Draft of 16 Jul 1998 3:34 p.m.

On the Design of Generalized Concatenated Coding Systems with Interleavers

Dariush Divsalar and Robert J. McEliece

Abstract.

In this paper we will study a class of concatenated coded communications systems which are built from convolutional codes and interleavers. These systems include as special cases both the classical tubo codes [1][2] and the serial concatentation of convolutional codes [4]. Our main results are a method for computing a conjectured interleaving gain exponent and for optimizing the effective free distance d_2. We include proofs of theorems about d_2 which were stated without proof in [6].

1. Introduction.

Our goal in this paper is to introduce a class of generalized concatenated convolutional coding systems with interleavers, which includes both parallel concatenated convolutional codes (turbo codes) [1][2] and serial concatenated convolutional codes [4] as special cases. We estimate the average (over all interleavers) performance of these codes using the classical union bound technique, which leads us to two important design parameters: the interleaving gain exponent α_M and the asymptotic coding gain exponent h_M. We shall see that these parameters are relatively easy to compute, and so they can serve the design engineer as preliminary figures of merit.

Here is an outline of the paper. In Section 2, we will briefly review the union bound, as it applies to block codes, and see the importance of the input-output weight enumerator (IOWE). In Section 3, we will introduce our generalized concatenation structure and give a formula for the average (over the ensemble of all interleavers) IOWE when the component codes are block codes. In Section 4, we will replace the block codes of Section 3 with truncated convolutional codes, and study the parameters α_M and h_M mentioned above. In particular, we will show that for both parallel and serial concatenation, h_M is maximized if the effective free distance d_2 of certain of the component codes is maximized. Then in Section 5, we will state and prove two uppper bounds on d_2 which can be used to identify component codes with optimal or near-optimal values of d_2. Finally in Section 6, we present tables of convolutional codes for which d_2 is as large as possible.

2. Union Bounds on the Performance of Block Codes.

Consider a linear (n, N) block code C with code rate $R_c = N/n$ and minimum distance h_m. The union bounds on the word error probability P_W, and bit-error probability P_b of the code C over a memoryless binary-input channel, using maximum likelihood decoding, have the well-known form

$$P_W \leq \sum_{h=h_m}^{n} \sum_{w=1}^{N} A_{w,h} z^h \tag{2.1}$$

$$P_b \leq \sum_{h=h_m}^{n} \sum_{w=1}^{N} \frac{w}{N} A_{w,h} z^h, \tag{2.2}$$

where $A_{w,h}$ denotes the number of codewords in C with input weight w and output weight h. $A_{w,h}$ is called the *input–output weight coefficient* (IOWC). The function z^h represents an upper bound on the pairwise error probability for two codewords separated by Hamming (output) distance h. For AWGN channels we have $z = e^{-R_c E_b/N_0}$ where E_b/N_0 is the signal-to-noise ratio per bit. For independent Rayleigh fading channels, assuming coherent detection, and perfect estimate of fading samples, we have $z = [1 + R_c E_b/N_0]^{-1}$. All these results apply to convolutional codes as well, if we construct an equivalent block code from the convolutional code. These results also apply to concatenated codes including parallel [1][2] and serial [4] concatenations and other types of code concatenations. As soon as we obtain the input–output weight coefficients $A_{w,h}$ for a particular concatenated code, we can compute its performance.

3. Computation of $A^C_{w,h}$ for Concatenated Codes with Random Interleavers.

In this section, we consider a general class of concatenated coding systems of the type depicted in Figure 1, with q encoders (circles) and $q - 1$ interleavers (boxes). The ith code C_i is an (n_i, N_i) linear block code, and the ith encoder is preceded by an interleaver (permuter) P_i of size N_i, except C_1 which is not preceded by an interleaver, but rather is connected to the input. The overall structure must have no loops, i.e., it must be a graph-theoretic tree.

Define $s_q = \{1, 2, \ldots, q\}$ and subsets of s_q by $s_I = \{i \in s_q : C_i \text{ connected to input}\}$, $s_O = \{i \in s_q : C_i \text{ connected to Channel }\}$, and its complement $\overline{s_O}$. The overall system depicted in Figure 1 is then an encoder for an (n, N) block code with $n = \sum_{i \in s_O} n_i$.

If we know the IOWC's $A^{(i)}_{w_i,h_i}$'s for the constituent codes C_i, we can calculate the *average* IOWC's $A_{w,h}$ for the overall system (averaged over the ensemble of all possible interleavers), using the **uniform interleaver** technique [2]. (A uniform interleaver is defined as a probabilistic device that maps a given input word of weight w into all distinct $\binom{N_i}{w}$ permutations of it with equal probability $p = 1/\binom{N_i}{w}$.) The result is

$$A_{w,h} = \sum_{\substack{h_i : i \in s_O \\ \Sigma h_i = h}} \sum_{h_i : i \in \overline{s}_O} A^{(1)}_{w_1,h_1} \prod_{i=2}^{q} \frac{A^{(i)}_{w_i,h_i}}{\binom{N_i}{w_i}} \tag{3.1}$$

In (3.1) we have $w_i = w$ if $i \in s_I$, and $w_i = h_j$ if C_i is preceeded by C_j (see Figure 2.). For example, for the $(n_2 + n_3 + n_4, N)$ encoder of Figure 1 the formula (3.1) becomes

$$A_{w,h} = \sum_{\substack{h_1,h_2,h_3,h_4 \\ (h_2+h_3+h_4=h)}} A^{(1)}_{w_1,h_1} \frac{A^{(2)}_{w_2,h_2}}{\binom{N_2}{w_2}} \frac{A^{(3)}_{w_3,h_3}}{\binom{N_3}{w_3}} \frac{A^{(4)}_{w_4,h_4}}{\binom{N_4}{w_4}} = \sum_{\substack{h_1,h_2,h_3,h_4 \\ (h_2+h_3+h_4=h)}} A^{(1)}_{w,h_1} \frac{A^{(2)}_{w,h_2}}{\binom{N}{w}} \frac{A^{(3)}_{h_1,h_3}}{\binom{n_1}{h_1}} \frac{A^{(4)}_{h_1,h_4}}{\binom{n_1}{h_1}}.$$

(The formula (3.1) is intuitively plausible if we note that the term $A^{(i)}_{w_i,h_i}/\binom{N_i}{w_i}$ is the probability that a random input word to C_i of weight w_i will produce an output word of weight h_i.)

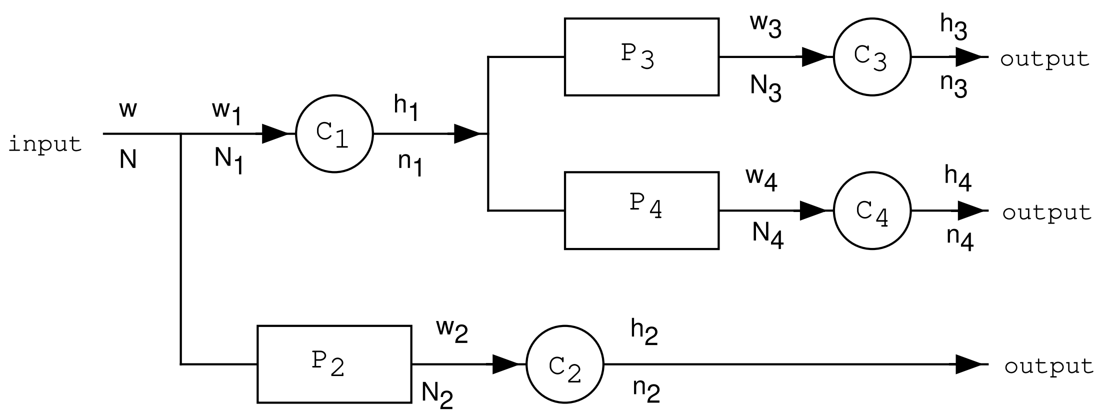

Figure 1. Example of concatenated codes with tree structure $s_I = \{1, 2\}$, $s_O = \{2, 3, 4\}$, $\overline{s}_O = \{1\}$.

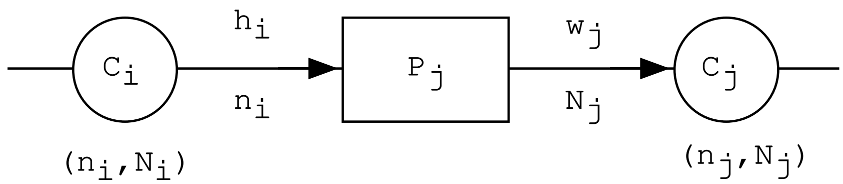

Figure 2. C_i (an (n_i, N_i) encoder) is connected to C_j (an (n_j, N_j) encoder) by an interleaver of size N_j. We have the "boundary conditions" $N_j = n_i$ and $w_j = h_i$.

4. Design of Concatenated Codes.

In this section we will consider systems of the form depicted in Figure 1, in which the individual encoders are truncated convolutional encoders, and study the behavior of the average IOWC's in formula (3.1) as the input block length N approaches infinity. To this end we define, for each fixed w and h,

$$\alpha(w,h) = \lim_{N\to\infty} \log_N A^C_{w,h}. \tag{4.1}$$

Two fundamental design parameters that we shall study here are

$$\alpha_M = \max_h \max_w \alpha(w,h), \tag{4.2}$$

and

$$h_M = \operatorname*{argmax}_h \max_w \alpha(w,h). \tag{4.3}$$

Extensive numerical simulations, and theoretical considerations that are not fully rigorous (see e.g. [2] and [4]) lead us to make the following conjecture about the behavior of the union bounds (2.1) and (2.2) for systems of the type shown in Figure 1, which we denote by P_W^{UB} and P_b^{UB}.

Conjecture. *There exists a positive number z_0, which depends on the q component convolutional codes and the tree structure of the overall system, but not on N, such that for any fixed $z < z_0$,*[1] *as the block length N becomes large,*

$$P_W^{UB} = O(N^{\alpha_M}) \tag{4.4}$$

$$P_b^{UB} = O(N^{\alpha_M - 1}). \tag{4.5}$$

[1] Because of the inherent properties of the union bound, z_0 will necessarily be less than the "cutoff rate," which for a binary code of rate r is $z_{\mathrm{cutoff}} = 2^{1-r} - 1$.

Eq. (4.4) says that if $\alpha_M < 0$, then for a given $z < z_0$ the *word* error probability of the concatenated code decreases to zero as the input block size is increased. Eq. (4.5) says that if the weaker condition $\alpha_M < 1$ is satisfied, the *bit* error probability dereases to zero as the input block size is increased. When $\alpha_M < 0$ we say we have *word error probability interleaving gain*, and when $\alpha_M < 1$ we say we have *bit error probability interleaving gain.* Furthermore, for N fixed, the parameter h_M defined in (4.3) determines the rate at which P_W approaches zero as $z \to 0$ ($E_b/N_0 \to \infty$). In designing a system of the type depicted in Figure 1, our primary concern will be to minimize α_M, and for a given value of α_M, our secondary concern will be to maximize h_M.

We shall now discuss the calculation of $\alpha(w, h)$, α_M, and h_M for a concatenated system of the type depicted in Figure 1, assuming that each of the encoders is a convolutional encoder. We use analytical tools introduced in [3], and [4].

Consider a (n, k, m) convolutional code C with rate $R = k/n$ and memory m, and the corresponding sequence of $(N/R, N - m)$ block codes $C^{(N)}$, for $N = k, 2k, \ldots$, obtained by truncating the convolutional code at depths $L = 1, 2, \ldots$. By definition, $C^{(N)}$ consists of the output labels of the trellis paths from the all-zeros state back to the all-zeros state of length N/k. Let $A^C_{w,h,j}$ be the input-output weight coefficients for those codepaths in the convolutional code C consisting of the concatenation of exactly j simple codepaths, with no gaps between them (see Figure 3). We can "expand" each codepath of the type depicted in Figure 3 into several codewords from $C^{(N)}$ by inserting zero-runs before each of the j simple codepaths, so that the overall path length is N/k. If the ith simple codepath starts in position l_i, then clearly

$$1 \le l_1 < l_2 < \cdots < l_j \le N/k. \tag{4.6}$$

Since there are $\binom{N/k}{j}$ set of indices satsfying (4.6), we have the following bound on the IOWE of the code $C^{(N)}$.

$$A^{(N)}_{w,h} \le \sum_{j \ge 1} \binom{N/k}{j} A^C_{w,h,j}. \tag{4.7}$$

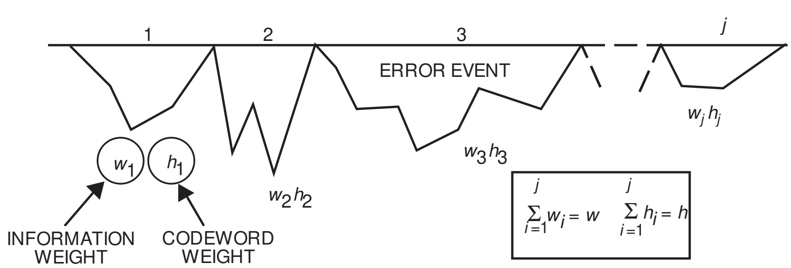

Figure 3. A code sequence in the convolutional code C consisting of the concatenation of j simple codepaths. The IOWC for such codepaths is denoted by $A^C_{w,h,j}$.

Since in (4.1), w and h are fixed and $N \to \infty$, we use the estimate

$$\binom{N}{j} \approx \frac{N^j}{j!}. \tag{4.8}$$

Substituting this estimate into (4.7) we obtain

$$A_{w,h}^{(N)} \approx \sum_{j=1}^{n_M} \frac{N^j}{j!k^j} A_{w,h,j}^{C}. \tag{4.9}$$

Using the estimate in (4.9), and (4.8) in (3.1) for each constituent code, we obtain

$$\alpha(w,h) = \max \left\{ \sum_{i=1}^{q} j_i(w_i,h_i) - \sum_{i=2}^{q} w_i : \sum_{i \in S_O} h_i = h \right\}, \tag{4.10}$$

where $j_i = j_i(w,h)$ is defined by

$$j_i(w,h) = \max_j \{ j : A_{w,h,j}^{C_i} \neq 0 \}.$$

Next we discuss the computation of the quantities $j_i(w_i,h_i)$. For any convolutional code C, given a j-segment code sequence of the form depicted in Figure 3, we have $w_1 + \cdots + w_j = w$ and $h_1 + \cdots + h_j = h$. Thus if w_m denotes the minimum possible nonzero input weight which produces a finite-weight codeword, we have $jw_m \leq w$, so that $j \leq w/w_m$. Similarly, if h_m denotes the minimum nonzero codeword weight, we have $jh_m \leq h$, so that $j \leq h/h_m$. Thus we have

$$j_C(w,h) \leq \min\{ \left\lfloor \frac{w}{w_m} \right\rfloor, \left\lfloor \frac{h}{h_m} \right\rfloor \}. \tag{4.11}$$

Note that if C is nonrecursive we have $w_m = 1$, whereas if C is recursive, $w_m = 2$. Also, h_m is the free distance of the code.

Our object is to select the component codes so as to minimize α_M. This will guarantee that for large N, the bounds in (2.1) and (2.2) will be as small as possible. Since in (4.10) for maximization there is no constraint on h, the contributions to h comes from those constituent codes connected to the channel. Therefore for α_M in (4.10) to be a minimum, at least those codes connected to the channel should be recursive. Thus maximizing the minimum output weights for input weight 2 sequences is important since it maximizes h_M.

For example, consider the parallel concatenation of q codes, with $q-1$ interleavers. If each of the component codes is recursive, then $j_i(w_i,h_i) \leq w_i/2$ and $w_i = w$, for $i = 1, \ldots, q$, and so by (4.11),

$$\alpha(w,h) \leq q[\frac{w}{2}] - (q-1)w.$$

Since this bound is a decreasing function of $w \geq 2$, its maximum occurs for $w = 2$, so that

$$\alpha_M \leq -q + 2.$$

(If any of the component codes is nonrecursive, a larger value of α_M results). For a classical turbo code with $q = 2$, we have $\alpha_M = 0$, so there is no word error probability interleaving gain. This suggests that the word error probability for classic turbo codes with a uniform interleaver will not improve with input block size, a fact that has been verified by numerical evaluations of the union bound. Thus when all codes are recursive, to maximize the corresponding h_M, we should maximize d_2, which is defined to be the minimum output weight for weight 2 input sequence for each constituent code.

As another example, consider the serial concatenation of two convolutional codes. If the inner code is recursive then we obtain from (4.10) $\alpha(w,h) = j^o + j^i - w^i$, where "$o$" denotes the outer code and "i" denotes the inner code. But by (4.11),

$$j^o + j^i - w^i \leq \lfloor \frac{h^o}{{d_{\text{free}}}^o} \rfloor + \lfloor \frac{h^o}{2} \rfloor - h^o,$$

where h^o is the output weight of the outer encoder. This is a decreasing function of $h^0 \geq {d_{\text{free}}}^o$, so the minimum occurs at $h^0 = {d_{\text{free}}}^o$, and so we have

$$\alpha(w,h) \leq - \left\lfloor \frac{{d_{\text{free}}}^o + 1}{2} \right\rfloor + 1$$

Therefore for serial concatenated codes if $d_f^o \geq 3$ there is interleaving gain for both bit and word error probabilities. (If the inner code is nonrecursive $\alpha_M \geq 0$ and there is no interleaving gain.)

Define d_2^i to be the minimum weight of codewords of the inner code generated by weight-2 input sequences. We obtain a different values for h_M for even and odd values of d_f^o. Then using arguments similar to those leading to (4.11), we find that for even d_f^o, the output weight h_M associated with the highest exponent of N is given by

$$h_M = \frac{d_f^o d_2^i}{2}, \tag{4.12}$$

whereas for d_f^o odd, the value of h_M is given by

$$h_M = \frac{(d_f^o - 3) d_2^i}{2} + d_3^i, \tag{4.13}$$

where d_3^i is the minimum weight of sequences of the inner code generated by a weight-3 input sequence. Thus among inner codes with maximum d_2, we should choose those with maximum d_3. Therefore for serial concatenated codes if $d_f^o \geq 3$ there is interleaving gain for both bit and word error probabilities.

It follows from the arguments in this section that in choosing the convolutional code fragments in Figure 1, it is wise to choose those connected to the output to have feedback, and to have the largest possible value of d_2. In the next section (Section 5), we will state and prove two theoretical results about the maximum possible value of d_2 for an IIR (infinite impulse response) convolutional code fragment with given values of n, k, and m. Then in Section 6, we will present some tables of IIR code fragments for which d_2 has been maximized, and, for a given maximum value of d_2, d_3 is maximized.

5. Two Upper Bounds on d_2.

In this section we will prove two theorems about the quantity "d_2" referred to in the previous section. (These theorems were stated without proof in [6].) We begin with some definitions.

An (r, k, m) binary convolutional code fragment[2] is defined [8] as the set of all output sequences of r-vectors $(x_0, x_1, \ldots,)$ which can be produced by a fixed (r, k, m) convolutional encoder. An (r, k, m) convolutional encoder, in turn, is a linear sequential finite-state machine defined by four matrices A, B, C, and D, with dimensions

$$\begin{aligned} A &: m \times m \\ B &: k \times m \\ C &: m \times r \\ D &: k \times r. \end{aligned}$$

When the (A, B, C, D) encoder is presented with a sequence of k-dimensional input vectors $u_0, u_1, \ldots$, in response it moves through a sequence of m-dimensional state vectors and produces a sequence of r-dimensional output vectors. The s_j's and the x_j's are defined by the state-space equations

$$\begin{aligned} s_{j+1} &= s_j A + u_j B \\ x_j &= s_j C + u_j D, \end{aligned} \tag{5.1}$$

for $j = 0, 1, \ldots$. (The initial state s_0 is defined to be 0.)

An output sequence $\mathbf{x} = (x_0, x_1, \ldots)$ is sometimes called a *codeword*. If the input sequence (u_j) and the state sequence (s_j) both become zero for all sufficiently large values of j, i.e, if $u_j = 0$ and $s_j = 0$ for all $j \geq L$, then because of the linearity of the state-space equations (5.1), it follows that $x_j = 0$ for all $j \geq L$. In that case the sequence $\mathbf{x} = (x_0, x_1, \ldots, x_{L-1})$ is called a *finite codeword*.

5.1 Definition. *If $\mathbf{x}$ is a binary vector or a finite sequence of binary vectors, the symbol $|\mathbf{x}|$ denotes the Hamming weight of $\mathbf{x}$, i.e., the number of nonzero components of $\mathbf{x}$. If $\mathbf{u}$ is the input to the encoder and $\mathbf{x}$ is the corresponding output, the codeword $\mathbf{x}$ is said to have* input weight *$|\mathbf{u}|$ and* output weight *$|\mathbf{x}|$.*

5.2 Definition. *For $i \geq 1$, the input-weight-i minimum distance of the (A, B, C, D) code fragment is defined as*

$$d_i = \min\{|\mathbf{x}| : |\mathbf{u}| = i\}, \tag{5.2}$$

where in (5.2), $\mathbf{u}$ is an input to the encoder and $\mathbf{x}$ is the corresponding output.

Here are our two main results.

---

[2] We use the term *fragment* to include the case of output dimension less than input dimension, i.e., $r \leq k$, which commonly occurs in the construction of turbo-like codes.

5.3 Theorem. *If $\mathcal{C}$ is a binary (r, k, m) convolutional code fragment, then*

$$d_2 \le \min\left(\left\lceil \frac{2^m}{k} \right\rceil r, 2r + \left\lfloor \frac{2^{m-1}r}{k} \right\rfloor\right). \tag{5.3}$$

5.4 Theorem. *If $\mathcal{C}$ is a binary $(r, 1, m)$ convolutional code fragment with $m \ge 2$, then*

$$d_2 \le (2 + 2^{m-1})r. \tag{5.4}$$

Equality holds in (5.4) if and only if the a canonical generator matrix for the code fragment is of the form

$$G(D) = \left(\frac{P_1(D)}{Q(D)}, \dots, \frac{P_r(D)}{Q(D)}\right),$$

where $Q(D)$ is a primitive polynomial of degree m, and $P_1(D), \dots, P_m(D)$ are polynomials of degree m, each having constant term 1, none of which are equal to $Q(D)$.

The proof is organized as follows. In section 2.1, we define the *state diagram* and the *zero-input state diagram* for a given (A, B, C, D) encoder. Then in section 2.2, we give a (state-diagram) characterization of input weight 2 codewords. In section 2.3, we give the heart of the proof of Theorems 5.3 and 5.4, in the case where the matrix A is nonsingular. Then in section 2.4 we handle the case of singular A. Then in sections 2.5 and 2.6 we put the pieces together and complete the proofs of the two main theorems. (We have relegated the proofs of two technical algebraic results to Appenidix A.)

5.1. The State Diagram and the Zero-Input State Diagram.

The *state diagram* corresponding to the (A, B, C, D) encoder defined in (5.1) is a finite, directed, labelled graph with 2^m vertices and 2^{m+k} (doubly-labelled) edges. The vertices are (labelled by) the 2^m possible states. If s is a state, and if u is an arbitrary k-vector, then there is a directed edge from s to $s' = sA + uB$ labelled (u, x), where $x = sC + uD$. Symbolically, we write

$$s \xrightarrow[x]{u} s'$$

where

$$\begin{aligned} s' &= sA + uB \\ x &= sC + uD. \end{aligned}$$

The *zero-input (ZI) subdiagram* of the state diagram has the same set of vertices as the full state diagram, but includes only those edges produced by inputs of the form $u = 0$. Thus the ZI state diagram has 2^m vertices and 2^m edges, symbolically written as

$$s \xrightarrow[x]{\mathbf{0}} s'$$

where

$$\begin{aligned} s' &= sA \\ x &= sC. \end{aligned}$$

For example, there is a simple $(2,1,2)$ convolutional code defined by the four matrices

$$A = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}, \; B = (\, 1 \quad 0\,), \; C = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}, \; D = (\, 1 \quad 1\,), \tag{5.1.1}$$

whose state diagram and zero-input state diagram are shown in Figures 4 and 5. (See also [8], Example 2.5.)

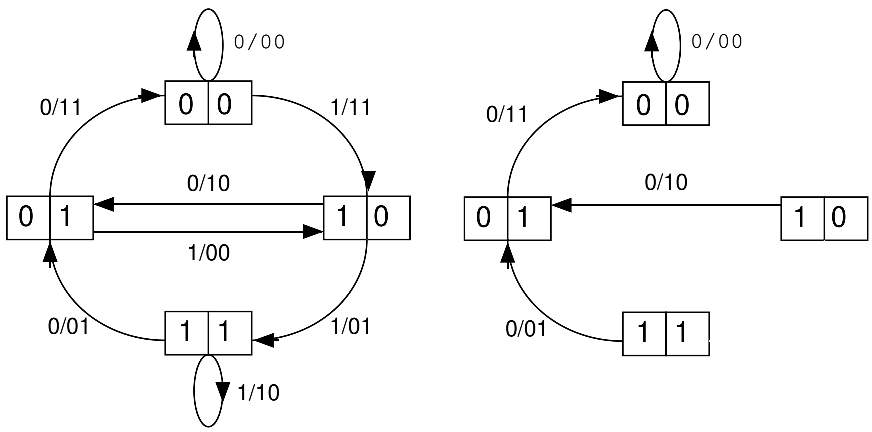

Figure 4. The state diagram and the zero-input state diagram for the code defined in (5.1.1).

Here is a simple lemma we will need later on.

5.1.1 Lemma. *The total output weight of the 2^m zero-input edges is at most 2^{m-1}. Equality holds if and only if each of the r columns of C has at least one nonzero entry.*

Proof: The set of output labels on the zero-input edges is exactly the set of 2^m vectors x of the form $x = sC$, where s is an arbitrary binary m-vector. The ith component of sC is $s \cdot c_i$, where c_i is the ith column on C. But the mapping $s \to s \cdot c$ achieves the values 0 and 1 equally often, except when c is identically 0. The result now follows. ▪

A viewpoint we shall find useful is that of a *codepath* in the state diagram. A codepath is defined to be a path of finite length in the state diagam which begins and ends in the zero state. A typical codepath can be represented as follows:

$$s_0 = \mathbf{0} \xrightarrow[x_0]{u_0} s_1 \xrightarrow[x_1]{u_1} s_2 \xrightarrow[x_2]{u_2} \cdots s_{L-1} \xrightarrow[x_{L-1}]{u_{L-1}} s_L = \mathbf{0}. \tag{5.1.2}$$

Note that the codepaths of length L are in one-to-one correspondence with the codewords of length L, the codeword associated with the codepath (5.1.2) being $(x_0, \ldots, x_{L-1})$.

5.2. Some General Results on d_2.

5.2.1 Theorem. *For a convolutional code fragment, $d_1 < \infty$ if and only if for some row b_i of B,*

$$b_i A^L = 0$$

for some nonnegative integer L.

Proof: A finite codepath of input weight 1 must be of the form

$$\mathbf{0} \xrightarrow[x_0]{e_i} b_i \xrightarrow[x_1]{\mathbf{0}} b_i A \cdots \xrightarrow[x_{L-1}]{\mathbf{0}} b_i A^L = \mathbf{0},$$

where e_i is a k-vector of weight 1, with the "1" in the ith position. ▪

The next theorem characterizes the codepaths of input weight 2.

5.2.2 Theorem. *The codepaths of input weight 2 are in one-to-one correspondence with the equations of the form*

$$b_i A^{t_i} = b_j A^{t_j}, \tag{5.2.1}$$

where b_i and b_j are rows of B and t_i and t_j are nonnegative integers. (In (5.2.1), we assume $(i, t_i) \neq (j, t_j)$.)

Proof: An input sequence of weight 2 which produces a finite codepath must be either of the form

$$(e_i, 0, \ldots, 0, e_j, 0, \ldots, 0),$$

or else of the form

$$(e_i + e_j, 0, \ldots, 0).$$

In the first case the corresponding state sequence is

$$\begin{aligned} \mathbf{0} \xrightarrow{e_i} b_i \xrightarrow{\mathbf{0}} b_i A \cdots \xrightarrow{\mathbf{0}} b_i A^{t-1} \xrightarrow{e_j} b_i A^t + b_j \\ \cdots \xrightarrow{\mathbf{0}} (b_i A^t + b_j) A^s = \mathbf{0}. \end{aligned}$$

From this it follows that $b_i A^{t+s} = b_j A^s$, as required.

In the second case the state sequence is

$$\mathbf{0} \xrightarrow{e_i + e_j} b_i + b_j \xrightarrow{\mathbf{0}} \cdots \xrightarrow{\mathbf{0}} (b_i + b_j) A^t = \mathbf{0},$$

which is equivalent to $b_i a^t = b_j A^t$. ▪

5.2.3 Theorem. *Suppose that b_i and b_j are rows of B and that (5.2.1) holds, for two pairs $(i, t_i) \neq (j, t_j)$. Then*

$$d_2 \leq r \cdot (\max(t_i, t_j) + 1).$$

Proof: From Theorem 5.2.2, we see that (5.2.1) implies the existence of an input weight 2 codpath of length $\max(t_i, t_j) + 1$. Naturally the output weight of such a codepath cannot exceed $r \cdot (\max(t_i, t_j) + 1)$. ▪

5.2.4 Corollary. *For any (r, k, m) convolutional code fragment with $d_1 = \infty$,*

$$d_2 \leq \left\lceil \frac{2^m}{k} \right\rceil r.$$

Proof: If $\{b_1, \ldots, b_k\}$ are the rows of B, consider the set of $k(L+1)$ states of the form $b_i A^j$, for $i = 1, \ldots, k$, and $j = 0, \ldots, L$. Each of these states is in the rowspace of A, and nonzero by Theorem 5.2.1, but since A is $m \times m$, this rowspace contains at most $2^m - 1$ nonzero elements. Thus if $k(L+1) \geq 2^m$, the pigeon-hole principle implies that two of these states must be equal, i.e., there is an equation of the form $b_i A^s = b_j A^t$, with $\max(s, t) \leq L$. Then by Theorem 5.2.3, $d_2 \leq (L+1)r$. Taking $L + 1 = \lceil 2^m/k \rceil$, we obtain the stated result. ▪

5.3. Nonsingular A.

In this section we will assume that the matrix A is nonsingular.

5.3.1 Theorem. *If the matrix A is nonsingular,*

$$d_2 \leq 2r + \left\lfloor \frac{2^{m-1}r}{k} \right\rfloor. \tag{5.3.1}$$

Proof: Let the k rows of B be denoted by $\mathcal{B} = \{b_1, \ldots, b_k\}$. If two rows of B are equal, e.g., $b_i = b_j$, then by Theorem 5.2.3, $d_2 \leq r$, and so (5.3.1) is certainly satisfied. Thus from now on, we assume the rows of B to be distinct.

If b_i is the ith row of B, we denote by t_i the least nonnegative integer t such that $b_i A^t \in \mathcal{B}$. (Such an integer must exist; e.g. $b_i A^n = b_i$ where n is the period of A.) If $b_i A^{t_i} = b_j$, we write $j = \pi(i)$. We now associate with b_i the following input-weight 2 codepath, which we call path $\mathcal{P}_i$:

$$\mathcal{P}_i : \qquad s_0 = \mathbf{0} \xrightarrow{e_i} b_i \xrightarrow{\mathbf{0}} \cdots \xrightarrow{\mathbf{0}} b_i A^{t_i - 1} \xrightarrow{e_{\pi(i)}} (b_i A^{t_i} + b_{\pi(i)}) = \mathbf{0}. \tag{5.3.2}$$

We first note that each zero-input edge in the code's state diagram occurs at most once among the k paths $\mathcal{P}_1, \ldots, \mathcal{P}_k$. To see that this is so, assume the contrary, i.e., the existence of an equation of the form

$$b_i A^s = b_j A^t, \tag{5.3.3}$$

where $i \neq j$, $s \leq t_i - 2$, $t \leq t_j - 2$, and $s \geq t$. Then since we are assuming that A is invertible, we can multiply both sides of (5.3.3) by A^{-t} and obtain

$$b_i A^{s-t} = b_j, \tag{5.3.4}$$

which contradicts the fact that t_i is the least integer such that $b_i A^{t_i} \in \mathcal{B}$.

Since each zero-input edge occurs at most once as an edge in $\{\mathcal{P}_1, \ldots, \mathcal{P}_k\}$, and since by Lemma 5.1.1 the total output weight of all the zero-input edges is at most $2^{m-1}r$, it follows that at least one of the k codepaths $\mathcal{P}_i$ has a total output weight corresponding to its zero input edges of at most $\lfloor 2^{m-1}r/k \rfloor$. Since there are only two edges in $\mathcal{P}_i$ with nonzero input, the total output weight of $\mathcal{P}_i$ is therefore at most $2r + \lfloor 2^{m-1}r/k \rfloor$. ▪

5.4. Singular A.

In this section we will assume that the matrix A is singular, i.e., $\mathrm{rank}(A) \le m-1$.

5.4.1 Theorem. *If the matrix A is singular,*

$$d_2 \le \left(1 + \left\lceil \frac{2^{m-1}}{k} \right\rceil\right) r.$$

Proof: If $\{b_1, \ldots, b_k\}$ are the rows of B, consider the set of kL states of the form $b_i A^j$, for $i = 1, \ldots, k$, and $j = 1, \ldots, L$. Each of these states is in the rowspace of A, but since A is singular, this rowspace contains at most $2^{m-1} - 1$ nonzero elements. Thus if $kL \ge 2^{m-1}$, the pigeon-hole principle implies that two of these states must be equal, i.e., there is an equation of the form $b_i A^s = b_j A^t$, with $\max(s,t) \le L$. Then by Theorem 5.2.3, $d_2 \le (1+L)r$. Taking $L = \lceil 2^{m-1}/k \rceil$, we obtain the desired result. ∎

5.4.2 Lemma. *If y is a real number, and if r is a positive integer, then*

$$\lceil y \rceil r \le r + \lfloor yr \rfloor. \tag{5.4.1}$$

Proof: Write $y = n + \theta$, where n is an integer and $0 < \theta \le 1$. Then $\lceil y \rceil = n+1$, $\lfloor yr \rfloor = nr + \lfloor \theta r \rfloor$, and the inequality (5.4.1) becomes

$$r - \lfloor \theta r \rfloor \le r,$$

which is plainly true. ∎

5.4.3 Corollary. *If the matrix A is singular,*

$$d_2 \le 2r + \lfloor \frac{2^{m-1} r}{k} \rfloor.$$

Proof: This follows from Theorem 5.4.1 and Lemma 5.4.2, where we set $y = 2^{m-1}/k$:

$$\begin{aligned} d_2 &\le \left(1 + \left\lceil \frac{2^{m-1}}{k} \right\rceil\right) r \quad \text{by Theorem 5.4.1} \\ &\le r + r + \lfloor \frac{2^{m-1} r}{k} \rfloor \quad \text{by Lemma 5.4.2} \\ &= 2r + \lfloor \frac{2^{m-1} r}{k} \rfloor. \end{aligned}$$

∎

5.5. Proof of Theorem 5.3.

Here we need only recapitulate. Corollary 5.2.4 tells us that

$$d_2 \le \left\lceil \frac{2^m}{k} \right\rceil r \tag{5.5.1}$$

for any (r, k, m) convolutional code fragment. Theorem 5.3.1 tells us that

$$d_2 \le 2r + \left\lfloor \frac{2^{m-1} r}{k} \right\rfloor \tag{5.5.2}$$

holds when A is nonsingular, and Corollary 5.4.3 tells us that the same bound holds when A is singular. Thus both (5.5.1) and (5.5.2) hold for for any (r, k, m) convolutional code fragment. This proves Theorem 5.3.

5.6. Proof of Theorem 5.4.

We first note that the bound (5.4) is simply the second alternative in the bound (5.3) when $k = 1$, so that if equality holds in (5.4), we must have

$$\left\lceil \frac{2^m}{k} \right\rceil r \ge 2r + \left\lfloor \frac{2^{m-1} r}{k} \right\rfloor,$$

for $k = 1$, which simplifies to $2^{m-1} \ge 2$, i.e., $m \ge 2$.[3]

Furthermore, if we apply the bound of Theorem 5.4.1 in the case $k = 1$, we obtain $d_2 \le (1 + 2^{m-1})r$, which means that if $d_2 = (2 + 2^{m-1})r$, the feedback matrix A must be nonsingular.

In summary, a *necessary* condition for an $(r, 1, m)$ code fragment to have $d_2 = (2 + 2^{m-1})r$ is that $m \ge 2$ and that A be nonsingular. In the remainder of this section, we will assume that these two conditions are satisfied.

Theorem 5.2.2 identifies the codepaths of input weight 2 in terms of equations of the form

$$b_i A^{t_i} = b_j A^{t_j}, \tag{5.6.1}$$

where b_i and b_j are rows of B. In the case $k = 1$, however, B has only one row, which we shall denote by b. Furthermore A is nonsingular, so that (5.6.1) can be simplified to

$$bA^t = b. \tag{5.6.2}$$

[3] If $k = 1$ and $m = 1$, the bound (5.4) can be improved to $d_2 \le r$, with equality if and only if $G(D) = (P_1(D)/(D + 1), \ldots, P_r(D)/(D + 1))$, where each $P_i(D)$ equals either 1 or D. We omit the simple proof of this fact.

If t is the least positive integer such that (5.6.2) holds, then the corresponding codepath is

$$s_0 = \mathbf{0} \xrightarrow[d]{1} b \xrightarrow[bC]{0} bA \xrightarrow[bAC]{0} \cdots \xrightarrow[bA^{t-2}C]{0} bA^{t-1} \xrightarrow[bA^{t-1}C+d]{1} \mathbf{0}. \tag{5.6.3}$$

The output weight of this codepath is

$$d_2 = |d| + |bA^{t-1}C + d| + |bC| + |bAC| + \cdots + |bA^{t-2}C|.$$

Since both d and $bA^{t-1}C + d$ are r-vectors, we have

$$|d| + |bA^{t-1}C + d| \le 2r.$$

Furthermore, since $bC, bAC, \ldots, bA^{t-2}C$ are output symbols on zero-input edges, it follows from Lemma 5.1.1 that

$$|bC| + |bAC| + \cdots + |bA^{t-2}C| \le 2^{m-1}r.$$

It therefore follows that $d_2 = 2r + 2^{m-1}r$ if and only if

$$d = (1, 1, \ldots, 1) \tag{5.6.4}$$

$$bA^{t-1}C = (0, 0, \ldots, 0) \tag{5.6.5}$$

and

$$|bC| + |bAC| + \cdots + |bA^{t-2}C| + |bA^{t-1}C| = 2^{m-1}r. \tag{5.6.6}$$

The sum on the left side of (5.6.6) can be written as

$$\sum_{j=1}^{r} \sum_{i=0}^{t-1} |bA^i c_j|,$$

where c_j is the jth column of C. Now according to Theorem A.1 in Appendix A, each of the sums $\sum_{i=0}^{t-1} |bA^i c_j|$ is at most 2^{m-1}, which means that (5.6.6) holds if and only if

$$\sum_{i=0}^{t-1} |bA^i c_j| = 2^{m-1} \qquad \text{for } j = 1, \ldots, r. \tag{5.6.7}$$

But Theorem A.1 also says that (5.6.7) holds if and only if A is a primitive matrix and b and each of the c_j's is nonzero.

It therefore follows that necessary and sufficient conditions for $d_2 = (2 + 2^{m-1})r$ are $m \ge 2$ and

$$\begin{aligned} A &= \text{a primitive } m \times m \text{ matrix} \\ b &\ne 0 \\ c_j &\ne 0, \qquad \text{for } j = 1, \ldots, r. \\ bA^{-1}c_j &= 0 \qquad \text{for } j = 1, \ldots, r. \\ d &= (1, 1, \ldots, 1). \end{aligned} \tag{5.6.8}$$

What remains now is to translate the conditions (5.6.8) into a statement about the code's generator matrix. To do this we use the fact that that a generator matrix for a convolutional code fragment defined by the matrices (A, B, C, D) is given by ([8], Eq. (2.19))

$$G(D) = b \cdot \left[D^{-1} I_m - A\right]^{-1} \cdot [c_1, \ldots, c_r] + [1, \ldots, 1], \tag{5.6.9}$$

where b denotes the single row of B and c_j is the jth column of C. Combining the fact that $\left[D^{-1} I_m - A\right]^{-1} = D\left[I_m - DA\right]^{-1}$ with Theorem A.2, it follows from (5.6.9) that $G(D)$ is of the form

$$G(D) = \left(\frac{P_1(D)}{Q(D)}, \ldots, \frac{P_r(D)}{Q(D)}\right),$$

where $Q(D)$ is a primitive polynomial of degree m (the characteristic polynomial of A), and $P_j(D) = DP_j'(D) + Q(D)$, where $P_j'(D)$ is a polynomial of degree $\le m-1$. However, by (5.6.8) and Theorem A.2, the coefficient of D^{m-1} in $P_j'(D)$ equals $q_m b A^{-1} c_j = 0$, so that $P_j(D)$ has degree exactly m. Also, since the constant term of $DP_j'(D)$ is 0, and the constant term of $Q(D)$ is 1, if follows that the constant term of $P_j(D)$ is also 1. Finally, since $b \ne 0$, and $c_j \ne 0$, it follows from Theorem A.2 that $P_j'(D) \ne 0$, so that $P_j(D) \ne Q(D)$. ▪

6. Tables of "d_2-Optimal" Convolutional Code Fragments.

In this Section, we present several tables, giving the generator matrices (in octal notation) for a number of convolutional code fragments with the largest possible values of d_2.[4] The entries in Table 1 for $m \le 3$ are identical to the corresponding entries in Table 2 of [3]. The entries in Tables 2, 4, and 7, with $m \le 4$, are taken from [5]. The entries in Table 5, and those in Tables 2, 4, and 7, with $m = 5$ or $m = 6$, are new. In every case, we have verified (theoretically in some cases, and numerically in others) that the given value of d_2 is maximal, even though that value may be strictly less than the bound of Theorem 5.3 or Theorem 5.4, which is given by the d_2^b entry in the tables. (For $(r, 1, 1)$ code fragments, the bound of Theorem 5.4 can be improved to $d_2 \le r$, and we have given this as the value of d_2^b in Tables 1, 3, and 6.)

In some cases, the maximum value of d_2 can be obtained only by code fragments with repeated columns in the generator matrix. Unfortunately, experiment shows that using such code fragments as components of a turbo code yields poor results. Thus when the optimal code fragment has repeated columns, we list it in *italics*, and immediately below we give the best code fragment with the same parameters, but which does not have repeated columns.

For two IIR code fregments with the same value of d_2, we would expect heuristically that the one with the highest value of d_3 will give the best performance. Thus in choosing

[4] In the Tables, we use the letter h to denote the entries of the generator matrices. This should not be confused with our previous use of h to denote the output weight of a codeword.

the entries for the tables, when we found two or more code fragments with the largest possible value of d_2, we chose code fragments with the largest value of d_3.

If the code fragment is used to build a systematic encoder, the free distance of the resulting code is $d_{\text{free}} = \min_i(i + d_i)$. In the tables we denote the optimizing vaule of i bu $i*^*$ and the corresponding d_i by d^*.

For example, consider the $m = 3$ entry from Table 2. There we have $h_0 = \texttt{13} = D^3 + D + 1$, $h_1 = \texttt{15} = D^3 + D^2 + 1$, and $h_2 = \texttt{17} = D^3 + D^2 + D + 1$. Thus the encoder fragment is

$$G(D) = \begin{pmatrix} (D^3 + D^2 + 1)/(D^3 + D + 1) \\ (D^3 + D^2 + D + 1)/(D^3 + D + 1) \end{pmatrix}, \tag{6.1}$$

and the corresponding $(3, 2, 3)$ systematic encoder is

$$G'(D) = \begin{pmatrix} 1 & 0 & (D^3 + D^2 + 1)/(D^3 + D + 1) \\ 0 & 1 & (D^3 + D^2 + D + 1)/(D^3 + D + 1) \end{pmatrix}. \tag{6.2}$$

Table 2 says that the value of d_2 for the encoder fragment is 3, so that the value of d_2 for the $(3, 2, 3)$ code with generator matrix given by (6.2) is $d_2' = 2 + d_2 = 5$. On the other hand, Table 2 says that d_3 for the fragment is 1, so that $d_{\text{free}} \leq 3 + 1 = 4$ for this code, and in fact $d_{\text{free}} = 4$ in this case.

Table 1. d_2-optimal $(1, 1, m)$ code fragments.

$$G = (\, h_1/h_0 \,)$$

m	h_0	h_1	d_2	d_2^b	d_3	(d^*, i^*)
1	3	2	1	1	∞	$(1, 2)$
2	7	5	4	4	2	$(2, 3)$
3	15	17	6	6	4	$(2, 4)$
4	31	37	10	10	5	$(2, 4)$
5	75	57	18	18	7	$(4, 4)$
6	147	115	34	34	10	$(4, 5)$

Table 2. d_2**-optimal** $(1, 2, m)$ **code fragments.**

$$G = \begin{pmatrix} h_1/h_0 \\ h_2/h_0 \end{pmatrix}$$

m	h_0	h_1	h_2	d_2	d_2^b	d_3	(d^*, i^*)
1	3	1	1	0	1	∞	$(0, 2)$
2	7	3	5	2	2	0	$(0, 3)$
3	13	15	17	3	4	1	$(1, 3)$
4	23	35	27	6	6	2	$(2, 3)$
5	45	43	61	10	10	3	$(3, 3)$

Table 3. d_2**-optimal** $(2, 1, m)$ **code fragments.**

$$G = \begin{pmatrix} h_1/h_0 & h_2/h_0 \end{pmatrix}$$

m	h_0	h_1	h_2	d_2	d_2^b	d_3	(d^*, i^*)
1	3	2	1	2	2	∞	$(2, 2)$
2	*7*	*5*	*5*	*8*	*8*	*4*	$(4, 3)$
2	7	5	3	6	8	4	$(4, 3)$
3	13	17	15	12	12	7	$(7, 3)$
4	23	33	37	20	20	9	$(6, 4)$
5	73	45	51	36	36	14	$(6, 5)$
6	147	115	101	68	68	20	$(6, 5)$

Table 4. d_2**-optimal** $(1, 3, m)$ **code fragments.**

$$G = \begin{pmatrix} h_1/h_0 \\ h_2/h_0 \\ h_2/h_0 \end{pmatrix}$$

m	h_0	h_1	h_2	h_3	d_2	d_2^b	d_3	(d^*, i^*)
2	7	5	3	1	1	2	0	$(1, 2)$
3	13	15	17	11	2	3	1	$(2, 2)$
4	23	35	33	25	3	4	1	$(1, 3)$
5	51	73	65	61	7	7	1	$(2, 3)$

Table 5. d_2-optimal $(2, 2, m)$ code fragments.

$$G = \begin{pmatrix} h_1/h_0 & h_2/h_0 \\ h_3/h_0 & h_4/h_0 \end{pmatrix}$$

m	h_0	h_1	h_2	h_3	h_4	d_2	d_2^b	d_3	(d^*, i^*)
1	3	1	2	2	1	2	2	∞	$(2, 2)$
2	*7*	*3*	*3*	*5*	*5*	*4*	*4*	*0*	$(0, 3)$
2	7	1	5	5	3	3	4	1	$(1, 3)$
3	15	3	11	11	13	7	8	3	$(1, 4)$
4	31	35	23	23	21	12	12	3	$(3, 3)$

Table 6. d_2-optimal $(3, 1, m)$ code fragments.

$$G = (\, h_1/h_0 \quad h_2/h_0 \quad h_3/h_0 \,)$$

m	h_0	h_1	h_2	h_3	d_2	d_2^b	d_3	(d^*, i^*)
1	*3*	*2*	*1*	*1*	*3*	*3*	∞	$(3, 2)$
2	*7*	*5*	*5*	*5*	*12*	*12*	*6*	$(6, 3)$
2	7	5	3	6	8	12	6	$(6, 3)$
3	13	17	15	11	18	18	9	$(9, 3)$
4	23	35	27	37	30	30	13	$(10, 4)$
5	73	45	51	47	54	54	20	$(10, 5)$
6	147	115	101	135	102	102	29	$(11, 5)$

Table 7. d_2-optimal $(1, 4, m)$ code fragments.

$$G = \begin{pmatrix} h_1/h_0 \\ h_2/h_0 \\ h_3/h_0 \\ h_4/h_0 \end{pmatrix}$$

m	h_0	h_1	h_2	h_3	h_4	d_2	d_2^b	d_3	(d^*, i^*)
1	*3*	*1*	*1*	*1*	*1*	*0*	*1*	∞	$(\mathit{0}, \mathit{2})$
2	*7*	*5*	*5*	*5*	*5*	*0*	*1*	*2*	$(\mathit{0}, \mathit{2})$
2	7	1	2	3	5	0	1	0	$(0, 2)$
3	13	15	17	11	7	2	2	0	$(0, 3)$
4	23	35	33	37	31	3	4	1	$(1, 3)$
5	51	47	61	63	75	5	6	1	$(1, 3)$

A. Appendix A.

In this Appendix we shall give proofs of two results that we used in the proofs in Section 2.

A.1 Theorem. *If A is a nonsingular binary $m \times m$ matrix with period n, and if b and c are arbitrary nonzero m-vectors, and if t is an integer $\leq n$, then*

$$|bc| + |bAc| + \cdots + |bA^{t-1}c| \leq 2^{m-1},$$

with equality possible only if A is similar to the companion matrix of an mth degree primitive polynomial.

Proof: Suppose that the minimal polynomial of A is $Q(x)$, where

$$Q(x) = x^k - h_1 x^{k-1} - \cdots - h_k,$$

where $h_k \neq 0$ and $k \leq m$. Then we have

$$A^j = \sum_{i=1}^{k} h_i A^{j-i}, \qquad \text{for all } j \geq k.$$

If we multiply this matrix equation on the left by b and on the right by c, and if we define $x_j = bA^j c$, we obtain

$$x_j = \sum_{i=1}^{k} h_i x_{j-1}, \qquad \text{for all } j \geq k.$$

Thus $\mathbf{x} = (x_0, x_1, \ldots, x_{n-1})$ is a nonzero codeword in the (n, k) cyclic code with parity-check polynomial equal to $h(x)$. Then by a known result about cyclic codes (`Book 9, pp. 81--83`), $|\mathbf{x}| \leq 2^{k-1}$, with equality iff $h(x)$ is primitive. Thus since $k \leq m$, we can have $|\mathbf{x}| = 2^{m-1}$ iff $k = m$ and $h(x)$ is a primitive polynomial of degree m. But since A is $m \times m$, it follows that both the minimal and characteristic polynomials of A are equal to $h(x)$, so that A is similar to the companion matrix of an mth degree primitive polynomial.
■

A.2 Theorem. *If A is a nonsingular nonderogatory* [5] *$m \times m$ matrix with charcteristic polynomial*

$$Q(D) = 1 - q_1 D - \cdots - q_m D^m,$$

and if b and c are arbitrary m-vectors, then the rational function

$$F_b(D) = b(I_m - DA)^{-1}c,$$

[5] *A nonderogatory matrix is one for which the characteristic and minimal polynomials are equal.*

is of the form

$$\frac{P_{b,c}(D)}{Q(D)},$$

where $P_{b,c}(D)$ is a polynomial of degree $\leq m-1$. The coefficient of 1 in $P_{b,c}(D)$ is bc, and the coefficient of of D^{m-1} in $P_{b,c}(D)$ is $q_m bA^{-1}c$. Finally, if $Q(D)$ is irreducible, then $P_{b,c}(D)$ is the zero polynomial if and only if one or both of b and c are zero.

Proof: We have

$$(I_m - DA)^{-1} = I + AD + A^2D^2 + \cdots,$$

so that

$$(A.1) \qquad \begin{aligned} P_{b,c}(D) &= \sum_{j\geq 0}(bA^jc)D^j \\ &= \sum_{j\geq 0} f_jD^j, \qquad \text{(say)}. \end{aligned}$$

Now since $Q(D)$ is the characteristic polynomial for A, we have

$$(A.2) \qquad A^t = q_1A^{t-1} + \cdots + q_mA^{t-m}, \qquad \text{for } t \geq m.$$

Combining (A.1) and (A.2), we obtain

$$f_t = q_1f_{t-1} + \cdots + q_mf_{t-m}, \qquad \text{for } t \geq m.$$

Hence if we multiply $P_{b,c}(D)$ by Q(D) we obtain a polynomial of degree $\leq m-1$:

$$(A.3) \qquad \begin{aligned} P_{b,c}(D)Q(D) &= f_0 + (f_1 - q_1f_0)D + \cdots + (f_{m-1} - q_1f_{m-2} - \cdots - q_{m-1}f_0)D^{m-1} \\ &= g_0 + g_1D + \cdots + g_{m-1}D^{m-1}. \end{aligned}$$

If we recall that $f_j = bA^jc$, it follows from (A.3) that

$$(A.4) \qquad \begin{aligned} g_0 &= bc \\ g_1 &= b(A - q_1I)c \\ &\vdots \end{aligned}$$

$$(A.5) \qquad g_{m-1} = b(A^{m-1} - q_1A^{m-2} - \cdots - q_{m-1}I_m)c.$$

Eq. (A.4) shows that the coefficient of 1 in $P_{b,c}(D)$, i.e., g_0, is bc. To prove the assertion about the coefficient of D^{m-1}, i.e., g_{m-1}, observe that if we multiply (A.2) with $t = m$ by A^{-1}, we obtain

$$q_mA^{-1} = A^{m-1} - q_1A^{m-2} - \cdots q_{m-1}I_m.$$

Substituting this expression into (A.5) we find that $g_m = q_mbA^{-1}c$, as asserted.

Lastly we assume that $Q(D)$ is irreducible and investigate the possibility that for a fixed pair (b, c), the polynomial $P_{b,c}(D)$ is identically zero. By (A.1), this is equivalent to

$$(A.6) \qquad bA^jc = 0 \qquad \text{for } j = 0, 1, \ldots.$$

But by a known result, if the minimal polynomial of A has degree m and is irreducible, and if $c \neq 0$, the vectors $c, Ac, \ldots, A^{m-1}c$ are linearly independent ([9], Ch. 7). Thus if (A.6) holds, it must be that $b = 0$. In summary, $P_{b,c}(D) = 0$ if and only if either b or c (or both) is zero. ∎

References.

1. C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon Limit Error-Correcting Coding: Turbo Codes," *Proc. 1993 IEEE International Conference on Communications*, Geneva, Switzerland, pp. 1064–1070, May 1993.
2. S. Benedetto and G. Montorsi, "Unveiling turbo codes: some results on parallel concatenated coding schemes", *IEEE Trans. on Inf. Theory*, vol. 42 (March 1996), pp. 409–429..
3. S. Benedetto and G. Montorsi, "Design of Parallel Concatenated Convolutional Codes," *IEEE Transactions on Communications*, vol. 44, no. 5, pp. 591–600, May 1996.
4. S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara, "Serial Concatenation of Interleaved Codes: Performance Analysis, Design, and Iterative Decoding," *IEEE Trans. on Information Theory*, Vol. 44, No. 3, May 1998, pp. 909–926.
5. D. Divsalar and F. Pollara, "On the design of turbo codes," *TDA Progress Report* vol. 42-123 (November 15, 1995), pp. 99–121.
6. D. Divsalar and R. J. McEliece, "On the Effective Free Distance of Turbo-Codes," *Electronics Letters*, vol. 32, no. 5 (29 Feb. 1996), pp. 445–446.
7. R. A. Horn and C. R. Johnson, *Matrix Analysis.* Cambridge: Cambridge University Press, 1988.
8. R. J. McEliece, "The Algebraic Theory of Convolutional Codes," Chapter 12 in *Handbook of Coding Theory*, V. S. Pless and W. C. Huffman, eds. Amsterdam: Elsevier Science B.V., 1998.
9. E. Selmer, *Linear Recurrence Relations over Finite Fields.* Bergen, Norway: University of Bergen Department of Mathematics, 1966.