Titre: Title:	A punctured coding approach to Trellis-Coded Modulation
Auteurs: Authors:	François Chan, & David Haccoun
Date:	1993
Type:	Rapport / Report
Référence: Citation:	Chan, F., & Haccoun, D. (1993). A punctured coding approach to Trellis-Coded Modulation. (Technical Report n° EPM-RT-93-22). https://publications.polymtl.ca/9502/

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/9502/
Version:	Version officielle de l'éditeur / Published version
Conditions d'utilisation: Terms of Use:	Tous droits réservés / All rights reserved

Institution:	École Polytechnique de Montréal
Numéro de rapport: Report number:	EPM-RT-93-22
URL officiel: Official URL:	
Mention légale: Legal notice:	

EPM/RT-93/22

A Punctured Coding Approach to Trellis-Coded Modulation

François Chan, étudiant Ph.D.
Dr. David Haccoun, Professeur Titulaire.

École Polytechnique de Montréal

Septembre 1993

Dépôt légal, 3^{e} trimestre 1993
Bibliothèque nationale du Québec
Bibliothèque nationale du Canada

Pour se procurer une copie de ce document, s'adresser au:

Éditions de l'École Polytechnique de Montréal
École Polytechnique de Montréal
Case Postale 6079, Succursale A
Montréal (Québec) H3C 3A7
(514) 340-4000

Compter 0,10$ par page (arrondir au dollar le plus près) et ajouter 3,00$ (Canada) pour la couverture, les frais de poste et la manutention. Régler en dollars canadiens par chèque ou mandat poste au nom de l'École Polytechnique de Montréal. Nous n'honorerons que les commandes accompagnées d'un paiement, sauf s'il y a eu entente préalable dans le cas d'établissements d'enseignement, de sociétés ou d'organismes canadiens.

A Punctured Coding Approach to Trellis-Coded Modulation*

François Chan and David Haccoun

Département de génie électrique et de génie informatique
Ecole Polytechnique de Montréal
C.P. 6079, succ. A
Montréal H3C 3A7

ABSTRACT

The puncturing technique is widely used for binary convolutional codes to provide high-rate codes from a low-rate code. The technique simplifies decoding and allows easy rate-variable coding and decoding. In this paper, this technique is applied to Trellis-Coded Modulation (TCM). Using extensive computer searches, the best punctured TCM codes have been found. Most of them have the same free Euclidean distance as Ungerboeck's optimum codes. Hence, when decoded in the same usual manner, the error performances of optimum codes and punctured codes are very close. However, using the fact that punctured codes are obtained from a low-rate code, decoding can be simplified at the expense of a small reduction in the coding gain. In addition to the decoding advantages, puncturing yields greater flexibility in rate changes and thus, variable bandwidth efficiency systems can be easily implemented.

* This research has been supported in part by the Natural Sciences and Engineering Research Council of Canada, the Fonds pour la formation des Chercheurs et l'Aide à la Recherche (FCAR) du Québec and by a grant from the Canadian Institute for Telecommunications Research under the National Centers of Excellence program of the Government of Canada. This paper was presented in part at the IEEE International Symposium on Information Theory, San Antonio, Texas, Jan. 17–22, 1993 and at the Canadian Workshop on Information Theory, Rockland, Ontario, May 30–June 2, 1993.

I. INTRODUCTION

Ungerboeck [1]–[3] showed that Trellis-Coded Modulation (TCM) can achieve a significant coding gain in the range of 3 to 6 dB over uncoded modulation without requiring more bandwidth. This is achieved by using a rate R=m/m+1 convolutional code and by introducing the redundancy in the signal set, i.e., by expanding the constellation set. TCM is an effective technique for increasing the bandwidth efficiency of a channel beyond 2 b/s/Hz without excessive signal power. Optimum codes providing the maximum free Euclidean distance have been found for several constellations and number of states [3].

Unfortunately, there are two major drawbacks. First, a different code is required for each signal constellation. Hence, implementing a system with various spectral efficiencies (e.g., 1, 2 or 3 b/s/Hz) would require several encoders and decoders. Second, decoding TCM signals with the Viterbi Algorithm [4] may require a huge complexity if the number of states is large. For a rate R=m/m+1 code, 2^m paths merge into each trellis state. Selecting the survivor, i.e., the most likely path, among the 2^m merging paths requires (2^m—1) binary comparisons per state at each trellis level. Hence, if the number of states is large and if the coding rate is high, Viterbi decoding may become prohibitively complex.

To circumvent these difficulties, Viterbi et al. introduced a so-called pragmatic approach to TCM [5]: a conventional 64–state rate-1/2 binary convolutional encoder is employed for QPSK, while for 8-PSK, one uncoded bit and the two output symbols of the same convolutional encoder are used to select a signal from the 8-PSK constellation. Similarly, for 16-PSK, two uncoded bits and the two coded output symbols are used. The trellises for 8-PSK or 16-PSK have the same connectivity as the basic rate-1/2 trellis with the exception of parallel branches. For 8-PSK, which has one uncoded input bit, each branch of the basic trellis is paired, while for 16-PSK, each branch is replaced by four parallel branches. Hence, a single encoder/decoder can implement various modulation schemes and achieve different spectral efficiencies. This pragmatic approach has, however, a limitation: it uses only a rate-1/2 convolutional code. The best known codes for one-, two-, four-, and eight-dimensional TCM have been found by Ungerboeck [3]. Some of these optimum schemes use codes with rate R=2/3, 3/4 and 4/5 and hence, limiting the pragmatic code to a rate-1/2 will entail sub-optimality. The pragmatic encoder for 16-PSK provides the same performance as the best

Ungerboeck scheme because the code used by Ungerboeck is also a rate-1/2 convolutional code. However, since the optimum 8-PSK code is a rate-2/3 code, the pragmatic 8-PSK code yields an inferior performance with a degradation at $P(B) = 1 \times 10^{-5}$ of 0.4 dB and an asymptotic degradation of 2 dB [5]. Our objective is to obtain the same flexibility (variable spectral efficiency) as Viterbi et al.'s pragmatic approach without being limited to using a rate-1/2 code in conjunction with parallel branches, by considering punctured codes [6].

High-rate binary convolutional codes present the same problems as TCM codes: different optimum codes for each coding rate and exponential increase of the decoding complexity with the coding rate. These problems have been alleviated by the "punctured coding" technique which was first used by Linkabit Corporation in 1972 as reported by Morley [7] but first published by Cain, Clark and Geist [8]. This technique allows the implementation of a high-rate code by periodically puncturing, i.e., deleting, some of the symbols generated by a low-rate encoder. Any rate b/v code can be obtained in this manner [9]–[12]. These codes result in a slight performance loss of only 0.1 to 0.2 dB [8] when compared to the best known conventional high-rate codes. With this technique, a variable-rate code can be derived from a single low-rate code by simply changing the puncturing pattern. This requires only a single encoder and more importantly, a single decoder operating on a low-rate trellis.

In addition, decoding is considerably simplified [8]. It has already been mentioned that decoding a rate m/m+1 code requires (2^{m}—1) binary comparisons per trellis state. On the other hand, with punctured codes, decoding can be realized with m binary comparisons by using the trellis of the underlying low-rate code: m consecutive branches of the low-rate trellis correspond to one branch of the high-rate trellis and two branches merge into each state of the low-rate trellis. Hence, the puncturing technique results in a significant simplification of the decoder. The savings (m instead of 2^{m}—1 comparisons) are particularly important when m is large, i.e, when the coding rate is high and apply to both Viterbi and sequential decoding [10, 13].

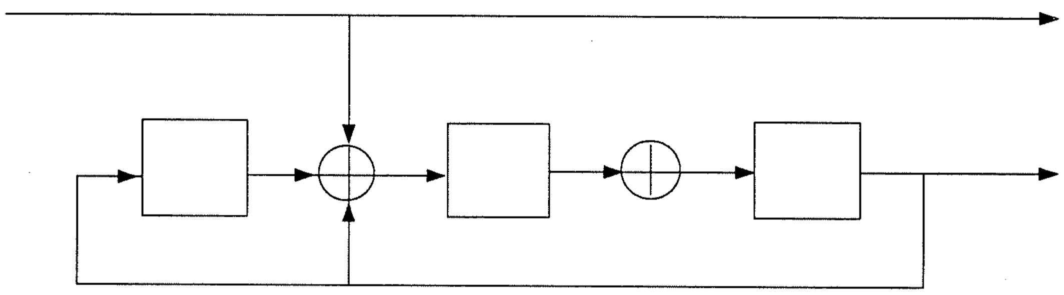
The advantages of punctured coding, that is, simplified decoding and variable rate, make the puncturing technique very attractive in practice and have led to its widespread implementation in commercial convolutional encoders/decoders [14], [15]. In this paper, this technique is applied to Trellis-Coded Modulation and new codes are presented. The paper is organized as follows. In Section II, we describe how Punctured Trellis-Coded Modulation (PTCM) codes are obtained and we list some of the best found codes. Decoding techniques, both optimum and suboptimum, are discussed in Section III. Performance results, obtained from computer simulations are given. In Section IV, we consider families of QPSK, 8-PSK and 16-PSK codes derived from a single rate-1/2 code. Although each PTCM code in the family may not necessarily be optimum for that rate, the families of codes are quite good compared to the best known codes and allow easy implementation of variable bandwidth efficiency systems.

II. PUNCTURED TRELLIS-CODED MODULATION (PTCM) CODES

The objective is to replace the high-rate convolutional code used in Ungerboeck's TCM scheme by a punctured code. This technique is referred to as *Punctured Trellis-Coded Modulation* (PTCM). It is hoped that the punctured code provides identical, or only slightly inferior, error performance as Ungerboeck's optimum code.

As an example, a rate–1/2, 8–state encoder, depicted in Fig. 1, is considered to illustrate this technique. Just like for a rate $R = b/v$ punctured code derived from a low-rate $R = 1/v_0$ code, a puncturing matrix with b columns, v_0 rows and v 1's is applied at the output of the encoder [8]–[12]. The i-th column of the matrix corresponds to the i-th branch of the original low-rate trellis and a 1 (or a 0) in the i-th column means that the corresponding code symbol of the i-th branch is kept (or deleted, respectively). Here, the puncturing matrix $\mathbf{P} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$ is used, as illustrated in Fig. 2. Hence, our matrix yields a rate-2/3 code and its trellis, which is obtained by combining two levels of the previous trellis is represented in Fig. 3. The code symbols are mapped onto the 8-PSK signal constellation of Fig. 4, which is the same as in Ungerboeck's scheme [1]. The shortest error-event path is shown in Fig. 3 and corresponds to the path with minimum free distance

$$d^2_{free} = 2.0 + 0.58 + 2.0 = 4.58$$



Parity Check Coefficients: 04, 15

Fig. 1. Rate-1/2, 8–state encoder

A. Search for good codes

In the above example, a feedback code has been selected as the original code and punctured to yield a rate R=2/3 code. Usually, punctured binary codes are obtained from feedforward codes [8]–[10]. However, it has been noticed that for 8-PSK, a feedback code provides a better bit error performance than a feedforward code with the same Euclidean distance, even though their error-event performances may be almost the same. As an example, the simulated bit and sequence error performances of the best 64–state feedforward code obtained from our computer search is compared to a feedback code in Fig. 5. The superiority of the feedback code is especially important at low signal-to-noise ratios. These results have been corroborated independently by others [16, p. 251], [17]. Furthermore, using a systematic code, such as our systematic feedback codes, guarantees that it is not catastrophic [18]. Consequently, in the remaining of our search, feedback codes are considered. As for their decoding, the procedure will be described in detail in the next section.

For binary convolutional codes, the best high-rate punctured codes are found by choosing as original codes the known optimum free distance (OFD) low-rate

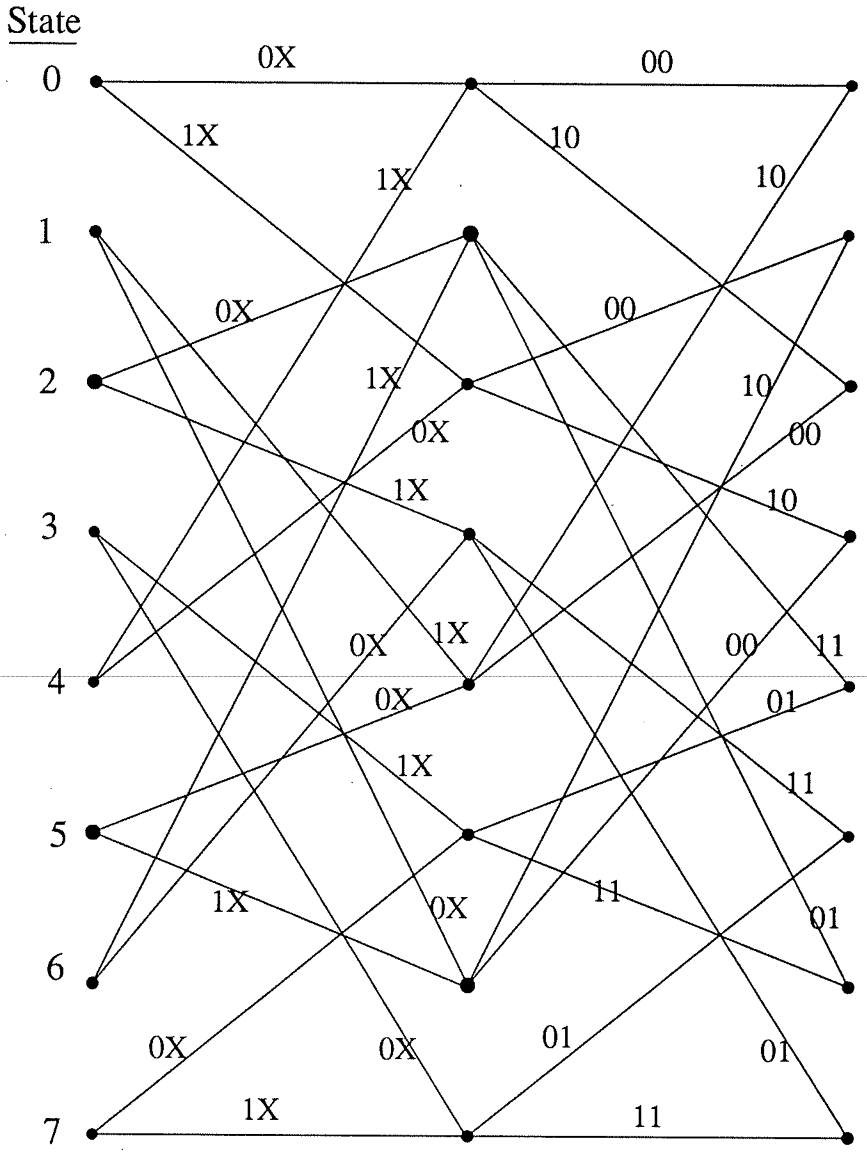


Fig. 2. Trellis for R=1/2, 8–state code

codes and then determining the puncturing pattern which will yield the maximum free distance. This approach is based on the often used intuition that "good codes generate good codes" [10]. For TCM, it will be shown in the next section that

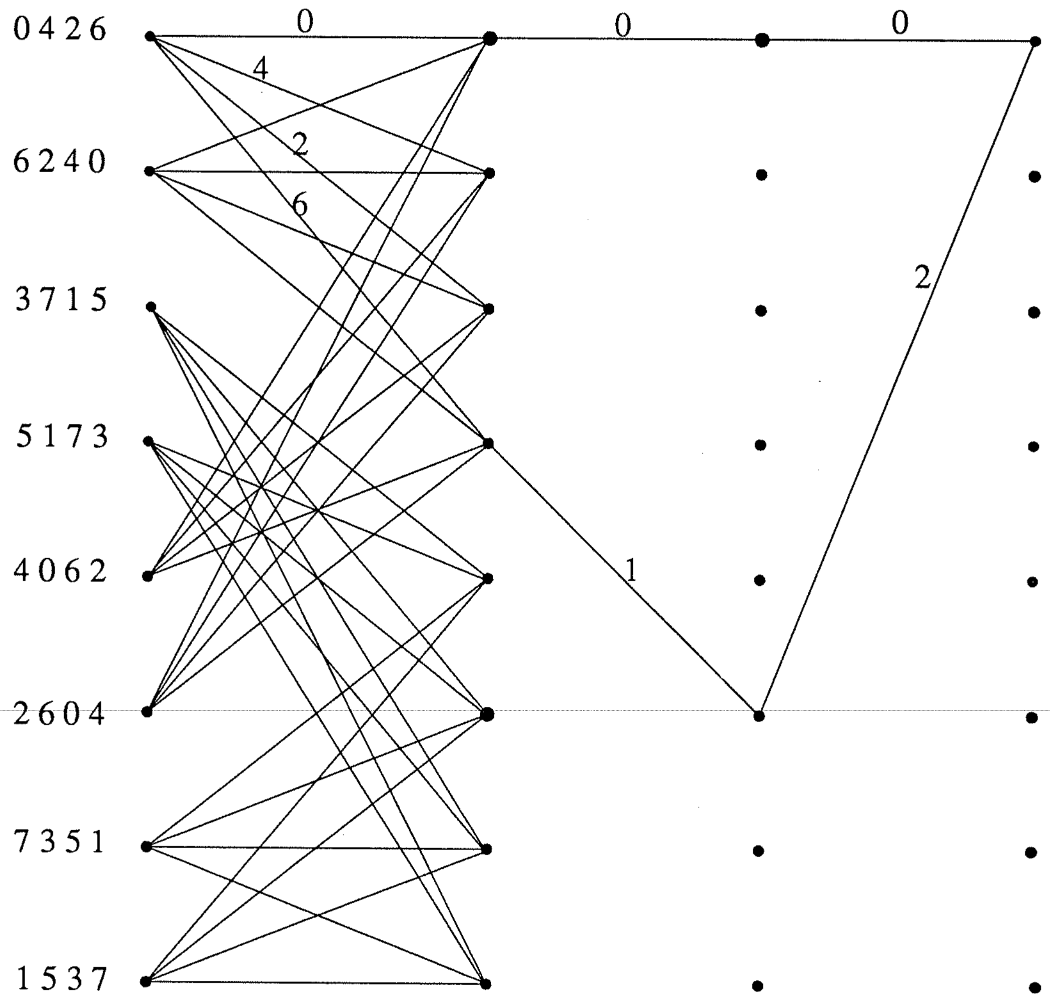


Fig. 3. Trellis for punctured R=2/3, 8–state code

the error performance depends on the free Euclidian distance[1] and therefore, it is the appropriate criterion for selecting codes. However, we cannot simply use the known binary convolutional codes with maximum free Hamming distance and Gray coding as a mapping function. As stated by Ungerboeck, the code with the largest Euclidean distance is not necessarily the code with the largest Hamming distance [1]. Therefore, a computer search must be performed to find the original codes which, when punctured, will provide the maximum free Euclidean distance.

[1] Ungerboeck [1] has defined the free Euclidean distance to be "the minimum distance between all pairs of channel-signal sequences which the encoder can produce".

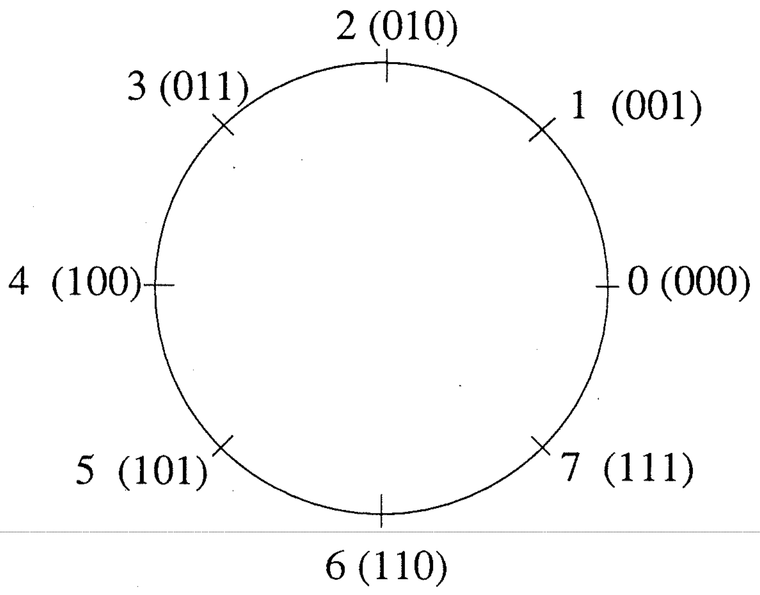


Fig. 4. 8-PSK constellation

We have selected the same mapping of coded symbols into channel signals as Ungerboeck and the goal is to get codes with the same Euclidean distance as Ungerboeck's optimum codes.

Ungerboeck [1] has used the following heuristic rules which are conjectured to provide the best TCM schemes:

1. Parallel transitions (when they occur) receive signals with the largest Euclidean distance.
2. Transitions originating from or merging into the same state receive signals with the second largest distance.
3. All the signals should appear with equal frequency.

The puncturing pattern that can provide the maximum distance is determined. From the trellis of a rate-1/2 code (see e.g. Fig. 2) and our mapping of Fig. 4, we can deduce that in order to satisfy rule 2, the last column of the perforation

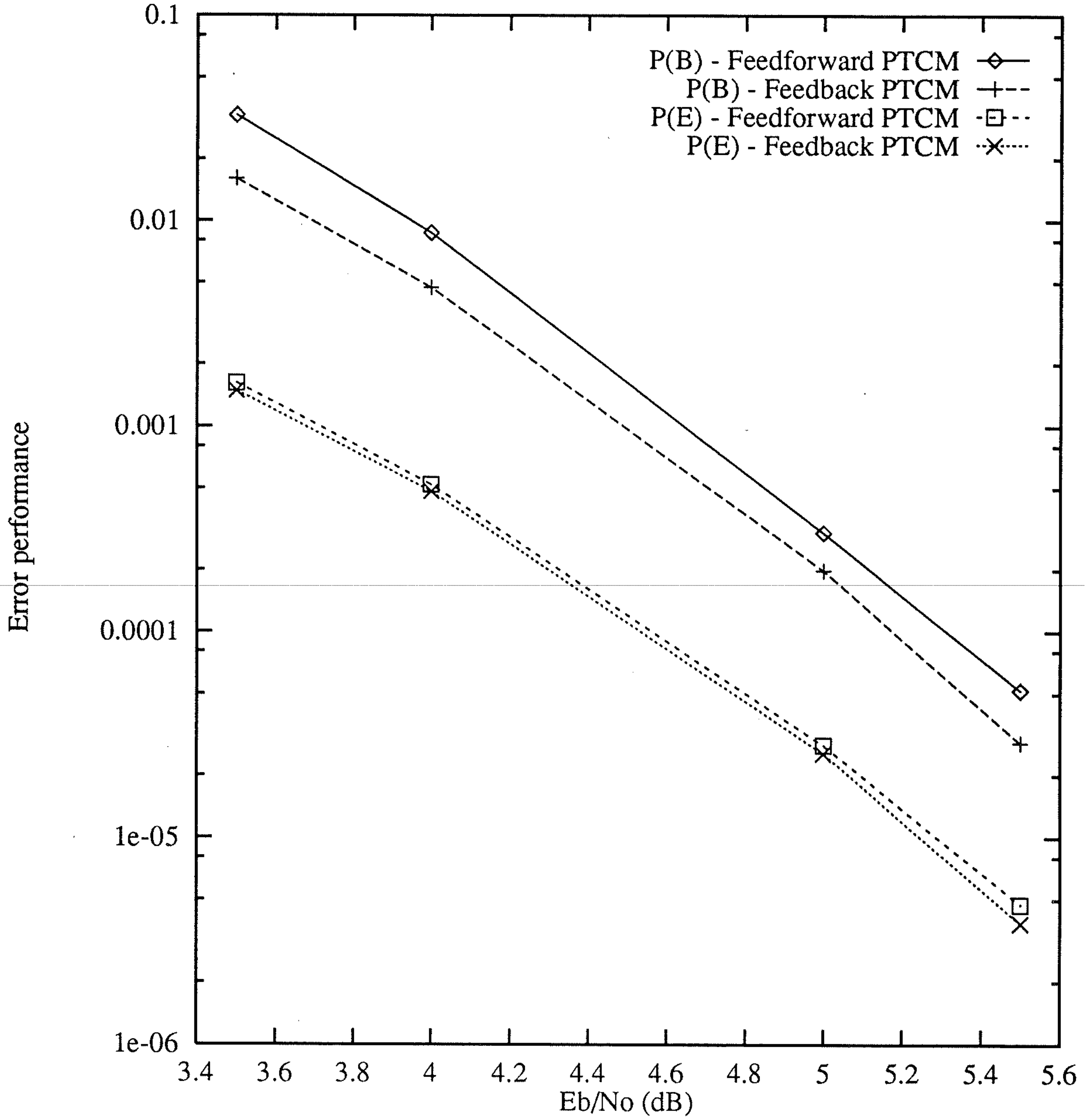


Fig. 5. Probability of bit error and probability of error-event of a feedforward and a feedback 64–state PTCM code — 8-PSK

Number of states	$\tilde{m}$	Punctured		Ungerboeck (1987)	
		Parity-check coefficients h^1, h^0	d^2_{free}	Parity-check coefficients h^2, h^1, h^0	d^2_{free}
8	2	04, 15	4.58	04, 02, 11	4.58
16	2	04, 23	5.17	16, 04, 23	5.17
32	2	34, 47	5.17	34, 16, 45	5.75
64	2	074, 163	6.34	066, 030,103	6.34
128	2	174, 311	6.34	122, 054, 277	6.58
256	2	240, 705	7.17	130, 072, 435	7.51
512	2	0024, 1225	7.51*	0260, 0164, 1007	7.51 (1982)

* Code search not complete

Table 1 Punctured and Ungerboeck TCM codes with largest d_{free} for 8-PSK

matrix should contain only 1's, i.e., the last branch symbols are not punctured. Otherwise, in our example, transitions originating from or merging into any state do not receive signals from one of the two QPSK subsets, {0,2,4,6} and {1,3,5,7}, but from a combination of both and hence, the distance between signals is reduced. Therefore, for a rate-2/3, 8-PSK PTCM code and the mapping of Fig. 4, the perforation matrix must be

$$\mathbf{P} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$$

The best 8-PSK PTCM codes obtained from a computer search are listed in Table 1, where $\tilde{m}$ represents the number of information bits that are actually used by the convolutional encoder and where the parity-check coefficients h_2, h_1, h_0 are expressed in octal notation. When several codes have the same free Euclidean distance, we select the code yielding the lowest error probability as obtained from computer simulations. When the number of codes with the same distance is very large (10 or more codes), we select one of them at random. If the distance spectrum of the codes is known, we could also make the selection based on the spectrum. Compared to Ungerboeck codes, PTCM codes offer the same, or only slightly smaller, free Euclidean distance.

The optimum codes for 8-PSK with 8 or more states [3] are "true" rate-2/3 codes, i.e., there are no uncoded bits and therefore, no parallel transitions in the trellis. With parallel transitions, the maximum achievable free Euclidean distance is limited by the distance between signals assigned to parallel transitions since the free squared Euclidean distance of a code is given by

$$d^2_{free} = Min\left[d^2_{par}, d^2_{mul}\right]$$

where d^2_{par} is the distance between parallel branches and d^2_{mul} is the minimum squared Euclidean distance between multi-branch paths. However, parallel transitions reduce the connectivity in the trellis and hence, multi-branch error-events are longer. For some codes, especially short-memory codes, d^2_{par} is larger than d^2_{mul} and hence, it is preferable to have parallel transitions, which increase the minimum length and the free distance of multi-branch error-events. As a result, a larger free distance is achieved with parallel transitions for, e.g., 4–state 8-PSK, 16-PSK with up to 128 states, 16-QAM. Therefore, to achieve the same distance as the optimum codes, the puncturing technique should be able to implement codes with parallel transitions. This can be easily done by using a systematic rate-1/2 encoder and activating the clock of the shift-register only $\tilde{m}$ times every m cycles (or branches) if an encoder with a rate $R = m/m+1$ and $(m - \tilde{m})$ uncoded bits is to be implemented. The puncturing pattern must, of course, be of the form

$$\begin{bmatrix} \underbrace{\begin{matrix} 1 & \cdots & 1 \\ 0 & \cdots & 0 \end{matrix}}_{(m-\tilde{m})\text{times}} & \begin{matrix} 1 \\ 1 \end{matrix} \end{bmatrix}$$

As an example, a 4–state punctured encoder for 16-PSK which is equivalent to an encoder with two uncoded bits is illustrated in Fig. 6. The trellis of the punctured code is depicted in Fig. 7(b). It can be verified by combining the three branches that this trellis is equivalent to the trellis of the Ungerboeck code depicted in Fig. 7(a), both have exactly the same transitions. Therefore, PTCM codes are quite general: they can include conventional high-rate codes or codes with parallel transitions.

Results of computer search for the largest Euclidean distance 16-PSK and 16-QAM PTCM codes are given in Tables 2 and 3. The best PTCM codes achieve the same free Euclidean distances as Ungerboeck's best known codes. In both cases, using the same reasoning as with the 8-PSK code given above, the perforation is

$$P = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

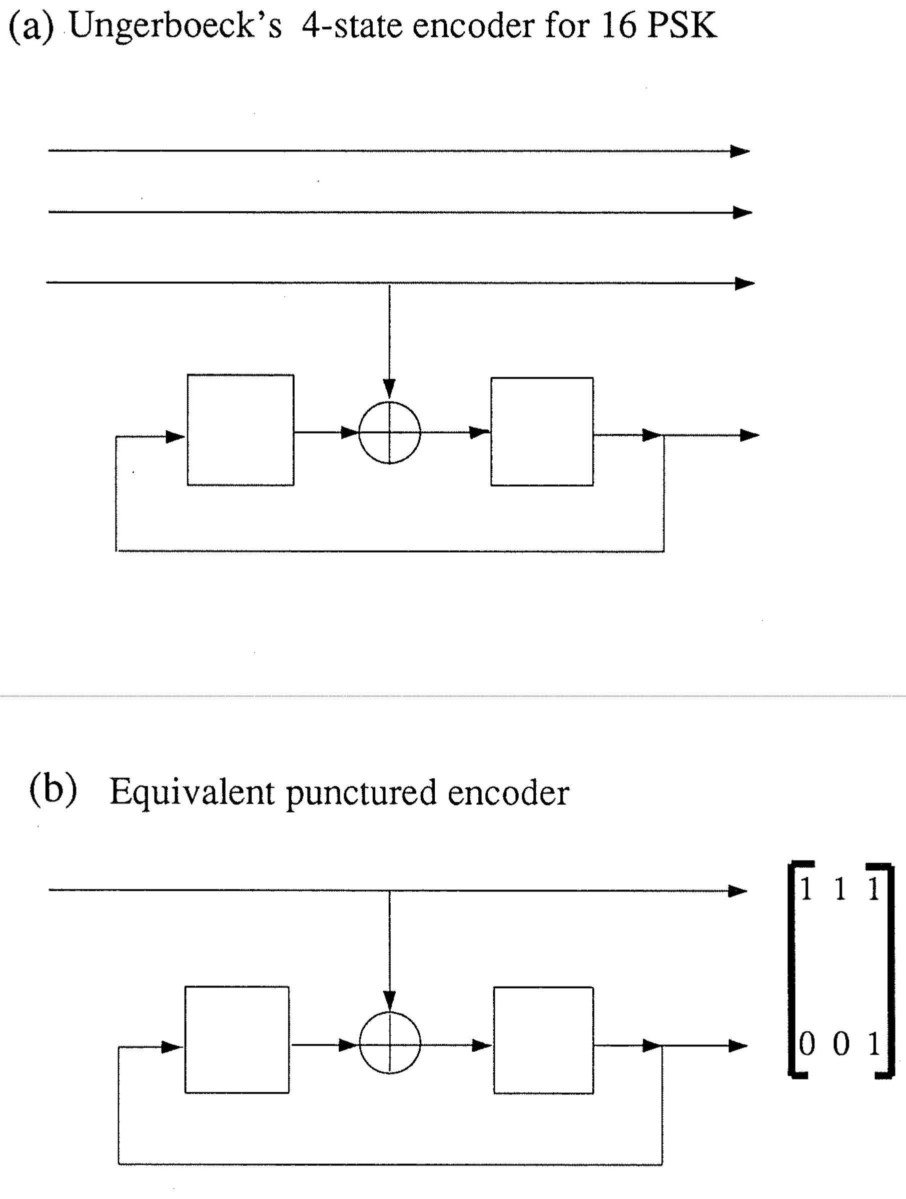
The rate of 16-PSK or 16-QAM codes should be R=3/4, but the true rate is $R = \tilde{m}/\tilde{m}+1$, where $\tilde{m}$, the number of bits actually encoded in the convolutional encoder, is given in the tables. This means that there are $(3 - \tilde{m})$ uncoded bits and $2^{(3-\tilde{m})}$ parallel branches in the trellis. The shift-register clock is activated $\tilde{m}$ times every m cycles (or branches) and these $\tilde{m}$ times occur at the end of the m cycles.

B. Extensions and discussion

We have found quasi-optimum PTCM codes for 8-PSK, 16-PSK and 16-QAM. The search could be extended to larger constellations (e.g., 32-QAM, 64-QAM, etc.), multidimensional constellations and coset codes. Codes with a larger number of states may also be investigated. However, this is an extremely time-consuming process. Furthermore, we have only examined the free Euclidean distance of our codes. Although it is a good criterion for code selection, as shown in the next section, it may be useful to compute the distance spectrum of our PTCM codes.

Most of the codes presented above have the same free Euclidean distance as the best known codes discovered by Ungerboeck [3]. Yet, some of them have a slightly smaller Euclidean distance. We believe that it may be possible to find PTCM codes with the same distance as the optimum codes. One possible approach would be to start with an original code of rate lower than R=1/2. A similar approach for punctured binary convolutional codes provided codes whose free distance is larger than the distance of codes obtained using R=1/2 original codes [19]. Another approach, which has been used successfully for binary convolutional codes [11], consists in constructing the low-rate original codes which, when punctured, duplicate given known high-rate codes. However, these codes are no longer rate-compatible.

For the 8-PSK code with 32 states, a larger distance $d^2_{free} = 5.75$, which is the distance of the optimum Ungerboeck code, can be obtained if we allow the trellis

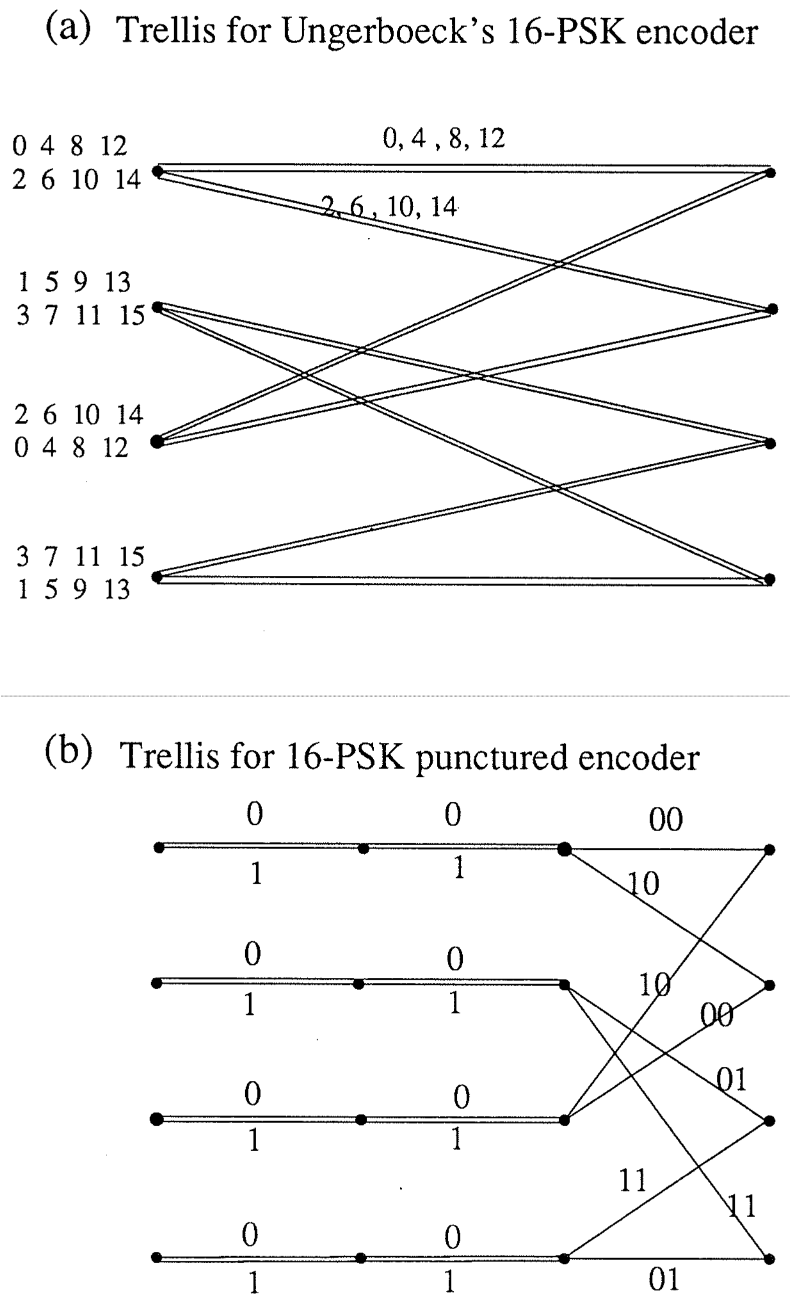


(a) Ungerboeck's 4-state encoder for 16 PSK

(b) Equivalent punctured encoder

Shift-register activated only once every 3 cycles (or branches)

Fig. 6. 16-PSK encoder with parallel transitions. (a) Ungerboeck's encoder. (b) Equivalent punctured encoder



(a) Trellis for Ungerboeck's 16-PSK encoder

(b) Trellis for 16-PSK punctured encoder

Fig. 7. Trellis for 16-PSK codes (a) Ungerboeck code. (b) Punctured code

Number of states	$\tilde{m}$	Punctured		Ungerboeck (1987)	
		Parity-check coefficients h^1,h^0	d^2_{free}	Parity-check coefficients h^1,h^0	d^2_{free}
4	1	2, 5	1.324	2, 5	1.324
8	1	04, 13	1.476	04, 13	1.476
16	1	04, 23	1.628	04, 23	1.628
32	1	10, 45	1.910	10, 45	1.910
64	1	024, 103	2.000	024, 103	2.000*
128	1	024, 203	2.000	024, 203	2.000*

* Code search not complete

Table 2 Punctured and Ungerboeck TCM codes with largest d_{free} for 16-PSK

Number of states	$\tilde{m}$	Punctured		Ungerboeck (1987)	
		Parity-check coefficients h^1, h^0	d^2_{free}	Parity-check coefficients h^2, h^1, h^0	d^2_{free}
4	1	2, 5	16.0	- , 2, 5	16.0*
8	2	04, 11	20.0	04, 02, 11	20.0
16	2	10, 27	24.0	16, 04, 23	24.0
32	2	4, 45	24.0	10, 06, 41	24.0
64	2	54, 117	28.0	064, 016, 101	28.0
128	2	74, 211	32.0	042, 014, 203	32.0
256	2	74, 575	32.0*	304, 056, 401	32.0
512	2	200, 1761	32.0*	0510, 0346, 1001	32.0*

* Code search not complete

Table 3 Punctured and Ungerboeck TCM codes with largest d_{free} for 16-QAM

connectivity to be different for punctured and non-punctured branches. This is realized by feeding back the second output symbol when it is not punctured and feeding back a 0 when it is punctured. This makes the decoder more complex since it must now operate on two trellis connectivities.

Recently, How [20] has independently designed practical TCM schemes for 16-QAM using punctured convolutional codes. However, the error performance of his 64–state, rate-2/3 punctured code is inferior to the performance of the 32–state Ungerboeck code. Furthermore, decoding the punctured 64–state code requires $2 \times 64 = 128$ binary comparisons at each trellis level while Ungerboeck code requires $(2^2 - 1) \times 32 = 96$ comparisons since, as already mentioned, it takes (2^m—1) comparisons per state to decode a rate R=m/m+1 code. Therefore, there is no advantage in using How's codes. We now consider the decoding problem of PTCM and show that our codes are amenable to attractive suboptimum decoding technique with a degradation smaller than with How's codes.

III. DECODING OF PTCM CODES

In Section II, it has been mentioned that some PTCM codes have the same free Euclidean distance as Ungerboeck codes. Therefore, using a bound on the error-event probability P(E), they should provide a similar error performance asymptotically, i.e., at high signal-to-noise ratios. In this section, we verify that even at moderate signal-to-noise ratios, for a bit error probability in the range of 10^{-5} to 10^{-3}, performances are also similar. Only Viterbi decoding [4], [21], which is a widely used decoding algorithm for TCM and convolutional codes, is considered in this paper. The error performances of PTCM codes over an Additive White Gaussian Noise (AWGN) channel, as obtained from computer simulations, are presented and compared to the performance of Ungerboeck codes. A punctured code can be considered as a true high-rate code and optimally decoded using its high-rate trellis. For the 8–state, 8-PSK code in the example of section II, this corresponds to the trellis of Fig. 3. This is referred to as "usual decoding" since this is the same type of decoding that is usually performed for TCM codes and it is optimum. On the other hand, a punctured code can also be viewed as a low-rate code with symbols periodically deleted, as illustrated in Fig. 2.

Using this low-rate trellis simplifies the decoding but unlike binary convolutional codes, for TCM codes, it results in some ambiguity which may be resolved by suboptimum decoding as shown next.

A. Optimum decoding (usual)

Considering the punctured TCM code as a true high-rate code with a rate $R = m/(m+1)$, we compare the performance of the PTCM code and of the Ungerboeck code. The Viterbi algorithm is an optimum decoding technique for convolutional codes [21] which can be used for TCM as well [4], [22]. Let **r** be the received vector, x_0 and x_1, two equally likely codewords. Following Clark & Cain [22], the optimum decision rule over an AWGN channel is to select codeword x_0 whenever

$$\|\mathrm{r} - \mathrm{x}_0\|^2 < \|\mathrm{r} - \mathrm{x}_1\|^2 \tag{6}$$

where $\|.\|^2$ denotes the squared Euclidean distance. Let the received symbol at trellis level l be r_l and let the coded symbol of a branch, i.e., the signal, be a^i, $i = 0, N-1$, where N is the cardinality of the signal set. Then, the branch metric $m(r_l, a^i)$ is

$$d^2(r_l, a^i) = \|r_l - a^i\|^2 \tag{7}$$

For each trellis state a Viterbi decoder adds this metric to the accumulated metric of the path leading to the state and selects the most likely path among the 2^m paths merging at the state. The decoding complexity is the same for both TCM and PTCM codes, that is, $2^m - 1$ binary comparisons per state are required at each trellis level.

A lower bound on the error-event probability of a TCM code is given by [22]

$$P(E) \geq Q(d_{free}/2\sigma) \tag{8}$$

where d_{free} is the free Euclidean distance and σ^2 is the Gaussian noise variance. The error performance approaches this bound asymptotically at high signal-to-noise ratios. Since the PTCM code is considered as a true high-rate code and decoding proceeds in the same fashion as an Ungerboeck code, bound (8) on the error-event probability is also valid for PTCM: the decoder simply views the

PTCM code as a code equivalent to Ungerboeck code. The bit error probability obtained by computer simulations of 8-PSK using Ungerboeck and our punctured codes with 64 and 256 states is illustrated in Fig. 8. With 64 states, both codes have the same free Euclidean distance as indicated in the previous section and it can be seen that their error performances are also very close. With the 256–state codes, there is a slight difference because the free Euclidean distance of the PTCM code is smaller than the distance of Ungerboeck code. Therefore, the free Euclidean distance not only defines the asymptotic coding gain, but is also an appropriate criterion for our code selection, even for a bit error probability in the range of $[10^{-3} - 10^{-5}]$. Similarly, for 16-PSK and 16-QAM, Fig. 9 shows that the error performance of PTCM codes is very close to that of Ungerboeck codes when decoded in the same manner since the free distances are identical.

Optimum decoding of rate R=m/m+1 TCM or PTCM codes requires $2^m - 1$ binary comparisons per state at each trellis level. Clearly, if the code has a large number of states and if m is large, then decoding may become prohibitively complex. We now present good suboptimum decoding techniques which circumvent this difficulty.

B. Simplified suboptimum decoding

Approximate decoding

When a PTCM code is considered as a true high-rate code, its performance is similar to that of Ungerboeck code with the same free distance but decoding is also similar and no simplification is obtained. Since simplifying the decoding is one of our objectives in developing PTCM codes, following the punctured binary convolutional codes, a PTCM code is now viewed as a low-rate code with symbols periodically deleted. It should be noted that at the transmitter side, the PTCM code is always viewed as a true high-rate code, that is, a channel signal is a mapping of the symbols on two or more branches of the low-rate code. Fig. 10 depicts a small part of the trellis for the 8-PSK punctured code of Fig. 2 and illustrates the problem: at intermediate node[2] c, metrics of the converging branches, i.e., branches at the first level of the trellis, are unknown in PTCM.

[2] When decoding a rate R=b/v punctured code using the trellis of the low-rate original code, a state at level b (and any multiple of b), corresponding to b (or multiple of b) information bits is denoted a "true" state since it corresponds to a state in the trellis of a conventional high-rate code. A state at any other trellis level is denoted an "intermediate" state.

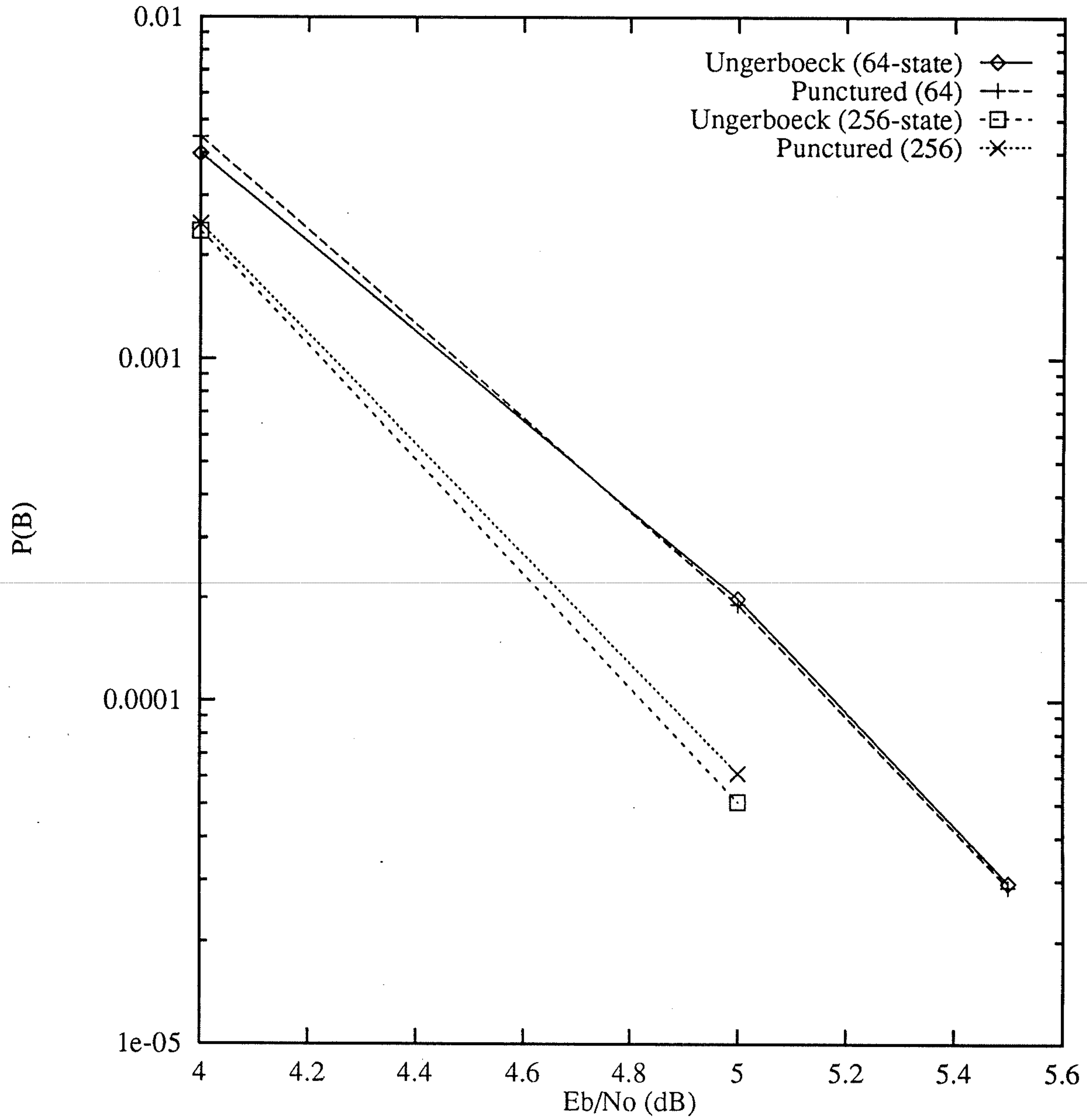


Fig. 8. Bit error performance of Ungerboeck and PTCM codes using same decoding method — 8-PSK

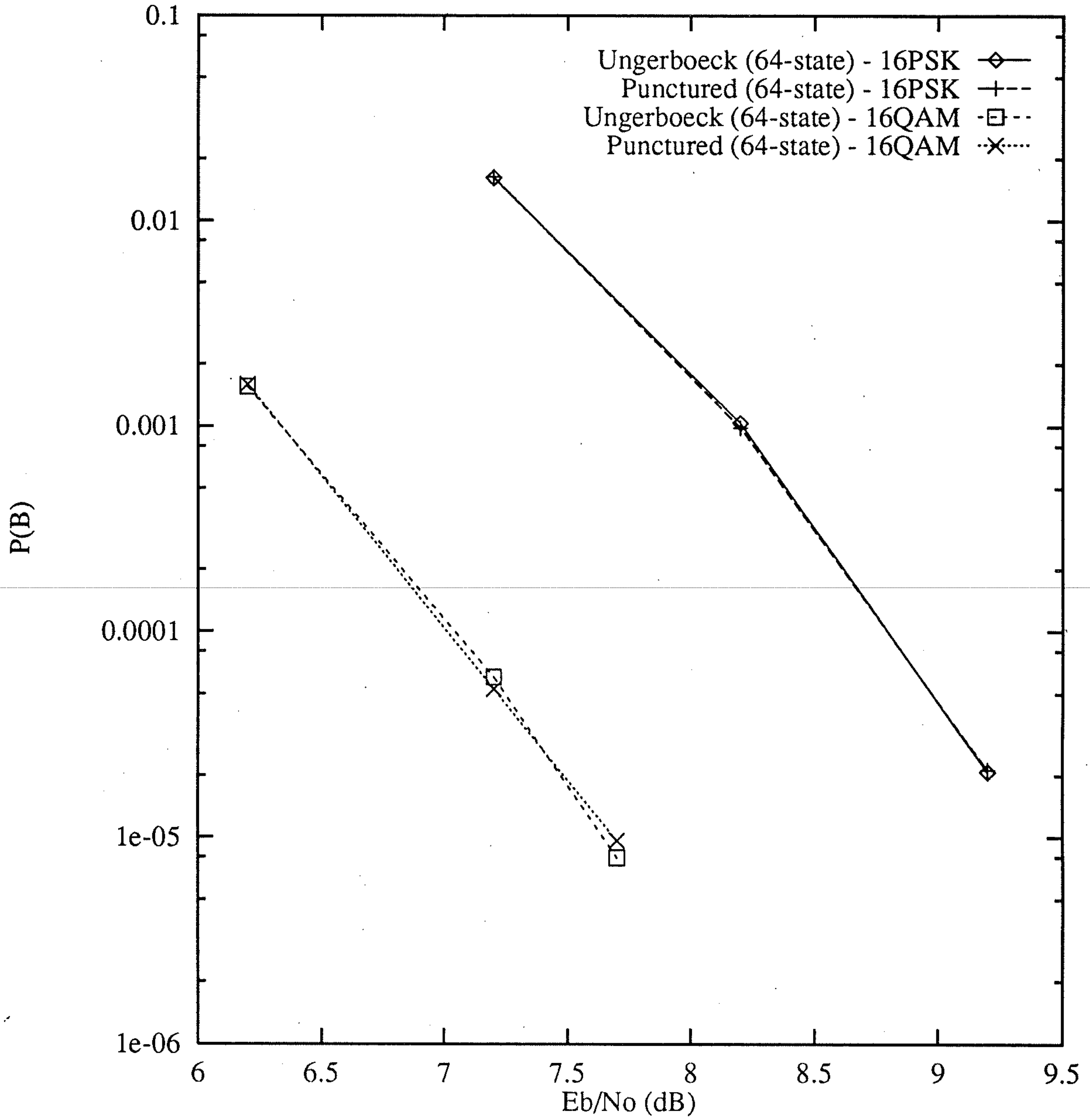


Fig. 9. Bit error performance of Ungerboeck and PTCM codes using same decoding method — 16-PSK and 16-QAM

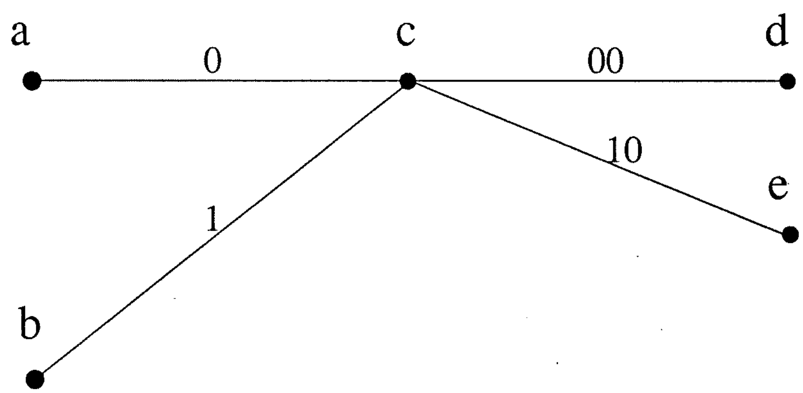


Fig. 10. Part of the trellis for 8-PSK punctured code

Before presenting the problem in detail, we first briefly review how decoding is performed for punctured binary convolutional codes.

For binary convolutional codes a transmitted signal corresponds to one trellis branch only. Hence, considering Fig. 10 as the trellis of a convolutional code, the metric of 0 on the ac branch (or 1 on the bc branch) is known: it is just the Hamming distance between 0 (or 1) and the received symbol. Decoding is simplified because at each state, whether it is an "intermediate" state or a "true" state, a decision about the survivor can be made.

For the PTCM code of Fig. 10, a channel signal is defined as the symbol on a branch at the first level in combination with the symbols on the following second level branch. In our example, 0 is the first symbol of the channel signal 000 or 010. As indicated previously, while the metric $m(r_l, 000)$ of 000 can be computed as $d^2(r_l, 000)$, where r_l is the received channel signal, the metric of symbol 0 on the first level branch, is not known. However, in order to simplify decoding, a final decision must be made at the intermediate node c. Therefore, the decision rule at state c is to select branch 0 if

$$M(a) + m(r_l, 0) < M(b) + m(r_l, 1) \tag{9}$$

where $M(i)$ is the accumulated metric at state i and where $m(r_l, s)$ is the metric of the branch with symbol s to be defined later. Without loss of generality let s

be 0. Then, $m(r_l, 0)$ can assume two values: $m(r_l, 000)$ or $m(r_l, 010)$ since, as already mentioned 0 is the first symbol of signals 000 or 010 (see Fig. 10). A solution is to use as $m(r_l, 0)$ an approximate metric and to base the decision at the intermediate node on that metric. This is accomplished as follows.

First, note that only four distinct signals can pass through an intermediate state. In our example, the four signals that can pass through state c are 0, 2, 4 and 6 (in octal representation). Signals 0 and 2 are grouped together since they both come from state a and have the same leading symbol 0 (branch from state a to state c). Similarly, signals 4 and 6 are grouped together. Hence, at state c, the choice between branch 0 and branch 1 is, in fact, a choice between group(0,2) and group(4,6). The metric of branch 0, i.e., of group(0,2) is defined as the Euclidean distance between the received signal r_l and a point equidistant from signals 0 and 2. (It is assumed, of course, that all input signals are equiprobable.) As illustrated in Fig. 11, we have selected signal 1 as this equidistant point. Similarly, for the group(4,6), point 5 is chosen. Thus, at node c, the metric of branch 0 is

$$m(r_l, 0) = d^2(r_l, 001) \tag{10}$$

and the metric of branch 1 is

$$m(r_l, 1) = d^2(r_l, 101) \tag{11}$$

It should be noted that the chosen point does not have to be an actual signal as long as it is equidistant from the two signals of interest and it has been verified through computer simulations that choosing another equidistant point makes almost no difference in the error performance of the decoder. The groups and the equidistant points for the four other signals (1, 3, 5 and 7) passing through another intermediate state are also shown in Fig. 11.

An optimum decoder uses two metrics for $m(r_l, 0)$, $m(r_l, 000)$ and $m(r_l, 010)$, since in a punctured high-rate trellis obtained by combining two levels of the low-rate trellis (see Figs. 2 and 3), branch 0 at the first level of the low-rate trellis corresponds to two branches of the high-rate trellis: branch 000 and branch 010. Therefore, $m(r_l, 0)$ is just an approximation to the two exact metrics and is used to simplify decoding by allowing a final decision at the intermediate state c. At the next trellis level, using as leading symbol the surviving branch at the intermediate state, the exact metrics can be computed.

The above procedure described for rate-2/3 codes can be generalized to any rate R=m/m+1. If the length, or period, of the puncturing pattern is m, i.e, if the rate of the code is R=m/(m+1) and if the cardinality of the signal set is N (N=2^{m+1}), then N/2 distinct signals can pass through a state at the first level, N/4 at the second level, ... , and N/2^{m-1} at level (m-1). The signal set is first partitioned into two subsets with N/2 signals each according to the least significant bit of the signal. Only one of these subsets passes through an intermediate state at the first trellis level. This subset is then partitioned into two subsets: one of these subsets or groups is associated with branch 0 and contains all N/4 signals starting with symbol 0 as most significant bit (MSB). This corresponds to trellis level 1 and the subsets are denoted level–1 subsets. This subset is again partitioned into two groups with N/8 signals according to the second MSB (level–2 subset), and so on, until the last subset contains only one signal. It corresponds to a branch converging to a true state at level m. It can be seen that after each trellis level, the number of surviving signals is halved since we keep only one of the groups (or subsets) merging at the state. Fig. 12 illustrates the partitioning for a 8-PSK signal set.

In summary, the approximate decoding technique consists in merging constellation signals that have the same leading symbol in a group and use as approximate branch metric the Euclidean distance between the received signal and a point equidistant from the signals in the group. This technique is clearly suboptimum since it does not use the exact metric but an approximation which reduces the metric separation, hence leading to an error performance degradation. Suppose that a sequence of all-zeros is transmitted and that the received signal r_l is indicated by a X in Fig. 11. Let

$$d^2(r_l, 100) - d^2(r_l, 000) = \Delta \tag{12}$$

On the other hand, with approximate decoding, we have

$$d^2(r_l, 101) - d^2(r_l, 001) = \delta \tag{13}$$

It can be easily verified from Fig. 11 that $\delta < \Delta$, i.e., the branch metric separation has been reduced. The optimum decoder compares $(M(a) + m(r_l, 000))$ with $(M(b) + m(r_l, 100))$. As long as

$$M(a) + m(r_l, 000) < M(b) + m(r_l, 100)$$

or

$$M(a) - M(b) < \Delta$$

the optimum decoder selects 000 as the most likely codeword. On the other hand, the approximate decoder compares $M(a) + m(0) = M(a) + m(r_l, 001)$ with $M(b) + m(1) = M(b) + m(r_l, 101)$. If

$$M(a) + m(r_l, 001) < M(b) + m(r_l, 101)$$

or

$$M(a) - M(b) < m(r_l, 101) - m(r_l, 001) = \delta$$

the approximate decoder selects group(0,2) and hence, makes the same decision as the optimum decoder. However, if $\delta < M(a) - M(b) < \Delta$, the approximate decoder selects group(4,6). This decision results in an error, whereas there was no error with the optimum decoder. Therefore, because of the reduced metric separation, the approximate decoder is more sensitive to the accumulated metrics accuracy than the optimum decoder. Instead of the accumulated metrics, which depend on the signals received at the previous trellis levels, the approximate decoder may be regarded as being less tolerant than the optimum decoder to noise in the current received signal.

It should be noted that if the decision were based only on the received signal r_l at trellis level l, then the decision regions depicted in Fig. 11 would be optimum, i.e., the approximate decoder would select the same group as an optimum decoder. However, the Viterbi decoder is a sequence decoder and therefore, the decision does not only depend on r_l but also on the entire sequence of previously received signals.

In our approximate decoder for a rate-2/3, 8-PSK PTCM code, the first error-event may be caused by a decision at a state of the first trellis level (intermediate state) or at a state of the second level (true state), given that the decision at the first level is correct. Let $P_1(E)$ be an upper bound on the first error-event probability for an intermediate state, and let $P_2(E)$ be an upper bound on the first error-event probability for a true state, assuming the decision at the intermediate state is correct. Then a bound on the first error-event probability $P(E)$ is given by

$$P(E) \leq P_1(E) + P_2(E) \tag{18}$$

Since $P_2(E)$ is the probability of error at a true state, given that there is no error at the previous intermediate state, $P_2(E)$ is the probability of error of an optimum decoder and is denoted $P(E)_{opt}$. A bound on $P(E)_{opt}$ has been given in (8). $P_1(E)$ is the degradation caused by the suboptimum decoding technique. Hence, $P(E)_{opt}$ is due to the code, while $P_1(E)$ is due to the suboptimum decoding method. A similar result has been obtained for the error probability of suboptimum decoding algorithms for binary convolutional codes [23]. The difference, however, is that here, the decoding algorithm itself is optimum, but the metrics are suboptimum.

Bit error performance obtained by simulation of PTCM 8-PSK codes with approximate decoding and with optimum decoding is shown in Fig. 13. It can be seen that approximate decoding leads to a slight degradation of approximately 0.15 dB at $P(B) = 1 \times 10^{-4}$ for a 64–state code. However, its complexity is reduced: it requires 2 comparisons per state, while optimum decoding necessitates 3 comparisons. In addition, the performance of this PTCM code is still superior to that of a 32–state Ungerboeck code. For a complexity lying between those of the 32–state and 64–state Ungerboeck codes, approximate decoding also yields a performance between both of them. Hence, PTCM codes allow a complexity/performance trade-off. The complexity reductions provided by the puncturing technique increase significantly with the coding rate R=m/m+1 since decoding the low-rate trellis requires only m comparisons per state instead of 2^m–1 for the high-rate trellis.

Staged decoding

Our approximate decoding technique for PTCM codes described above provides good performance for 8–PSK, allowing a reduction of the decoding complexity at a cost of a very small performance degradation. With rate–2/3, 16-QAM PTCM codes, however, this technique does not provide satisfactory results. For example, the error performance of the 64–state code with approximate decoding, shown in Fig. 14, is inferior to the performance of the 32–state Ungerboeck code with optimum decoding, while requiring a larger decoding complexity. Hence, the approximate metrics lead to an unacceptable degradation. We now present another decoding technique, called staged decoding, which circumvents this difficulty.

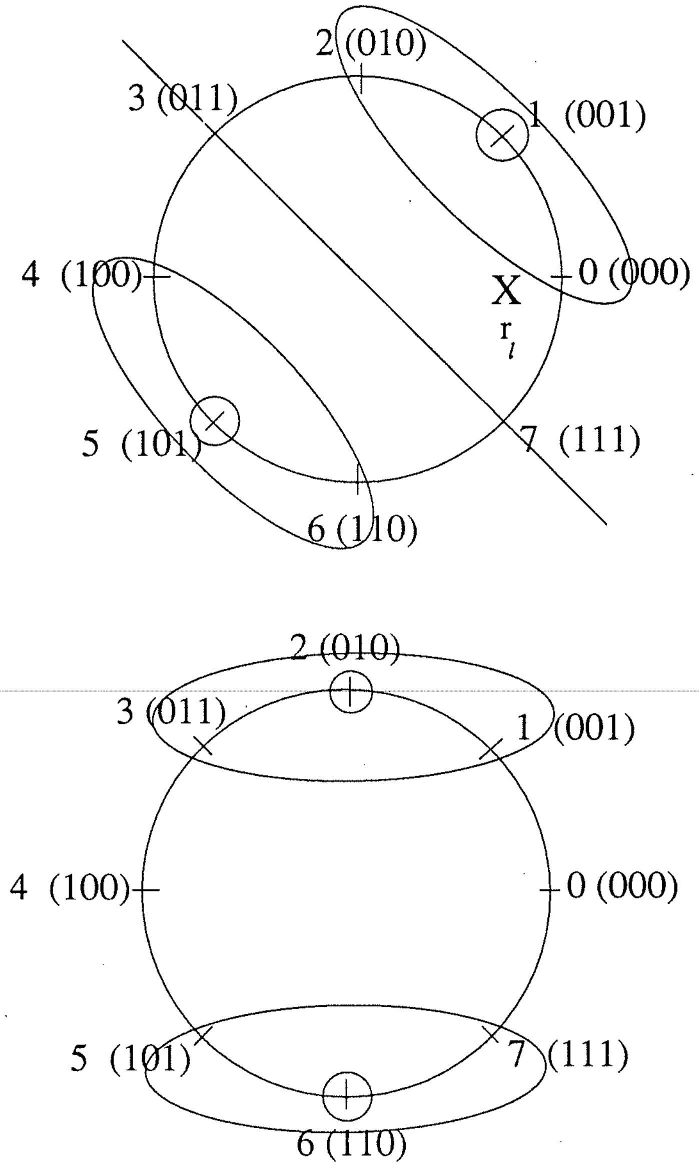


Fig. 11. 8-PSK constellation with merged points for approximate decoding

It is clear that in order to improve performance, the degradation caused by the decision at the intermediate states must be reduced and hence, "better" metrics must be chosen. With approximate decoding the branch metric at the intermediate state is the Euclidean distance between the received signal and a

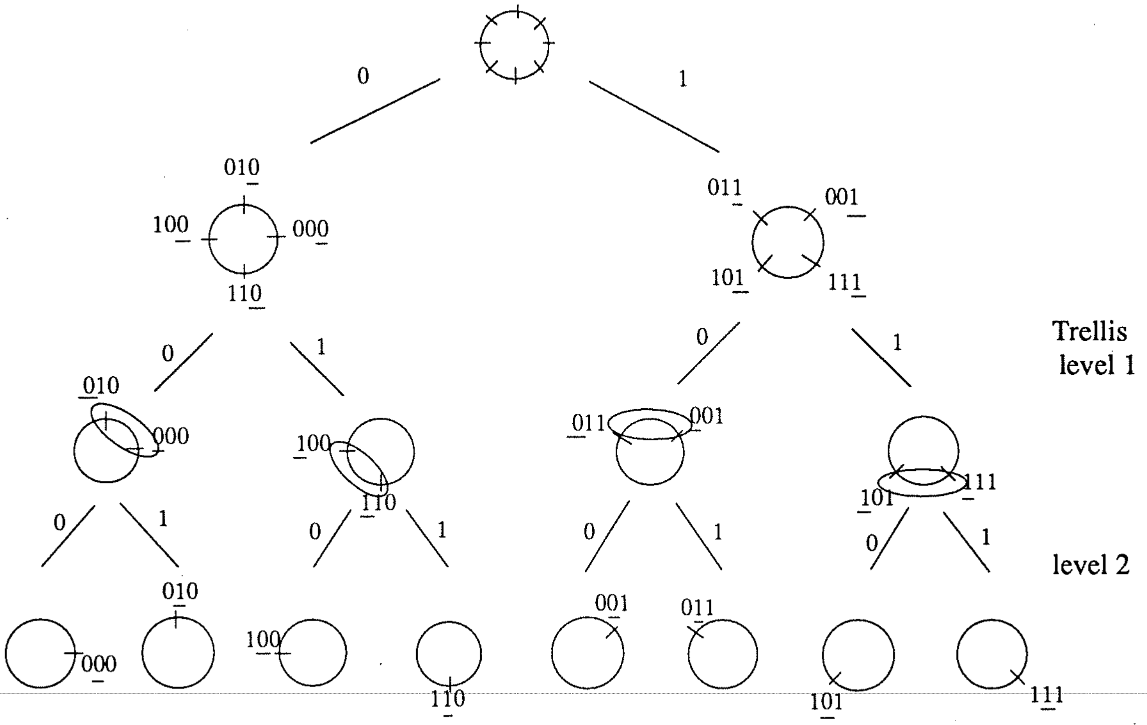


Fig. 12. Partitioning of an 8-PSK signal constellation

point equidistant from all signals in the group or subset of that trellis level. This is just an approximate metric since the optimum metrics must be the distances between the received signals and each signal in the group. The second decoding technique we propose is similar to approximate decoding, but it uses as branch metric at the intermediate state the minimum distance between the received signal and each signal in the group

$$m(r_l, c) = \min_{a_i^c} \|r_l - a_i^c\|^2 \ , \quad c = 0, 1, \quad i = 1, 2, \ldots, n$$

where a_i^c is the ith signal in the subset corresponding to branch c of the trellis level and n is the number of signals in the subset. Note that as described previously for approximate decoding, the number of signals n is halved at each trellis level.

To illustrate the technique, the example of Fig. 10 is used again. The metric of branch 0 at state c is now

$$m(r_l, 0) = \min\left(\|r_l - 000\|^2, \|r_l - 010\|^2\right)$$

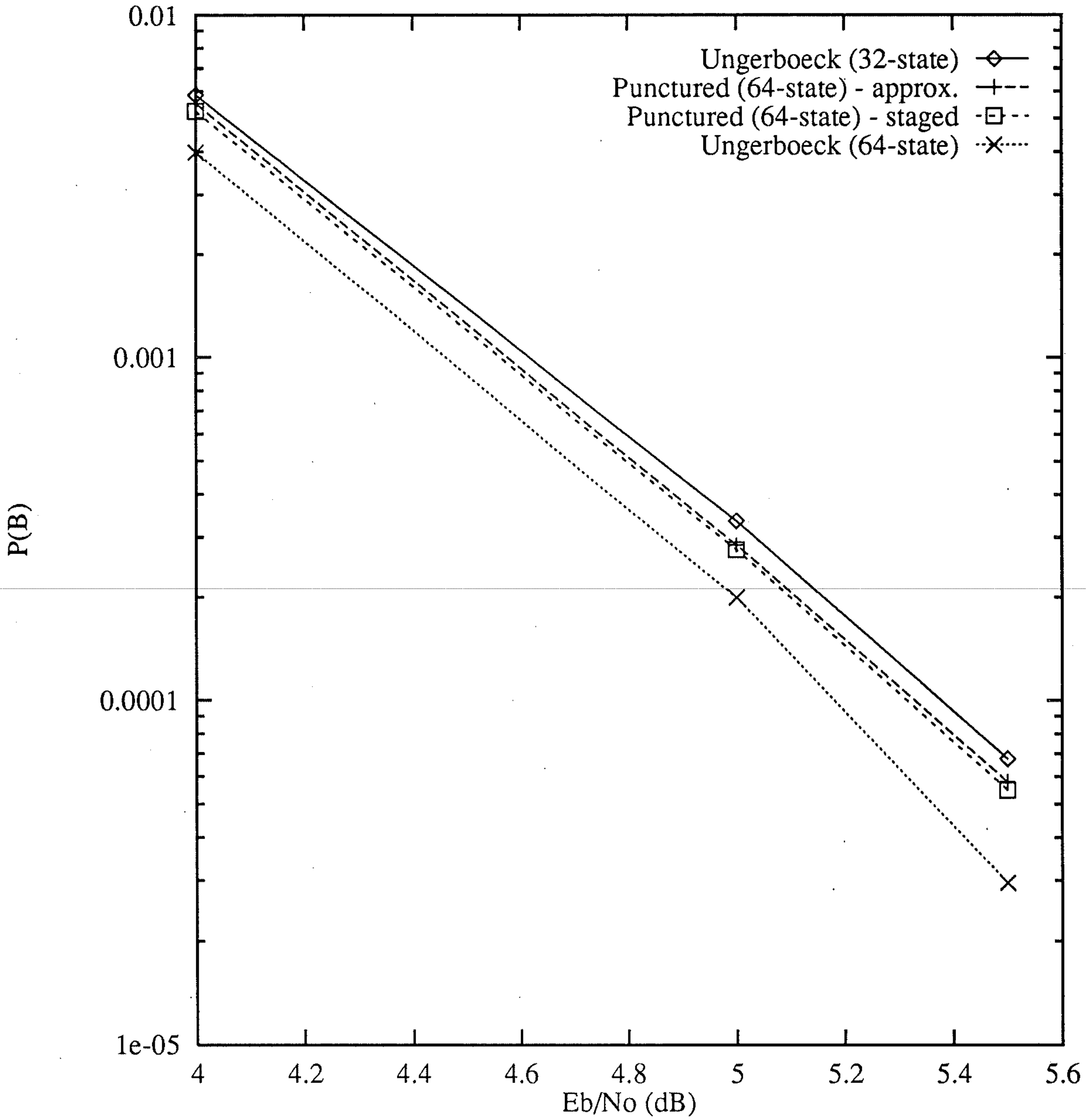


Fig. 13. Comparison of the bit error performance of three decoding techniques: optimum (Ungerboeck), approximate and staged decoding — 8-PSK

and the metric of branch 1 is

$$m(r_l, 1) = \min\left(\|r_l - 100\|^2, \|r_l - 110\|^2\right)$$

Hence, at each trellis level, the decoder first computes the branch metrics and uses them to select the survivor among the two merging branches. It should be noted that the above branch metric computation is done only once at each trellis level, i.e., it is not repeated at each state, since it depends only on the received signal r_l.

The bit error probability of the 64–state QAM code using this technique is illustrated in Fig. 14. Only 2 comparisons per state are needed to decode this rate-2/3 code, while optimum decoding requires 3 comparisons. At $P(B) = 1 \times 10^{-4}$, compared to optimum decoding, the degradation is smaller than 0.1 dB and the performance of the 64–state code with staged decoding is superior to that of the 32–state Ungerboeck code decoded optimally.

This technique is called "staged decoding" since it is similar to the suboptimal staged decoding procedure of Imai and Hirakawa [24] for their multilevel codes. Staged decoding, which is much simpler than optimum decoding, has since been widely used, in particular by Calderbank [25] and by Pottie and Taylor [26]. With multilevel coding the transmitted signal is not the mapping of the output of a single encoder, as in Ungerboeck's scheme [1], but of several encoders. Optimum decoding considers all the codes as a unique code and decodes them together, while staged decoding decodes the component codes in sequence [24]. This leads to significant complexity reductions but it is suboptimum since the first decoder receives no help, i.e., information, from the second decoder, and so on.

At first sight, it may seem that Imai and Hirakawa's procedure has nothing in common with our technique. First, even the coding is completely different: while they have several codes, we use a single one. Similarly, they need several decoders, while we require only one. However, we share the same objective of reducing the complexity of optimum decoding. The problems are essentially identical, namely: knowing only the received signal, what are the metrics of the symbols of component codes (for multilevel codes) or of the low-rate trellis branches (for PTCM codes)? It is clear that the metrics should be computed in the same manner in both cases, i.e., they are the minimum Euclidean distances between the received signal and a point in the subset at that level of partitioning (see Fig. 12). Hence, our technique can be considered as staged decoding, where

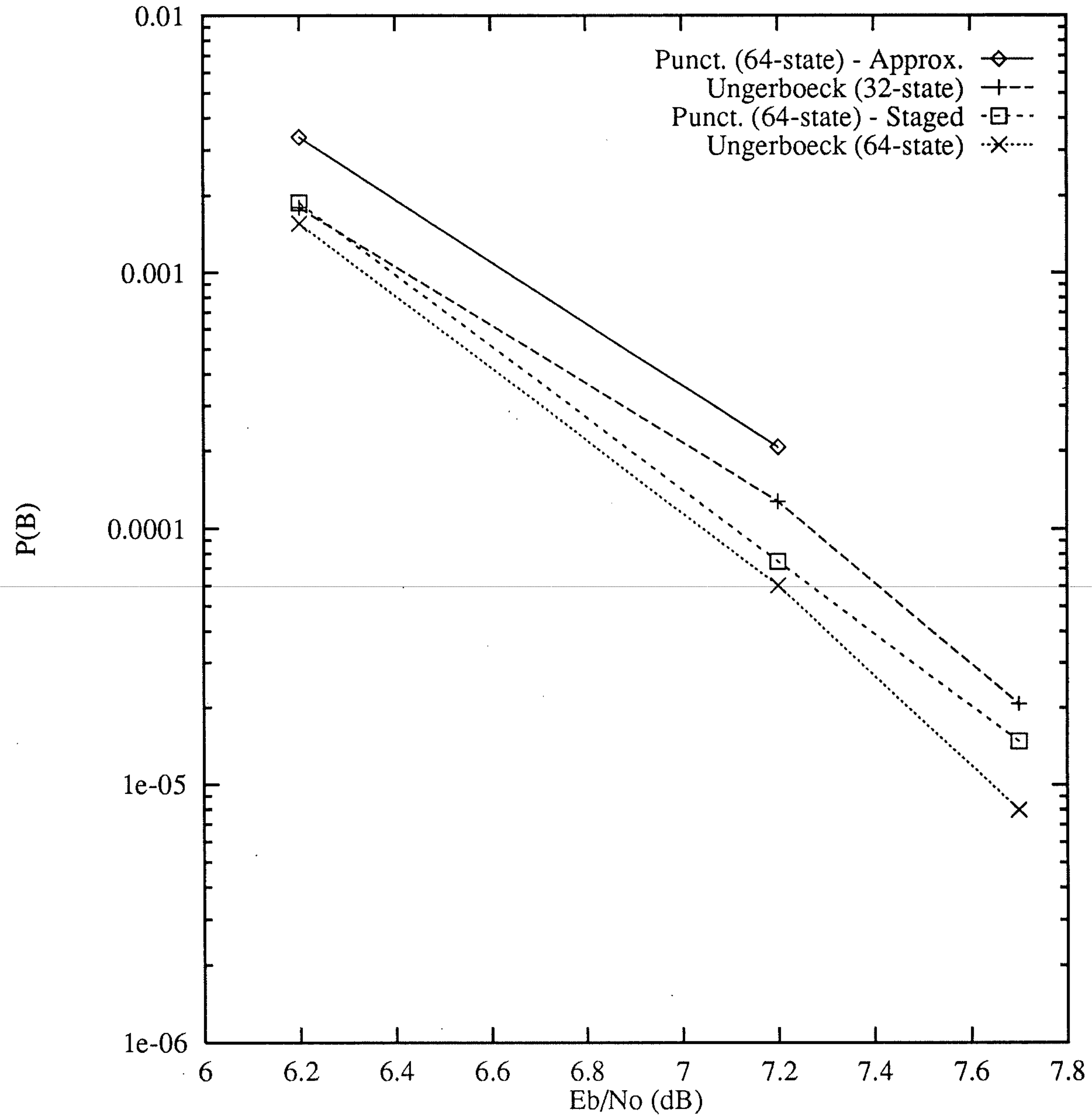


Fig. 14. Comparison of the bit error performance of three decoding techniques: optimum, approximate and staged decoding — 16-QAM

a stage is a trellis level and the number of stages is m, the period of the puncturing pattern. Furthermore, in a different application, Zehavi [27] needed the metrics of each of the three bits of a coded 8-PSK signal when bit interleaving is used. He also computed his metrics in the same manner.

Simplified decoding techniques for PTCM codes have been described. Approximate decoding and staged decoding yield almost the same error performance for 8-PSK codes. Therefore, approximate decoding may be preferred since it is slightly simpler. For 16-QAM codes, on the other hand, staged decoding is superior. For higher rate codes, which offer a significant decoding complexity reduction (m comparisons instead of 2^m-1 at each state), we may consider decoding techniques that slightly increase complexity in order to improve error performance.

As a conclusion, the error performance of PTCM codes with suboptimum decoding is slightly inferior to that of Ungerboeck codes but decoding complexity is reduced. The puncturing technique yields the same complexity savings with TCM codes as with binary convolutional codes. It should be noted that the small performance degradation is not caused by the code but by the decoding technique. Hence, if the need arises, the performance can be improved by changing the decoder, while no modification is required at the transmitter side.

IV. VARIABLE-RATE PTCM CODES

It has been shown that the best PTCM codes found using an extensive computer search can be as good as the best known codes discovered by Ungerboeck [3]. Furthermore, the decoding complexity savings with our proposed techniques are the same as with punctured convolutional codes and these savings entail only a very slight degradation of the error performance. We show in this section that the second advantage of punctured convolutional codes, rate variability, is maintained for PTCM.

As already mentioned, the rate of a convolutional code can be easily changed by modifying only the puncturing pattern. For example, a single rate-1/2 code can provide punctured codes with rate R=2/3 or 3/4. By mapping the output of these codes onto the appropriate constellations, a family of QPSK, 8-PSK and

Number of states	Variable-rate PTCM codes				Best known codes		
	h^1, h^0	QPSK	8PSK	16PSK	QPSK	8PSK	16PSK
16	4, 23	7	5.17	1.628	7	5.17	1.628
32	10, 45	7	5.17	1.910	8	5.76	1.910
64	74, 163	8	6.34	2.0	10	6.34	2.0
128	154, 325	10	6.34	2.0	10	6.58	2.0

Table 4 Free distances of variable-rate PTCM codes

16-PSK PTCM codes is obtained from a single low-rate code. In Section II, the best code with a given number of states for a constellation may be different from the best code for another constellation. We now search for rate-1/2 codes with the largest free Euclidean distances for QPSK, 8–PSK and 16–PSK and not only for one of these constellations. The families of such codes obtained from rate-1/2 codes are listed in Table 4, where h^1 and h^0 are the parity-check coefficients. Although each PTCM code in the family may not necessarily be the best for that rate, its free Euclidean distance is equal, or only slightly smaller than the distance of the optimum code. Table 4 presents a family of variable-rate codes or variable modulation schemes, also called rate-compatible codes.

The advantage is that a single encoder/decoder is sufficient to achieve variable bandwidth efficiencies and hence, variable throughputs. The puncturing technique is more flexible than Viterbi et al's pragmatic approach [5], [28] since it allows either a true high-rate code or a code with parallel branches. Since many good codes for the gaussian channel [3] and especially for fading channels (see e.g. [29], [4]) do not have parallel branches, the puncturing technique may be able to reproduce such codes whereas the pragmatic approach cannot. The free squared Euclidean distance of our 64–state code is $d^2_{free} = 6.34$ for 8-PSK and thus, the asymptotic coding gain is 5.0 dB, while the distance of the pragmatic code is $d^2_{free} = 4.0$, resulting in an asymptotic coding gain of 3.0 dB [5]. It can be seen that when parallel branches become a limitation, the puncturing technique can provide codes without parallel branches which have a larger free distance than pragmatic codes.

Depending on the application for which the variable-rate code is intended, another family may be more suitable than the one listed in Table 4. As an example, for the 64–state code in Table 4, we have assumed that 8-PSK or 16-PSK are used most of the time and QPSK is seldom used. Therefore, it is more advantageous to select a code providing a larger free distance with 8-PSK and 16-PSK, even though the distance for QPSK may be reduced. The average error performance is improved since QPSK is used to transmit very few bits and hence, its free distance has a smaller effect on the probability of error. On the other hand, if QPSK is used more often than 8-PSK or 16-PSK, we may choose a code with a larger distance for QPSK.

The 64–state code listed in Table 4 has the same distance as the best known codes for 8-PSK and 16-PSK. Its QPSK distance is, however, slightly inferior. This distance can be improved without affecting the trellis connectivity and hence, rate variability. This is achieved by using the method already suggested in Section II, i.e., considering that the rate of the original code is lower than R=1/2. An encoder providing a free Hamming distance $d_{free} = 10$ has been obtained by computer search and is illustrated in Fig. 15. Notice that it is the same encoder that the one in Table 4, except that a third output has been added. State transitions are not modified, the trellis connectivity remains the same. The perforation matrix for QPSK is

$$\mathbf{P} = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$$

yielding a rate R=1/2 code. For 8-PSK (rate–2/3), the matrix is

$$\mathbf{P} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}$$

This means that the third output symbol is simply ignored or punctured and we have the same encoder as in Table 4. The same procedure is repeated for 16-PSK, i.e., the third symbol is always punctured.

A variable-rate communications system may be useful when several types of information have to be transmitted: for example, QPSK can be used for data, 8-PSK for digitized voice and 16-PSK for video, which in general consists of a large number of bits. Furthermore, this technique can be used for the same applications as punctured convolutional codes [30]. In type I hybrid ARQ/FEC

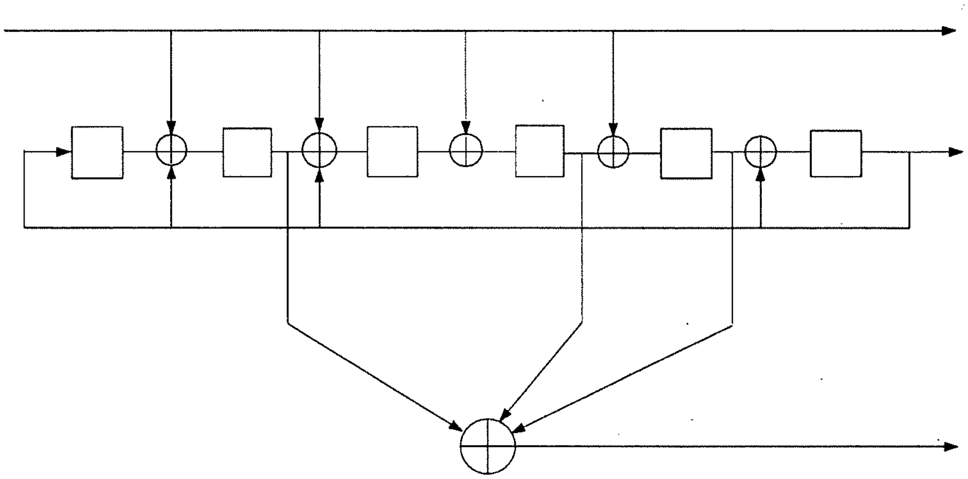


Fig. 15. Rate R=1/3 encoder for QPSK, 8-PSK or 16-PSK

transmission schemes [31], 16-PSK, which provides the highest throughput, can be used first and, should an error be detected, 8-PSK, or even QPSK may be used in subsequent retransmissions.

PTCM may also be well suited for unequal error protection (UEP), assuming some information about the susceptibility to errors of the information bits is supplied by the source: the most sensitive data may be transmitted using QPSK, the less sensitive ones may be transmitted using 8-PSK or 16-PSK. Hence, by changing the rate of the code, three levels of error protection are available; the bit error probability is improved approximately by more than three orders of magnitude from one level to the next one, e.g., from 8-PSK to QPSK.

V. CONCLUSION

Clark and Cain [22, p. 383] claimed that "the advantages of punctured codes are of no utility in this application" (Trellis-Coded Modulation). We have applied the puncturing technique to Trellis-Coded Modulation and arrived at a different conclusion: the advantages puncturing provides to binary convolutional codes are maintained. First, simplified decoding can be performed at the expense of a small performance degradation. The puncturing technique provides the same complexity reduction as obtained with convolutional codes. A complexity/error performance trade-off is available with PTCM. Furthermore, variable rate using a single code is possible by changing the puncturing pattern. Hence, variable bandwidth efficiency systems can be easily implemented. Finally, PTCM is very flexible and the code is not limited to a rate R=1/2 with parallel branches, it can be a true high-rate code. However, a code with parallel branches is still available. It is expected that the potential for PTCM is tremendous since it can duplicate almost any TCM code (although we have considered here only the simplest ones) providing simplified decoding and variable bandwidth efficiency systems.

ACKNOWLEDGEMENT

Stimulating discussions about this work with Professor H. Leib of McGill University are gratefully acknowledged.

REFERENCES

[1] G. Ungerboeck, "Channel coding with multilevel/phase signals," *IEEE Trans. Inform. Theory*, vol. IT-28, pp. 55–67, Jan. 1982.

[2] G. Ungerboeck, "Trellis-coded modulation with redundant signal sets — part I: Introduction," *IEEE Commun. Mag.*, pp. 5–11, Feb. 1987.

[3] G. Ungerboeck, "Trellis-coded modulation with redundant signal sets — part II: State of the art," *IEEE Commun Mag.*, pp. 12–21, Feb. 1987.

[4] E. Biglieri, D. Divsalar, P. J. Mclane, and M. K. Simon, *Introduction to Trellis-Coded Modulation with Applications*. Macmillan Publishing Company, New York, 1991.

[5] A. J. Viterbi, J. Wolf, E. Zehavi, and R. Padovani, "A pragmatic approach to trellis-coded modulation," *IEEE Commun. Mag.*, pp. 11–19, Jul. 1989.

[6] F. Chan and D. Haccoun, "High-rate punctured convolutional codes for trellis-coded modulation," in *Proc. IEEE International Symposium on Information Theory*, p. 414, San Antonio, Texas, 1993.

[7] S. A. Morley, "Forward error correction applied to Intelsat IDR carriers," *International Journal of Satellite Communications*, vol. 6, pp. 445–454, 1988.

[8] J. B. Cain, G. C. Clark, and J. M. Geist, "Punctured convolutional codes of rate (n-1)/n and simplified maximum likelihood decoding," *IEEE Trans. Inform. Theory*, vol. IT-25, pp. 97–100, Jan. 1979.

[9] Y. Yasuda, K. Kashiki, and Y. Hirata, "High-rate punctured convolutional codes for soft decision Viterbi decoding," *IEEE Trans. Commun.*, vol. COM-32, pp. 315–319, Mar. 1984.

[10] D. Haccoun and G. Bégin, "High-rate punctured convolutional codes for Viterbi and sequential decoding," *IEEE Trans. Commun.*, vol. 37, pp.1113–1125, Nov. 1989.

[11] G. Bégin and D. Haccoun, "High-rate punctured convolutional codes: Structure properties and construction technique," *IEEE Trans. Commun.*, vol. 37, pp. 1381–1385, Dec. 1989.

[12] G. Bégin, D. Haccoun, and C. Paquin, "Further results on high-rate punctured convolutional codes for Viterbi and sequential decoding," *IEEE Trans. Commun.*, vol. 38, pp. 1922–1928, Nov. 1990.

[13] G. Bégin and D. Haccoun, "Performance of sequential decoding of high-rate punctured convolutional codes," *accepted for publication in IEEE Trans. Commun.*

[14] "Q0256 — k=7 multi-code rate Viterbi decoder technical data sheet," Qualcomm, Incorporated, San Diego, CA 92121, 1990.

[15] "Custom ASIC products short form catalog," Stanford Telecommunications, Incorporated, Santa Clara, CA, 1989.

[16] J. B. Anderson and S. Mohan, *Source and Channel Coding: An Algorithmic Approach.* Kluwer Academic Publishers, 1991.

[17] J.-E. Porath, "On trellis coded modulation for gaussian and band-limited channels," Tech. Rep. 221, School of Elec. and Comp. Eng., Chalmers Univ. of Technology, Göteborg, Sweden, Nov. 1991.

[18] G. D. Forney, "Convolutional codes i: Algebraic structure," *IEEE Trans. Inform. Theory*, vol. IT-16, pp. 720–738, Nov. 1970.

[19] P. CharnKeitKong, K. Yamaguchi, and H. Imai, "New constructions of $k/(k+1)$ rate-variable punctured convolutional codes," in *Proc. IEEE International Symposium on Information Theory*, San Antonio, Texas, 1993.

[20] S. K. How, "Practical trellis coded modulation with punctured rate-2/3 convolutional codes," in *Proc. IEEE International Symposium on Information Theory*, p. 416, San Antonio, Texas, 1993.

[21] A. J. Viterbi, "Convolutional codes and their performance in communication systems," *IEEE Trans. Commun. Technol.*, vol. COM-19, pp. 751–772, Oct. 1971.

[22] G. C. Clark and J. B. Cain, *Error-Correction Coding for Digital Communications.* Plenum Press, 1981.

[23] F. Chan and D. Haccoun, "Adaptive decoding of convolutional codes over memoryless channels," *submitted for publication in IEEE Trans. Commun.*, Feb. 1992.

[24] H. Imai and S. Hirakawa, "A new multilevel coding method using error-correcting codes," *IEEE Trans. Inform. Theory*, vol. IT-23, pp. 371–377, May 1977.

[25] A. R. Calderbank, "Multilevel codes and multistage decoding," *IEEE Trans. Commun.*, vol. 37, pp. 222–229, March 1989.

[26] G. J. Pottie and D. P. Taylor, "Multilevel codes based on partitioning," *IEEE Trans. Inform. Theory*, vol. 35, pp. 87–98, Jan. 1989.

[27] E. Zehavi, "8–PSK trellis codes for a Rayleigh channel," *IEEE Trans. Commun.*, vol. 40, No. 5, pp. 873–884.

[28] "Q1875 pragmatic trellis decoder technical data sheet," Qualcomm, Incorporated, San Diego, CA 92121, 1992.

[29] D. Divsalar and M. K. Simon, "The design of trellis coded MPSK for fading channels: Performance criteria," *IEEE Trans. Commun.*, vol. 36, No. 9, pp. 1004–1012, Sep. 1988.

[30] J. Hagenauer, "Rate-compatible punctured convolutional codes (RCPC codes) and their applications," *IEEE Trans. Commun.*, vol. 36, pp. 389–400, Apr. 1988.

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