# An Analysis of Alternate Symbol Inversion for Improved Symbol Synchronization in Convolutionally Coded Systems

L. D. Baumert and R. J. McEliece
Communications Systems Research Section

H. van Tilborg
Technological University, Eindhoven, The Netherlands

*In the current NASA Planetary Program Flight/Ground Data System Standard, it is proposed that alternate symbols of the output of a convolutional encoder be inverted in order to guarantee the symbol synchronizer a certain richness of symbol transition.*

*In this paper we analyze this technique; in particular we characterize those convolutional codes with the property that even if alternate symbols are inverted, arbitrarily long transition free symbol streams may occur. For codes which do not exhibit this pathological behavior, we give an upper bound on the largest possible transition-free run.*

## I. Introduction

Many modern digital communication systems derive symbol synchronization from the data itself rather than from a separate synchronization channel. A common type of symbol synchronizer, and one that has become a NASA standard, is one that includes a clock whose phase and frequency are governed by timing estimates derived from the received data. The performance of this kind of symbol synchronizer depends in part on the "richness" of symbol transitions in the received data. An unusually long sequence of all 0's or all 1's, for example, could cause the local clock to lose synchronization, and so also data, temporarily.

If the data were uncoded, one possible method of increasing the transition density would perhaps be to add the sequence $\cdots 10101010 \cdots$ to the data stream, i.e., by inverting alternate symbols. The new data stream would then contain a long transition-free string only if the original data stream contained a long alternating string. Presumably a long alternating string is less likely than a long constant string, and so this method would probably have the desired effect.

If the data is encoded prior to transmission, one could again try to increase the transition density by inverting alternate symbols, and indeed this symbol inversion is now part of NASA's data system standards (Ref. 3). And whereas for uncoded data one can only assert that symbol inversion will tend to increase the transition density, for convolutionally encoded data a much stronger statement can be made: Provided the code does not suffer from a certain improbable property, there is an absolute upper bound one can place on

the length of the largest possible transition-free symbol run from the encoder, independent of the data being encoded. For example, the NASA standard rate 1/2 constraint length 7 has maximum run 14; the rate 1/3, constraint length 7 has maximum run 12.

In this paper we shall give a general approach to the problem of finding the largest possible encoded output of the form $\cdots$ 10101010 $\cdots$ from any convolutional code. In Section 2 we will classify all codes which admit an infinite run of this type; in Section 3 we will give an upper bound on the largest runs for codes which do not admit an infinite run; in Section 4 we work some specific examples; and in the Appendix we collect some known results about convolutional codes which we need to derive our results.

Finally, a word about notation is needed. The convolutional encoders of concern operate on binary sequences of the form

$$a = (\cdots, a_{-1}, a_0, a_1, \cdots)$$

which, theoretically at least, extend infinitely in both directions. The index refers to discrete time intervals. In practice, however, each sequence "starts" at some finite time; i.e., there is an index $s$ such that $t < s$ implies $a_t = 0$. The codewords produced by the encoder are of the same type; indeed some are also of finite length ($a_t = 0$ for $t > m$). Using $x$ as a place holder it is sometimes convenient to write

$$a = \sum_{-\infty}^{\infty} a_i x^i = \sum_{s}^{\infty} a_i x^i$$

We also use certain algebraic properties of these formal power series, e.g., $x^r + x^{r+1} + \cdots = x^r/(1+x)$.

## II. Convolutional Codes with an Infinite Run of Alternating Symbols

**Theorem 1.** Let $C$ be an $(n, k)$ convolutional code over $GF(2)$ with basic generator matrix $G$. Then $C$ contains a codeword with an infinite run of alternating symbols if and only if there exists a linear combination $v = [v_1, \cdots, v_n]$ of the rows of $G$ such that

$$[v_1, \cdots, v_n] \equiv \begin{cases} [0, 1, \cdots, 0, 1] \quad \text{or } [1, 0, \cdots, 1, 0] \\ \text{modulo } 1 + x, \ n \text{ even} \end{cases}$$

$$[v_1, \cdots, v_n] \equiv \begin{cases} [1, x, \cdots, x, 1] \quad \text{or } [x, 1, \cdots, 1, x] \\ \text{modulo } 1 + x^2, \ n \text{ odd} \end{cases}$$

**Proof.** We prove sufficiency first. When $n$ is even consider the codeword produced by the inputs $X_i = a_i/1 + x$ applied to each of the generators $g_i$, where $v = \Sigma\, a_i g_i$. After an initial transient the output will be $v_1(1), v_2(1), \cdots, v_n(1)$ and since $v_i(1) \equiv v_i(x)$ modulo $1 + x$ the result follows. For $n$ odd note that $v_i(x) \equiv x$ modulo $1 + x^2$ means that the sum of its even coefficients is 0 and the sum of its odd coefficients is 1, whereas the situation is reversed for $v_i(x) \equiv 1$ modulo $1 + x^2$. Thus after an initial transient the input sequences $X_i = a_i/1 + x^2$ will produce an infinite run of alternating symbols.

We now prove necessity. When $n$ is even an infinite run of alternating symbols results from the juxtaposition of $n$-tuples of the form 10...10 or 01...01. For definiteness, assume the former occurs. Then, if a codeword of $C$ contains such an infinite run, there exists a codeword $c$ and input sequences $X_1, \cdots, X_k$, none starting earlier than $t = 0$, such that

$$X_1 g_1 + \ldots + X_k g_k = h + \frac{x^s}{1+x}\,[1, 0, \ldots, 1, 0] \quad s \geqslant 0$$

Here $g_i$ is the $i^{\text{th}}$ row of $G$ and $h$ is an $n$-tuple of polynomials (of degrees $< s$) which describes the initial segment of $c$. Since $G$ is a basic encoder, there exists a basis for the module of $n$-tuples of polynomials over $GF(2)$ which is of the form $\{g_1, \cdots, g_k, g_{k+1}, \cdots, g_n\}$. Let $\{g_j^{\perp}\}$ be the dual basis, i.e., the $g_j^{\perp}$ are $n$-tuples of polynomials such that $g_i \cdot g_j^{\perp} = 0$ unless $i = j$ and $g_i^{\perp} \cdot g_i = 1$. Taking inner products on both sides of the equation yields

$$X_j = h \cdot g_j^{\perp} + \frac{x^s}{1+x}\,[1, 0, \cdots, 1, 0] \cdot g_j^{\perp} = A_j + \frac{\epsilon_j}{1+x}$$

where $A_j$ is a polynomial and $\epsilon_j = 0$ or 1. Thus $X_1 g_1 + \cdots + X_k g_k$ is

$$\sum_{j=1}^{k} \left\{ A_j g_j + \frac{\epsilon_j g_j}{1+x} \right\} = h + \frac{x^s}{1+x}\,[1, 0, \cdots, 1, 0]$$

Multiplying through $1 + x$ and reducing modulo $1 + x$ yields

$$\Sigma\, \epsilon_j g_j \equiv [1, 0, \cdots, 1, 0] \text{ modulo } 1 + x$$

so this is the vector $v$ as promised.

Similarly, for $n$ odd,

$$\Sigma X_j g_j = h + \frac{x^s}{1+x^2}\,[1, x, \cdots, x, 1]$$

$$X_j = A_j + \frac{\epsilon_j}{1+x^2}$$

where $A_j$ is a polynomial and $\epsilon_j = 0, 1, x$ or $1+x$. As before, this yields

$$\Sigma \epsilon_j g_j \equiv x^s\ [1, x, \cdots, x, 1] \text{ modulo } 1+x^2$$

and the proof is complete. [Q.E.D.]

In the proof above the condition that $G$ be basic was not required for sufficiency; thus if the congruence is satisfied an infinite run of alternating symbols does indeed occur in the code. Note that since the $\epsilon_j$ are restricted at most $2^k$ (resp., $4^k$) linear combinations need be tried when $n$ is even (resp., $n$ is odd). For modest values of $k$ this is not too large a task.

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 $1+x$ or $1+x^2$.

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). Of course one does not usually have a generator matrix for the dual code at hand, but when such is available the test is simpler than the one above, for large $k$. (A generator matrix for the dual code can always be computed, however; see the appendix for this).

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

($n$ even) $\Sigma f_{2i+\alpha} \equiv 0$ modulo $1+x$ for $\alpha = 0$ or $\alpha = 1$

($n$ odd) $\Sigma f_{2i} + x\Sigma f_{2i+1} \equiv 0$ modulo $1+x^2$

**Proof.** Since $(f_1, \cdots, f_n) = 1$ all codewords of the dual code are multiples of

$$\cdots 0 f_{10} f_{20} \cdots f_{n0} f_{11} f_{21} \cdots f_{n1} \cdots f_{1d} f_{2d} \cdots f_{nd} 0 \cdots$$

where $d = \max(\deg f_i)$. Thus it is sufficent to check the inner products of this codeword of $C^{\perp}$ with an infinite alternating run.

*n even.*

$$\cdots 0\ 1\ 0\ 1\ 0\ \cdots 0\ 1\ 0\ 1\ 0\ \cdots$$

($\alpha = 1$) $\quad f_{10} f_{20} f_{30} f_{40} \cdots f_{n0} f_{11} f_{21} f_{31} f_{41} \cdots$

($\alpha = 0$) $\quad f_{10} f_{20} f_{30} \cdots \quad f_{n0} f_{11} f_{21} f_{31} \cdots$

*n odd:*

$$\cdots 0\ 1\ 0\ 1\ 0\ \cdots 1\ 0\ 1\ 0\ 1\ \cdots 0\ 1\ 0\ 1\ 0\ \cdots$$

(coef. of $x$)

$$f_{10} f_{20} f_{30} f_{40} \cdots f_{n0} f_{11} f_{21} f_{31} f_{41} \cdots f_{n1} f_{12} f_{22} f_{32} f_{42} \cdots$$

(constant) $\quad f_{10} f_{20} f_{30} f_{40} \cdots f_{n0} f_{11} f_{21} f_{31} f_{41} \cdots$

In both cases the necessity of the above conditions is immediate. (For $n$ odd the coefficients referred to are $a, b$ from

$$\Sigma f_{2i} + x\Sigma f_{2i+1} \equiv ax + b \text{ modulo } 1+x^2$$

On the other hand the above conditions obviously guarantee the existence of a codeword $(\cdots 1010 \cdots 10 \cdots)$ 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 input sequences truncated to start at some time $t_0$ each preceded by a finite number of initial symbols which set the encoder's memory units properly.

[Q.E.D.]

Suppose an $(n, k)$ convolutional code $C$ over $GF(2)$ with generator matrix $F$ for its dual code is given. Suppose $F$ is a basic encoder, i.e., the g.c.d. of its $n-k$ by $n-k$ subdeterminants is 1, then, if $[f_1, \cdots, f_n]$ 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 $i^{\text{th}}$ row of $F$. Clearly

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

and the maximum run of alternating symbols $n$ any codeword of $C$ has length $L = L(C) \leqslant \min L(C_i)$.

**Corollary**. When $n$ is odd an $(n, k)$ convolutional code $C$ over $GF(2)$ 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 $L(C_i) = L(C_j) = \infty$ with $\alpha \neq 1$ for $C_i$ and $\alpha \neq 0$ for $C_j$. Add row $j$ to row $i$ in $F$; this gives an equivalent basic encoder which has $L(C_i) < \infty$.

## 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 4 these bounds are applied to some specific examples.

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

$$f_{10}f_{20} \cdots f_{n0}f_{11}f_{21} \cdots f_{n1} \cdots f_{1d}f_{2d} \cdots f_{nd}$$

is its associated bit pattern. Let $s$ be the number of symbols occurring between the first and last nonzero symbols $f_{ij}$ inclusively. If $(f_1, \cdots, f_n) = 1$, $s$ is the minimum length of any nonzero codeword of $C$ and

$$n(d-1) + 2 \leqslant s \leqslant n(d+1)$$

**Theorem 3**. Let $C$ be an $(n, n-1)$ convolutional code over $GF(2)$ with generator matrix for its dual code given by $[f_1, \cdots, f_n]$, where $(f_1, \cdots, f_n) = 1$. 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

$$h(x) = \Sigma f_{2i} + x \Sigma f_{2i+1} \equiv 1 + x \text{ modulo } 1 + x^2$$

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

Combining this with the limits given above for $s$ yields

$$\left.\begin{matrix} nd \\ \\ n(d+1) \end{matrix}\right\} \leqslant L \leqslant \begin{cases} n(d+2) - 2 & n \text{ even or } n \text{ odd}, \\ & h(x) \equiv 1 + x \\ n(d+3) - 2 & n \text{ odd}, h(x) \equiv \\ & 1 \text{ or } x \end{cases}$$

**Proof.** Suppose $n$ is even. Then from Theorem 2 above $\Sigma f_{2i} \equiv \Sigma f_{2i+1} \equiv 1$ modulo $1 + x$. If there were an alternating run of length $\geqslant s + n - 1$ it would have $s$ consecutive symbols which would have inner product zero with the bit pattern of the $f$'s. This contradicts $\Sigma f_{2i} \equiv \Sigma f_{2i+1} \equiv 1$, so $L \leqslant s + n - 2$. 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 they match up with the symbols $1, \cdots, s$ and $n + 1, \cdots, n + s$. Clearly this run can be extended to the right and the left to form a codeword of $C$; it is merely a matter of selecting symbols $1 \pm jn$ so that the inner products are zero. Such a codeword could conceivably extend infinitely in both directions; however using the same argument as at the end of Theorem 2 it follows that there is a finite codeword with an alternating run of this length.

If $n$ is odd then, from Theorem 2, $h(x) \not\equiv 0$ modulo $1 + x^2$. If $h(x) \equiv 1 + x$ the proof above applies, so $L = s + n - 2$. If $h(x) \equiv 1$ or $x$ then one of the inner products is zero but the other is not (see the display shown in the proof of Theorem 2). If there were a run of length $\geqslant s + 2n - 1$ there would have to be a run of $s$ consecutive symbols where the inner product was zero. On one side or the other of these $s$ 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 $L \leqslant s + 2n - 2$. 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 $n, \cdots, n + s - 1$ of this run have inner product zero with the bit pattern of the $f$'s. [Q.E.D.]

Recall from the previous section 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 $L = L(C) \leqslant \min L(C_i)$. 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 $i^{\text{th}}$ row of $F$ it follows that

$$L(C) \leqslant L(C_i) \leqslant \begin{cases} n(d+2) - 2 & n \text{ even} \\ n(d+3) - 2 & n \text{ odd} \end{cases}$$

**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

$$L \leqslant \begin{cases} n(\mu+2) - 2 & n \text{ even} \\ n(\mu+3) - 2 & n \text{ odd} \end{cases}$$

**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 $\leqslant \mu$. Thus the result follows immediately except when $n$ is even and $L(C_i) = \infty$ for $i = 1, \cdots, n-k$. 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 $j = 1, \cdots, n-k$ $(j \neq i)$. 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. Some Examples

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

$$\begin{bmatrix} x^3+x & x^3+1 & x^4+x^2+x+1 \\ x^2 & x^3+x+1 & x^3+x^2+1 \end{bmatrix} \equiv$$

$$\begin{bmatrix} 0 & x+1 & x+1 \\ 1 & 1 & x \end{bmatrix} \pmod{1+x^2}$$

Note that the sum of its rows is congruent to $[1, x, 1]$ modulo $1 + x^2$ thus, by Theorem 1, $C$ contains a codeword with an infinite run of alternating symbols. As mentioned in Section 2 this conclusion is valid even though $G$ is not a basic encoder ($x + x^2$ divides its $2 \times 2$ subdeterminants). Applying Theorem 2 to the dual encoder $F = [x^5 + x^3 + x^2 + x, x^3 + x^2 + 1, x^4 + x + 1]$ note that $(f_1, f_2, f_3) = 1$ and that $f_2 + xf_1 + xf_3 = x^6 + x^5 + x^4 + x^2 + x + 1 \equiv 0$ modulo $1 + x^2$ so that again the existence of a codeword in $C$ with an infinite run of alternating symbols is assured. $C^{\perp}$ does not contain such a codeword since $F$ is a basic encoder and $F \equiv [1 + x, x, x]$ modulo $1 + x^2$.

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

$$\begin{bmatrix} x & x^3+x+1 & x+1 & x^2+x+1 \\ x^2+x+1 & x^3+1 & x^3 & x^2+1 \\ x^2 & x^2+x+1 & x^2 & x^3+1 \end{bmatrix} \equiv$$

$$\begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 \end{bmatrix} \begin{matrix} \alpha = 0 \\ \alpha = 0, 1 \\ \alpha = 1 \end{matrix}$$

Thus each row of $F'$ satisfies the congruences of Theorem 2 for some value of $\alpha$. But row 1 satisfies the congruence only for $\alpha = 0$ and row 3 only for $\alpha = 1$. Thus $C$ does not contain a codeword with an infinite run of alternating symbols. In fact since the sum of rows 1 and 3 of $F'$ has degree $d = 3$ it follows that the maximum run of alternating symbols in any codeword of $C$ is bounded above by $n(d+2) - 2 = 18$. A basic generator for $C$ is $[1 + x^2 + x^4 + x^5 + x^6 + x^7 + x^8, x^3 + x^4 + x^5 + x^9, x + x^7 + x^8, x + x^2 + x^3 + x^6 + x^7 + x^8 + x^9]$ thus $\mu = 9$ and the Corollary to Theorem 3 gives only the weaker bound $n(\mu + 2) - 2 = 42$.

Up to this point the distinction (mentioned in the appendix) between the dual generators $F$ and $F'$ has been overlooked. Actually it is $F'$ that is used in the proofs of Theorems 2 and 3. There is no problem with this because either $F$ and $F'$ both satisfy the congruences of Theorem 2 or they

both do not. Similarly in Theorem 3 $h(x) \equiv 1 + x$ for $F$ if and only if $h(x) \equiv 1 + x$ for $F'$. However if the actual value of $s$ is to be used to establish a bound the bit patterns of $F'$ should be examined as $s$ can differ for corresponding rows of $F$ and $F'$. In the case at hand the sum of rows 1 and 3 of $F'$ has $s = 14$; so $L \leqslant s + n - 2 = 16$.

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 $L \leqslant 16$. (Even 16 may be too high; 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 $n\mu + k + 1$ which compares reasonably well with the bounds given by the Corollary. E.g., consider the $(n, k)$ convolutional encoder

$$G = \left[\begin{array}{c|c} I & 0' \\ \hline 0 \cdots 0 & p\ q\ p\ q \cdots \end{array}\right]$$

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

Here $p = p(x) = 1 + x + x^{\mu}$ and for $n$ even $q = q(x) = 1 + x^2 + x^{\mu}$ $(\mu \geqslant 3)$ while for $n$ odd $q(x) = 1 + x^3 + x^{\mu}$ $(\mu \geqslant 4)$. $G$ is obviously basic and minimal. Further 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 $n\mu + k + 1$ can be confirmed by selecting the inputs $X_1, \cdots, X_k$ properly. E.g., let $n = 8, k = 4$ and $\mu = 3$ then the bit pattern associated with the bottom row of $G$ is

00011111 00010101 00001010 00011111

So if $X_4 = 1 + x^2 + x^3$ $(= 10110 \cdots)$ and $X_2 = x + x^2 + x^3 + x^4$ with $X_1 = X_3 = 0$, the codeword generated by $G$ is

00011111 01010101 01010101 01010101 010111 $\cdots$

which starting with its 8th symbol has an alternating run of length $29 = 8 \cdot 3 + 5$. Obviously $X_1, \cdots, X_{k-1}$ 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 $X_k$ is the critical one. For $n$ even, $k$ even and $\mu$ odd $X_k = 1 + x^2 + x^4 + \cdots + x^{\mu-1} + x^{\mu}$. 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 (Ref. 3). Here $G = [g_1, g_2]$ or $[g_1, g_2, g_3]$ with $g_1 = 1 + x^2 + x^3 + x^5 + x^6, g_2 = 1 + x + x^2 + x^3 + x^6, g_3 = 1 + x + x^2 + x^4 + x^6$. These both are basic minimal encoders which do not possess infinite alternating runs in any codeword as Theorem 1 easily shows. (Note that $[g_1, g_3, g_2]$ 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 $L \leqslant 2 \cdot 8 - 2 = 14$ and Theorem 3 itself guarantees the existence of finite codewords with alternating runs of this length, since $s = 14$ in this case. The rate 1/3 code has a dual generator $F'$ given by

$$F' = \left[\begin{array}{ccc} x & 1 + x^2 + x^3 & 1 + x + x^2 + x^3 \\ 1 + x^3 & x^3 & 1 + x + x^2 \end{array}\right] \begin{array}{l} h(x) \equiv 1 + x \\ h(x) \equiv 0 \end{array}$$

Apply Theorem 3 to the first row of $F'$. Here $s = 11$ so $L \leqslant s + n - 2 = 12$. A finite codeword with an alternating run of length 12 is generated from $G$ by the input $X_1 = 1 + x + x^2 + x^4 + x^7$ $(= \cdots 0111010010 \cdots)$; so this bound is achieved.

# Appendix
# Convolutional Encoders

Proofs of most of the results mentioned here may be found in Forney (Ref. 1). Computations are restricted to $GF(2)$; however everything is easily generalized to any finite field.

A $k \times n$ matrix $G$ of polynomials $g_{ij}$ determines a rate $k/n$ ($k \leqslant n$) convolutional encoder with input sequences $X_i$ ($i = 1, \cdots, k$) and output sequences $Y_j$ ($j = 1, \cdots, n$), where

$$Y_j = \sum_{i=1}^{k} X_i \, g_{ij}$$

provided that $G$ is of rank $k$. These output sequences $Y_j$ are interleaved to produce a single codeword

$$Y_{11} \, Y_{21} \cdots Y_{n1} \, Y_{12} \, Y_{22} \cdots Y_{n2} \cdots$$

The collection of all such codewords (i.e., the "row space" of $G$) is the rate $k/n$ convolutional code generated by $G$. Such an encoder may be realized by $k$ shift registers the $i^{\text{th}}$ of which contains $\nu_i$ memory units where $\nu_i = \max_j(\deg g_{ij})$; $\nu_i$ is called the constraint length of the $i^{\text{th}}$ register. This is said to be the obvious realization of the encoder $G$ and thus requires $\nu = \Sigma \nu_i$ memory units in all ($\nu$ is the overall constraint length of the realization).

Two convolutional encoders are equivalent if they generate the same code. An encoder is called basic if there is no polynomial $h$ ($\deg h \geqslant 1$) which divides all the $k \times k$ subdeterminants of $G$. Basic encoders do not suffer from catastrophic error propagation and thus they are preferred over others. Fortunately every code can be generated by some basic encoder. I.e., there exists a basic encoder equivalent to any given encoder $G$.

In general a basic encoder that has maximum degree $\mu$ among its $k \times k$ subdeterminants requires at least $\mu$ memory units for its implementation. Sometimes it requires more. When $\mu$ is obviously sufficient, i.e. when $\mu = \Sigma \nu_i$, the encoder is said to be a minimal encoder. Since equivalent basic encoders have the same value of $\mu$, a minimal encoder requires as few memory units as any equivalent basic encoder. In fact, a minimal encoder requires as few memory units as any equivalent encoder, basic or not. Again, every encoder is equivalent to some minimal encoder. So, theoretically at least, there is no loss in assuming that any particular code at hand is generated by a minimal encoder. (Finding a minimal encoder equivalent to a given encoder can be a computational chore however; see below). It is a direct consequence of the minimality condition that any linear combination of the generators of a minimal convolutional encoder has degree greater than or equal to the degrees of all the generators occurring in the combination. This implies that equivalent minimal encoders not only have the same overall constraint length $\nu = \Sigma \nu_i$, but that they also have the same number of generators of each degree. I.e., the set of degrees $\nu_i$ ($i = 1, \cdots, k$ with multiplicities counted) is an invariant of minimal encoders under equivalence. Closely related to these last two comments is a property of minimal encoders called the predictable degree property which allows easy ennumeration of the short codewords generated by a minimal encoder, see Ref. 1.

Associated with any $(n, k)$ convolutional code $C$ is its dual code $C^{\perp}$. $C^{\perp}$ is the $(n, n-k)$ convolutional code which consists of all sequences orthogonal to every codeword of $C$. If $C$ is generated by a minimal encoder with overall constraint length $\nu$ then $C^{\perp}$ can also be generated by a minimal encoder of the same overall constraint length.

If $[g_1, \cdots, g_n]$ is a generator matrix for the $(n, 1)$ code $C$ and if $\Sigma f_i g_i = 0$ for polynomials $f_1, \cdots, f_n$ it is algebraically convenient to consider $[f_1, \cdots, f_n]$ as a row of a generator matrix $F$ for $C^{\perp}$. But if one considers the codewords the coefficients of the $f_i$'s must be reversed. For example, let $n = 2$; then one codeword of $C$ is

$$\cdots 0 \, g_{10} \, g_{20} \, g_{11} \, g_{21} \, g_{12} \, g_{22} \cdots$$

This codeword is orthogonal to the symbol sequences

$$\cdots f_{10} \, f_{20} \, 0 \cdots \qquad \ell = 0$$

$$\cdots f_{11} \, f_{21} \, f_{10} \, f_{20} \, 0 \cdots \qquad \ell = 1$$

$$\vdots \qquad\qquad \vdots$$

since the respective inner products are the coefficient of $x^{\ell}$ in $\Sigma f_i g_i = 0$. Thus each $f$ in $F$ should be replaced by $x^{\sigma} f(1/x)$ where $\sigma$ is the maximum degree of any element of $F$. An

equivalent generator matrix $F'$ is given by $x^{\nu_i} f(1/x)$ where $\nu_i$ is the maximum degree of any element in row $i$ of $F$. If $F$ is a minimal encoder then so is $F'$. The sequence $\{\nu_i\}$ of maximum degrees of the rows is the same for $F$ and $F'$, thus they have the same overall constraint length also. Thus for most purposes $F$ can be taken to be the generator matrix of the dual code $C^{\perp}$.

Every generator matrix $G$ has an invariant factor decomposition $G = A\Gamma B$. Here $A$ and $B$ are square matrices of determinant 1 with polynomial elements. $A$ is $k \times k$ and $B$ is $n \times n$. $\Gamma$ is a $k \times n$ diagonal matrix whose diagonal elements $\gamma_i$ ($i = 1, \cdots, k$) are nonzero polynomials. The $\gamma_i$ are called the invariant factors of $G$ and $\gamma_i$ divides $\gamma_{i+1}$. Over $GF(2)$ an encoder is basic if and only if $\gamma_k = 1$. This decomposition of $G$ can be produced by elementary row and column operations on $G$; see, for example, Gantmacher (Ref. 2). Now the first $k$ rows of $B$ constitute a basic encoder equivalent to $G$. Furthermore $B^{-1}$ exists and has polynomial elements. If $F^T$ denotes the last $n - k$ columns of $B^{-1}$, then $F$ is a $n - k \times n$ polynomial matrix which is a basic encoder for the dual code $C^{\perp}$.

Thus given any encoder $G$ it is possible to find an equivalent basic encoder by computing the invariant factor decomposition. However simpler methods often suffice. If $G$ is not basic, i.e., if the greatest common divisor of the $k \times k$ subdeterminants of $G$ is a polynomial $h$ of degree $\geq 1$, let $\psi$ be any irreducible polynomial dividing $h$. Then some linear combination of the rows of $G$ is divisible by $\psi$. By performing a row reduction of $G$ modulo $\psi$ one determines a transformation matrix $T$ of determinant 1 such that $TG$ has a row divisible by $\psi$. Divide this row by $\psi$; this produces an encoder equivalent to $G$ with $h$ replaced by $h/\psi$. Eventually this process terminates in an encoder equivalent to $G$ with $h = 1$; i.e., an equivalent basic encoder.

Similarly, if $G$ is basic but not minimal the matrix of $\nu_i^{\text{th}}$ order terms of $G$ will have rank less than $k$; thus a row reduction of this matrix leads to a transformation $T$ of determinant 1 such that $TG$ has smaller overall constraint length than $G$. Clearly, after at most $\Sigma \nu_i - \mu$ of these steps a minimal encoder equivalent to $G$ will be produced.

An alternate method of finding generators for the dual code also exists. After all, any $n - k$ linearly independent vectors orthogonal to $G$ will form such a generator matrix, so the following process can be used to produce them one at a time. Suppose an $(n, k)$ encoder $G$ is given. Since $G$ has rank $k$ some $k \times k$ subdeterminant is not zero. Let $H$ be the matrix formed from these $k$ columns plus one other. Then $H$ has dimension $k \times k + 1$ and is of rank $k$. Consider the $k + 1 \times k + 1$ matrix $H'$ whose first $k$ rows are $H$ and whose $k + 1^{\text{st}}$ row is row $i$ of $H$. Expanding $|H'|$ in terms of this last row shows that the vector of cofactors is orthogonal to every row of $H$, since $|H'| = 0$. Thus using this vector of cofactors to specify $k + 1$ components and setting the other components equal to zero we have a generator for the dual. Adjoin this generator to $G$ and repeat the process until $n - k$ generators have been found. These $n - k$ generators will generate the dual code. Note that after $j$ rows have been added to $G$ it will have rank $k + j$ so the process does not break down. Further, in the special case where $k = n - 1$, the single generator for the dual code will be basic if $G$ was. (In general this will not be the case for $k < n - 1$ however).

## Acknowledgement

The authors wish to thank M. K. Simon and J. G. Smith for bringing this problem to their attention and for suggesting several possible approaches.

## References

1. G. D. Forney, Jr., "Convolutional Codes I: Algebraic Structure", *IEEE Trans. Inform. Theory*, Vol. IT-16, November 1970, pp. 720-738 (See also correction: same journal, May 1971, page 360).

2. F. R. Gantmacher, *The Theory of Matrices*, Chelsea, New York, 1959.

3. *NASA Planetary Program Flight/Ground Data System Standards.* Revision 5, June 1, 1977. National Aeronautics and Space Administration, Washington, D.C.