Bandwidth-Efficient Turbo Trellis-Coded Modulation Using Punctured Component Codes

Patrick Robertson, *Member, IEEE*, and Thomas Wörz, *Member, IEEE*

***Abstract*— We present a bandwidth-efficient channel coding scheme that has an overall structure similar to binary turbo codes, but employs trellis-coded modulation (TCM) codes (including multidimensional codes) as component codes. The combination of turbo codes with powerful bandwidth-efficient component codes leads to a straightforward encoder structure, and allows iterative decoding in analogy to the binary turbo decoder. However, certain special conditions may need to be met at the encoder, and the iterative decoder needs to be adapted to the decoding of the component TCM codes. The scheme has been investigated for 8-PSK, 16-QAM, and 64-QAM modulation schemes with varying overall bandwidth efficiencies. A simple code choice based on the minimal distance of the punctured component code has also been performed. The interset distances of the partitioning tree can be used to fix the number of coded and uncoded bits. We derive the symbol-by-symbol MAP component decoder operating in the log domain, and apply methods of reducing decoder complexity. Simulation results are presented and compare the scheme with traditional TCM as well as turbo codes with Gray mapping. The results show that the novel scheme is very powerful, yet of modest complexity since simple component codes are used.**

***Index Terms*—Decoding, iterative methods, trellis-coded modulation.**

I. Introduction

In 1993, powerful so-called turbo codes were introduced [1] which achieve good bit-error rates (BER's) ($10^{-3} \cdots 10^{-5}$) at low SNR. They are of interest in a wide range of telecommunications applications, and comprise two binary component codes and an interleaver. They were originally proposed for binary modulation (BPSK). Successful attempts were soon undertaken to combine binary turbo codes with higher order modulation (e.g., 8-PSK, 16-QAM) using Gray mapping [2], and alternatively as component codes within multilevel codes [3]. In contrast, in our approach—called turbo trellis-coded modulation (TTCM)—we have employed two Ungerboeck-type codes [4] in combination with trellis-coded modulation (TCM) in their recursive systematic form as component codes in an overall structure rather similar to binary turbo codes [5], [6]. A different approach for bandwidth-efficient coding using recursive parallel concatenation was proposed in [7] and [8] where there is no puncturing of coded bits or symbols. TCM codes by themselves combine modulation and coding by optimizing the Euclidean distance between codewords; they can be decoded with the Viterbi or the Bahl–Jelinek (symbol-by-symbol MAP) algorithm [9]. Multidimensional TCM allows even higher bandwidth efficiency than traditional Ungerboeck TCM by assigning more than one symbol per trellis transition or step [10]. In this case, the set partitioning takes into account the union of more than one two-dimensional signal set.

The basic principle of turbo codes is applied to TCM by retaining the important properties and advantages of both of their structures. Essentially, TCM codes can be seen as systematic feedback convolutional codes followed by one (or more for multidimensional codes) signal mapper(s). Just as binary turbo codes use a parallel concatenation of two binary recursive convolutional encoders, we have concatenated two recursive TCM encoders, and adapted the interleaving and puncturing. Naturally, this has consequences at the decoding side.

In this paper, we also extend the basic concept of TTCM to incorporate multidimensional component codes which allows a higher overall bandwidth efficiency for a given signal constellation than ordinary TTCM. As a further possibility of increasing the bandwidth efficiency, we employ higher order modulation constellations (for example, 64-QAM). These two approaches require us to retain parallel transitions in the trellis for complexity reasons; in other words, some of the information bits are completely uncoded in both component codes. In [5], we did not allow parallel transitions for 8-PSK and 16-QAM modulation with two, respectively three, information bits per symbol since the corresponding uncoded bits would not benefit from the interleaver and the parallel concatenation. However, due to the higher operating SNR for very high bandwidth-efficient schemes and the large Euclidean distance that separates the subsets of signal points that carry these uncoded bits, the restriction of not allowing parallel transitions to TTCM can be broken without loss of performance at least in schemes with 8-PSK transmitting 2.5 information bits/symbol and 64-QAM with 5 bits/symbol which were investigated here.

By applying the technique to 8-PSK, 16-QAM, and 64-QAM modulation formats, we have shown its viability over a large range of bandwidth efficiency and signal-to-noise ratios. In all cases, low BER's ($10^{-4} \cdots 10^{-5}$) could be achieved within 1 dB or less from Shannon's limit—a finding that, in the context of binary turbo codes, was responsible for the interest they generated.

The paper begins by describing the generic encoder (beginning with a motivation for its structure); an encoder with 8-PSK signaling will serve as a salient example. We then

Manuscript received September 1, 1996; revised April 22, 1997. This work was presented in part at IEEE ICC'96, Dallas, TX, June 1996, and at IEEE ICC'97, Montreal, P.Q., Canada, June 1997.

The authors are with the Institute for Communications Technology, German Aerospace Center (DLR), D-82230 Wessling, Germany.

Publisher Item Identifier S 0733-8716(98)00229-7.

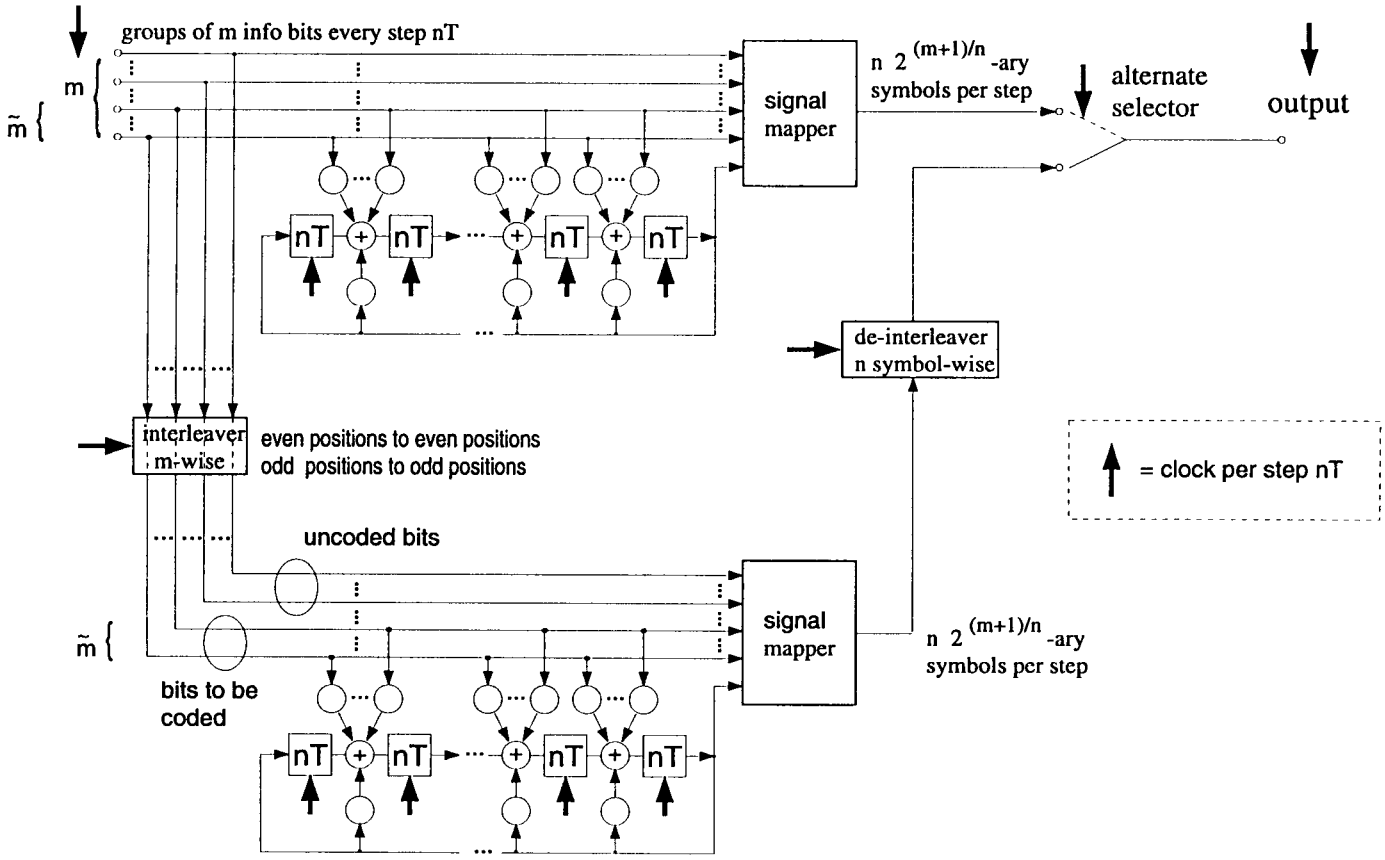

Fig. 1. Generic encoder that treats uncoded bits as coded bits from a structural point of view.

present the results of a search for component codes for 8-PSK and Z^2 signal sets, taking into consideration the puncturing at the encoder. This is followed by a section on the iterative decoder using symbol-by-symbol MAP component decoders whose structures are derived for our case of nonbinary trellises and special metric calculation. Finally, we present simulation results of the new scheme with two- and four-dimensional 8-PSK, as well as two-dimensional 16-QAM and 64-QAM. The influence of varying the block size—of important practical relevance—is also a subject of investigation. For reference, we judge the new schemes against classical TCM and binary turbo codes with Gray mapping, as well as their BER performance with respect to channel capacity.

II. The Encoder

A. Motivation for the Structure

Let us recall that two important characteristics of turbo codes are their simple use of recursive systematic component codes in a parallel concatenation scheme. Pseudorandom bit-wise interleaving between encoders ensures a small bit-error probability [11]. What is crucial to their practical suitability is the fact that they can be decoded iteratively with good performance [1]. It is well known that Ungerboeck codes combine coding and modulation by optimizing the Euclidean distance between codewords and achieve high spectral efficiency (m bits per 2^{m+1}-ary symbol from the two-dimensional signal space) through signal set expansion. The encoder can be represented as combination of a systematic recursive convolutional encoder and symbol mapper. If $\tilde{m}$ out of m bits are encoded, the resulting trellis diagram consists of $2^{\tilde{m}}$ branches per state, not counting parallel transitions. This results in more than two branches per state for $\tilde{m} > 1$—we call this a nonbinary trellis.

We have employed Ungerboeck codes (and multidimensional TCM codes) as building blocks in a turbo coding scheme in a similar way as binary codes were used [1]. The major differences are: 1) the interleaving now operates on short groups of m bits (e.g., pairs for 8-PSK with two-dimensional TCM schemes) instead of single bits; 2) to achieve the desired spectral efficiency, puncturing the parity information is not quite as straightforward as in the binary turbo coding case; and 3) there are special constraints on both the component encoders as well as the structure of the interleaver.

Let the size of the interleaver be N. The number of modulated symbols per block is $N \cdot n$, with $n = D/2$, where D is the signal set dimensionality. The number of information bits transmitted per block is $N \cdot m$. The encoder is clocked in steps of $n \cdot T$ where T is the symbol duration of each transmitted $2^{((m+1)/n)}$-ary symbol. In each step, m information bits are input and n symbols are transmitted, yielding a spectral efficiency of m/n bits per symbol usage. Fig. 1 shows the generic encoder, comprising two TCM encoders linked by the interleaver. A signal mapper follows each recursive systematic convolutional encoder where the latter each produce one parity bit in addition to retaining the m information bits at their

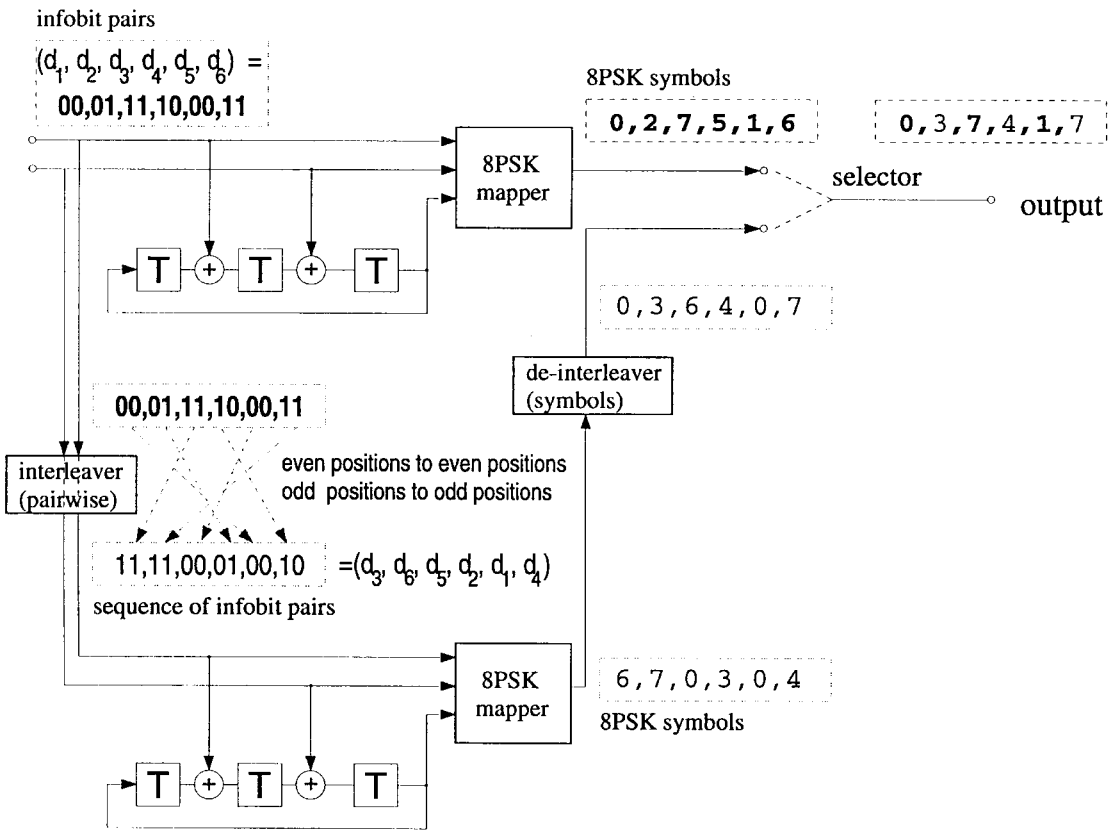

Fig. 2. Encoder shown for 8-PSK with two-dimensional component codes memory 3. An example of interleaving with $N = 6$ is shown. Bold letters indicate that symbols or pairs of bits correspond to the upper encoder.

inputs. For clarity, we have not depicted any special treatment of the $m - \tilde{m}$ uncoded bits as opposed to the $\tilde{m}$ bits to be encoded: in practice, uncoded bits would not need to be passed through the interleaver but would be simply used to choose the final signal point from a subset of points after the selector. We will return to the problem of parallel transitions shortly. For the moment, the interleaver is restricted to keeping each group of m bits unchanged within itself (as visualized by the dashed lines passing through the interleaver in Fig. 1). The output of the bottom encoder/mapper is deinterleaved according to the inverse operation of the interleaver. This ensures that at the input of the selector, the m information bits partly defining each group of n symbols of both the upper and lower input are identical. Therefore, if the selector is switched such that a group of n symbols is chosen alternately from the upper and lower inputs, then the sequence of $N \cdot n$ symbols at the output has the important property that each of the N groups of m information bits defines part of each group of n output symbols. The remaining bit which is needed to define each group of n symbols is the parity bit taken alternatively from the upper and lower encoder.

A simple example will now serve to clarify the operation of the encoder for the case $n = 1, m = 2, N = 6$, and 8-PSK signaling: it is illustrated in Fig. 2. The set partitioning is shown in Fig. 3. The 6-long sequence $(d_1, d_2, \cdots, d_6) = (\mathbf{00}, \mathbf{01}, \mathbf{11}, \mathbf{10}, \mathbf{00}, \mathbf{11})$ of information bit pairs $(m = 2)$ is encoded in an Ungerboeck style encoder to yield the 8-PSK sequence $(\mathbf{0}, \mathbf{2}, \mathbf{7}, \mathbf{5}, \mathbf{1}, \mathbf{6})$. The information bits are interleaved—on a pairwise basis—and encoded again into the sequence (6, 7, 0, 3, 0, 4). We deinterleave the second encoder's output symbols to ensure that the ordering of the two information bits partly defining each symbol corresponds to that of the first encoder, i.e., we now have the sequence (0, 3, 6, 4, 0, 7). Finally, we transmit the first symbol of the first encoder, the second symbol of the second encoder, the third of the first encoder, the fourth symbol of the second encoder, etc., $(\mathbf{0}, 3, \mathbf{7}, 4, \mathbf{1}, 7)$. Thus, the parity bit is alternately chosen from the first and second encoder (bold, notbold, bold, etc.). Also, the kth information bit pair exactly determines two of the three bits of the kth symbol x_k. This ensures that each information bit pair defines part of the constellation of an 8-PSK symbol exactly once.

B. Interleaver and Code Constraints

By deinterleaving the output of the second decoder, each symbol index k before the selector in Fig. 1 has the property of being associated with input information bit group index k, regardless of the actual interleaving rule. However, from the standpoint of the second component *decoder*, it will become evident (see Section III) that with the alternate selection chosen, the interleaver must map even positions to even positions and odd ones to odd ones (or even–odd, odd–even). Other than this constraint, the interleaver can be chosen to be pseudorandom or modified to avoid low distance error events.

A constraint on the component code was made in [5] such that the corresponding trellis diagram of the convolutional

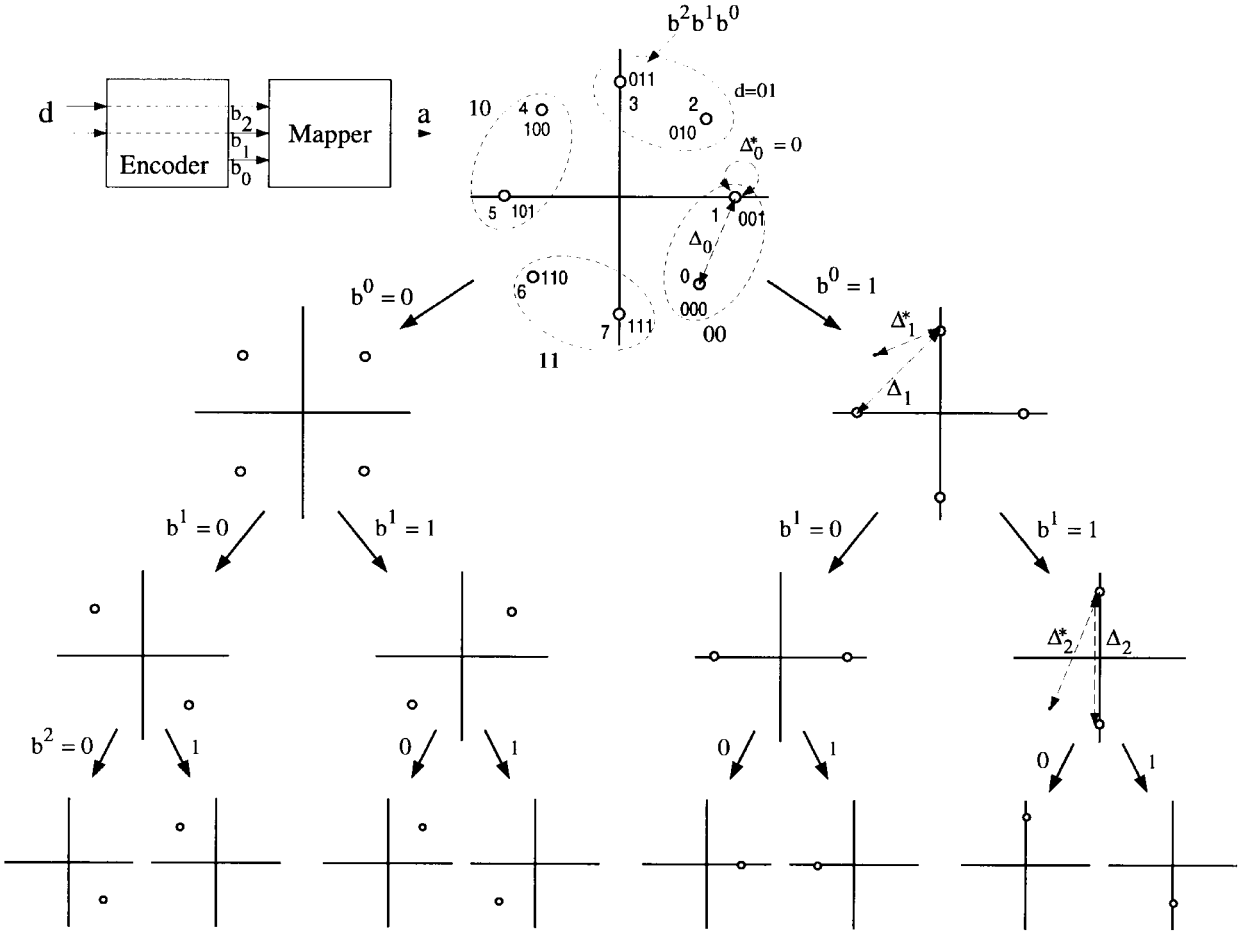

Fig. 3. Set partitioning for 8-PSK. Dotted ovals denote subsets corresponding to the different combinations of d. The distances Δ_i are relevant for code design.

encoders should have no parallel transitions. This ensures that each information bit benefits from the parallel concatenation and interleaving. This condition can be relaxed under a number of conditions. The first, proposed in [12], applies if the interleaver no longer keeps each group of m bits unchanged during interleaving. Remember that we have so far assumed that the interleaver keeps the input unchanged within each group of information bits, and the corresponding symbol deinterleaver does not modify its symbol inputs (except for the actual re-ordering of their positions, of course). In [12], the above condition was relaxed for 8-PSK with $m = 2$ where the interleaver swapped the two information bits and the code allowed two parallel transitions per state. For 8-PSK with $m = 2$, this ensures that each information bit influences either the states of the upper or lower encoder—but never both. A slight advantage for a small number of decoding iterations was reported. Unless otherwise stated, the examples in this paper assume a nonmodifying interleaver. The second case in which we allow parallel transitions is when we desire a very high bandwidth efficiency. Due to the higher operating SNR and the large Euclidean distance that separates the subsets of signal points that define parallel transitions (assuming sensible set partitioning and mapping), uncoded information bits receive ample protection at least in the cases of 8-PSK transmitting 2.5 information bits/symbol and 64-QAM with 5 bits/symbol. The transmission of uncoded bits has been proposed for the multilevel approach of [3] where channel capacity arguments show that these two bits theoretically need only minimal (if any) coding protection when five information bits are sent using one 64-QAM symbol.

In the following, a heuristic rule is given in order to determine the number of uncoded bits per symbol. It is based on the experience that the BER of TTCM schemes (with large block lengths) reaches a value of $P_b \approx 10^{-5}$ at a signal-to-noise ratio E_s/N_0^* which is approximately 1 dB above the corresponding channel capacity [5]. Let us consider the sequence of increasing inner-set distances Δ_i when following down the partitioning of the corresponding signal set (for an example of partitioning an 8-PSK constellation, refer to Fig. 3). For each distance, we can evaluate a rough approximation of the BER in the uncoded case, by applying the well-known formula

$$P_b(\Delta_i) = \frac{1}{2}\ \text{erfc}\ \sqrt{\frac{E_s\Delta_i}{4N_0}}. \tag{1}$$

By using the above formula to approximate the BER of the uncoded bits with $P_b(\Delta_i)$, two approximations are included.

- The error propagation from the partition levels which include coded bits into the partition levels with uncoded bits is neglected.

- Moreover, the number of nearest neighbors is not included in the calculation, only the pure distance is used to evaluate (1).

As a result, we can identify at which level of the partition chain the corresponding uncoded bits have enough protection based on the distance Δ_i and the given SNR E_s/N_0^* to bring the BER below $P_b = 10^{-5}$. Two examples are given in the following.

- *Example 1:*
  - — Signal set: four-dimensional 8-PSK.
  - — Desired information rate: 2.5 bits/symbol.
  - — The two 8-PSK symbols are generated by the rule [10] $\binom{y_1}{y_2} = z^5\binom{4}{4} + z^4\binom{0}{4} + z^3\binom{2}{2} + z^2\binom{0}{2} + z^1\binom{1}{1} + z^0\binom{0}{1}$ modulo 8. The parity bit is z^0; the information bits are z^1–z^5.
  - — Corresponding channel capacity: 8.8 dB [4] $\Longrightarrow$ $E_s/N_0^* = 9.8$ dB.
  - — Sequence of distances Δ_i for the partition chain of the signal set [10] and corresponding uncoded BER's shown in (a), at the bottom of the page.
  - — Conclusion: three encoded bits (including the parity bit) are necessary to reach the desired BER for the uncoded bits (hence, $\tilde{m} = 2$).
- *Example 2:*
  - — Signal set: two-dimensional 64-QAM.
  - — Desired information rate: 5 bits/symbol.
  - — Corresponding channel capacity: 16.2 dB [4] $\Longrightarrow$ $E_s/N_0^* = 17.2$ dB.
  - — Sequence of distances Δ_i for the partition chain of the signal set [4] and corresponding uncoded BER's given in (b), found at the bottom of the page.
  - — Conclusion: again, three encoded bits are necessary to reach the desired BER for the uncoded bits ($\tilde{m} = 2$).

A further condition on the code, which has its origins at the decoder [(8) in Section III-B], is that the information bits in step k do not affect the value of the parity bits at step k; this condition was also proposed for good TCM codes in [4].

In [13], an algorithm was presented that modifies an interleaver for binary turbo codes in a controlled, but random fashion. It tries to maximize the minimal distance between codewords whose corresponding information difference vectors have a small weight (typically, 1–5). The algorithm is based on the distance properties of the component codes, and works by attempting to break interleaver patterns leading to small codeword distances. In principle, the algorithm can be used for TTCM interleaver optimization as well, even though the interleaver no longer maps single bits. Modifying the interleaver might be especially useful for very small block sizes where a random interleaver is likely not to be the best choice.

C. Component Code Design

In an initial attempt to find good component codes, we have used an exhaustive computer search similar to [4] that maximizes the minimal distance of each component code under consideration of randomly selecting the parity bits of each second symbol.

In [4, eq. (15b)], it is stated that the minimal distance is bounded by

$$d_{\text{free}}^2 \geq \Delta_{\text{free}}^2 = \min \sum_{i=k}^{k+L} \Delta_{q(\boldsymbol{e}_i)}^2 \equiv \min\, \Delta^2[\boldsymbol{e}(D)] \qquad (2)$$

minimizing over all nonzero code sequences $\boldsymbol{e}(D)$. The variable $q(\boldsymbol{e}_i)$ is the number of trailing zeros in $\boldsymbol{e}_i$. The values $\Delta_0^2, \Delta_1^2, \Delta_2^2, \cdots$, are the squared minimal Euclidean distances between signals of each subset, and must be replaced by $\Delta_0^{*2}, \Delta_1^{*2}, \Delta_2^{*2}, \cdots$, when the corresponding transmitted symbol was "punctured"; the distances are shown in Fig. 3. These new distances can be calculated by assuming that the "random" parity bit takes its worst case value and minimizes the distance between elements of the subsets. We obtained the results of Table I, where the parity check polynomials in octal notation are given as in [4]. Note that in the case of 8-PSK, the punctured code has a loss compared to uncoded QPSK ($d_{\text{free}}^2/d_{\text{QPSK}}^2 = d_{\text{free}}^2/2 = 0.878$), but we must not forget that we are able to transmit an *additional* (parity) bit every $2 \cdot n$ 8-PSK symbols, albeit with little protection within the signal constellation.

It should be noted that better results might be obtained if the code search maximizes the smallest distance between subsets

Part. Level	0	1	2	**3**	4	5	6
Δ_i	0.586	1.172	2	**4**	4	8	∞
$P_b(\Delta_i)$	0.05	0.009	0.001	$6.5 \cdot 10^{-6}$	$6.5 \cdot 10^{-6}$	$3.5 \cdot 10^{-10}$	–

(a)

Part. level	0	1	2	**3**	4	5	6
Δ_i	0.095	0.19	0.38	**0.76**	1.52	3.05	∞
$P_b(\Delta_i)$	0.06	0.013	$8 \cdot 10^{-4}$	$4 \cdot 10^{-6}$	$1.3 \cdot 10^{-10}$	$1.8 \cdot 10^{-19}$	–

(b)

TABLE I
"PUNCTURED" TCM CODES WITH BEST MINIMAL DISTANCE FOR 8-PSK AND QAM (IN OCTAL NOTATION)

Code	$\bar{m}$	$H^0(D)$	$H^1(D)$	$H^2(D)$	$H^3(D)$	$d^2_{\text{free}}/\Delta_0^2$
2-dim. 8-PSK, 8 states	2	11	02	04		3
4-dim. 8-PSK, 8 states	2	11	06	04		3
2-dim. 8-PSK, 16 states	2	23	02	10		3
4-dim. 8-PSK, 16 states	2	23	14	06		3
2-dim. Z^2, 8 states	3	11	02	04	10	2
2-dim. Z^2, 16 states	3	21	02	04	10	3
2-dim. Z^2, 8 states	2	11	04	02		3
2-dim. Z^2, 16 states	2	21	04	10		4

of the component code corresponding to small input Hamming weights.

III. THE DECODER

The iterative decoder is similar to that used to decode binary turbo codes, except that there is a difference in the nature of the information passed from one decoder to the other, and in the treatment of the very first decoding step (half iteration). A major novelty is the fact that each decoder alternately sees its corresponding encoder's noisy output symbol(s), and then the other encoder's noisy output symbol(s). The information bits, i.e., systematic bits that partly resulted in the mapping of each of these symbols, are correct—in the sense of being identical to the corresponding encoder output—in both cases. However, this is not so for the parity bits since these belong to the other encoder every other group of n symbol—we have indexed these symbols with "*" and will call these symbols "punctured" for brevity. Note that in the following, the attribute "*" or "punctured" refers to the pertinent component decoder only.

In the binary turbo coding scheme, it can be shown that the component decoder's output can be split into three additive parts (when in the logarithmic or log-likelihood ratio domain [14]) for each information bit k: the systematic component (corresponding to the received systematic value for bit k), the *a priori* component (the information given by the other decoder for bit k), and the extrinsic component (that part that depends on all other inputs). Only the so-called extrinsic component may be given to the next decoder; otherwise, information will be used more than once in the next decoder [1], [15]. Furthermore, these three components are disturbed by independent noise.

Here, the situation is complicated by the fact that the systematic component cannot be separated from the extrinsic one since the noise that affects the parity component also affects the systematic one because—unlike in the binary case—the systematic information is transmitted together with parity information in the same symbol(s). However, we can split the output into two different components: 1) *a priori* and 2) (extrinsic and systematic). Each decoder must now pass just the latter to the next decoder, and care is taken not to use the systematic information more than once in each decoder. Note that we have written (extrinsic and systematic) in parentheses to stress their inseparability. In the Appendix, we have derived the symbol-by-symbol MAP decoder for nonbinary trellises.

A. Extrinsic, A Priori, and Systematic Components

Because we will now take a close look at the way the iterative decoder works, we have decided to write logarithms of probabilities, denoted by $L(\,)$, for brevity and clarity. We had stated above that we wish to pass the component (extrinsic and systematic) to the next decoder in which it is used as *a priori* information. We shall define the component (extrinsic and systematic) as that part of the MAP output that does not depend on the *a priori* information $\Pr\{d_k = i\}$. In other words, we must subtract the *a priori* term (A4)

$$L_a(d_k = i) = \log \Pr\{d_k = i\} \tag{3}$$

from the logarithm of (A10) to obtain a term independent of the *a priori* information $\Pr\{d_k = i\}$

$$L_{e\&s}(d_k = i) = \log \Pr\{d_k = i|\underline{\boldsymbol{y}}\} - \log \Pr\{d_k = i\} \tag{4}$$

$\forall i \in \{0, \cdots, 2^m - 1\}$. This can be done since $\Pr\{d_k = i\}$ is a factor in γ_i that does not depend on M or M' and can be written outside the summations in (A10). We will abbreviate $L_{e\&s}(d_k = i)$ in diagrams and when written in text by $(e\&s)$.

However, the decoder must be formulated in such a way that it correctly uses the channel observation $\boldsymbol{y}_k$ and the *a priori* information $\Pr\{d_k = i\}$ at each step k. This is best illustrated in a diagram: see Fig. 4. Shown on the left is the interrelation of both MAP decoders for one information bit in a binary turbo coding scheme. We have denoted the extrinsic component—omitting the index k—by e, the *a priori* component by a, and the systematic and parity ones by s and p. Bold letters indicate that the variables correspond directly to the upper decoder, not bold ones correspond directly to the lower decoder. Of course, the decoders have memory (indicated by inputs α and β), so each input will affect many neighboring outputs; we have only shown the relationships for one bit. Both decoders are symmetrical as they only pass the newly generated extrinsic information to the next decoder.

The right side shows the decoders for TTCM where the upper decoder sees a punctured symbol (which was output by the other decoder: "$*$-mode"); in the example of our encoder in Fig. 2, it might have received a noisy observation of symbol $x_2 = 3$. The corresponding symbol from the upper encoder ($\mathbf{2}$) was not transmitted. The upper decoder now ignores this symbol—indicated by the position of the upper switch—as far as the direct channel input is concerned: in (A3), we set

$$L_{p\&s} = \log p(\boldsymbol{y}_k|d_k = i, S_k = M, S_{k-1} = M') \to 0 \tag{5}$$

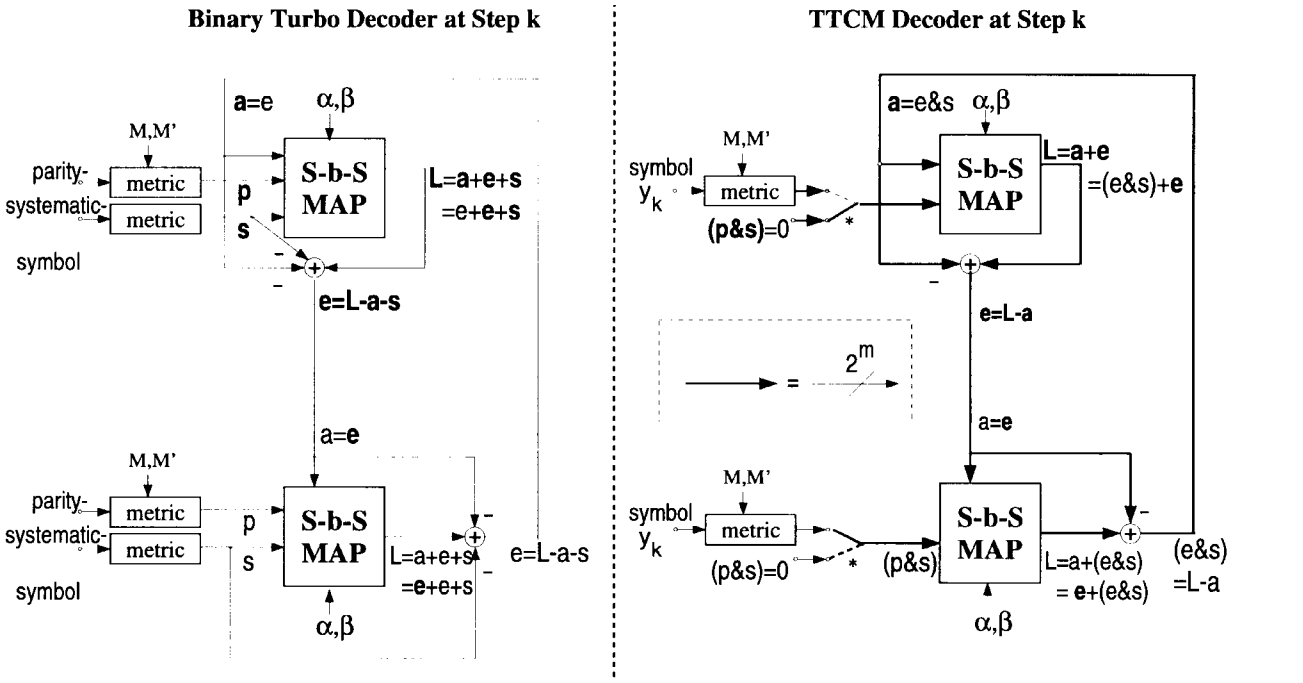

Fig. 4. Decoders for binary turbo codes and TTCM. Note that the labels and arrows apply only to one specific info bit (left) or group of m info bits (right). The interleavers/deinterleavers are not shown.

illustrated in Fig. 4 by $(p\&s) = 0$. The only input for this step in the trellis is *a priori* information $\boldsymbol{a}$ from the other decoder, and this includes the systematic information s. The output of the MAP, for this transition, is the sum of this *a priori* information $\boldsymbol{a}$ and newly computed extrinsic information $\boldsymbol{e}$, which is

$$L_e = L_{e\&s} \tag{6}$$

since we have set $p\&s$ to zero. The *a priori* information $\boldsymbol{a}$ is subtracted, and the extrinsic information $\boldsymbol{e}$ is passed to the second decoder as its *a priori* information a (see the equations written in Fig. 4). The second decoder, however, sees a symbol that *was* generated by its encoder; hence, it can compute

$$L_{p\&s}(d_k = i) = \log p(\boldsymbol{y}_k | d_k = i, S_k = M, S_{k-1} = M') \tag{7}$$

for each i, and subsequently $L_{e\&s}(d_k = i)$ which is used as the *a priori* input of the upper decoder in the next iteration. The setting of the switches will alternate from one group of bits (index k) to another.

B. Metric Calculation in the First Decoding Stage

The above applies only to the decoding process where *a priori* information for the upper decoder is already available, which is the case in all but the very first decoding stage. We had relied on the fact that if the upper decoder sees a group of n punctured symbols, we had embedded the systematic information, so to speak, in the *a priori* input. Before the first decoding pass of the upper decoder, we need to set the *a priori* information to contain the systematic information for the $*$ transitions, where the transmitted symbol was determined partly by the information group d_k, but also by the unknown parity bit $b^{0,*} \in \{0, 1\}$ produced by the *other* encoder. We thus set the *a priori* information, by applying the mixed Bayes' rule, to

$$\begin{aligned} \Pr\{d_k = i\} \leftarrow \Pr\{d_k = i | \boldsymbol{y}_k\} &= \text{const} \cdot p(\boldsymbol{y}_k | d_k = i) \\ &= \text{const} \cdot \sum_{j \in \{0,1\}} p\Big(\boldsymbol{y}_k, b_k^{0,*} = j | d_k = i\Big) \\ &= \frac{\text{const}}{2} \cdot \sum_{j \in \{0,1\}} p\Big(\boldsymbol{y}_k | d_k = i, b_k^{0,*} = j\Big) \end{aligned} \tag{8}$$

where it is assumed that $\Pr\{b_k^{0,*} = j | d_k\} = \Pr\{b_k^{0,*} = j\} = 1/2$, i.e., the parity bit in the symbol x_k is statistically independent of the information bit group d_k and equally likely to be zero or one. Furthermore, the initial *a priori* probability of d_k—prior to any decoding—is assumed to be constant for all i. Above, it is not necessary to calculate the value of the constant since the value of $\Pr\{d_k = i | \boldsymbol{y}_k\}$ can be determined by dividing the summation $\sum_{j \in \{0,1\}}$ by its sum over all i (normalization). If the upper decoder is not at a $*$ transition, then we simply set $\Pr\{d_k = i\}$ to $(1/2^m)$.

C. The Complete Decoder

The complete decoder is shown in Fig. 5. By "metric s," we mean the evaluation of (8). All thin signal paths are channel outputs or values of $\log p(\boldsymbol{y}_k | d_k = i, S_k = M, S_{k-1} = M')$; thick paths represent a group of 2^m values of logarithms of probabilities.

We would like to ensure that punctured and unpunctured symbols are uniformly spread, i.e., occur alternately at both of the decoders' inputs. With our encoder's selector, the interleaver must be chosen as in Section II-B.

1) Avoiding Calculation of Logarithms and Exponentials: Since we work with logarithms of probabilities, it

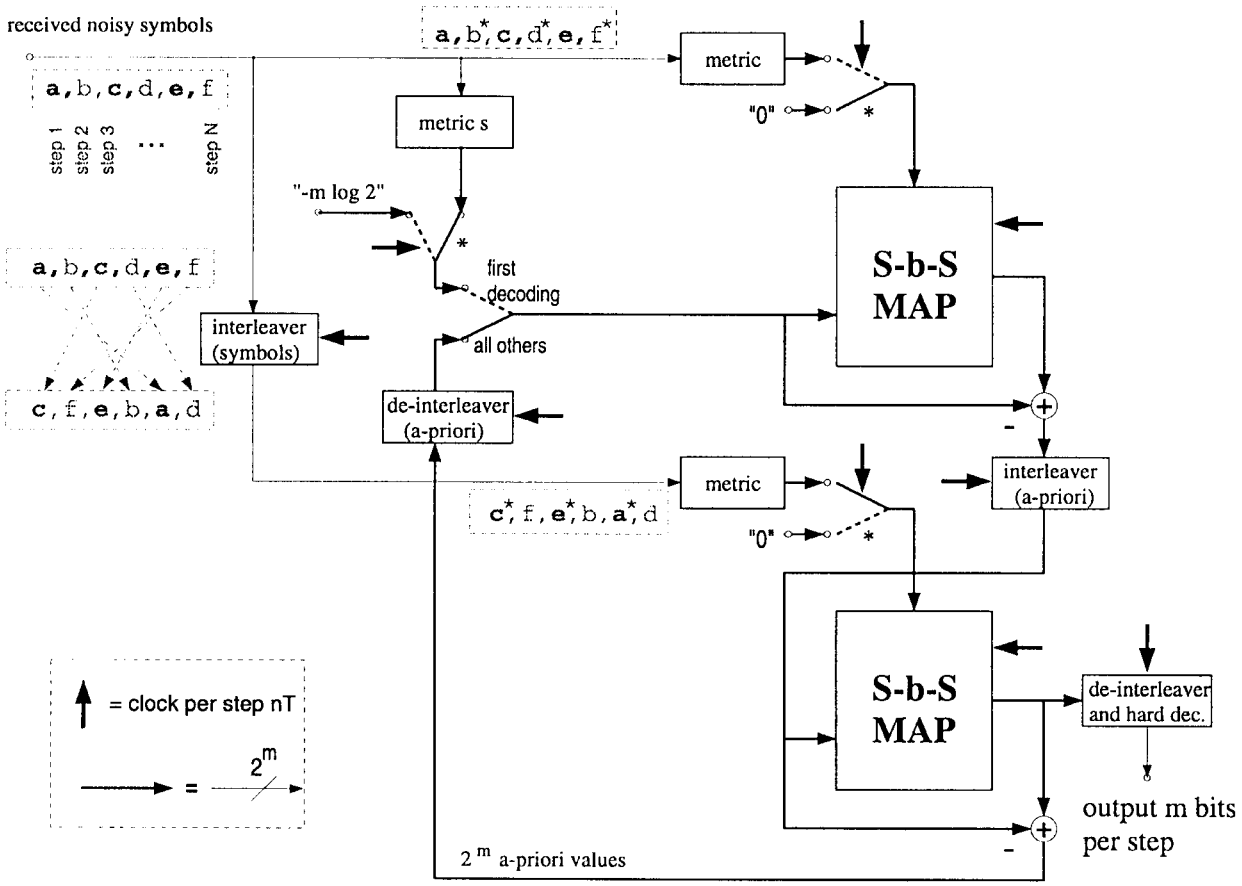

Fig. 5. Complete decoder.

is undesirable to switch between probabilities and their logarithms. This becomes necessary, however, at the following four stages in the decoder.

1) In (8), when we sum over probabilities $(\sum_{j\in\{0,1\}} p(\boldsymbol{y}_k|d_k=i,b_k^{0,*}=j))$, but the demodulator provides us with $\log(p(\boldsymbol{y}_k|d_k=i,b_k^{0,*}=j))$.
2) When evaluating $\sum_i \sum_{j\in\{0,1\}} p(\boldsymbol{y}_k|d_k=i,b_k^{0,*}=j)$ to normalize (8) to unity.
3) When normalizing the sum of (A10) to unity.
4) When calculating the hard decision of each individual bit given the values of (A10).

All of the above mandate the calculation of the logarithm of the sum over exponentials (when the decoder otherwise operates in the log domain). By recursively applying the relation [14]

$$\begin{aligned}\ln(e^{\delta_1}+e^{\delta_2}) &= \max(\delta_1,\delta_2)+\ln\left(1+e^{-|\delta_2-\delta_1|}\right)\\ &= \max(\delta_1,\delta_2)+f_c(|\delta_1-\delta_2|)\end{aligned} \tag{9}$$

the problem can be solved for an arbitrary number of exponentials. The correction function $f_c(\cdot)$ can be realized with a one-dimensional table with as few as eight stored values [14]. When implementing the above, we noticed negligible degradation.

2) Subset Decoding: When the component code's trellis contains parallel transitions, this reduces the required decoding complexity: during the iterations, it is not necessary to decide on, or calculate soft outputs for, the uncoded bits that cause these parallel transitions. In the MAP decoders, the parallel transitions can be merged, which mathematically corresponds to adding the path transition probabilities $\gamma_i\{\boldsymbol{y_k}, M', M)$ of the parallel transitions. It is clear that the sum is over just those $2^{(m-\tilde{m})}$ values of i which represent all combinations of the statistically independent uncoded bits. There is one such sum for every particular combination of the remaining $\tilde{m}$ bits which are encoded. From then on, the MAP decoder calculates and passes on only the likelihoods of these $\tilde{m}$ bits. Hence, the (de-)interleaver needs to operate only on groups of $\tilde{m}$ bits. During the very last decoding stage, decisions (and if desired, reliabilities) for the $(m-\tilde{m})$ uncoded bits can be generated by the MAP decoder, either optimally or suboptimally, e.g., by taking into account only those transitions between the most likely states along the trellis.

IV. Examples and Simulations

As examples, we have used 2-D 8-PSK (with $N = 1024$ and 5000), 2-D 16-QAM (with $N = 683$ and 5000), 4-D 8PSK (with $N = 40$, 200, and 3000), and 2-D 64-QAM (with $N = 40$, 200, and 3000). The interleavers were chosen to be pseudorandom, and identical for each transmitted block. In all cases, the component decoders were symbol-by-symbol MAP decoders operating in the log domain. The number of trellis states was eight. To help the reader compare curves for different values of N, the x axes of the respective curves

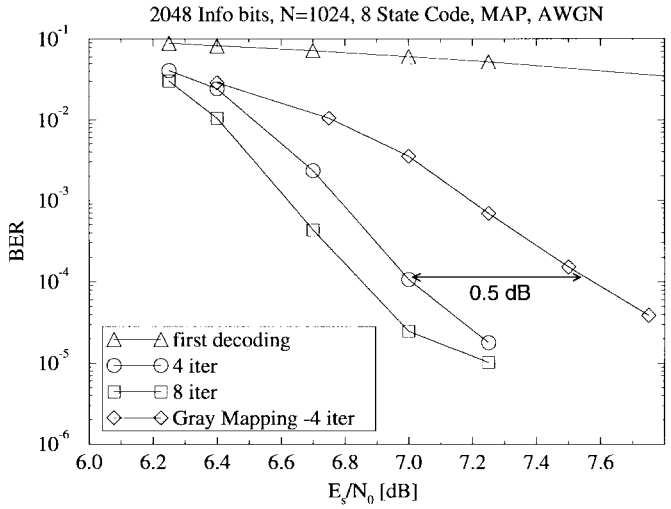

Fig. 6. TTCM for 2-D 8-PSK, 2 bits/symbol. Channel capacity: 2 bits/symbol at 5.9 dB.

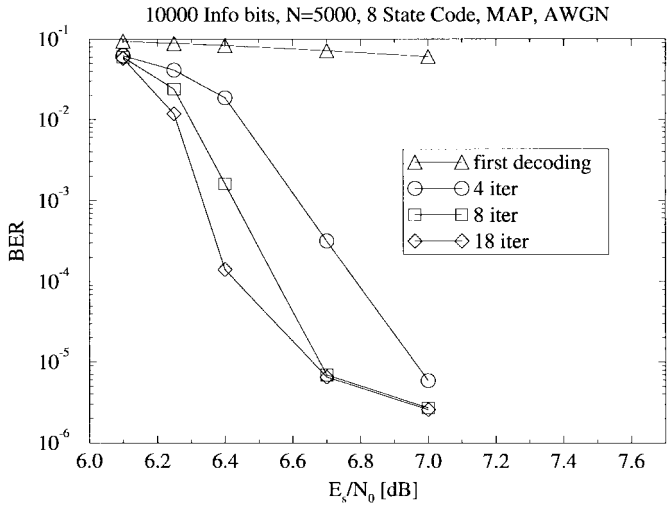

Fig. 7. TTCM for 2-D 8-PSK, 2 bits/symbol. Channel capacity: 2 bits/symbol at 5.9 dB.

were chosen to show the same range of SNR. The channel was modeled to be AWGN, where N_0 is the one-sided noise power spectral density. The small block sizes of 200, 1000, and roughly 2050 information bits were included to verify that the schemes work well in applications that tolerate only short end-to-end delays. In general, it must be borne in mind that when comparing different approaches to channel coding, the block size (or other measure of fundamental delay) must be kept constant.

The BER curves are shown in Figs. 6 and 7 for 8-PSK with 2 bits/symbol (bps), in Figs. 8 and 9 for 16-QAM with 3 bps, in Figs. 10–12 for 8-PSK with 2.5 bps, and finally in Figs. 13–15 for 64-QAM with 5 bps.

One iteration is defined as comprising two decoding steps: one in each dimension. The weak asymptotic performance of the component code (evident after from the high BER after the very first decoding step) seems not to affect the performance of the turbo code after a few iterations since good BER can be achieved at less than 1 dB from Shannon's limit for large

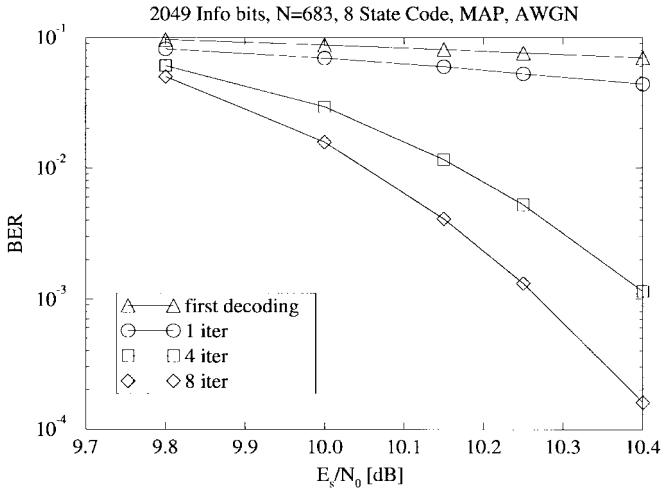

Fig. 8. TTCM for 2-D 16-QAM, 3 bits/symbol. Channel capacity: 3 bits/symbol at 9.3 dB.

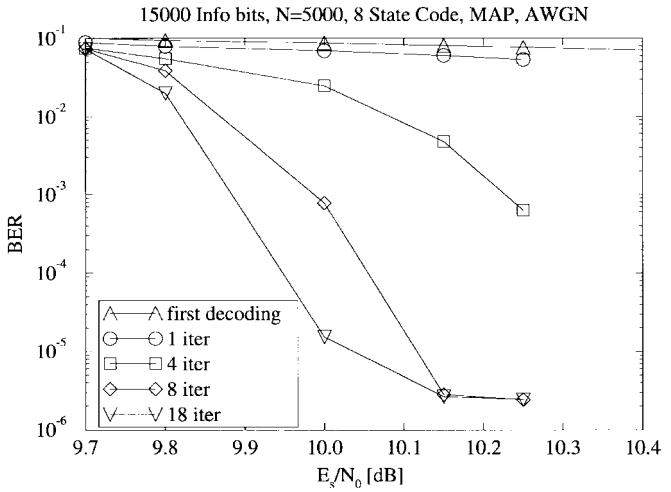

Fig. 9. TTCM for 2-D 16-QAM, 3 bits/symbol. Channel capacity: 3 bits/symbol at 9.3 dB.

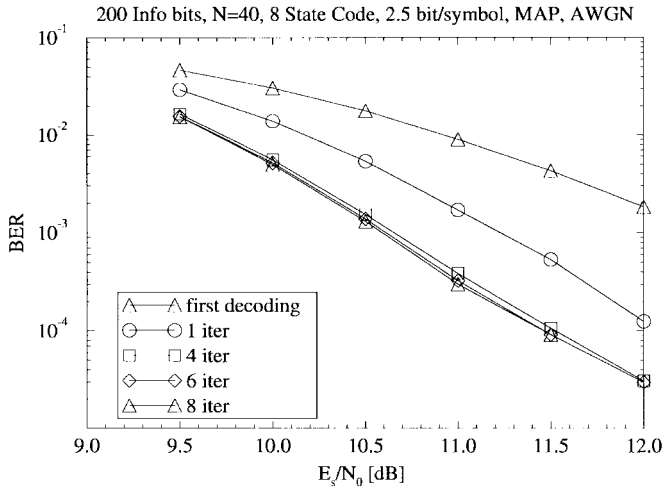

Fig. 10. TTCM for 4-D 8-PSK, 2.5 bits/symbol. Channel capacity: 2.5 bits/symbol at 8.8 dB.

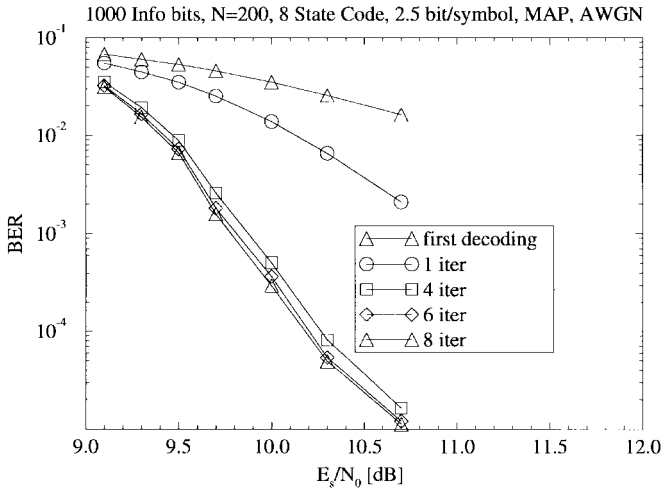

Fig. 11. TTCM for 4-D 8-PSK, 2.5 bits/symbol. Channel capacity: 2.5 bits/symbol at 8.8 dB.

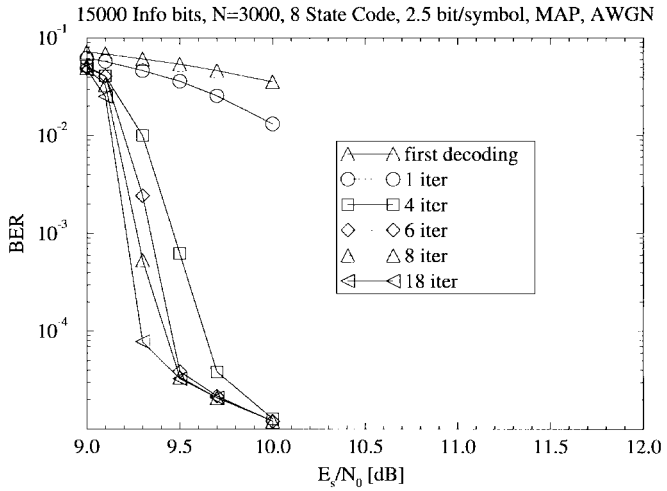

Fig. 12. TTCM for 4-D 8-PSK, 2.5 bits/symbol. Channel capacity: 2.5 bits/symbol at 8.8 dB.

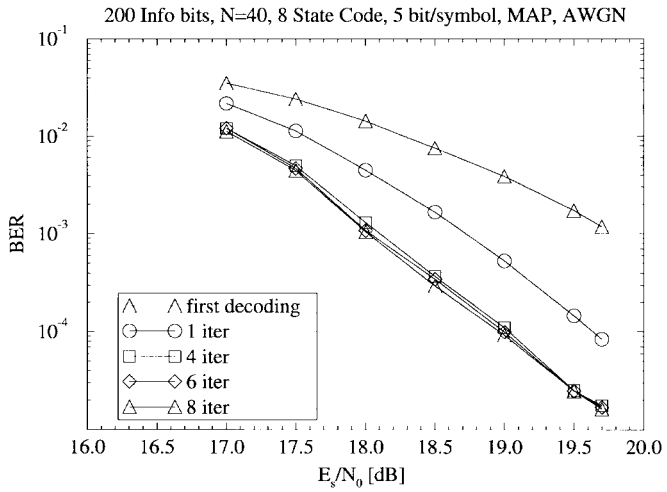

Fig. 13. TTCM for 2-D 64-QAM, 5 bits/symbol. Channel capacity: 5 bits/symbol at 16.2 dB.

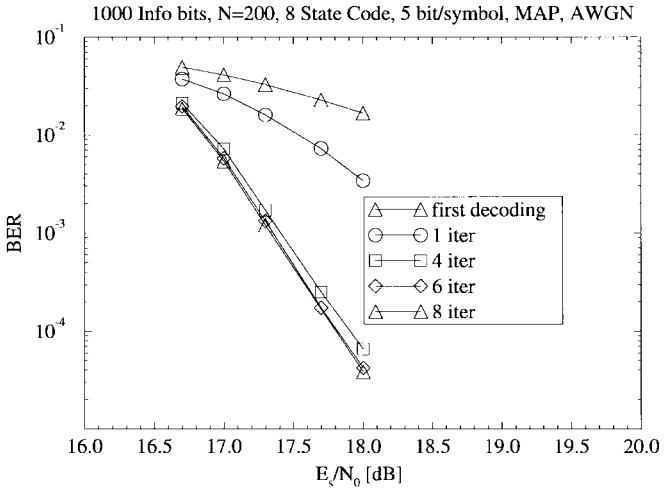

Fig. 14. TTCM for 2-D 64-QAM, 5 bits/symbol. Channel capacity: 5 bits/symbol at 16.2 dB.

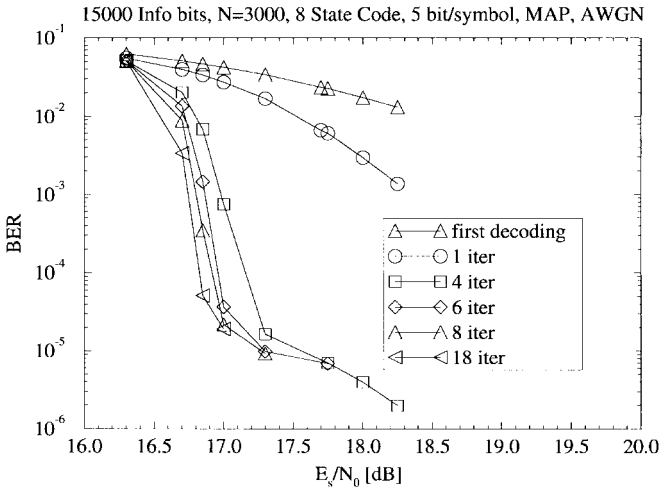

Fig. 15. TTCM for 2-D 64-QAM, 5 bits/symbol. Channel capacity: 5 bits/symbol at 16.2 dB.

interleaver sizes N. For comparison, Fig. 6 includes the results for a Gray mapping scheme for 2-D 8-PSK as presented in [2]; it has the same complexity (when measured as the number of trellis branches per information bit) as our four-iteration scheme and the same number of information bits per block: 2048. The number of states of the binary trellis for the Gray mapping scheme is eight, hence, there are 2048 $\times$ 8 $\times$ 2 trellis branches per decoding in each dimension; in our TTCM scheme, there are 1024 $\times$ 8 $\times$ 4 branches. Compared to TCM with 64-state Ungerboeck codes and 8-PSK (not included in the figures), we achieve a gain of 1.7 dB at a BER of 10^{-4}. At this BER, our proposed TTCM system has a 0.5 dB advantage over the Gray mapping scheme after four iterations. Rather than comparing all of our examples with other coding techniques, we simply point out that good BER can be achieved within 1 dB from Shannon's limit as long as the block size is sufficiently large.

The results for the higher bandwidth-efficient examples are also encouraging, except for the fact that the characteristic flattening of the BER curves comes into effect at higher BER:

in the case of the two-dimensional schemes with 8-PSK and 16-QAM, this happens between 10^{-5} and 10^{-6}, whereas the BER curve begins to flatten at roughly a factor of 10 higher for the bandwidth-efficient schemes with 8-PSK and 64-QAM. However, turbo-coded systems will often be employed as an inner coding stage by concatenating a block code (e.g., RS or BCH code) with a turbo code in order to reach very low BER; in these cases, BER's of around 10^{-4} are sufficient.

V. Conclusions

We have presented a channel coding scheme (TTCM) that is bandwidth efficient and allows iterative turbo decoding of codes built around punctured parallel concatenated trellis codes together with higher order signaling. In contrast to using binary turbo codes and subsequent Gray mapping onto the constellation, we have designed the turbo code directly around two recursive TCM component codes. Thereby, the bitwise interleaver known from classical binary turbo codes is replaced by an interleaver operating on a group of bits. By adhering to a set of constraints for the component code and interleaver, the resulting code can be decoded iteratively using, e.g., symbol-by-symbol MAP component decoders working in the logarithmic domain to avoid numerical problems and reduce the decoding complexity. We outlined the structure of the iterative decoder, and derived the symbol-by-symbol MAP algorithm for nonbinary trellises. Furthermore, we illustrated the differences compared to the binary case as far as the definitions of extrinsic, systematic, and extrinsic components of the symbol-by-symbol MAP output are concerned. In the case of a TTCM decoder, it was shown that it is necessary to group the systematic and extrinsic components together.

A search for good component codes was performed, taking into account the puncturing at the transmitter. The selection criterion was their minimal distance. Using these simplest of these codes (memory three), simulations were undertaken, and the results indicate a marked improvement over classical TCM with Ungerboeck codes, and performs better than turbo codes and Gray mapping at comparable complexity. Most importantly, error correction close to Shannon's limit is possible for highly bandwidth-efficient schemes that are of relatively low complexity.

Possible further areas of study could be better overall code design (taking into account the interleaver and the component codes), analytical performance evaluation, as well as a comprehensive study of implementation issues.

Appendix
the Symbol-by-Symbol Map Algorithm for Nonbinary Trellises

We will briefly rederive the symbol-by-symbol MAP algorithm [9] (MAP for short) for nonbinary trellises. At the moment, we consider just a classical TCM scheme, with *a priori* information—on each group of info bits d_k—to be used in the decoder. Let the number of states be 2^ν, and the state at step k be denoted by $S_k \in \{0, 1, \cdots, 2^\nu - 1\}$. The group of m information bits d_k can be represented by an integer in the range $(0 \cdots 2^m - 1)$ and is associated with the transition from step $k-1$ to k. The receiver observes N sets of n noisy symbols, where n such symbols are associated with each step in the trellis, i.e., from step $k-1$ to step k the receiver observes $\boldsymbol{y}_k = (y_k^0, \cdots, y_k^{(n-1)})$. The total received sequence be $\underline{\boldsymbol{y}} = \underline{\boldsymbol{y}}_1^N = (\boldsymbol{y}_1, \cdots \boldsymbol{y}_N)$. It is the TCM encoder output sequence $(\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N)$ that has been disturbed by additive white Gaussian noise with one-sided noise-power spectral density N_0. Each $\boldsymbol{x}_k = (x_k^0, \cdots, x_k^{(n-1)})$ is the group of n symbols output by the mapper at step k.

The goal of the decoder is to evaluate $\Pr\{d_k | \underline{\boldsymbol{y}}_1^N\}$ for each d_k, and for all k. Let us define the forward and backward variables

$$\alpha_{k-1}(M') = \frac{p\left(S_{k-1} = M', \underline{\boldsymbol{y}}_1^{k-1}\right)}{p(\underline{\boldsymbol{y}}_1^{k-1})} \tag{A1}$$

$$\beta_k(M) = \frac{p\left(\underline{\boldsymbol{y}}_{k+1}^N | S_k = M\right)}{p\left(\underline{\boldsymbol{y}}_{k+1}^N | \underline{\boldsymbol{y}}_1^k\right)}. \tag{A2}$$

The branch transition probability for step k, $p(d_k = i, \boldsymbol{y}_k, S_k = M | S_{k-1} = M')$, is denoted by and calculated as

$$\begin{aligned}\gamma_i(\boldsymbol{y}_k, M', M) = & p(\boldsymbol{y}_k | d_k = i, S_k = M, S_{k-1} = M') \\ & \cdot q(d_k = i | S_k = M, S_{k-1} = M') \\ & \cdot \Pr\{S_k = M | S_{k-1} = M'\}.\end{aligned} \tag{A3}$$

$q(d_k = i | S_k = M, S_{k-1} = M')$ is either zero or one, depending on whether encoder input $i \in \{0, 1, \cdots, 2^m - 1\}$ is associated with the transition from state $S_{k-1} = M'$ to $S_k = M$ or not. In the last component of (A3), we use the *a priori* information

$$\begin{aligned}&\Pr\{S_k = M | S_{k-1} = M'\} \\ &\quad = \begin{cases} \Pr\{d_k = 0\}, & \text{if } q(d_k = 0 | S_k = M, S_{k-1} = M') = 1 \\ \Pr\{d_k = 1\}, & \text{if } q(d_k = 1 | S_k = M, S_{k-1} = M') = 1 \\ \quad \cdots & \quad \cdots \\ \Pr\{d_k = 2^m - 1\}, & \\ & \text{if } q(d_k = 2^m - 1 | S_k = M, \\ & \quad S_{k-1} = M') = 1 \end{cases} \\ &\quad = \Pr\{d_k = j\}\end{aligned} \tag{A4}$$

where j: $q(d_k = j | S_k = M, S_{k-1} = M') = 1$. If there does not exist a j such that $q(d_k = j | S_k = M, S_{k-1} = M') = 1$, then $\Pr\{S_k = M | S_{k-1} = M'\}$ is set to zero.

We must bear in mind that the event $(d_k = i, \boldsymbol{y}_k, S_{k-1} = M')$ has no influence on $\underline{\boldsymbol{y}}_{k+1}^N$ if S_k is known, and hence

$$\begin{aligned}&p\left(\underline{\boldsymbol{y}}_{k+1}^N | S_k = M\right) \\ &\quad = p(\underline{\boldsymbol{y}}_{k+1}^N | d_k = i, \boldsymbol{y}_k, S_k = M, S_{k-1} = M').\end{aligned} \tag{A5}$$

Using (A5) and the fact that

$$p\left(\underline{\boldsymbol{y}}_1^{k-1}\right) = \frac{p\left(\underline{\boldsymbol{y}}_1^k\right)}{p\left(\boldsymbol{y}_k | \underline{\boldsymbol{y}}_1^{k-1}\right)} \tag{A6}$$

the product of (A1), (A2), and (A4) can be shown to be

$$\begin{aligned}&\alpha_{k-1}(M')\cdot\beta_k(M)\cdot\gamma_i(\boldsymbol{y}_k,M',M)\\&\quad=p\Big(S_{k-1}=M',\underline{\boldsymbol{y}_1^{k-1}}\Big)\\&\qquad\cdot p\Big(\underline{\boldsymbol{y}_{k+1}^{N}},d_k=i,\boldsymbol{y}_k,S_k=M|S_{k-1}=M'\Big)\\&\qquad\cdot\frac{p\Big(\boldsymbol{y}_k|\underline{\boldsymbol{y}_1^{k-1}}\Big)}{p\Big(\underline{\boldsymbol{y}_1^{N}}\Big)}.\end{aligned}\tag{A7}$$

Obviously

$$\begin{aligned}&p\Big(\underline{\boldsymbol{y}_{k+1}^{N}},d_k=i,\boldsymbol{y}_k,S_k=M|S_{k-1}=M'\Big)\\&\quad=p\Big(\underline{\boldsymbol{y}_{k+1}^{N}},d_k=i,\boldsymbol{y}_k,S_k=M|S_{k-1}=M',\underline{\boldsymbol{y}_1^{k-1}}\Big)\end{aligned}\tag{A8}$$

so we can rewrite (A7) as

$$\begin{aligned}&\alpha_{k-1}(M')\cdot\beta_k(M)\cdot\gamma_i(\boldsymbol{y}_k,M',M)\cdot\frac{1}{p\Big(\boldsymbol{y}_k|\underline{\boldsymbol{y}_1^{k-1}}\Big)}\\&\quad=p\Big(S_{k-1}=M',S_k=M,d_k=i,\underline{\boldsymbol{y}_1^{N}}\Big)\frac{1}{p\Big(\underline{\boldsymbol{y}_1^{N}}\Big)}\\&\quad=p\Big(S_{k-1}=M',S_k=M,d_k=i|\underline{\boldsymbol{y}_1^{N}}\Big).\end{aligned}\tag{A9}$$

Therefore, the desired output of the MAP decoder is

$$\begin{aligned}\Pr\{d_k=i|\underline{\boldsymbol{y}}\}=\text{const}\cdot\sum_{M}\sum_{M'}&\gamma_i(\boldsymbol{y}_k,M',M)\\&\cdot\alpha_{k-1}(M')\cdot\beta_k(M)\end{aligned}\tag{A10}$$

$\forall\ i\in\{0,\cdots,2^m-1\}$. The constant can be eliminated by normalizing the sum of (A10) over all i to unity. The probability $\Pr\{d_k=i|\underline{\boldsymbol{y}}\}$ comprises *a priori*, systematic, and extrinsic components since it depends on the complete received sequence as well as the *a priori* likelihoods of d_k.

All that remains now is to recursively define $\alpha_{k-1}(M')$ and $\beta_k(M)$. We begin by writing

$$\Pr\Big\{S_k=M|\underline{\boldsymbol{y}_1^{k-1}},\boldsymbol{y}_k\Big\}\cdot p\Big(\boldsymbol{y}_k|\underline{\boldsymbol{y}_1^{k-1}}\Big)=p\Big(\boldsymbol{y}_k,S_k=M|\underline{\boldsymbol{y}_1^{k-1}}\Big)\tag{A11}$$

and dividing both sides by $p(\boldsymbol{y}_k|\underline{\boldsymbol{y}_1^{k-1}})$ and expanding into the form

$$\begin{aligned}&\Pr\Big\{S_k=M|\underline{\boldsymbol{y}_1^{k}}\Big\}\\&\quad=\alpha_k(M)=\frac{\sum_{M'}p\Big(\boldsymbol{y}_k,\,S_k=M,S_{k-1}=M'|\underline{\boldsymbol{y}_1^{k-1}}\Big)}{\sum_{M}\sum_{M'}p\Big(S_k=M,S_{k-1}=M',\boldsymbol{y}_k|\underline{\boldsymbol{y}_1^{k-1}}\Big)}.\end{aligned}\tag{A12}$$

Because of (A8), we can write (A13), as shown at the bottom of the page. Defining

$$\gamma_T(\boldsymbol{y}_k,M',M)=\sum_{i=0}^{2^m-1}\gamma_i(\boldsymbol{y}_k,M',M)\tag{A14}$$

yields

$$\alpha_k(M)=\frac{\sum_{M'}\gamma_T(\boldsymbol{y}_k,M',M)\cdot\alpha_{k-1}(M')}{\sum_{M}\sum_{M'}\gamma_T(\boldsymbol{y}_k,M',M)\cdot\alpha_{k-1}(M')}.\tag{A15}$$

Similarly

$$\begin{aligned}&\beta_k(M)\\&=\frac{\sum_{M''}p\Big(S_{k+1}=M'',\underline{\boldsymbol{y}_{k+1}^{N}}|S_k=M\Big)}{p(\underline{\boldsymbol{y}_{k+1}^{N}}|\underline{\boldsymbol{y}_1^{k}})}\\&=\frac{\sum_{M''}p\big(S_{k+1}=M'',\boldsymbol{y}_{k+1}|S_k=M\big)\cdot p\Big(\underline{\boldsymbol{y}_{k+2}^{N}}|S_{k+1}=M''\Big)}{p\Big(\underline{\boldsymbol{y}_{k+1}^{N}}|\underline{\boldsymbol{y}_1^{k}}\Big)}\end{aligned}\tag{A16}$$

since $p(\underline{\boldsymbol{y}_{k+2}^{N}}|S_{k+1}=M'')=p(\underline{\boldsymbol{y}_{k+2}^{N}}|S_{k+1}=M'',\ \boldsymbol{y}_{k+1},\ S_k=M)$. Finally, we can calculate $\beta_k(M)$ recursively using

$$\begin{aligned}&\beta_k(M)\\&=\frac{\sum_{M''}p(S_{k+1}=M'',\boldsymbol{y}_{k+1}|S_k=M)\cdot\dfrac{p\Big(\underline{\boldsymbol{y}_{k+2}^{N}}|S_{k+1}=M''\Big)}{p\Big(\underline{\boldsymbol{y}_{k+2}^{N}}|\underline{\boldsymbol{y}_1^{k+1}}\Big)}}{\sum_{M''}\sum_{M}p\Big(S_{k+1}=M'',S_k=M,\boldsymbol{y}_{k+1}|\underline{\boldsymbol{y}_1^{k}}\Big)}\\&=\frac{\sum_{M''}\gamma_T(\boldsymbol{y}_{k+1},M,M'')\cdot\beta_{k+1}(M'')}{\sum_{M''}\sum_{M}\gamma_T(\boldsymbol{y}_{k+1},M,M'')\cdot\alpha_k(M)}.\end{aligned}\tag{A17}$$

In our implementation of the above algorithm, we have used logarithms of probabilities and logarithms of $\alpha_{k-1}(M')$, $\beta_k(M)$, and $\gamma_i(\boldsymbol{y}_k,M',M)$ employing the quasioptimal log-MAP algorithm [14] that uses the `max` function in conjunction with a table lookup to compute the logarithm of a sum of exponentials. The loss incurred through the use of the log-MAP algorithm is less than 1/10 dB, even when using a lookup table with eight stored values.

$$\alpha_k(M)=\frac{\sum_{M'}\Pr\{\boldsymbol{y}_k,S_k=M|S_{k-1}=M'\}\cdot p\Big(S_{k-1}=M'|\underline{\boldsymbol{y}_1^{k-1}}\Big)}{\sum_{M}\sum_{M'}p(\boldsymbol{y}_k,S_k=M|S_{k-1}=M')\cdot\Pr\Big\{S_{k-1}=M'|\underline{\boldsymbol{y}_1^{k-1}}\Big\}}.\tag{A13}$$

ACKNOWLEDGMENT

The authors would like to thank Dr. J. Hagenauer for valuable discussions.

REFERENCES

[1] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo-codes," in *Proc. ICC'93*, May 1993, pp. 1064–1070.
[2] S. Le Goff, A. Glavieux, and C. Berrou, "Turbo-codes and high spectral efficiency modulation," in *Proc. ICC'94*, May 1994, pp. 645–649.
[3] U. Wachsmann and J. Huber, "Power and bandwidth efficient digital communication using turbo codes in multilevel codes," *European Trans. Telecommun.*, vol. 6, no. 5, 1995.
[4] G. Ungerboeck, "Channel coding with multilevel/phase signals," *IEEE Trans. Inform. Theory*, vol. IT-28, pp. 55–67, Jan. 1982.
[5] P. Robertson and T. Woerz, "Coded modulation scheme employing turbo codes," *Electron. Lett.*, vol. 31, pp. 1546–1547, Aug. 1995.
[6] ———, "A novel bandwidth efficient coding scheme employing turbo codes," in *Proc. ICC'96*, June 1996, pp. 962–967.
[7] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara, "Bandwidth efficient parallel concatenated coding schemes," *Electron. Lett.*, vol. 31, no. 24, pp. 2067–2069, 1995.
[8] ———, "Parallel concatenated trellis coded modulation," in *Proc. ICC'96*, June 1996, pp. 974–978.
[9] L. 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, Mar. 1974.
[10] S. Pietrobon *et al.*, "Trellis-coded multidimensional phase modulation," *IEEE Trans. Inform. Theory*, vol. 36, pp. 63–89, Jan. 1990.
[11] S. Benedetto and G. Montorsi, "Performance evaluation of parallel concatenated codes," in *Proc. ICC'95*, June 1995, pp. 663–667.
[12] W. Blackert and S. Wilson, "Turbo trellis coded modulation," in *Proc. CISS'96*, 1996.
[13] P. Robertson, "Improving the structure of code and decoder for parallel concatenated recursive systematic (turbo) codes," in *Proc. ICUPC '94*, Sept. 1994, pp. 183–187.
[14] P. Robertson, E. Villebrun, and P. Hoeher, "A comparison of optimal and sub-optimal MAP decoding algorithms operating in the log domain," in *Proc. ICC'95*, June 1995, pp. 1009–1013.
[15] J. H. Lodge, R. Young, P. Hoeher, and J. Hagenauer, "Separable MAP 'filters' for the decoding of product and concatenated codes," in *Proc. ICC'93*, May 1993, pp. 1740–1745.

Patrick Robertson (M'97) was born in Edinburgh, Scotland, in 1966. He received the Dipl.-Ing. degree in electrical engineering from the Technical University of Munich in 1989, and the Ph.D. degree from the University of the Federal Armed Forces, Munich, in 1995.

Since 1990, he has been working at the Institute for Communications Technology, German Aerospace Research Establishment (DLR), Oberpfaffenhofen, Germany. In 1993, he spent three months as a Visiting Researcher with the Communications Research Centre, Ottawa. His current research interests include modulation, synchronization, and channel coding applied to radio communications.

Thomas Wörz (M'86) received the Dipl.-Ing. degree in electrical engineering from the Technical University of Stuttgart, Germany, in 1988 and the Ph.D. degree from the Technical University of Munich in 1995.

Since 1988, he has been with the Institute of Communications Technology, German Aerospace Research Establishment (DLR), Oberpfaffenhofen. In 1991, he spent a three-month period as a Guest Scientist at the Communications Research Centre (CRC), Ottawa. His research interests include classical coding, coded modulation, synchronization, and signal processing. Currently, he is involved in several projects considering the signal design for future satellite-based navigation systems.