Minimality for Punctured Convolutional Codes

C. Fragouli, C. Komninakis and R. D. Wesel
Electrical Engineering Department, University of California at Los Angeles
christin@ee.ucla.edu, chkomn@ee.ucla.edu, wesel@ee.ucla.edu

Abstract— **This paper investigates encoders optimization for Hamming weight after periodic puncturing, and discusses minimality issues that may affect the performance of the punctured encoders. Periodically puncturing a minimal encoder produces a higher rate encoder that may or may not be minimal. If it is not minimal, it may have a zero-output loop and it may be catastrophic. A code search can use a fast algorithm to determine whether an encoder's state diagram has a zero-output loop under periodic symbol puncturing, and a proposed method to assess the performance of codes with a zero-output loop that are not catastrophic. As an example, the paper optimizes rate-$1/4$ unpuctured codes for Hamming weight under both bit-wise and symbol-wise periodic puncturing. Code tables and simulation results are included.**

I. Introduction

Trellis codes can be designed to offer reliable performance under periodic puncturing, that may occur either at the transmitter to achieve rate variability, or over periodic erasure channels. Such channels arise for example from partial-band interference in frequency-hopped or multicarrier transmission, that is dispersed by a block interleaver. Lapidoth [1] and Wesel [2] describe techniques for the analysis and design of convolutional codes and trellis codes respectively, that can offer consistent performance over a set of periodic erasure patterns. This paper deals with minimality issues that may arise when puncturing periodically. These issues are of interest during exhaustive searches.

A general description of a convolutional encoder with k inputs, n outputs, and m memory elements is given by the state-space equations over $GF(2)$

$$\mathbf{s}_{j+1} = \mathbf{s}_j\mathbf{A} + \mathbf{u}_j\mathbf{B}$$
$$\mathbf{x}_j = \mathbf{s}_j\mathbf{C} + \mathbf{u}_j\mathbf{D},$$

where $\mathbf{s}_j$ is the state vector of dimension $1 \times m$, $\mathbf{x}_j$ is the output vector of dimension $1 \times n$ and $\mathbf{u}_j$ is the input vector of dimension $1 \times k$.

Periodic symbol puncturing [3] with period p can be described by a p-element vector $\tilde{\mathbf{a}} = [a_1 \dots a_p]$, $a_i \in \{0, 1\}$, applied to the output of a convolutional encoder

$$\mathbf{y}_j = a_{(j\%p)}\mathbf{x}_j, \tag{1}$$

where $\%$ denotes the modulo operation, and $\mathbf{y}_j$ is the output after puncturing. A $p-q$ erasure pattern has q zero values $a_i = 0$ (erased), and $p-q$ nonzero values $a_i = 1$ (unerased). A code under periodic symbol puncturing is periodically time-variant, in the sense that the output corresponding to a specific encoder state and input depends on the element i of the puncturing pattern.

This work was supported by NSF CAREER award #9733089, and the Xetron Corporation.

Periodic bit puncturing with the period equal to the number of output bits n can be described by

$$\mathbf{y}_j = \mathbf{x}_j \odot \tilde{\mathbf{a}}, \tag{2}$$

where $\odot$ stands for the element by element multiplication of vectors $\mathbf{x}_j$ and $\tilde{\mathbf{a}}$. Bit puncturing with a $p-q$ puncturing pattern amounts to ignoring q out of p outputs of the initial code, and thus leads to a time-invariant code of a higher rate.

Among all encoders that produce the same set of output sequences, the encoder that uses the smallest number of memory elements is called minimal [4]. A minimal encoder is *output observable* (and vice versa) [5], [6], [7], that is, the knowledge of the output sequence for a finite number of steps, is sufficient to determine the initial state $\mathbf{s}_0$ and thus the corresponding state sequence.

Assume that the encoder $\{\mathbf{A},\mathbf{B},\mathbf{C},\mathbf{D}\}$ is minimal. Puncturing creates a higher rate encoder, that may or may not be output observable, depending on whether puncturing has left enough structure to the output sequences to determine the corresponding state sequences. If the punctured encoder is not output observable, then two different semi-infinite state sequences are mapped to the same output sequence. From linearity this implies that either the state diagram has a zero-output loop different than the self-loop around the zero state, or there exists a non-zero state with a zero-output path (forward or backward) to the zero state [8].

If there exists a zero-output loop with non-zero input, the encoder is catastrophic. A catastrophic code maps an infinite input weight sequence to a finite output weight sequence, which causes the code to fail. If there exists a zero-input zero-output loop, special care is needed to calculate the distance characteristics of this code. Thus, during a search for codes under periodic puncturing, we need to examine whether an encoder has a zero-output loop under the different puncturing patterns.

This paper deals with only this form of non-minimality (zero-output loop) that is of special interest during exhaustive searches. Section II discusses the appropriate search space for the code search. Section III introduces a fast algorithm to check whether an encoder has a zero-output loop under a specific puncturing pattern. Section IV proposes a method to assess the performance of the encoders containing such a loop. As an application, Section V optimizes rate $1/4$ encoders employing BPSK for Hamming weight, under both symbol-wise and bit-wise periodic puncturing with period $p = 4$. Code tables

0-7803-7097-1/01/$10.00 ©2001 IEEE

and simulation results are included. Finally Section VI concludes the paper.

II. Search space

An exhaustive search maximizing free distance over all minimal encoders of a given rate and number of memory elements yields codes with better distance properties than any non-minimal encoder with the same complexity can achieve. To achieve optimal performance under a single specific periodic puncturing pattern, an exhaustive search can safely exclude any non-minimal encoders under this pattern.

However, searching for a single code that performs well under a family of periodic puncturing patterns is a multi-criterion optimization problem [2], that does not necessarily have a unique solution. Often, no single code gives the best possible performance for all puncturing patterns. The search may instead identify a set of encoders that offer reasonable performance over the whole family of puncturing patterns. This set may include encoders that become non-minimal under one or more of the puncturing patterns.

As a result, an exhaustive search considering several puncturing patterns, cannot safely exclude encoders that do not retain minimality under all puncturing patterns. For example, when punctured to rate one (no redundancy) an encoder is always non-minimal, since the minimal encoder in this case contains no memory elements. Still such a pattern might be of interest in a partial-band jamming application.

A typical search space for trellis codes is the set of encoders that are range distinct to each other. Two encoders are called range equivalent [9] if they have the same set of output sequences, and range distinct if they are not range equivalent. This is Forney's notion of equivalence [4]. For non-minimal encoders the mapping from input to output sequences plays an important role. If the non-minimal encoder has a zero-output loop, the input sequence mapped to the zero-output loop determines whether the encoder is catastrophic or not, which has a dramatic effect in performance. Thus the set of range distinct encoders is not sufficient for an exhaustive search that considers a family of puncturing patterns.

A sufficient search space that includes all encoders of interest is an exhaustive set of encoders that are strictly distinct. Two encoders are called *strictly equivalent* [9] if they map the same input sequence to the same output sequence. If two encoders are not strictly equivalent we say they are strictly distinct. The rational form theorem in [10] provides a method for identifying an exhaustive set of strictly distinct encoders. A group theoretic approach in [11] provides a different way of identifying such a set.

III. Algorithm to determine zero-output loops under symbol puncturing

The straightforward method described in [2] to determine whether a puncturing pattern causes an encoder's state diagram to have a zero-output loop, is to start from all the states, coupled with all the the distinct phases of the puncturing pattern, and check if they belong to a zero-output loop. As phases we refer to the p possible cyclic shifts of a $p-q$ puncturing pattern.

Assume that after puncturing the state-space diagram contains a loop that has only zero outputs. This loop, before puncturing, must have at least one zero output. Otherwise, if all the loop outputs were different than zero, since they cannot all be punctured (at least one a_i is nonzero) a non-punctured output would eventually add distance to the loop. Thus the non-punctured outputs of the zero-output loop have to be zero.

This observation leads to the proposed algorithm. To check for zero-output loops, start from the states that have a zero output before puncturing, coupled with the distinct phases of the puncturing pattern such that $a_1 = 1$, i.e. the first (every p) symbol is not punctured. That is, align the zero output with a nonzero a_i.

For example, for an encoder with six memory elements under the $p - q = 4 - 2$ puncturing pattern [0 1 0 1], instead of starting from all the 64 states coupled with the two distinct phases [0 1 0 1] and [1 0 1 0], it is sufficient to start from the three states that have a zero output (don't need to include the zero-output at the self-loop around the zero state) coupled with phase [1 0 1 0]. This amounts to a reduction by approximately 97% of the search space.

IV. Performance of encoders with a zero-input zero-output loop

The performance of a code over an AWGN channel is determined by its free distance. For codes optimized for periodic puncturing, the corresponding metric is residual distance. Residual distance [2] indicates how much output distance is provided by the code after periodic attenuation. In other words, residual distance is the free distance of the punctured code.

Consider a non-minimal encoder with a zero-input zero-output loop, decoded with the standard Viterbi algorithm. The free distance is not necessarily equal to the minimum output distance associated with the encoder's error events. As error events we refer to the trellis paths of finite length that leave the zero state once and return to it only once ([12] pg. 61).

However, the free distance is equal to that of a strictly equivalent minimal encoder, and for minimal encoders free distance is the minimum distance associated with the encoder's error events. This minimal strictly equivalent encoder may also be used to compute transfer function bounds. A reduced complexity transfer function bound and a method for calculating the transfer function bound for codes under periodic erasures, are described in [2], [13].

As an example, consider the non-minimal encoder $\{\mathbf{A_1}, \mathbf{B_1}, \mathbf{C_1}, \mathbf{D_1}\}$ that contains a zero-input zero-output loop

0-7803-7097-1/01/$10.00 ©2001 IEEE

$$\mathbf{s}_{j+1} = \mathbf{s}_j \underbrace{\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix}}_{\mathbf{A_1}} + \mathbf{u}_j \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}}_{\mathbf{B_1}}$$

$$\mathbf{x}_j = \mathbf{s}_j \underbrace{\begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\mathbf{C_1}} + \mathbf{u}_j \underbrace{\begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}}_{\mathbf{D_1}},$$

where $\mathbf{s} = [s_1\ s_2\ s_3\ s_4]$, $\mathbf{u} = [u_1\ u_2\ u_3]$ and $\mathbf{x} = [x_1\ x_2\ x_3\ x_4]$. This encoder is strictly equivalent to the minimal encoder $\{\mathbf{A_2}, \mathbf{B_2}, \mathbf{C_2}, \mathbf{D_2}\}$

$$\mathbf{s}_{j+1} = \mathbf{s}_j \underbrace{\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}}_{\mathbf{A_2}} + \mathbf{u}_j \underbrace{\begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix}}_{\mathbf{B_2}}$$

$$\mathbf{x}_j = \mathbf{s}_j \underbrace{\begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}}_{\mathbf{C_2}} + \mathbf{u}_j \underbrace{\begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}}_{\mathbf{D_2}},$$

where $\mathbf{s} = [s_1\ s_2]$. The encoder $\{\mathbf{A_2}, \mathbf{B_2}, \mathbf{C_2}, \mathbf{D_2}\}$ may be used to calculate the free distance and the transfer function bounds for the encoder $\{\mathbf{A_1}, \mathbf{B_1}, \mathbf{C_1}, \mathbf{D_1}\}$.

The following theorem allows determination of the free distance of an encoder with a zero-output loop in its state diagram without converting it to a minimal equivalent encoder:

Theorem 1 *The free distance of a code with a zero-output loop, is equal to the minimum output distance of all paths that start and end in the zero state, and all paths that start in the zero state and end in the zero-output loop.*

Proof: The non-minimal encoder $\{\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}\}$ is not output observable. Thus there exists a state vector equivalence transformation $\hat{\mathbf{s}}_j = \mathbf{P}\mathbf{s}_j$, where $\mathbf{P}$ is a nonsingular matrix determined by the observability matrix $O = [\mathbf{C}\ \mathbf{CA}\ \ldots\ \mathbf{CA}^{m-1}]$ [14], that leads to an algebraically equivalent encoder of the form

$$[\mathbf{s}_o\ \mathbf{s}_{no}]_{j+1} = [\mathbf{s}_o\ \mathbf{s}_{no}]_j \underbrace{\begin{bmatrix} \mathbf{A}_o & \mathbf{0} \\ \mathbf{A}_{no1} & \mathbf{A}_{no2} \end{bmatrix}}_{\mathbf{A}} + \mathbf{u}_j \underbrace{\begin{bmatrix} \mathbf{B}_o \\ \mathbf{B}_{no} \end{bmatrix}}_{\mathbf{B}}$$

$$\mathbf{x}_j = [\mathbf{s}_o\ \mathbf{s}_{no}]_j \underbrace{\begin{bmatrix} \mathbf{C}_o \\ \mathbf{C}_{no} \end{bmatrix}}_{\mathbf{C}} + \mathbf{u}_j\mathbf{D},$$

where the m-dimensional state vector $\mathbf{s}$ is divided into an observable part $\mathbf{s}_o$ and and a not observable part $\mathbf{s}_{no}$. Dropping the not observable state vector $\mathbf{s}_{no}$, we obtain an observable state equation of lower dimension, that corresponds to a strictly equivalent observable encoder.

Assuming $\mathbf{C}$ is full rank, the zero-input zero-output loop is described by the state equations $\mathbf{s}_o = \mathbf{0}$, $\mathbf{s}_{no} = \mathbf{A}_{no2}\mathbf{s}_{no}$. That is, for the strictly equivalent observable encoder, the zero-input zero-output loop is mapped to the self-loop around the zero state. Error events that start and end in the zero state $\mathbf{s}_o = \mathbf{0}$ for the minimal encoder, may start or end in the zero loop for the non-minimal encoder. ■

V. Code Search and Simulation Results

In this section we present code tables and simulation results for rate $1/4$ codes employing BPSK with $m = 6$ memory elements, optimized for periodic puncturing with period $p = 4$. To identify good codes we examined both symbol-wise and bit-wise puncturing.

To describe an encoder we give in octal notation the feedback polynomial $f(D)$, the $k = 1$ row $\mathbf{b}_1$ of matrix $\mathbf{B}$, the $n = 4$ columns $\{\mathbf{c}_1 \ldots \mathbf{c}_4\}$ of matrix $\mathbf{C}$, and the $k = 1$ row $\mathbf{d}_1$ of matrix $\mathbf{D}$. The last row of matrix $\mathbf{A}$ contains the coefficients: $[f_0 \ldots f_5]$ of the encoders feedback polynomial $f(D) = D^6 + f_5D^5 + \ldots + f_0$. We use this notation for the codes presented in all the following tables. For example, code C_9 in Table III, described by the polynomials {0140, 040, 054, 062, 052, 013, 07} has the $\{\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}\}$ state-space description

$$\mathbf{s}_{j+1} = \mathbf{s}_j \underbrace{\begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}}_{\mathbf{A}} + \mathbf{u}_j \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}}_{\mathbf{B}}$$

$$\mathbf{x}_j = \mathbf{s}_j \underbrace{\begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\mathbf{C}} + \mathbf{u}_j \underbrace{\begin{bmatrix} 0 & 1 & 1 & 1 \end{bmatrix}}_{\mathbf{D}}.$$

A. Bit-wise Puncturing

For periodic bit puncturing an exhaustive search examined all strictly distinct rate $1/4$ feedback encoders such that, when punctured to uncoded, all resulting $1/1$ encoders are non-catastrophic. The search was restricted to feedback encoders, since all rate $1/1$ feedforward encoders are catastrophic, (with the exception of the $1/1$ codes with generator polynomials $[D^k]$, $k = 1 \ldots m$).

Table I presents three codes, each achieving the highest residual Hamming distance for a puncturing pattern, in octal notation as described previously.

TABLE I

Codes with $m = 6$ memory elements, $k = 1$ input, and $n = 4$ outputs, optimized for Hamming distance under bit-wise periodic puncturing, with period P=4.

Code	$\{f, \mathbf{b}_1, \mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3, \mathbf{c}_4, \mathbf{d}_1\}$
B_1	{0101, 040, 010, 052, 066, 077, 01}
B_2	{0123, 040, 044, 064, 011, 027, 02}
B_3	{0123, 040, 050, 066, 076, 077, 010}

Table II provides in detail the distance characteristics of the codes presented in Table I. In the second row, the d_* subscript denotes the unpunctured output bits. For example, d_{12} stands for the residual distance when the two most significant (MSB) bit outputs are not punctured and

the two least significant (LSB) outputs are punctured and d_{1234} stands for the free distance when no output bit is punctured.

The number given in parenthesis is N_b. For no puncturing N_b would be the total number of input bits of all error events that have output distance equal to the free distance, divided by the number of input bits. For codes under periodic puncturing, N_b is the natural extension of this idea, as is described in [2]. The notation "(zl)" indicates that the encoder has a zero-input zero-output loop, and the distance from the zero state to this loop determines the residual distance.

TABLE II

ANALYTIC DISTANCE PROPERTIES OF CODES IN TABLE I.

Code	d^2_{free}					
	d_{12}	d_{13}	d_{14}	d_{23}	d_{24}	d_{34}
B_1	4(7)	5(9)	4(4)	5(zl)	3(zl)	6(zl)
B_2	6(30)	2(zl)	5 (4)	6(30)	3(zl)	5(4)
B_3	5(6)	4(6)	4(zl)	4(zl)	4(zl)	5(4)

Code	d^1_{free}				d^0_{free}
	d_{123}	d_{124}	d_{134}	d_{234}	d_{1234}
B_1	8(2)	8(4)	8(4)	7(zl)	14(12)
B_2	8(4)	8(4)	8(4)	8(4)	11(4)
B_3	8(6)	10(11)	8(11)	9(4)	12(11))

B. *Symbol-wise puncturing*

For periodic symbol puncturing, we performed a partial search over a set of strictly distinct feedback encoders. Feedback encoders were chosen as more resilient to catastrophicity. As noted in [2] a catastrophic feedforward encoder may lead to a non-catastrophic range-equivalent feedback code, while a catastrophic feedback code always leads to a catastrophic range-equivalent feedforward code. The search was partial in that it did not exhaust the set of strictly distinct encoders due to computational limitations, but it included a full set of range distinct encoders.

Puncturing with period four leads to one $4-3$ pattern (0001), two $4-2$ puncturing patterns (0011 and 0101), and one $4-1$ puncturing pattern (0111), with associated minimum (among all phases) free distance denoted by d^{01}_{free}, d^{03}_{free}, d^{05}_{free} and d^{07}_{free} respectively. Puncturing to rate one (uncoded) for non-catastrophic encoders always implies distance $d^{01}_{free} = 1$. The free distance of the unpuctured code is denoted by d^{17}_{free}.

Table III presents encoders that are non-catastrophic when punctured to uncoded, and offer consistent performance under the different puncturing patterns. The metrics presented in the third and fourth column are calculated [2] as $J_{dB} = \sum_j 10\log_{10}(4(d^j_{free})^2)$ and $J_{MI} = \sum_j \frac{p-q_j}{p}\log_2(4(d^j_{free})^2)$, where the summation is over all puncturing patterns $\tilde{\mathbf{a}}_j$, $4(d^j_{free})^2$ is the squared Euclidean distance corresponding to Hamming distance d_j and BPSK constellation normalized to unit energy ($E_s = 1$), and q_j is the number of punctured symbols for the $p - q_j$ puncturing pattern $\tilde{\mathbf{a}}_j$. Table IV gives the residual distance characteristics of these codes. L_D is the natural extension of traceback depth for codes under periodic erasures. Both N_b and L_D are described in detail in [2].

TABLE III

CODES WITH $m = 6$ MEMORY ELEMENTS, $k = 1$ INPUT, AND $n = 4$ OUTPUTS, OPTIMIZED FOR HAMMING DISTANCE UNDER SYMBOL-WISE PERIODIC PUNCTURING, WITH PERIOD P=4.

C	$\{f, \mathbf{b}_1, \mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3, \mathbf{c}_4, \mathbf{d}_1\}$	J_{MI}	J_{dB}
C_1	$\{170, 40, 66, 56, 13, 33, 17\}$	70.3	15.21
C_2	$\{140, 40, 42, 16, 11, 25, 17\}$	70.3	15.21
C_3	$\{140, 40, 52, 06, 65, 15, 74\}$	70.3	15.21
C_4	$\{113, 40, 24, 74, 22, 65, 64\}$	70.8	15.25
C_5	$\{151, 40, 40, 60, 76, 07, 13\}$	70.8	15.25
C_6	$\{124, 60, 64, 22, 06, 43, 16\}$	70.8	15.25
C_7	$\{136, 60, 40, 62, 01, 27, 7\}$	70.8	15.25
C_8	$\{160, 40, 74, 16, 41, 71, 15\}$	70.8	15.25
C_9	$\{140, 40, 54, 62, 52, 13, 7\}$	70.9	15.26
C_{10}	$\{140, 40, 12, 56, 35, 63, 17\}$	70.6	15.24
C_{11}	$\{120, 40, 64, 06, 43, 67, 16\}$	70.5	15.23
C_{12}	$\{140, 40, 26, 45, 65, 13, 17\}$	70.5	15.23
C_{13}	$\{174, 40, 15, 55, 07, 47, 16\}$	70.5	15.23
C_{14}	$\{110, 40, 45, 65, 63, 07, 15\}$	69.5	15.09
C_{15}	$\{146, 40, 24, 42, 62, 11, 17\}$	69.5	15.09
C_{16}	$\{150, 60, 25, 65, 37, 77, 15\}$	69.5	15.09

TABLE IV

RESIDUAL DISTANCE FOR CODES IN TABLE III UNDER $p = 4$.

Code	1111			0111		
	d_{free}	N_b	L_D	d^{07}_{free}	N_b	L_D
C_1	20	71	11	11	4	9
C_2	20	40	10	11	2	9
C_3	20	24	10	11	2	9
C_4	19	38	9	11	6	9
C_5	19	38	9	11	6	9
C_6	19	32	9	11	2	9
C_7	19	40	9	11	4	9
C_8	19	24	9	11	4	9
C_9	18	8	8	12	2	9
C_{10}	18	8	7	13	20	13
C_{11}	19	40	10	12	10	10
C_{12}	19	24	11	12	14	13
C_{13}	19	40	11	12	20	13
C_{14}	19	24	10	13	32	13
C_{15}	19	16	7	13	42	13
C_{16}	19	24	10	13	29	13

Code	0011			0101		
	d^{03}_{free}	N_b	L_D	d^{05}_{free}	N_b	L_D
C_1	6	12	10	8	59	15
C_2	6	10	10	8	101	15
C_3	6	28	17	8	99	15
C_4	7	32	11	8	32	12
C_5	7	33	11	8	35	12
C_6	7	49	14	8	68	15
C_7	7	41	14	8	63	15
C_8	7	44	14	8	64	15
C_9	7	18	11	8	28	13
C_{10}	6	8	10	8	71	17
C_{11}	6	4	10	8	55	17
C_{12}	6	8	13	8	48	15
C_{13}	6	9	13	8	57	15
C_{14}	5	2	10	7	12	13
C_{15}	5	6	10	7	8	13
C_{16}	5	4	10	7	12	13

The codes in Table IV achieve much higher residual Hamming distance than the codes in Table II: for $4-0$ puncturing (unpunctured) they achieve distance up to 20 as opposed to 14, for $4-1$ puncturing they achieve distance up to 14 as opposed to 8, and for $4-2$ puncturing they achieve distance up to 8 as opposed to 4. The reason is that the set of non-catastrophic codes under bit-wise puncturing is much smaller than the set of non-catastrophic codes under symbol-wise puncturing. The limited number of non-catastrophic codes incurs much smaller achievable distance for the rest of the puncturing patterns.

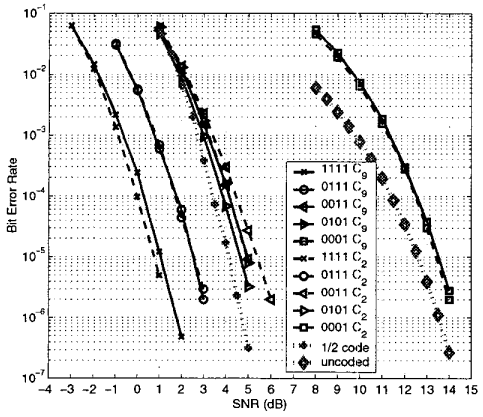

Fig. 1. Performance of codes C_2 and C_9 under periodic symbol puncturing with period $p=4$, over AWGN channel.

Fig. 1 plots the performance of codes C_2 and C_9 in BER vs. SNR (in dB), for all possible periodic puncturing patterns. Code C_2 offers better performance when unpunctured (pattern 1111), and code C_9 better performance for the $4-2$ puncturing patterns (0101 and 0011). The same figure plots the performance of uncoded BPSK, and of the best rate-$1/2$ code with six memory elements. The Viterbi decoder used traceback depth $L_D=40$.

Fig. 2 plots BER vs. mutual information (in bits per channel use) for the code C_2. Mutual information is calculated as $\frac{p-q}{p}\log_2(1+\mathrm{SNR})$ for a $p-q$ puncturing pattern, This plot allows one to examine proximity to capacity for each puncturing pattern. Code C_2 for BER=10^{-5} requires a consistent excess mutual information between $0.77-0.885$ for all five erasure patterns. This amount of excess mutual information is very similar to the one required in [2] for rate-$1/3$ 8-PSK codes.

VI. Conclusions

A minimal encoder when periodically punctured leads to a higher rate encoder that may or may not be minimal. This paper, discusses the appropriate search space for codes optimized to offer consistent performance under periodic puncturing. We propose a fast algorithm to determine whether an encoder under a specific puncturing pattern has a zero-output loop, and introduce a method to

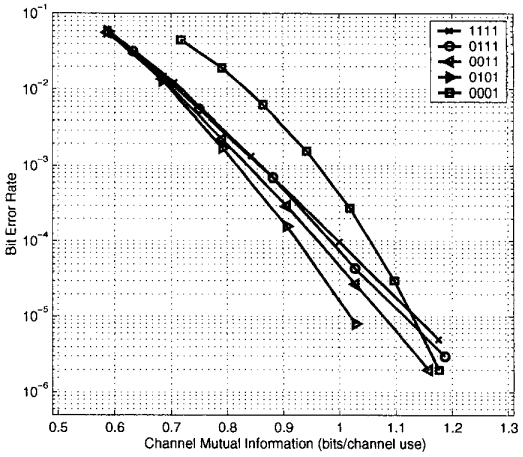

Fig. 2. Performance of code C_2 vs. mutual information under periodic symbol puncturing with period $p=4$. Transmitted rate is 0.25 bits per channel use.

assess the performance of codes containing such a loop. The paper also provides simulation results and code tables for rate-$1/4$ codes optimized for Hamming weight under both bit-wise and symbol-wise puncturing.

References

[1] A. Lapidoth. The performance of convolutional codes on the block erasure channel using various finite interleaving techniques. *IEEE Transactions on Info. Theory*, 40:1459–1473, September 1994.

[2] R. D. Wesel, X. Liu, and W. Shi. Trellis codes for periodic erasures. *IEEE Transactions on Communications*, 48(6):938–947, June 2000.

[3] R. D. Wesel, X. Liu, and W. Shi. Periodic symbol puncturing of trellis code. In *Thirty-first Asilomar conference on signals, systems and computers*, Nov. 1997.

[4] G. D. Forney. Convolutional codes I: algebraic structure. *IEEE Transactions on Info. Theory*, 26(6):720–738, November 1970.

[5] Hans-Andrea Loeliger, G. D. Forney, Thomas Mittelholzer, and Mitchell D. Trott. Minimality and observability of group systems. *Linear Algebra And Its Applications*, 205-206:937–963, July 1994.

[6] G. D. Forney, Rolf Johannesson, and Zhe-xian Wan. Minimal and canonical rational generator matrices for convolutional codes. *IEEE Transactions on Info. Theory*, 42(6):1865–1880, November 1996.

[7] C. Fragouli and R. D. Wesel. Convolutional codes and control theory. In *ComCon99*, Athens, Greece, June 1999.

[8] H. A. Loeliger and T. Mittelholzer. Convolutional codes over groups. *IEEE Transactions on Info. Theory*, 42:1660–1686, Nov. 1996.

[9] R. D. Wesel. Trellis Code Design For Correlated Fading And Achievable Rates For Tomlinson-Harashima Precoding. *PhD Dissertation*, August 1996.

[10] C. Fragouli and R. D. Wesel. Turbo encoder design for symbol-interleaved parallel concatenated trellis-coded modulation. In *IEEE Transactions on Communications*, March 2001.

[11] S.B enedetto , R. Garello, and G. Montorsi. A search for good convolutional codes to be used in the construction of turbo codes. *IEEE Transactions on Communications*, 46(9):1101–1105, September 1998.

[12] E. Biglieri, D. Divsalar, P. J. McLane, and M. Simon. *Introduction to Trellis-Coded Modulation.* Macmillan Publishing Company, New York, 1991.

[13] R. D. Wesel. Reduced complexity transfer function computation. In *ICC99*, Vancouver, Canada, June 1999.

[14] Chi-Tsong Chen. Linear System Theory and Design. *Oxford University Press*, 1999.

0-7803-7097-1/01/$10.00 ©2001 IEEE