On Saving Decoder States for Some Trellis Codes and Partial Response Channels

EPHRAIM ZEHAVI AND JACK K. WOLF

Abstract—**It is shown that under certain conditions, when convolutional codes with precoding are used in conjunction with a partial response channel, the number of decoder states in a maximum likelihood decoder matched to both the code and channel is 1/2 of that predicted by Wolf and Ungerboeck [1].**

I. Introduction

Wolf and Ungerboeck [1] considered constructing good trellis codes for transmitting information over a partial response channel where the decoder is matched both to the redundancy of the code and to the intersymbol interference generated by the channel. A method of constructing these trellis codes suggested by Wolf and Ungerboeck was to utilize a binary convolutional code with 2^v states and then to pass the encoded output sequence through a precoder prior to transmission over a partial response channel. In particular, for a class IV partial response channel [2] with transfer function $(1 - D^2)$, Wolf and Ungerboeck suggested treating the channel as the interleaving of two $(1 - D)$ channels and using a precoder with transfer function $(1 \oplus D)^{-1}$. In that case, the decoder would in general require 2^{v+1} states.

Here, we show that for a class of convolutional codes [with a $(1 \oplus D)^{-1}$ precoder and a $(1 - D)$ channel] the number of states in the decoder matched to the code, precoder, and channel is only 2^v. Within the class are several well-known codes including the rate 1/2 convolutional code of constraint length 7 and free distance 10. For this code, we will show how to modify the 64-state Viterbi decoder matched to the code so that it matches both the code, precoder, and the intersymbol interference of the $(1 - D)$ channel.

The codes discussed in this paper have similar properties to the trellis codes found by other authors [3]–[4]. The approach taken here, however, is different and is a natural generalization of the work of Wolf and Ungerboeck [1]. In particular, the results of this paper give further evidence to the remarkable fact that good binary convolutional codes for the binary symmetric channel, when properly applied, yield good binary convolution codes for the $1 \pm D$ [or $1 \pm D^n$] partial response channel.

An outline of this correspondence follows. In the next section, we summarize the applicable work of Wolf and Ungerboeck and describe the class of codes for which we can save decoder states. This is followed by a section containing examples of codes in the class. Included in this section are details of the decoder design for a rate 1/2 constraint length 7 code of free Hamming distance 10. Finally, we conclude with a brief summary.

II. Convolutional Codes and the $(1 - D)$ Channel

We consider the coding scheme proposed by Wolf and Ungerboeck shown in Fig. 1. A binary sequence U is encoded into another binary sequence V using a rate R convolutional code having a 2^v-state encoder. The free Hamming distance of the code is d_H. The output of the encoder is passed through a precoder with transfer function $(1 \oplus D)^{-1}$ resulting in the binary sequence W. The elements of U, V, and W all are assumed to be elements from the finite field $GF(2)$. The symbols of W are term by term converted to the real number binary sequence X with elements from the set $(+1, -1)$ by converting the finite field symbol 0 to the real number $+1$ and the finite field symbol 1 to the real number -1. (This is not necessary for this discussion, but is included since this was part of the scheme proposed by Wolf and Ungerboeck.) The symbols X are passed through a $(1 - D)$ partial response channel resulting in the real number sequence Y [with elements from the set $(+2, 0, -2)$]. Noise N is added to Y resulting in the sequence Z. For purposes of analysis, we assume the noise to be i.i.d. Gaussian random variables with mean 0 and variance σ^2. A decoder produces a maximum likelihood estimate of the sequence U which we write as $\hat{U}$.

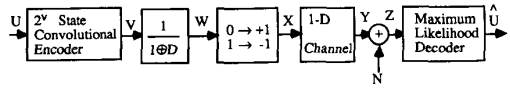

Fig. 1. Block diagram of encoded $(1 - D)$ system.

Wolf and Ungerboeck showed that:

a) the free squared Euclidean distance between output sequences Y of the $(1 - D)$ channel is at least $8\lfloor(d_H + 1)/2\rfloor$ where $\lfloor x \rfloor$ is the integer value of x,

b) the maximum likelihood decoder requires, in general, 2^{v+1} states.

Here, we show that for a certain class of convolutional codes, the maximum likelihood decoder requires only 2^v states. This class is characterized by a property of the generator polynomials of the code.

Consider a rate $R = k/n$ binary convolutional code which converts k input symbols into n output symbols. Thus, although we show in Fig. 1 an encoder with one input stream U and one output stream V, a more detailed description of this encoder would be as shown in Fig. 2.

Letting $U_i(D)$ and $V_j(D)$ be the D transform of the ith input stream and jth output stream, respectively, we describe the encoder by the set of equations

$$V_j(D) = \sum_{i=1}^{k} G_{ij}(D)\, U_i(D), \qquad j = 1, 2, \cdots, n.$$

Note that if we denote the D transform of the overall output sequence V, as $V(D)$, then $V(D)$ can be written as

$$V(D) = \sum_{j=1}^{n} D^{j-1} V_j(D^n).$$

Defining $G_i(D)$ as

$$G_i(D) \equiv \sum_{j=1}^{n} D^{j-1} G_{ij}(D^n),$$

we then have

$$V(D) = \sum_{i=1}^{k} G_i(D)\, U_i(D^n).$$

We define C as the class of codes for which $G_i(D)$ is divisible by $(1 \oplus D)$ for $i = 1, 2, \cdots, k$. We next show that for all binary convolutional codes in class C having 2^v states, the number of states in the maximum likelihood decoder matched to the code, precoder, and $(1 - D)$ channel is no more than

Paper approved by the Editor for Coding Theory and Applications of the IEEE Communications Society. Manuscript received November 19, 1986; revised June 24, 1987.

E. Zehavi is with QUALCOM, Inc., San Diego, CA 92121.

J. K. Wolf is with the Center for Magnetic Recording Research, University of California, La Jolla, CA 92093, and QUALCOM, Inc., San Diego, CA 92121.

IEEE Log Number 8718646.

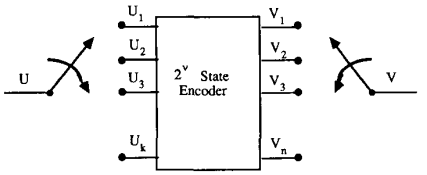

Fig. 2. Rate k/n encoder.

2^v. We prove this statement by noting that the D transform of W, $W(D)$, can be written as

$$W(D)=\sum_{i=1}^{k} G_i'(D)U_i(D^n)$$

where

$$G_i'(D)=\frac{G_i(D)}{1 \oplus D}=\sum_{j=1}^{n} D^{j-1}G_{ij}'(D^n).$$

An encoder defined by the polynomials $G_{ij}'(D)$ takes the uncoded input and directly produces the precoder output W. This encoder has no more states than the original encoder (i.e., 2^v).

Since, this encoder produces the output sequence of the precoder, the state of the precoder must be incorporated in these 2^v states. However, if one knows the precoder state, then one also knows the state of the $(1 - D)$ channel. Thus, the 2^v encoder states are sufficient for the decoder to track the states of the original encoder, precoder, and $(1 - D)$ channel.

The free Hamming distance for all codes in class C is even since total number of ones in all of the generator polynomials is even. Thus, the free squared Euclidean distance between output sequences of the $(1 - D)$ channel is at least $4d_H$ for all codes in class C.

III. Examples

An examination of convolutional codes with maximal free distance [5, Table 11.1] of rates 1/2, 2/3, 3/4, having no more than 1024 states, produced the codes in class C that are listed in Table I. More codes in class C exist than those listed in the table. For example, the rate 1/2, 32-state code with $d_H = 8$ and generator polynomials, $G_1(D) = 1 + D^2 + D^5$, and $G_2(D) = 1 + D + D^3 + D^4 + D^5$, which is available in a gate array device [6], is also in this class.

Finally, we consider the details of the rate 1/2 code with 64 states. The ordinary encoder for the code is shown in Fig. 3. Since $G(D) = G_1(D^2) + DG_2(D^2) = 1 + D + D^3 + D^4 + D^5 + D^6 + D^7 + D^{10} + D^{12} + D^{13}$, then $G'(D) = 1 + D^3 + D^5 + D^7 + D^8 + D^9 + D^{12}$ so that $G_1'(D) = 1 + D^4 + D^6$ and $G_2'(D) = D + D^2 + D^3 + D^4$. Thus, an encoder which produces the output W of the precoder is shown in Fig. 4. It is easy to see in this case why knowledge of the states of the encoder is sufficient to specify the state of the $(1 - D)$ channel. If one knows the present state of the encoder, one knows the previous output W_2 since W_2 does not depend upon the content of the last (rightmost) storage element. But this previous value of W_2 is just the content of storage element in the $(1 - D)$ channel (after making the transformation $0 \rightarrow +1$, $1 \rightarrow -1$).

Finally, we comment on the structure of the Viterbi maximum likelihood decoder for this code when used for transmission over a $(1 - D)$ channel. The trellis for describing the noiseless output sequences from the channel is the same as that for the code itself except for labeling of the branches. Where the code itself has only four possible branch labelings $\{(0, 0), (0, 1), (1, 0), (1, 1)\}$ there are now seven different branch labels $\{(0, 0), (2, 0), (-2, 0), (0, 2), (0, -2), (2, -2), (-2, 2)\}$. These branch labels effect the branch metrics which must be added to the state metrics before the best path to a state is chosen. If the branch metrics are stored in a ROM, the only modification that need be made to the Viterbi decoder for the code alone to adapt it for use with the $(1 - D)$ channel is to replace one ROM table with another ROM table.

TABLE I
List of Convolutional Codes in Class C

Rate	# of decoder states	Generators	d_H
1/2	64	$G_1(D) = 1+D^2+D^3+D^5+D^6$ $G_2(D) = 1+D+D^2+D^3+D^6$	10
1/2	256	$G_1(D) = 1+D^2+D^3+D^4+D^8$ $G_2(D) = 1+D+D^2+D^3+D^5+D^7+D^8$	12
1/2	1024	$G_1(D) = 1+D^3+D^4+D^6+D^7+D^8+D^{10}$ $G_2(D) = 1+D+D^2+D^3+D^5+D^6+D^{10}$	14
2/3	128	$G_{11}(D) = 1+D$ $G_{12}(D) = D+D^2+D^3$ $G_{13}(D) = 1+D^2+D^3$ $G_{21}(D) = D^2+D^3+D^4$ $G_{22}(D) = 1+D^3+D^4$ $G_{23}(D) = 1+D+D^2+D^3$	8
2/3	1024	$G_{11}(D) = 1+D+D^4+D^5$ $G_{12}(D) = D^2+D^3+D^5$ $G_{13}(D) = 1+D^3+D^4$ $G_{21}(D) = D+D^2+D^4$ $G_{22}(D) = 1+D+D^3+D^5$ $G_{23}(D) = 1+D+D^5$	10
3/4	512	$G_{11}(D) = 1$; $G_{23}(D) = D$ $G_{12}(D) = 1+D^3$; $G_{24}(D) = 1+D+D^2$ $G_{13}(D) = D+D^2+D^3$; $G_{31}(D) = D+D^2+D^3$ $G_{14}(D) = 1+D$; $G_{32}(D) = 0$ $G_{21}(D) = D^3$; $G_{33}(D) = 1+D$ $G_{22}(D) = 1+D+D^3$; $G_{34}(D) = 1+D+D^3$	8

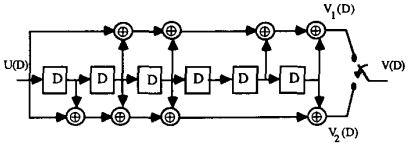

Fig. 3. Ordinary encoder for rate 1/2, 64-state code.

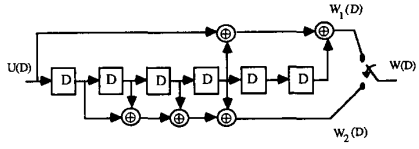

Fig. 4. Combined encoder/precoder.

IV. Summary

In this correspondence, we have described a class of codes for a partial response channel utilizing a maximum likelihood decoder matched to the code and the channel which has no more states than a decoder matched to the code itself.

References

[1] J. K. Wolf and G. Ungerboeck, "Trellis coding for partial-response channels," *IEEE Trans. Commun.*, vol. COM-34, pp. 751-772, July 1986.

[2] E. R. Kretzmer, "Generalization of a technique for binary data

communication," *IEEE Trans. Commun. Technol.*, vol. COM-14, pp. 67-68, Feb. 1966.
[3] A. R. Calderbank, C. Heegard, and T.-A. Lee, "Binary convolutional codes with application to magnetic recording," *IEEE Trans. Inform. Theory*, vol. IT-32, pp. 797-815, Nov. 1986.
[4] T.-A. Lee and C. Heegard, "An inversion technique for the design of binary convolutional codes for the $1 - D^n$ channel," in *Proc. IEEE Regional Meet.*, The Johns Hopkins Univ., Baltimore, MD, Feb. 1985.
[5] S. Lin and D. J. Costello, Jr., *Error Control Coding: Fundamentals and Application*. Englewood Cliffs, NJ: Prentice-Hall, 1983.
[6] Stanford Telecommunications, Inc., New Product Announcement, ST-2010, Viterbi Decoder.

Comment on "Performance of Single Access Classes on the IEEE 802.4 Token Bus"

ANURA P. JAYASUMANA

Abstract—**In the above correspondence [1], throughput bounds are derived for the IEEE 802.4 token passing scheme, in the presence of a single class of messages. It does not consider the fact that even after the token hold_timer expires, a node is allowed to complete its message transmission. In this correspondence, we consider this fact and modify the upper bounds on throughput given in [1] accordingly.**

I. Introduction

The IEEE 802.4 standard [2] specifies an optional priority mechanism which uses a set of timers to allocate bandwidth among different access classes. It works as follows: any station which receives the token loads its token_hold_timer with the value of hi_pri_token_hold_time. Then the station is allowed to initiate transmission of messages of highest access_class, until the timer expires or the messages of that queue are exhausted. If the token_hold_timer expires during a transmission, that transmission is completed before servicing the messages of the next access_class. The token hold time of the highest priority class can therefore exceed the value of hi_pri_token_hold_time. For each of the lower access classes, the token_hold_timer is loaded with the residual time from the token_rotation_timer. The station is allowed to initiate transmission of messages of current access_class, until the token_hold_timer expires or the messages in the corresponding queue are exhausted. If the timer expires during a transmission, then the current transmission is completed before servicing the next access_class or transmitting the token to the next station.

In [1], it is assumed that the message transmission of a given access_class is terminated at the very moment the token_hold_timer expires, i.e., it neglects the residual transmission time. The residual transmission time is the time required to complete a message transmission after the token_hold_timer has expired. This assumption will provide reasonable results when the average number of messages transmitted per token rotation is large. The residual transmission time, in general, depends on the message length distribution and the initial value of the token_hold_timer. Reference [1] assumes a constant message transmission time. In this case, if the initial token_hold_timer value is a multiple of the message transmission time, then the residual transmission time is zero. For asynchronous classes, however, the intial value of the token_hold_timer varies from one token rotation to the next. Below, we present an analysis in which we take into consideration the completion of message transmission after the expiration of token_hold_timer. The message transmission times are assumed to be exponentially distributed.

Paper approved by the Editor for Local Area Networks of the IEEE Communications Society. Manuscript received March 6, 1987; revised May 11, 1987.

The author is with the Department of Electrical Engineering, Colorado State University, Fort Collins, CO 80523.

IEEE Log Number 8718640.

II. Bounds

The notation used in [1] is summarized below:

N — number of stations
u — mean utilization of the network by a station
U — total network utilization
$\bar{C}$ — mean token rotation time
C_{max} — upper bound on mean token rotation time
X_{msg} — mean message transmission time
X_{token} — mean token transmission time
T_{sync} — hi_pri_token_hold_time
T_{async} — target_rotation_time.

The message transmission times are assumed to be exponentially distrubuted with mean X_{msg}. The relationship between U and $\bar{C}$ is given by [1]

$$\bar{C} = \frac{NX_{token}}{[1-U]}. \tag{1}$$

Consider the case where only synchronous messages are present in a network. A station then transmits high priority messages until either the token_hold_timer expires or the frames in the queue are exhausted. When a queue is heavily loaded, i.e., when it has more frames than it as allowed to transmit, the token hold time of a queue is limited by the first condition. In this case, the station attempts to transmit more frames, provided the token_hold_timer has not expired. If the token_hold_timer expires during the transmission of a message, the node will still complete message transmission before it transfers control to the next access_class. The memoryless property of exponential message length distribution ensures that the mean length of message portion left to be transmitted after token_hold_timer expires is also X_{msg}. Thus, under very high loads, when the station tries to send as many synchronous messages as possible in a given cycle, the mean token hold time is given by $(T_{sync} + X_{msg})$. If the value of hi_pri_token_hold_time is zero, no synchronous messages are transmitted. Under these conditions, [1, equat. (2)] would be modified as follows:

$$\begin{aligned} C_{max} &= NX_{token} && \text{if } T_{sync} \le 0 \\ &= N[T_{sync} + X_{token} + X_{msg}] && \text{if } T_{sync} > 0. \end{aligned} \tag{2}$$

If $T_{sync} \le 0$, no messages of the synchronous class are transmitted and hence the network utilization is zero. For $T_{sync} > 0$, the maximum utilization can be obtained using (1) and

0090-6778/88/0200-0224$01.00