A t-error-correcting code is perfect if the covering radius is t. The code is quasi-perfect if the covering radius is t + 1.

Let β be an element of order n = 2^m - 1. The largest cyclic code whose generator polynomial g(x) ∈ GF(2)[x] has the zeros β, β^2, ..., β^{r - 1} but not β^r is defined to be a primitive BCH code of designed distance d and is here denoted by B(d). Note that d must be odd if B(d) exists.

The code B(3) is the Hamming code, which is a one-error-correcting perfect code. Gorestein, Peterson, and Zierler [1] proved that B(5) is a two-error-correcting quasi-perfect code. They also proved that B(7) is a three-error-correcting code which has covering radius at least five, and thus B(7) is not quasi-perfect. Later Van der Horst and Berger [2], Assmus and Mattson [3], and Helleseth [4] proved that B(7) has covering radius five.

In this correspondence we will prove a conjecture due to Gorestein, Peterson, and Zierler [1], which says that B(d) is never quasi-perfect when d > 7.

Leon'tev [5] proved that B(d) is not quasi-perfect when 2 < (d - 1)/2 < √n/log n and m > 7.

We will need the following lemmas.

Lemma 1: If d = 2^r - r < m, then B(d) exists and has actual minimum distance d.

Lemma 2: If d = 2^r - 2^s - 1, where 0 < (r - 1)/2 < s < r < m, then B(d) exists and has actual minimum distance d.

Lemma 1 is theorem 5.4 in Peterson and Weldon [6]. Lemma 2 is proved by Kasami and Lin [7].

Theorem 1: No primitive binary t-error-correcting BCH code is quasi-perfect when t > 2.

Before proving Theorem 1 we prove the following stronger result.

Theorem 2: Let ρ_d and τ_d denote the covering radius and actual error correcting ability of B(d), respectively, for 3 < r < m - 1.

i) If 2^r = 2^s - 1 < d < 2^r - 2^s - 1 where s is one of the numbers [r], [r + 1], ..., r - 2, then

\[ ρ_d - τ_d > \frac{2^r - 3}{2^{r-1}} (τ_d + 1). \]

ii) If 2^r = 2^s - 1 < d < 2^r - 1, then

\[ ρ_d - τ_d > \frac{2^{r-1}}{2^r - 2^s} (τ_d + 1). \]

Proof:

i) Let 2^r = 2^s - 1 < d < 2^r - 2^s - 1 for some \( s=\left[\frac{r}{2}\right], \left[\frac{r}{2}\right] + 1, ..., r - 2 \), where 3 < r < m - 1. By Lemma 2, B(2^r = 2^s - 1 - 1) and B(2^r - 2^s - 1) exist, and we have

\[ B(2^r - 2^s - 1) \subseteq B(d) \subseteq B(2^r - 1). \]

Since B(d) ⊆ B(2^r = 2^s - 1), we can choose α ∈ B(2^r = 2^s - 1 - 1) − B(d). Here α has distance at least 2^r = 2^s - 1 - 1 from every element in B(d). From the definition of the covering radius it follows that

\[ ρ_d > 2^r = 2^s - 1 - 1. \]

Since B(2^r - 2^s - 1) ⊆ B(d), we get by Lemma 2

\[ τ_d < 2^r - 1 - 2^s - 1. \]

Combining (1) and (2) we have

\[ ρ_d - τ_d > 2^r - 1 - 2^s - 3. \]

which combined with (2) gives

\[ ρ_d - τ_d > \frac{(τ_d + 1)(2^r - 3)}{(2^r - 1)}. \]

This proves i).

ii) This is proved using the same method as in the proof of i).

Proof of Theorem 1: Since the only B(d) with d > 2^r - 1 - 1 is the perfect binary repetition code B(2^r - 1), it is sufficient to prove that ρ_d - τ_d > 1 when 3 < d < 2^r - 1 - 1.

Let 5 < d < 2^r - 1 - 1. We can choose r such that 3 < r < m - 1 and 2^r - 1 - 1 < d < 2^r - 1 - 1. Further d belong to one of the two cases i) or ii) of Theorem 2.

Note that we have

\[ ρ_d - τ_d > \frac{1}{2} (τ_d + 1), \]

when d belongs to case i)

\[ ρ_d - τ_d > \frac{1}{2} (τ_d + 1), \]

when d belongs to case ii).

Hence we always have ρ_d - τ_d > 1 since τ_d > 3, and therefore B(d) is not quasi-perfect except when d = 5.

From the proof above we get the following corollary.

Corollary: If τ_d > 2 and τ_d ≠ 2^r - 1 - 1, then ρ_d - τ_d > \frac{1}{2} (τ_d + 1).

REFERENCES


Symbol Synchronization in Convolutionally Coded Systems

LEONARD D. BAUMERT, ROBERT J. McELIECE, MEMBER, IEEE, AND HENK C. A. VAN TILBORG

Abstract—Alternate symbol inversion is sometimes applied to the output of convolutional encoders to guarantee sufficient richness of symbol transitions in which arbitrarily long sequences of all zeros or all ones can cause temporary loss of synchronization and thus data loss. To avoid this problem, alternate symbols of the data stream are usually replaced by alternating symbols. In this correspondence we explore the length of the transition-free symbol stream in such systems, and those convolutional codes are characterized in which arbitrarily long transition free runs occur.

I. INTRODUCTION

Many digital communication systems derive symbol synchronization from the transitions in the received symbol stream. In such systems unusually long sequences of all zeros or all ones can cause temporary loss of synchroniztion and thus data loss. To avoid this problem, alternate symbols of the data stream are inverted; presumably a long alternating string is less likely than a long constant string.

Suppose the symbol stream is the alternately inverted output of a convolutional encoder. How long a constant string occurs?
then? That is, how long a run of alternating symbols
\[ \cdots 0101010 \cdots \] occurs in some codeword of a convolutional code? As we shall see, arbitrarily long alternating runs do occur in some codes; we characterize these codes in Section II. In Section III, for codes which do not have arbitrarily long alternating runs, we give upper bounds for the length of the longest run. In Section IV we consider examples which illustrate the use of these results and indicate how good the various upper bounds can be expected to be.

The reader is assumed to be familiar with the theory of convolutional codes and encoders as it appears, say, in Forney [1]. Thus terms like “overall constraint length,” “minimal encoder,” “dual code and dual encoder,” etc., are assumed known and used without definition. However, we remind the reader that the convolutional encoders of concern operate on binary sequences of the form \( x = (\cdots, x_{-1}, x_0, x_1, \cdots) \) which, theoretically at least, extend to infinity in both directions. The index \( t \) refers to discrete time intervals. In practice each sequence “starts” at some finite time; i.e., there is an index \( s \) such that \( t < s \) implies \( x_t = 0 \). The codewords produced by the encoders are of the same type. Using the delay operator \( D \), it is sometimes convenient to write \( x = x_0 D + x_1 D^2 + \cdots \). \( D \) is the generator matrix \( G \) of the convolutional code \( C \), and \( D \) is a polynomial generating a codeword with an infinite run of alternating symbols in some codeword of \( C \) if and only if

\[ u = \sum_{i=0}^{k-1} u_i \] or \( u_i = 1 \) for all \( i \), \( \forall i \in C \). Hence, a convolutional code \( C \) possesses an infinite run of alternating symbols if and only if the generator matrix \( G \) of \( C \) contains a codeword with an infinite run of alternating symbols. We also use certain algebraic properties of these formal power series, e.g., \( D^* + D^2 + \cdots = D/(1 + D) \).

II. CONVOLUTIONAL CODES WITH AN INFINITE RUN OF ALTERNATING SYMBOLS

Theorem I: Let \( C \) be an \((n,k)\) convolutional code over \( \text{GF}(2) \) with generator matrix \( G \). Then \( C \) contains a codeword with an infinite run of alternating symbols if and only if there exists a linear combination \( v = \sum_{i=0}^{k-1} v_i \) of the rows \( g_i \) of \( G \) such that

\[ \begin{align*}
& [v_0, \cdots, v_{n-1}] \\
& = [1, D, \cdots, D, D] \mod (1 + D^2) \quad \text{if } n \text{ odd} \\
& = [1, D, \cdots, D, 1] \mod (1 + D^2) \quad \text{if } n \text{ even}
\end{align*} \]

Proof: (Sufficiency): When \( n \) is even, consider the codeword produced by the inputs \( a_1/(1 + D) \) applied to the rows \( g_i \), where \( v = \sum_{i=0}^{k-1} a_i g_i \). Note that this code is produced by applying \( 1/(1 + D) \) to each row of the equivalent encoder whose rows are \( a_i g_i \). Thus after an initial transient the output will be \( v_1, v_2, \cdots, v_{n-1} \) and since \( v(D) = \sum_{i=0}^{k-1} a_i g_i(D) \mod (1 + D) \), the output \( v(D) \mod (1 + D) \) means that the sum of its even coefficients is 0 and the sum of its odd coefficients is 1. The situation is reversed for \( v(D) \mod (1 + D) \). Thus after an initial transient the input sequence \( a_1/(1 + D) \) will produce an infinite run of alternating symbols.

(Necessity): When \( n \) is even an infinite run of alternating symbols results from the juxtaposition of \( n \)-tuples of the form \( 10 \cdots 10 \) for \( 0 \leq i \leq 1 \). For definiteness, assume the former occurs. Then, if a codeword of \( C \) contains such an infinite run, there exists a codeword \( u \) such that

\[ u = b + D^3 [1, 0, \cdots, 1, 0] \]

Here \( b \) is an \( n \)-tuple of polynomials (of degrees \( < s \)) which describes the initial segment of \( u \). Let \( v(D) = (1 + D) u(D) \). Obviously, \( v(D) \) is polynomial and \( v(D) \mod (1 + D) \) is the codeword produced by \( a_1/(1 + D) \). Similarly, for \( n \) odd, \( C \) contains

\[ w = h' + D^3 [1, D, \cdots, D, 1] \]

Define \( v(D) = (1 + D^3) w(D) \). It follows as above that \( v(D) \mod (1 + D^3) \) is a polynomial with a codeword of \( C \) containing a codeword \( u \). The proof is complete. □

If a basic encoder \( G \) is known for \( C \) then only \( 2^k \) (respectively, \( 4^k \)) linear combinations \( v = \sum_{i=0}^{k-1} a_i g_i \) need be tried, for then the \( a_i \) can be restricted to \( 0, 1 \) (respectively, \( 0, 1, D, 1 + D \)) when \( n \) is even (respectively, \( n \) is odd). Even more efficiently, a row reduction could be used to determine whether or not the required vector was in the row space of \( G \) modulo \( 1 + D \) (or \( 1 + D^2 \)).

The case \( k = 1 \) is particularly important. Here, basic just means that the \( n \) polynomials making up the single generator \( g_1 \) have no common polynomial divisor and the test amounts to reducing \( g_1 \) modulo \( 1 + D \) or \( 1 + D^2 \).

It is also possible to test for the presence of an infinite alternating run in terms of the dual code (see Corollary to Theorem 2 below)

Theorem 2: Suppose an \((n, n-1)\) convolutional code \( C \) over \( \text{GF}(2) \) is given and \( f = [f_1, \cdots, f_n] \) generates the dual code, where \( (f_1, \cdots, f_n) = 1 \). Then there is an infinite run of alternating symbols in some codeword of \( C \) if and only if

\[ \begin{align*}
& (n \text{ even}) \\
& (n \text{ odd})
\end{align*} \]

\[ \begin{align*}
& f_n f_{n-1} \cdots f_1 f_0 = 0 \quad \text{mod } (1 + D^3) \\
& f_n f_{n-1} \cdots f_1 f_0 = 0 \quad \text{mod } (1 + D^3)
\end{align*} \]

Proof: Since \( f_1, \cdots, f_n \) are codewords of the dual code, they are linear combinations of shifts of

\[ \begin{align*}
& f_0 f_n f_{n-1} \cdots f_1 f_0 = 0 \quad \text{mod } (1 + D^3) \\
& f_0 f_n f_{n-1} \cdots f_1 f_0 = 0 \quad \text{mod } (1 + D^3)
\end{align*} \]

where \( d = \max (\deg f_j) \). Thus it is sufficient to check the inner products of this codeword of \( C \) with an infinite alternating run.

\[ \begin{align*}
& (n \text{ even}) \\
& (n \text{ odd})
\end{align*} \]

\[ \begin{align*}
& \cdots 0101010 \cdots 01010 \cdots \\
& \cdots 0101010 \cdots 01010 \cdots
\end{align*} \]

In both cases the necessity of the above conditions is immediate. For \( n \) odd the coefficients referred to are \( a, b \) from \( \Sigma f_2 + D \Sigma f_2 + 1 \equiv (a + b) \mod (1 + D^3) \). On the other hand, the above conditions obviously guarantee the existence of a codeword \( \cdots 01010 \cdots \) extending infinitely in both directions. However, only codewords “starting” at some finite time are of concern, and it remains to be shown that such a codeword is in the code. But this is trivial; it amounts to using the same single sequences truncated at some time \( t_0 \) exactly as they were when generating the doubly infinite sequence. Thus from \( t_0 + \delta \) on the output will be an infinite alternating run. □

Suppose an \((n, k)\) convolutional code \( C \) over \( \text{GF}(2) \) with generator matrix \( F \) for its dual code is given. Suppose \( F \) is a basic encoder, i.e., the \( \gcd \) of its \( n - k \) subdeterminants is 1. Then, if \( F = [f_1, \cdots, f_n] \) is any row of \( F \) it follows that \( (f_1, \cdots, f_n) = 1 \).

Let \( C_i \) \( (i = 1, \cdots, n - k) \) be the \((n, n-1)\) convolutional code dual to the \( i \)-th row of \( F \). Clearly,

\[ C = \bigcap_{i=1}^{n-k} C_i \]

and the maximum run of alternating symbols in any codeword of \( C \), has length \( L = L(C) < \min \{ L(C_i) \} \).

Corollary: When \( n \) is odd, an \((n, k)\) convolutional code \( C \) over \( \text{GF}(2) \) contains a codeword with an infinite run of alternating
symbols if and only if every row of a basic generator matrix \( F \) for \( C^\perp \) satisfies the congruences of Theorem 2. When \( n \) is even it is further necessary that this be true for the same value of \( \alpha \) (0 or 1).

Note: Suppose \( n \) is even and \( L(C) = L(C^\perp) = \infty \) with \( \alpha \neq 1 \) for \( C^\perp \) and \( \alpha \neq 0 \) for \( C \). Add row \( j \) to row \( i \) in \( F \); this gives an equivalent basic encoder which has \( L(C^\perp) < \infty \).

III. BOUNDS FOR FINITE RUNS OF ALTERNATING SYMBOLS

If no codeword contains an infinite run of alternating symbols the question arises as to the maximum length \( L \) of such a finite run. It is easy to give a bound for \( L \) in terms of the generators for the dual code. From this bound it is possible to derive another bound (in general, weaker) which has the advantage that it can be applied directly without knowledge of the dual (see the Corollary to Theorem 3, below). In Section IV these bounds are applied to some specific examples.

Suppose \([f_1, \ldots, f_n]\) is a generator matrix for an \((n,1)\) convolutional code \( C \) over GF(2) with \( d = \max(\deg f_i) \). Then

\[
\begin{align*}
  f_0 f_0^* \cdots f_{s-1} f_{s-1}^* \cdots f_m f_m^*
\end{align*}
\]

is its associated bit pattern. Let \( s \) be the number of symbols occurring between the first and last nonzero symbols \( f_i \) and \( f_m \) inclusively. If \((f_1, \ldots, f_n) = 1, s \) is the minimum length of any nonzero codeword of \( C \) and

\[
 n(d-1) + 2 \leq s < n(d+1).
\]

Theorem 3: Let \( C \) be an \((n,1)\) convolutional code over GF(2) with generator matrix \( F \) for its dual code generated by \([f_1, \ldots, f_n]\). Suppose no codeword of \( C \) contains an infinite run of alternating symbols. Then the maximum run of alternating symbols in any codeword of \( C \) has length \( L = s + n - 2 \), when \( n \) is even or when \( n \) is odd and

\[
 h(D) = \Sigma f_{2i} D^{2i} + \Sigma f_{2i+1} D^{2i+1} \equiv 1 + D \pmod{1 + D^2}.
\]

If \( n \) is odd and \( h(D) = 1 \) or \( D \) modulo \( 1 + D^2 \), the maximum run of alternating symbols has length \( L = s + 2n - 2 \).

Combining this with the limits given above for \( s \) yields

\[
 n(d-1) + 2 \leq s < n(d+1) - 2, \quad n \text{ even or } n \text{ odd},
\]

\[
 h(D) \equiv 1 + D \pmod{(1 + D^2)},
\]

\[
 n(d+1) - 2 < L < n(d+3) - 2, \quad n \text{ odd},
\]

\[
 h(D) \equiv 1 \text{ or } D \pmod{(1 + D^2)}.
\]

Proof: Suppose \( n \) is even. Then, from Theorem 2, \( \Sigma f_{2i} \equiv f_{2i+1} = 1 \pmod{1 + D} \). If there were an alternating run of length \( s + n - 1 \) it would have \( s \) consecutive symbols which would have inner product zero with the bit pattern of the \( f_i \). This is a contradiction since the sum of its rows is congruent to \([1, D, 1]\) modulo \( 1 + D^2 \). On the other hand, consider an alternating run of length \( s + n \). Change the first and last of these symbols; the inner products will be correct provided that the inner product was zero. On one side or the other of these symbols there would have to be \( n \) more symbols from the alternating run of size \( s + 2n - 1 \). These \( n \) symbols together with \( s - n \) of the original \( s \) symbols would also have to have inner product zero contrary to the hypothesis.

So \( L \leq s + 2n - 2 \). As above, a finite codeword of \( C \) can be constructed containing an alternating run of length \( L = s + 2n - 2 \). It is merely necessary that positions \( n + 1, \ldots, n + s + 1 \) of this run have inner product zero with the bit pattern of the \( f_i \).

Recall from the previous section that the codes \( C_i \) \((n,n-1)\) convolutional codes dual to the rows of \( F \), where \( F \) was a basic generator matrix for \( C^\perp \) and the obvious property

\[
 C = \bigcap_{i=1}^{n-k} C_i
\]

from which it follows that the maximum run of alternating symbols in any codeword of \( C \) has length \( L = L(C) < \min L(C_i) \). Suppose \( L(C_i) \) is finite for at least one value of \( i \). Then, if \( d \) is the maximum degree of any element in the \( i \)th row of \( F \), it follows that

\[
 L(C) < L(C_i) \leq \begin{cases} (n(d+2)-2, & n \text{ even} \\ (n(d+3)-2, & n \text{ odd} \end{cases}
\]

Corollary: Suppose an \((n,k)\) convolutional code \( C \) over GF(2) is given with basic generator matrix \( G \). Let \( \mu \) be the maximum degree of the \( k \times k \) subdeterminants of \( G \). Then either \( L = L(C) = \infty \) or

\[
 L \leq \begin{cases} (n(\mu+2)-2, & n \text{ even} \\ (n(\mu+3)-2, & n \text{ odd} \end{cases}
\]

Proof: Under these conditions \( C^\perp \) has a generator matrix \( F \) (a so-called minimal encoder for \( C^\perp \)) all of whose entries are of degree \( < \mu \). Thus the result follows immediately except when \( n \) is even and \( L(C_i) = \infty \) for \( i = 1, \ldots, n-k \). Here if \( L \) is finite, a finite bound for it can be determined by replacing row \( i \) of \( F \) in turn by the sum of row \( i \) and row \( j \), for \( j = 1, \ldots, n-k \). Of course in general all this work will not be required but the point is that such transformations do not increase the maximum degree of the elements of the dual encoder and so the bound given above is valid here also.

IV. EXAMPLES

Consider the \((3,2)\) code \( C \) generated by the encoder \( G \):

\[
\begin{bmatrix}
  D^3 + D & D^3 + 1 & D^4 + D^2 + D + 1 \\
  D^2 & D^3 + D + 1 & D^3 + D^2 + 1 \\
  1 & D + 1 & D \\
\end{bmatrix} \pmod{1 + D^2}.
\]

Note that the sum of its rows is congruent to \([1, D, 1]\) modulo \( 1 + D^2 \) and thus, by Theorem 1, \( C \) contains a codeword with an infinite run of alternating symbols.

As a second example, consider the \((4,1)\) code \( C \) with generator \( F \) of its dual code given by

\[
\begin{bmatrix}
  D & D^3 + 1 & D + 1 & D^2 + D + 1 \\
  D^2 + D + 1 & D^3 + 1 & D^2 + 1 & D^3 + 1 \\
\end{bmatrix} \equiv \begin{bmatrix}
  1 & 1 & 0 & 1 \\
  0 & 1 & 0 & 1 \\
  1 & 1 & 1 & 0 \\
\end{bmatrix} \pmod{1 + D^2}.
\]

Thus each row of \( F \) satisfies the congruences of Theorem 2 for some value of \( a \). But row 1 satisfies the congruence only for \( a = 0 \) and row 2 only for \( a = 1 \). Thus \( C \) does not contain a codeword with an infinite run of alternating symbols. In fact since the sum of rows 1 and 2 of \( F \) has degree \( d = 3 \) it follows that the maximum run of alternating symbols in any codeword of \( C \) is bounded above by \( n(d+2)-2 = 18 \). If we compute \( s \) here we get \( s = 14 \); so \( L = s + 2n - 2 = 16 \) is a little sharper. A basic generator for \( C \) is \([1 + D^2 + D^4 + D^5 + D^6 + D^7 + D^8, D^3 + D^4 \]

...
+\varepsilon^4 + \varepsilon^9, D + \varepsilon^2 + \varepsilon^3, D + \varepsilon^2 + \varepsilon^3 + \varepsilon^6 + \varepsilon^7 + \varepsilon^8 + \varepsilon^9)\)

thus \(\mu = 9\) and the Corollary to Theorem 3 gives only the weaker bound \(n(\mu + 2) - 2 = 42\).

In the example above, the Corollary to Theorem 3 was a little disappointing in that it gave a bound of 42 whereas more careful examination yielded \(L < 16\) (even 16 may be too high, for a cursory examination of the bit pattern associated with the basic generator for \(C\) given above indicates that 13 may be the answer). When \(k = n - 1\) it is clear from Theorem 3 that encoders do exist for which the bound given by the Corollary is tight. In general there are minimal encoders whose codes have no infinite alternating run but do possess codewords with finite alternating runs of length \(n_k + k + 1\) which compares reasonably well with the bounds given by the Corollary. For example, consider the \((n, k)\) convolutional encoder

\[
G = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

where \(I\) is an identity matrix or order \(k - 1\) and \(0'\) is a \(k - 1\) by \(n - k + 1\) matrix of zeros.

Here \(p = p(D) = 1 + D + D^3\) and, for \(n\) even, \(q = q(D) = 1 + D^2 + D^n\) (\(\mu \geq 2\)) while for \(n\) odd \(q(D) = 1 + D^3 + D^n\) (\(\mu > 2\)). \(G\) is obviously basic and minimal. Further that Theorem 1 guarantees that no codeword generated by \(G\) contains an infinite run of alternating symbols. That \(G\) generates a codeword with a run of alternating symbols of length \(n_k + k + 1\) can be confirmed by selecting the inputs \(x^{(1)}, \ldots, x^{(6)}\) properly. For example, let \(n = 8, k = 4, \mu = 3\), then the bit pattern associated with the bottom row of \(G\) is

\[
00011111 00010101 00001010 00000000 00001111.
\]

So if \(x^{(6)} = 1 + D^2 + D^3 \varepsilon (-1011\cdots)\) and \(x^{(5)} = D^2 + D^3 + D^{14}\) with \(x^{(4)} = x^{(3)} = 0\) the codeword generated by \(G\) is

\[
00011111 01010101 01010101 01010101 01011111
\]

which, starting with its 8th symbol, has an alternating run of length \(79 = 8\cdot 9 + 5\). Obviously \(x^{(1)}, \ldots, x^{(k - 1)}\) can always be adjusted to fill in the first \(k - 1\) symbols of each block of \(n\) symbols in the proper fashion. So the input \(x^{(6)}\) is the critical one. For \(n\) even, \(k\) even, and \(\mu\) odd, \(x^{(\mu - 1)} = 1 + D^3 + D^{29} + \cdots + D^n\varepsilon^{-1}D^n\). Similar formulas exist for the other cases—when \(n\) is odd these vary with \(\mu\) modulo 4.

As final examples consider the NASA Planetary Standard encoders of rates \(1/2\) and \(1/3\). Here \(G = [g_1, g_2, g_3]\) or \([g_1, g_2, g_3, g_4]\) with \(g_1 = 1 + D + D^2 + D^3 + D^6, g_2 = 1 + D + D^3 + D^4 + D^6, g_3 = 1 + D + D^2 + D^3 + D^6\). These both are basic minimal encoders which do not possess infinite alternating runs in any codeword as Theorem 1 easily shows. (Note that \([g_1, g_2, g_3]\) and \([g_2, g_3, g_1]\) do possess such runs, thus if infinite alternating runs are to be avoided the outputs in \([g_1, g_2, g_3]\) must be interleaved properly). For the rate \(1/2\) code the Corollary of Theorem 3 yields \(L < 2 \cdot 8 - 2 = 14\), and Theorem 3 itself guarantees the existence of finite codewords with alternating runs in this case. The rate \(1/3\) code has a dual generator \(F\) given by

\[
F = \begin{bmatrix}
D & 1 + D^2 + D^3 & 1 + D + D^2 + D^3 \\
D^2 & 1 + D + D^2 & 0
\end{bmatrix},
\]

Apply Theorem 3 to the first row of \(F\). Here \(s = 11\) so \(L = s + n - 2 = 12\). A finite codeword with an alternating run of length 12 is generated from \(G\) by the input \(x^{(1)} = 1 + D + D^2 + D^4 + D^7\) (\(\cdots 0011010010\cdots\)) so this bound is achieved.

Note: It is easy to see that, for \(k = 1\) (\(n > 2\)), it is always possible to rearrange the columns of a basic generator matrix to avoid infinite alternating runs. However, this is not true in general. Consider a basic \((4, 2)\) convolutional code whose generator matrix modulo \(1 + D\) is

\[
[1 1 1 1] [0 0 1 1].
\]

Every permutation of the columns of this matrix yields a matrix whose row space contains \([1, 0, 1, 0]\) or \([0, 1, 0, 1]\).