid pairing low-weight codewords from one encoder with low-weight words from the other encoders. In this section we examine the effects of random interleavers on the low-weight input sequences which may produce low output codeword weights. The input sequences with a single “1” will appear again in the other encoders, for any choice of interleavers. This motivates the use of recursive encoders, since the output weight due to weight-1 input sequences $\mathbf{u} = (\dots 001000\dots)$ is large. Now we briefly examine the issue of whether one or more random interleavers can avoid matching small separations between the 1’s of a weight-2 data sequence with equally small separations between the 1’s of its permuted version(s). Consider, for example, a particular weight-2 data sequence $(\dots 001001000\dots)$, which corresponds to a low-weight codeword in each of the encoders of Fig. 1. If we randomly select an interleaver of size N , the probability that this sequence will be permuted into another sequence of the same form is roughly $2/N$ (assuming that N is large and ignoring minor edge effects). The probability that such an unfortunate pairing happens for at least one possible position of the original sequence $(\dots 001001000\dots)$ within the block size of N is approximately $1 - (1 - 2/N)^N \approx 1 - e^{-2}$. This implies that the minimum distance of a two-code turbo code constructed with a random permutation is not likely to be much higher than the encoded weight of such an unpermuted weight-2 data sequence. By contrast, if we use three codes and two different interleavers, the probability that a particular sequence $(\dots 001001000\dots)$ will be reproduced by both interleavers is only $(2/N)^2$. Now the probability of finding such an unfortunate data sequence somewhere within the block of size N is roughly $1 - [1 - (2/N)^2]^N \approx 4/N$. Thus, it is probable that a three-code turbo code using two random interleavers will see an increase in its minimum distance beyond the encoded weight of an unpermuted weight-2 data sequence. This argument can be extended to account for other weight-2 data sequences that may also produce low-weight codewords. For comparison, let us consider a weight-3 data sequence such as $(\dots 0011100\dots)$. The probability that this sequence is reproduced with one random interleaver is roughly $6/N^2$, and the probability that some sequence of the form $(\dots 0011100\dots)$ is paired with another of the same form is $1 - (1 - 6/N^2)^N \approx 6/N$. Thus, for large block sizes, the bad weight-3 data sequences have a small probability of being matched with bad weight-3 permuted data sequences, even in a two-code system. For a turbo code using three codes and two random interleavers, this probability is even smaller, $1 - [1 - (6/N^2)^2]^N \approx 36/N^3$. This implies that the minimum

distance codeword of the turbo code in Fig. 1 is more likely to result from a weight-2 data sequence of the form $(\dots 001001000\dots)$ than from the weight-3 sequence $(\dots 0011100\dots)$. Higher weight sequences have an even smaller probability of reproducing themselves after being passed through the random interleavers.

For a turbo code using q codes and $q - 1$ interleavers, the probability that a weight- n data sequence will be reproduced somewhere within the block by all $q - 1$ permutations is of the form $1 - [1 - (\beta/N^{n-1})^{q-1}]^N$, where β is a number that depends on the weight- n data sequence but does not increase with block size N . For large N , this probability is proportional to $(1/N)^{nq-n-q}$, which falls off rapidly with N , when n and q are greater than two. Furthermore, the symmetry of this expression indicates that increasing either the weight of the data sequence n or the number of codes q has roughly the same effect on lowering this probability.

In summary, from the above arguments, we conclude that weight-2 data sequences are an important factor in the design of the constituent codes, and that higher weight sequences have successively decreasing importance [12, 11]. Also, increasing the number of codes and, correspondingly, the number of interleavers, makes it more and more likely that the bad input sequences will be broken up by one or more of the permutations.

The overall minimum distance is not the most important characteristic of the turbo code if it is due to weight- n data sequences with $n > 2$. The performance of turbo codes with random interleavers can be obtained by transfer function bounding techniques [6, 4, 12, 13].

IV. DESIGN OF PARTIALLY RANDOM INTERLEAVERS

Interleavers should be capable of spreading low-weight input sequences so that the resulting codeword has high weight. In order to break low-weight sequences, random interleavers are desirable.

We have designed semirandom permutations (interleavers) by generating random integers i , $1 \leq i \leq N$, without replacement. We define an “ S -random” permutation as follows: Each randomly selected integer is compared to S previously selected integers. If the current selection is equal to any S previous selections within a distance of $\pm S$, then the current selection is rejected. This process is repeated until all N integers are selected. The searching time for this algorithm increases with S and is not guaranteed to finish successfully. However, we have observed that choosing $S < \sqrt{N/2}$ usually produces a solution in a reasonable time. Note that for $S = 1$, we have a purely random interleaver.

V. DESIGN OF CONSTITUENT ENCODERS

As discussed in Section III, maximizing the weight of output codewords corresponding to weight-2 data sequences gives the best BER performance for moderate bit SNR as the random interleaver size N gets large. In this region the dominant term in the expression for bit error probability of turbo codes is

$$P_b \approx \frac{\beta}{N^{q-1}} Q \left(\sqrt{2r \frac{E_b}{N_o} \left(\sum_{j=1}^q d_{j,2}^p + 2 \right)} \right)$$

where $d_{j,2}^p$ is the minimum parity-weight (weight due to parity checks only) of the codewords at the output of the j^{th} constituent code due to weight-2 data sequences, and β is a constant independent of N . Define $d_{j,2} = d_{j,2}^p + 2$ as the minimum output weight including parity and systematic bits.

Theorem. For any $r = \frac{b}{b+1}$ recursive systematic convolutional encoder with generator matrix

$$G = \begin{bmatrix} & \frac{h_1(D)}{h_0(D)} \\ & \frac{h_2(D)}{h_0(D)} \\ \mathbf{I}_{b \times b} & \cdot \\ & \cdot \\ & \frac{h_b(D)}{h_0(D)} \end{bmatrix}$$

where $I_{b \times b}$ is a $b \times b$ identity matrix, $\deg[h_i(D)] \leq m_j$, $h_i(D) \neq h_0(D)$, $i = 1, 2, \dots, b$ and $h_0(D)$ is a primitive polynomial of degree m_j , the following upper bound holds

$$d_{j,2}^p \leq \lfloor \frac{2^{m_j-1}}{b} \rfloor + 2$$

Proof. In the state diagram of any recursive systematic convolutional encoder with generator matrix G , there exists at least two non-overlapping loops corresponding to all-zero input sequences. If $h_0(D)$ is a primitive polynomial there are two loops: one corresponding to zero-input, zero-output sequences with branch length one, and the other corresponding to zero-input but non-zero-output sequences with branch length $2^{m_j} - 1$, which is the period of maximal length (ML) linear feedback shift registers (FSR) with degree m_j . The parity codeword weight of this loop is 2^{m_j-1} due to the balance property of ML sequences. This weight depends only on the degree of the primitive polynomial and is independent of $h_i(D)$ due to the invariance to initial conditions of ML FSR sequences. In general, the output of the encoder is a linear function of its input and current state. So, for any output we may consider, provided it depends at least on one component of the state and it is not $h_0(D)$, then the weight of a zero input loop is 2^{m_j-1} , by the shift-and-add property of ML FSRs.

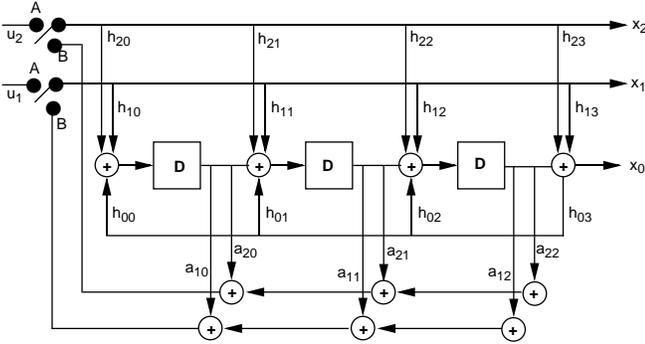


Figure 2: Canonical representation of a rate $\frac{b+1}{b}$ encoder ($b = 2$, $m_j = 3$).

Consider the canonical representation of a rate $b + 1/b$ encoder as shown in Fig. 2, when the switch is in position A. Let $S^k(D)$ be the state of the encoder at time k with coefficients $S_0^k, S_1^k, \dots, S_{m_j-1}^k$, where the output of the encoder at time k is

$$X = S_{m_j-1}^{k-1} + \sum_{i=1}^b u_i^k h_{i,m_j} \quad (1)$$

The state transition for input u_1^k, \dots, u_b^k at time k is given by

$$S^k(D) = \left[\sum_{i=1}^b u_i^k h_i(D) + DS^{k-1}(D) \right] \text{mod } h_0(D) \quad (2)$$

From the all-zero state we can enter the zero-input loop with non-zero input symbols u_1, \dots, u_b at state

$$S^1(D) = \sum_{i=1}^b u_i h_i(D) \text{mod } h_0(D) \quad (3)$$

From the same non-zero input symbol we leave exactly at state $S^{2^{m_j}-1}(D)$ back to the all zero state where $S^{2^{m_j}-1}(D)$ satisfies

$$S^1(D) = DS^{2^{m_j}-1}(D) \text{mod } h_0(D) \quad (4)$$

i.e., $S^{2^{m_j}-1}(D)$ is the “predecessor” to state $S^1(D)$ in the zero-input loop. If the most significant bit of the predecessor state is zero, i.e., $S_{m_j-1}^{2^{m_j}-1} = 0$, then the branch output for the transition from $S^{2^{m_j}-1}(D)$ to $S^1(D)$ is zero for zero input symbol. Now consider any weight 1 input symbol, i.e., $u_j = 1$ for $j = i$ and $u_j = 0$ for $j \neq i$, $j = 1, 2, \dots, b$. The question is: what are the conditions on the coefficients $h_i(D)$ such that, if we enter with weight 1 input symbol into the zero-input loop at state $S^1(D)$, the most significant bit of the “predecessor” state $S^{2^{m_j}-1}(D)$ be zero. Using eqs. 3 and 4 we can establish that

$$h_{i0} + h_{i,m_j} = 0 \quad (5)$$

Obviously, when we enter the zero-input loop from the all-zero state and when we leave this loop to go back to the all-zero state we would like the parity output to be equal to 1. From eq. 1 and 5 we require

$$h_{i0} = 1 \quad h_{i,m_j} = 1 \quad (6)$$

With this condition we can enter the zero-input loop with a weight-1 symbol at state $S^1(D)$ and then leave this loop from state $S^{2^{m_j}-1}(D)$ back to the all-zero state, for the same weight-1 input. The parity-weight of the codeword corresponding to weight-2 data sequences is then $2^{m_j-1} + 2$, where the first term is the weight of the zero-input loop and the second term is due to the parity bit appearing when entering and leaving the loop. If $b = 1$ the proof is complete and the condition to achieve the upper bound is given by 6. For $b = 2$ we may enter the zero-input loop with $\mathbf{u} = 10$ at state $S^1(D)$ and leave the loop to the zero state with $\mathbf{u} = 01$ at some state $S^j(D)$. If we can choose $S^j(D)$ such that the output weight of the zero input loop from $S^1(D)$ to $S^j(D)$ is exactly $2^{m_j-1}/2$ then the output weight of the zero-input loop from $S^{j+1}(D)$ to $S^{2^{m_j}-1}(D)$ is exactly $2^{m_j-1}/2$, and the minimum weight of codewords corresponding to some weight-2 data sequences is

$$\frac{2^{m_j-1}}{2} + 2$$

In general, for any b if we extend the procedure for $b = 2$, the minimum weight of the codewords corresponding to weight-2 data sequences is

$$\lfloor \frac{2^{m_j-1}}{b} \rfloor + 2 \quad (7)$$

where $\lfloor x \rfloor$ is the largest integer less than or equal to x . Clearly this is the best achievable weight for the minimum weight codeword corresponding to weight-2 data sequences. This upper bound can be achieved if the maximum run length of 1’s (m_j) in the zero-input loop does not exceed $\lfloor \frac{2^{m_j-1}}{b} \rfloor$, where b is a power of 2.

The run property of ML FSRs can help us in designing codes achieving this upper bound. Consider only runs of 1’s with length l , for $0 < l < m_j - 1$, then there are 2^{m_j-2-l} runs of length l , no runs of length $m_j - 1$, and only one run of length m_j . For a more

detailed proof and conditions when b is not a power of 2 see [14]
 \square

Corollary. For any $r = b/n$ recursive systematic convolutional code with b inputs, b systematic outputs and $n - b$ parity output bits using a primitive feedback generator, we have

$$d_{j,2}^p \leq (n - b) \left[\frac{2^{m_j - 1}}{b} + 2 \right] \quad (8)$$

Proof. A trivial solution is to repeat the parity output of a rate $\frac{b}{b+1}$ code. Then if this code achieves the upper bound so does a rate b/n code. \square There is an advantage in using $b > 1$ since the bound in eq.(8) for rate b/bn codes is larger than the bound for rate $1/n$ codes.

Best Rate 2/3 Constituent Codes. We obtained the best rate 2/3 codes as shown in Table 1, where $d_{j,2}^p$ is simply denoted by d_2^p and $d_2 = d_2^p + 2$. Minimum weight codewords corresponding to weight-3 data sequences are denoted by d_3 , d_{min} is the minimum distance of the code, and $k = m_j + 1$ in all tables. By ‘‘best’’ we only mean codes with large d_2 for a given m_j .

Best Rate 4/5, 16-State Constituent Codes. All three codes found have four common generators $h_0 = 23$, $h_1 = 35$, $h_2 = 31$, $h_3 = 37$, plus an additional generator $h_4 = 27$, or $h_4 = 21$, or $h_4 = 33$, all yielding $d_2 = 5$ and $d_{min} = 4$.

Trellis Termination for b/n codes with canonical realization. Trellis termination is performed (for $b = 2$) by setting the switch shown in Fig. 2 in position B. The tap coefficients $a_{i0}, \dots, a_{i,m_j-1}$ for $i = 1, 2, \dots, b$ can be obtained by repeated use of eq. (2), and by solving the resulting equations. The trellis can be terminated in state zero with at least m_j/b and at most m_j clock cycles (see [14] for details). When Fig. 3 is extended to multiple input bits (b parallel feedback shift registers), a switch should be used for each input bit.

Best Punctured Rate 1/2 Constituent Codes. A rate 2/3 constituent code can be derived by puncturing the parity bit of a rate 1/2 recursive systematic convolutional code. If the parity puncturing pattern is $P = [10]$ or $P = [01]$ then we show in [14] that it is impossible to achieve the upper bound on $d_2 = d_2^p + 2$ for rate 2/3 codes. (A puncturing pattern P has zeros where symbols are removed. The best rate 1/2 constituent codes with puncturing pattern $P = [10]$ are given in Table 2.

Best Rate 1/3 Constituent Codes. For rate $1/n$ codes the upper bound in eq. 7 for $b = 1$ reduces to

$$d_{j,2}^p \leq (n - 1)(2^{m_j - 1} + 2)$$

This upper bound was originally derived in [5], where the best rate 1/2 constituent codes meeting the bound were obtained. Here we present a simple proof based on our previous general result on rate b/n codes. Then we obtain the best rate 1/3 codes without parity repetition. In [14] we illustrate how parity repetition is undesirable for codes to be decoded with turbo decoders.

Consider a rate $1/n$ code shown in Fig. 3. In this figure $g_0(D)$ is assumed to be a primitive polynomial. As discussed above, the output weight of the zero-input loop per parity bit is $2^{m_j - 1}$ independent of the choice of $g_i(D)$, $i = 1, 2, \dots, n - 1$, provided that $g_i(D) \neq 0$ and that $g_i(D) \neq g_0(D)$, by the shift-and-add and balance properties of ML FSRs. If $S(D)$ represents the state polynomial, then we can enter the zero input loop only at state $S^1(D) = 1$ and leave the loop to the all-zero state at state $S^{2^{m_j - 1}}(D) = D^{m_j - 1}$. The i^{th} parity output on the transition $S^1(D) \rightarrow S^{2^{m_j - 1}}(D)$ with zero input bit is

$$x_i = g_{i0} + g_{i,m_j}$$

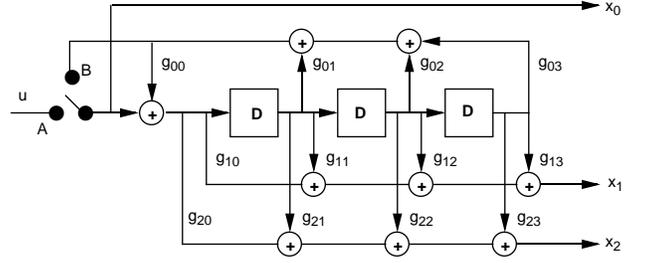


Figure 3: Rate $1/n$ code.

k	Code Generator	d_2	d_3	d_{min}
3	$h_0 = 7 \ h_1 = 3 \ h_2 = 5$	4	3	3
4	$h_0 = 13 \ h_1 = 15 \ h_2 = 17$	5	4	4
5	$h_0 = 23 \ h_1 = 35 \ h_2 = 27$	8	5	5
	$h_0 = 23 \ h_1 = 35 \ h_2 = 33$	8	5	5
6	$h_0 = 45 \ h_1 = 43 \ h_2 = 61$	12	6	6

Table 1: Best rate 2/3 constituent codes.

k	Code Generator	d_2	d_3	d_{min}
3	$g_0 = 7 \ g_1 = 5$	4	3	3
4	$g_0 = 13 \ g_1 = 15$	5	4	4
5	$g_0 = 23 \ g_1 = 37$	7	4	4
	$g_0 = 23 \ g_1 = 31$	7	4	4
	$g_0 = 23 \ g_1 = 33$	6	5	5
	$g_0 = 23 \ g_1 = 35$	6	4	4
	$g_0 = 23 \ g_1 = 27$	6	4	4

Table 2: Best rate 1/2 punctured constituent codes.

k	Code Generator	d_2	d_3	d_{min}
2	$g_0 = 3 \ g_1 = 2 \ g_2 = 1$	4	∞	4
3	$g_0 = 7 \ g_1 = 5 \ g_2 = 3$	8	7	7
4	$g_0 = 13 \ g_1 = 17 \ g_2 = 15$	14	10	10
5	$g_0 = 23 \ g_1 = 33 \ g_2 = 37$	22	12	10
	$g_0 = 23 \ g_1 = 35 \ g_2 = 27$	22	11	11

Table 3: Best rate 1/3 constituent codes (without parity repetition).

If $g_{i0} = 1$ and $g_{i,m_j} = 1$ for $i = 1, \dots, n-1$, the output weight of the encoder for that transition is zero. The output weight when entering and leaving the zero-input loop is $(n-1)$ for each case. In addition, the output weight of the zero-input loop will be $(n-1)2^{m_j-1}$. Thus we can achieve the upper bound.

We obtained the best rate $1/3$ codes without parity repetition as shown in Table 3, where $d_2 = d_2^p + 2$ represents the minimum output weight given by weight-2 data sequences. The best rate $1/2$ constituent codes are given by g_0 and g_1 in this table, as was also reported in [5].

VI. TURBO DECODING FOR MULTIPLE CODES

In this section, we consider decoding algorithms for multiple-code turbo codes. In general, the advantage of using three or more constituent codes is that the corresponding two or more interleavers have a better chance to break sequences that were not broken by another interleaver. The disadvantage is that, for an overall desired code rate, each code must be punctured more, resulting in weaker constituent codes. Also shorter constraint length codes should be used for successful operation of the turbo decoder. In our experiments, we have used randomly selected interleavers and S-random interleavers.

Let u_k be a binary random variable taking values in $\{0, 1\}$, representing the sequence of information bits $\mathbf{u} = (u_1, \dots, u_N)$. The MAP algorithm [1] provides the log likelihood ratio L_k , given the received symbols \mathbf{y} :

$$\begin{aligned} L_k &= \log \frac{P(u_k=1|\mathbf{y})}{P(u_k=0|\mathbf{y})} \\ &= \log \frac{\sum_{\mathbf{u}:u_k=1} P(\mathbf{y}|\mathbf{u}) \prod_{j \neq k} P(u_j)}{\sum_{\mathbf{u}:u_k=0} P(\mathbf{y}|\mathbf{u}) \prod_{j \neq k} P(u_j)} + \log \frac{P(u_k=1)}{P(u_k=0)} \end{aligned} \quad (9)$$

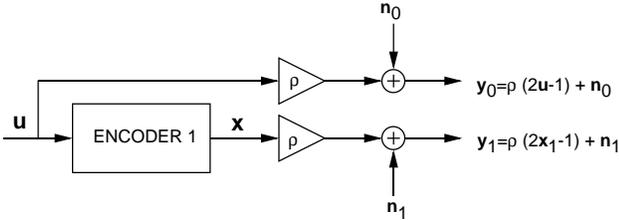


Figure 4: Channel Model

For efficient computation of Eq. (9) when the a priori probabilities $P(u_j)$ are nonuniform, the modified MAP algorithm in [15] is simpler to use than the version considered in [7]. Therefore, in this paper, we use the modified MAP algorithm of [15].

If the rate b/n constituent code is not equivalent to a punctured rate $1/n'$ code or if turbo trellis coded modulation is used, we can first use the symbol MAP algorithm [1] to compute the log-likelihood ratio of a symbol $\mathbf{u} = u_1, u_2, \dots, u_b$ given the observation \mathbf{y} as

$$\lambda(\mathbf{u}) = \log \frac{P(\mathbf{u}|\mathbf{y})}{P(\mathbf{0}|\mathbf{y})}$$

where $\mathbf{0}$ corresponds to the all-zero symbol. Then we obtain the log-likelihood ratios of the j th bit within the symbol by

$$L(u_j) = \log \frac{\sum_{\mathbf{u}:u_j=1} e^{\lambda(\mathbf{u})}}{\sum_{\mathbf{u}:u_j=0} e^{\lambda(\mathbf{u})}}$$

In this way the turbo decoder operates on bits and bit, rather than symbol, interleaving is used.

The channel model is shown in Fig. 4, where the n_{0k} 's and the n_{1k} 's are independent identically distributed (i.i.d.) zero-mean Gaussian random variables with unit variance, and $\rho = \sqrt{2rE_b/N_o}$ is the SNR. The same model is used for each encoder. To explain the basic decoding concept, we restrict ourselves to three codes, but extension to several codes is straightforward. In order to simplify the notation, consider the combination of puncturer and encoder as a block code with input \mathbf{u} and outputs \mathbf{x}_i , $i = 0, 1, 2, 3$ ($\mathbf{x}_0 = \mathbf{u}$) and the corresponding received sequences \mathbf{y}_i , $i = 0, 1, 2, 3$. The optimum bit decision metric on each bit is (for data with uniform a priori probabilities)

$$L_k = \log \frac{\sum_{\mathbf{u}:u_k=1} P(\mathbf{y}_0|\mathbf{u})P(\mathbf{y}_1|\mathbf{u})P(\mathbf{y}_2|\mathbf{u})P(\mathbf{y}_3|\mathbf{u})}{\sum_{\mathbf{u}:u_k=0} P(\mathbf{y}_0|\mathbf{u})P(\mathbf{y}_1|\mathbf{u})P(\mathbf{y}_2|\mathbf{u})P(\mathbf{y}_3|\mathbf{u})} \quad (10)$$

but in practice, we cannot compute Eq. (10) for large N because the permutations π_2, π_3 imply that \mathbf{y}_2 and \mathbf{y}_3 are no longer simple convolutional encodings of \mathbf{u} . Suppose that we evaluate $P(\mathbf{y}_i|\mathbf{u})$, $i = 0, 2, 3$ in Eq. (10) using Bayes' rule and using the following approximation:

$$P(\mathbf{u}|\mathbf{y}_i) \approx \prod_{k=1}^N \tilde{P}_i(u_k) \quad (11)$$

Note that $P(\mathbf{u}|\mathbf{y}_i)$ is not separable in general. However, for $i = 0$, $P(\mathbf{u}|\mathbf{y}_0)$ is separable; hence, Eq. (11) holds with equality. If such an approximation, i.e., Eq. (11), can be obtained, we can use it in Eq. (10) for $i = 2$ and $i = 3$ (by Bayes' rule) to complete the algorithm. A reasonable criterion for this approximation is to choose $\prod_{k=1}^N \tilde{P}_i(u_k)$ such that it minimizes the Kullback distance or free energy [3, 16]. Define \tilde{L}_{ik} by

$$\tilde{P}_i(u_k) = \frac{e^{u_k \tilde{L}_{ik}}}{1 + e^{\tilde{L}_{ik}}} \quad (12)$$

where $u_k \in \{0, 1\}$. Then the Kullback distance is given by

$$F(\tilde{\mathbf{L}}_i) = \sum_{\mathbf{u}} \frac{e^{\sum_{k=1}^N u_k \tilde{L}_{ik}}}{\prod_{k=1}^N (1 + e^{\tilde{L}_{ik}})} \log \frac{e^{\sum_{k=1}^N u_k \tilde{L}_{ik}}}{\prod_{k=1}^N (1 + e^{\tilde{L}_{ik}}) P(\mathbf{u}|\mathbf{y}_i)} \quad (13)$$

Minimizing $F(\tilde{\mathbf{L}}_i)$ involves forward and backward recursions analogous to the MAP decoding algorithm, but we have not attempted this approach in this work. Instead of using Eq. (13) to obtain $\{\tilde{P}_i\}$ or, equivalently, $\{\tilde{L}_{ik}\}$, we use Eqs. (11) and (12) for $i = 0, 2, 3$ (by Bayes' rule) to express Eq. (10) as

$$L_k = f(\mathbf{y}_1, \tilde{\mathbf{L}}_0, \tilde{\mathbf{L}}_2, \tilde{\mathbf{L}}_3, k) + \tilde{L}_{0k} + \tilde{L}_{2k} + \tilde{L}_{3k} \quad (14)$$

where $\tilde{L}_{0k} = 2\rho\mathbf{y}_{0k}$ (for binary modulation) and

$$f(\mathbf{y}_1, \tilde{\mathbf{L}}_0, \tilde{\mathbf{L}}_2, \tilde{\mathbf{L}}_3, k) = \log \frac{\sum_{\mathbf{u}:u_k=1} P(\mathbf{y}_1|\mathbf{u}) \prod_{j \neq k} e^{u_j (\tilde{L}_{0j} + \tilde{L}_{2j} + \tilde{L}_{3j})}}{\sum_{\mathbf{u}:u_k=0} P(\mathbf{y}_1|\mathbf{u}) \prod_{j \neq k} e^{u_j (\tilde{L}_{0j} + \tilde{L}_{2j} + \tilde{L}_{3j})}} \quad (15)$$

We can use Eqs. (11) and (12) again, but this time for $i = 0, 1, 3$, to express Eq. (10) as

$$L_k = f(\mathbf{y}_2, \tilde{\mathbf{L}}_0, \tilde{\mathbf{L}}_1, \tilde{\mathbf{L}}_3, k) + \tilde{L}_{0k} + \tilde{L}_{1k} + \tilde{L}_{3k} \quad (16)$$

and similarly,

$$L_k = f(\mathbf{y}_3, \tilde{\mathbf{L}}_0, \tilde{\mathbf{L}}_1, \tilde{\mathbf{L}}_2, k) + \tilde{L}_{0k} + \tilde{L}_{1k} + \tilde{L}_{2k} \quad (17)$$

A solution to Eqs. (14), (16), and (17) is

$$\begin{aligned} \tilde{L}_{1k} &= f(\mathbf{y}_1, \tilde{\mathbf{L}}_0, \tilde{\mathbf{L}}_2, \tilde{\mathbf{L}}_3, k) \\ \tilde{L}_{2k} &= f(\mathbf{y}_2, \tilde{\mathbf{L}}_0, \tilde{\mathbf{L}}_1, \tilde{\mathbf{L}}_3, k) \\ \tilde{L}_{3k} &= f(\mathbf{y}_3, \tilde{\mathbf{L}}_0, \tilde{\mathbf{L}}_1, \tilde{\mathbf{L}}_2, k) \end{aligned} \quad (18)$$

for $k = 1, 2, \dots, N$, provided that a solution to Eq. (18) does indeed exist. The final decision is then based on

$$L_k = \tilde{L}_{0k} + \tilde{L}_{1k} + \tilde{L}_{2k} + \tilde{L}_{3k} \quad (19)$$

which is passed through a hard limiter with zero threshold. We attempted to solve the nonlinear equations in Eq. (18) for $\tilde{\mathbf{L}}_1$, $\tilde{\mathbf{L}}_2$, and $\tilde{\mathbf{L}}_3$ by using the iterative procedure

$$\tilde{L}_{1k}^{(m+1)} = \alpha_1^{(m)} f(\mathbf{y}_1, \tilde{\mathbf{L}}_0, \tilde{\mathbf{L}}_2^{(m)}, \tilde{\mathbf{L}}_3^{(m)}, k) \quad (20)$$

for $k = 1, 2, \dots, N$, iterating on m . Similar recursions hold for $\tilde{L}_{2k}^{(m)}$ and $\tilde{L}_{3k}^{(m)}$. The gain $\alpha_1^{(m)}$ should be equal to one, but we noticed experimentally that better convergence can be obtained by optimizing this gain for each iteration, starting from a value slightly less than one and increasing toward one with the iterations, as is often done in simulated annealing methods. We start the recursion with the initial condition² $\tilde{\mathbf{L}}_1^{(0)} = \tilde{\mathbf{L}}_2^{(0)} = \tilde{\mathbf{L}}_3^{(0)} = \tilde{\mathbf{L}}_0$. For the computation of $f(\cdot)$, we use the modified MAP algorithm as described in [9] with permuters (direct and inverse) where needed, as shown in Fig. 5. The MAP algorithm always starts and ends at the all-zero state since we always terminate the trellis as described in [9]. We assumed $\pi_1 = I$ identity; however, any π_1 can be used. The overall decoder is composed of block decoders

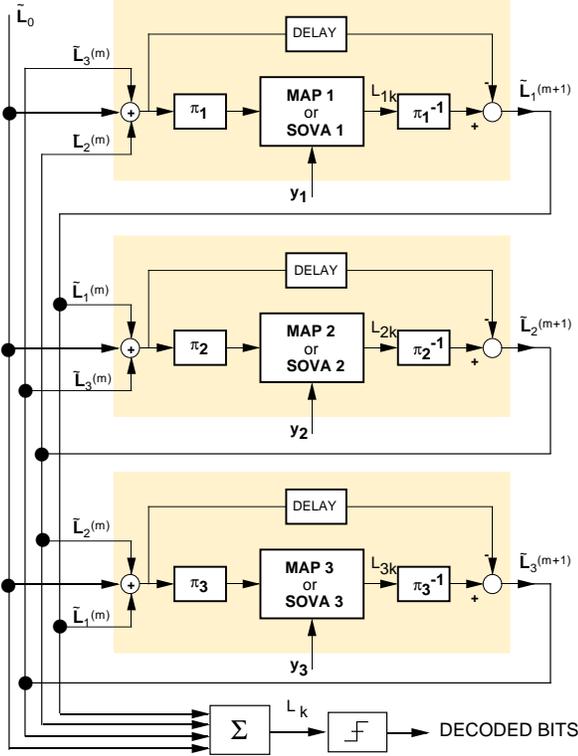


Figure 5: Multiple Turbo Decoder Structure

connected as in Fig. 5, which can be implemented as a pipeline or by feedback. In [11] we proposed an alternative version of the above decoder which is more appropriate for use in turbo trellis coded modulation, i.e., set $\tilde{L}_0 = 0$ and consider \mathbf{y}_0 as part of \mathbf{y}_1 . If the systematic bits are distributed among encoders, we use the same distribution for \mathbf{y}_0 among the MAP decoders.

²Note that the components of the $\tilde{\mathbf{L}}_i$'s corresponding to the tail bits, i.e., \tilde{L}_{ik} , for $k = N + 1, \dots, N + M$, are set to zero for all iterations.

At this point, further approximation for turbo decoding is possible if one term corresponding to a sequence \mathbf{u} dominates other terms in the summation in the numerator and denominator of Eq. (15). Then the summations in Eq. (15) can be replaced by “maximum” operations with the same indices, i.e., replacing $\sum_{\mathbf{u}:u_k=i}$ with $\max_{\mathbf{u}:u_k=i}$ for $i = 0, 1$. A similar approximation can be used for \tilde{L}_{2k} and \tilde{L}_{3k} in Eq. (18). This suboptimum decoder then corresponds to a turbo decoder that uses soft output Viterbi (SOVA)-type decoders rather than MAP decoders. Further approximations, i.e., replacing \sum with \max can also be used in the MAP algorithm.

VII. MULTIPLE-CODE ALGORITHM APPLIED TO TWO CODES

For turbo codes with only two constituent codes, Eq. (20) reduces to

$$\begin{aligned} \tilde{L}_{1k}^{(m+1)} &= \alpha_1^{(m)} f(\mathbf{y}_1, \tilde{\mathbf{L}}_0, \tilde{\mathbf{L}}_2^{(m)}, k) \\ \tilde{L}_{2k}^{(m+1)} &= \alpha_2^{(m)} f(\mathbf{y}_2, \tilde{\mathbf{L}}_0, \tilde{\mathbf{L}}_1^{(m)}, k) \end{aligned}$$

for $k = 1, 2, \dots, N$ and $m = 1, 2, \dots$, where, for each iteration, $\alpha_1^{(m)}$ and $\alpha_2^{(m)}$ can be optimized (simulated annealing) or set to 1 for simplicity. The decoding configuration for two codes, according to the previous section, can be obtained from Fig. 5. In this special case, since the paths in Fig. 5 are disjoint, the decoder structure can be reduced to a serial mode structure if desired.

If we optimize $\alpha_1^{(m)}$ and $\alpha_2^{(m)}$, our method for two codes is similar to the decoding method proposed in [7], which requires estimates of the variances of \tilde{L}_{1k} and \tilde{L}_{2k} for each iteration in the presence of errors. In the method proposed in [15], the received “systematic” observation was subtracted from \tilde{L}_{1k} , which may result in performance degradation. In [18] the method proposed in [15] was used but the received “systematic” observation was interleaved and provided to decoder 2. In [9], we argued that there is no need to interleave the received “systematic” observation and provide it to decoder 2, since \tilde{L}_{0k} does this job. It seems that our proposed method with $\alpha_1^{(m)}$ and $\alpha_2^{(m)}$ equal to 1 is simple and achieves the same performance reported in [18] for rate 1/2 codes.

VIII. PERFORMANCE AND SIMULATION RESULTS

The bit error rate performance of these codes was evaluated by using transfer function bounds [6] [13]. In [13] it was shown that transfer function bounds are very useful for signal-to-noise ratios above the cutoff rate threshold and that they cannot accurately predict performance in the region between cutoff rate and capacity. In this region, the performance was computed by simulation.

Figure. 6 shows the performance of turbo codes with m iterations and the following generators: For two $K = 5$ constituent codes, $(1, g_1/g_0, g_2/g_0)$ and (g_1/g_0) , with $g_0 = (37)_{octal}$, $g_1 = (33)_{octal}$ and $g_2 = (25)_{octal}$; For three $K = 3$ codes, $(1, g_1/g_0)$ and (g_1/g_0) with $g_0 = (7)_{octal}$ and $g_1 = (5)_{octal}$; For three $K = 4$ codes, $(1, g_1/g_0)$ and (g_1/g_0) with $g_0 = (17)_{octal}$ and $g_1 = (11)_{octal}$.

Further results at $\text{BER}=10^{-5}$ were obtained for two constituent codes with interleaving size $N = 16384$ as follows. For a rate 1/2 turbo code using two codes, $K = 2$ (differential encoder) with (g_1/g_0) where $g_0 = (3)_{octal}$ and $g_1 = (1)_{octal}$, and $K = 5$ with (g_1/g_0) where $g_0 = (23)_{octal}$ and $g_1 = (33)_{octal}$ the required bit SNR was 0.85 dB. This is an example where the systematic bits are not transmitted. For rate 1/3, we used two $K = 5$ codes, $(1, g_1/g_0)$ and (g_1/g_0) with $g_0 = (23)_{octal}$ and $g_1 = (33)_{octal}$ and obtained bit SNR= 0.25 dB. For rate 1/4, we

used two $K = 5$ codes with $(1, g_1/g_0, g_2/g_0)$ and (g_1/g_0) with $g_0 = (23)_{octal}$, $g_1 = (33)_{octal}$ and $g_2 = (25)_{octal}$ and obtained bit SNR = 0 dB. A fixed number of iterations $m = 20$ was used for all cases. Many of these codes may actually require a smaller number of iterations for BER=10⁻⁵ or below.

The simulation performance of other codes reported in this paper is still in progress.

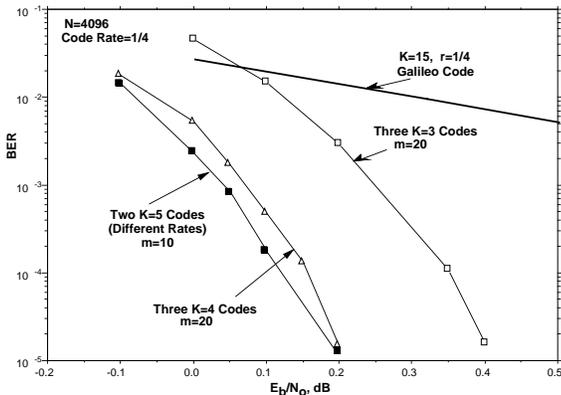


Figure 6: Performance of turbo codes

IX. TURBO TRELLIS CODED MODULATION

A pragmatic approach for turbo codes with multilevel modulation was proposed in [8]. Here we propose a different approach that outperforms the results in [8] when M-QAM modulation is used. A straightforward method to use turbo codes for multilevel modulation is first to select a rate $\frac{b}{b+1}$ constituent code where the outputs are mapped to a 2^{b+1} -level modulation based on Ungerboeck's set partitioning method (i.e., we can use Ungerboeck's codes with feedback). If MPSK modulation is used, for every b bits at the input of the turbo encoder we transmit two consecutive 2^{b+1} PSK signals, one per each encoder output. This results in a throughput of $b/2$ bits/sec/Hz. If M-QAM modulation is used, we map the $b+1$ outputs of the first component code to the 2^{b+1} in-phase levels (I-channel) of a 2^{2b+2} -QAM signal set, and the $b+1$ outputs of the second component code to the 2^{b+1} quadrature levels (Q-channel). The throughput of this system is b bits/sec/Hz.

First, we note that these methods require more levels of modulation than conventional TCM, which is not desirable in practice. Second, the input information sequences are used twice in the output modulation symbols, which is also not desirable. An obvious remedy is to puncture the output symbols of each trellis code and select the puncturing pattern such that the output symbols of the turbo code contain the input information only once. If the output symbols of the first encoder is punctured, for example as 101010..., the puncturing pattern of the second trellis code is non-uniform and depends on the particular choice of interleaver. Now for example, for 2^{b+1} -PSK a throughput b can be achieved. This method has two drawbacks, it complicates the encoder and decoder and the reliability of punctured symbols may not be reproducible at the decoder. A better remedy, for $\frac{b}{b+1}$ (b even), is to select the $b/2$ systematic outputs and puncture the rest of the systematic outputs, but keep the parity bit of the $\frac{b}{b+1}$ code (Note that the $\frac{b}{b+1}$ may have been already obtained by puncturing a rate $1/2$ code). Then do the same to the second constituent code but select only those systematic bits which were punctured in the first encoder. This method requires at least two interleavers: the first interleaver permutes the bits selected

by the first encoder and the second interleaver those punctured by the first encoder. For MPSK (or MQAM) we can use $2^{1+b/2}$ PSK symbols (or $2^{1+b/2}$ QAM symbols) per encoder and achieve throughput $b/2$. For M-QAM we can also use $2^{1+b/2}$ levels in the I-channel and $2^{1+b/2}$ levels in the Q-channel, and achieve a throughput of b bits/sec/Hz. These methods are equivalent to a multi-dimensional trellis coded modulation scheme (in this case, two multi-level symbols per branch) which uses $2^{b/2} \times 2^{1+b/2}$ symbols per branch, where the first symbol in the branch (which only depends on uncoded information) is punctured. Now, with these methods the reliability of the punctured symbols is reproducible at the decoder. Obviously, the constituent codes for a given modulation should be redesigned based on the Euclidean distance. In this paper we give one example for $b = 2$ with 16QAM modulation where for simplicity we can use the $2/3$ codes in Table 1 with Gray code mapping. Note that this may result in suboptimum constituent codes for multi-level modulation. The turbo encoder with 16QAM and two clock cycle trellis termination is shown in Fig. 7. The BER performance of this code with the turbo decoding structure for two codes discussed in Sec. VI is given in Fig. 8. For permutations π_1 and π_2 we used S-random permutations with $S=40$ and $S=32$ with block size of 16384 bits. For 8PSK we used the best 16-state rate $4/5$ code given in Sec. V to achieve throughput 2. More examples for 8PSK and 16QAM are given in [14].

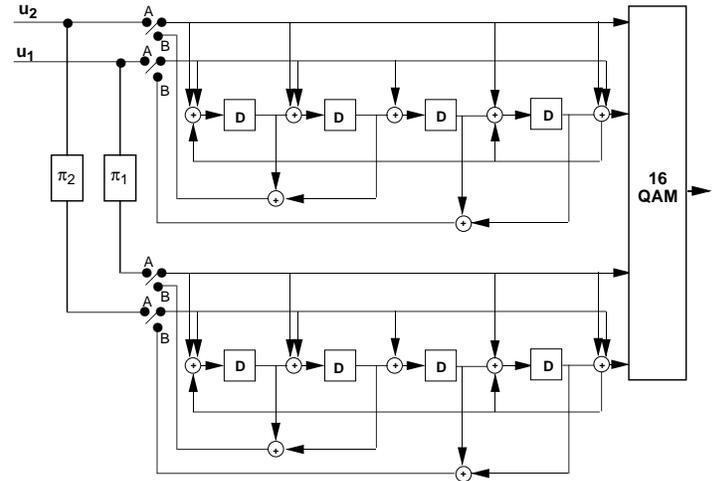


Figure 7: Turbo Trellis Coded Modulation, 16QAM, 2 bits/sec/Hz.

X. CONCLUSIONS

In this paper we have shown that powerful turbo codes can be obtained if multiple constituent codes are used. We proposed an iterative decoding method for multiple turbo codes by approximating the optimum bit decision rule. Construction of a partially random interleaver was discussed. A probabilistic argument was used to show the importance of maximizing the minimum output weight of constituent codes due to weight-2 input sequences in the design of turbo codes. We obtained an upper bound on this minimum output weight for rate b/n constituent codes. We found the best rate $2/3$, $1/3$ and 16-state rate $4/5$ constituent codes that can be used in the design of multiple turbo codes. We proposed new schemes that can be used for power and bandwidth efficient turbo trellis coded modulation.

XI. ACKNOWLEDGMENTS

The authors are grateful to S. Dolinar for his contributions to

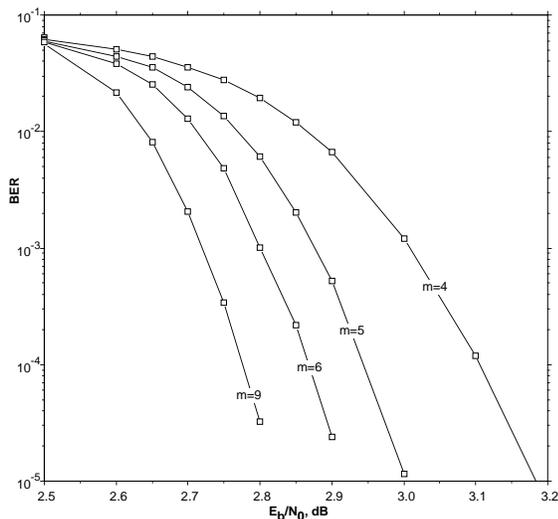


Figure 8: BER Performance of Turbo Trellis Coded Modulation, 16QAM, 2 bits/sec/Hz.

random interleavers and to R. J. McEliece for helpful comments.

REFERENCES

- [1] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal Decoding of Linear Codes for Minimizing Symbol Error Rate," *IEEE Trans. Inform. Theory*, vol. IT-20, pp. 284-287, 1974.
- [2] G. Battail, C. Berrou, and A. Glavieux, "Pseudo-Random Recursive Convolutional Coding for Near-Capacity Performance," *Comm. Theory Mini-Conference, GLOBECOM '93*, Houston, Texas, December 1993.
- [3] G. Battail and R. Sfez, "Suboptimum Decoding Using the Kullback Principle," *Lecture Notes in Computer Science*, vol. 313, pp. 93-101, 1988.
- [4] S. Benedetto, "Unveiling Turbo Codes", IEEE Communication Theory Workshop, April 23-26, 1995, Santa Cruz, CA
- [5] S. Benedetto and G. Montorsi, "Design of Parallel Concatenated Convolutional Codes", submitted to IEEE Transactions on Communications.
- [6] S. Benedetto and G. Montorsi, "Performance evaluation of turbo-codes", *Electronics Letters*, Feb. 2, 1995, Vol. 31, No. 3, pp. 163-165.
- [7] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon Limit Error-Correcting Coding: Turbo Codes," *Proc. 1993 IEEE International Conference on Communications*, Geneva, Switzerland, pp. 1064-1070, May 1993.
- [8] S. LeGoff, A. Glavieux, and C. Berrou, "Turbo Codes and High Spectral Efficiency Modulation", *Proceedings of IEEE ICC'94*, May 1-5, 1994, New Orleans, LA.
- [9] D. Divsalar and F. Pollara, "Turbo Codes for Deep-Space Communications", JPL TDA Progress Report 42-120, Feb. 15, 1995.
- [10] D. Divsalar and F. Pollara, "Multiple Turbo Codes for Deep-Space Communications", JPL TDA Progress Report 42-121, May 15, 1995.
- [11] D. Divsalar and F. Pollara, "Turbo Codes for PCS Applications," *Proceedings of IEEE ICC'95*, Seattle, Washington, June 1995.
- [12] D. Divsalar and F. Pollara, "Turbo Codes for Deep-Space Communications", IEEE Communication Theory Workshop, April 23-26, 1995, Santa Cruz, CA
- [13] D. Divsalar, S. Dolinar, R.J. McEliece, F. Pollara. "Transfer Function Bounds on the Performance of Turbo Codes", MILCOM 95, Nov. 5-8, 1995, San Diego, CA.
- [14] D. Divsalar and F. Pollara, "On the Design of Turbo Codes", JPL TDA Progress Report 42-123, Nov 15, 1995 (To be published).
- [15] J. Hagenauer and P. Robertson, "Iterative (Turbo) Decoding of Systematic Convolutional Codes With the MAP and SOVA Algorithms," *Proc. of the ITG Conference on Source and Channel Coding*, Frankfurt, Germany, October 1994.
- [16] M. Moher, "Decoding Via Cross-Entropy Minimization," *Proceedings GLOBECOM '93*, pp. 809-813, December 1993.
- [17] A.S. Barbuiescu and S.S. Pietrobon, "Terminating the Trellis of Turbo-Codes in the Same State", *Electronics Letters*, Vol. 31, no. 1, pp. 22-23, Jan. 1995.
- [18] P. Robertson, "Illuminating the Structure of Code and Decoder of Parallel Concatenated Recursive Systematic (Turbo) Codes, *Proceedings GLOBECOM '94*, San Francisco, California, pp. 1298-1303, December 1994.