N68-27228

A CONSTRUCTION TECHNIQUE FOR RANDOM

ERROR CORRECTING CONVOLUTIONAL CODES

by

Daniel J. Costello, Jr.

Technical Report No. EE-685

May 1, 1968

Department of Electrical Engineering,
University of Notre Dame,
Notre Dame, Indiana 46556

This work was supported by the National Aeronautics and Space Administration (NASA Grant NGR 15-004-026) in liaison with the Flight Data Systems Branch of the Goddard Space Flight Center.

A CONSTRUCTION TECHNIQUE FOR RANDOM ERROR CORRECTING CONVOLUTIONAL CODES

by

Daniel J. Costello, Jr., Student Member, IEEE

Abstract

A simple algorithm is presented for finding rate $\frac{1}{n}$ random-error correcting convolutional codes. Good codes considerably longer than any now known are obtained. A discussion of a new distance measure for convolutional codes, called the free distance, is included. Free distance is particularly useful when considering decoding schemes, such as sequential decoding, which are not restricted to a fixed constraint length. It is also shown how the above algorithm can be modified slightly to produce codes with known free distance.

A CONSTRUCTION TECHNIQUE FOR RANDOM-ERROR CORRECTING CONVOLUTIONAL CODES*

by

Daniel J. Costello, Jr., Student Member, IEEE

I. Introduction

Following Wozencraft and Reiffen [1], we can represent a rate $R = \frac{1}{n}$ binary convolutional code of memory order m as follows:

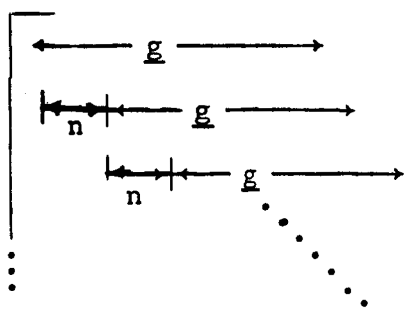
$$\underline{t} = \underline{i}\, G \tag{1}$$

where

$$\underline{t} = [t_o^{(1)}, t_o^{(2)}, \ldots, t_o^{(n)}, t_1^{(1)}, t_1^{(2)}, \ldots, t_1^{(n)}, \ldots] \text{ and}$$

$$\underline{i} = [i_o, i_1, i_2, \ldots]$$

are semi-infinite row vectors and

$G =$ 

is a semi-infinite matrix and where

$$\underline{g} = [g_o^{(1)}, \ldots, g_o^{(n)}, g_1^{(1)}, \ldots, g_1^{(n)}, \ldots, g_m^{(1)}, \ldots, g_m^{(n)}].$$

$\underline{g}$ is called the generator of the code. All blanks in G are assumed to be filled with "zeros". i_j is the $(j+1)^{st}$ information digit and $t_j^{(1)}$, $t_j^{(2)}, \ldots, t_j^{(n)}$ is the subblock of encoded digits corresponding to i_j.

* This work was supported by the National Aeronautics and Space Administration (NASA Grant NGR 15-004-026) in liaison with the Flight Data Systems Branch of the Goddard Space Flight Center, and forms part of a dissertation to be submitted in partial fulfillment of the requirements for the doctor's degree in electrical engineering at the University of Notre Dame, Notre Dame, Indiana.

The code is specified by G and is said to be in canonic systematic form if $g_o^{(1)} = 1$, $g_1^{(1)} = g_2^{(1)} = \ldots. = g_m^{(1)} = 0$. Then

$\underline{g} = [1, g_o^{(2)}, \ldots, g_o^{(n)}, 0, g_1^{(2)}, \ldots, g_1^{(n)}, \ldots, 0, g_m^{(2)}, \ldots, g_m^{(n)}]$ and

$\underline{t} = [i_o, t_o^{(2)}, \ldots, t_o^{(n)}, i_1, t_1^{(2)}, \ldots, t_1^{(n)}, \ldots]$.

The first j + 1 subblocks of transmitted digits can be represented as follows:

$$\underline{t}_j = \underline{i}_j G_j \quad , \quad j = 0,1,2,\ldots \tag{2}$$

where $\underline{t}_j = [t_o^{(1)}, \ldots, t_o^{(n)}, t_1^{(1)}, \ldots, t_1^{(n)}, \ldots, t_j^{(1)}, \ldots, t_j^{(n)}]$,

$\underline{i}_j = [i_o, i_1, \ldots, i_j]$,

and G_j contains only the first $(j + 1)n$ columns of G, e.g. for canonic systematic codes,

$$G_j = \begin{bmatrix} 1g_o^{(2)}\ldots g_o^{(n)}0g_1^{(2)}\ldots g_1^{(n)}0g_2^{(2)}\ldots g_2^{(n)}\ldots\ldots 0g_j^{(2)}\ldots g_j^{(n)} \\ \qquad\qquad 1g_o^{(1)}\ldots g_o^{(n)}0g_1^{(2)}\ldots g_1^{(n)}\ldots\ldots 0g_{j-1}^{(2)}\ldots g_{j-1}^{(n)} \\ \qquad\qquad\qquad\qquad 1g_o^{(2)}\ldots g_o^{(n)}\ldots\ldots 0g_{j-2}^{(2)}\ldots g_{j-2}^{(n)} \\ \ddots \\ \qquad\qquad\qquad\qquad\qquad\qquad 1g_o^{(2)}\ldots g_o^{(n)} \end{bmatrix} \tag{3}$$

where $g_i^{(k)} = 0$, $k = 1,2,\ldots,n$, if $i > m$. The first row of G_j will be denoted $\underline{g}_j$.

Note that each information digit affects a span of at most $(m+1)n$ transmitted digits since each row of G has non-zero entries over at most this span. Hence $n_A = (m+1)n$ is called the <u>constraint length</u> of a convolutional code of memory order m. Decoding is often assumed to be done by looking at only one constraint length of received digits commencing with the information digit to be decoded. The first constraint length of transmitted digits is given by

$$\underline{t}_m = \underline{i}_m \; G_m \quad . \tag{4}$$

The minimum distance, d_{min}, of a convolutional code of memory order m is defined [1] as

$$d_{min} = \min_{i_0 \neq i_0'} d_H \left(\underline{i}_m \; G_m, \underline{i}_m' G_m\right) \tag{5}$$

where $d_H(.,.)$ denotes the Hamming distance between the two vectors and the minimization is over all $\underline{i}_m = [i_o, i_1, \ldots, i_m]$ and $\underline{i}_m' = [i_o', i_1', \ldots$ $\ldots, i_m']$ with $i_o \neq i_o'$. It follows readily [1] that

$$d_{min} = \min_{i_0 \neq 0} W_H(\underline{i}_m G_m) \tag{6}$$

where $W_H(.)$ denotes the Hamming weight of the enclosed vector.

For the purposes of this paper, it is convenient to define a new quantity, called the column distance.

Definition 1. Given a rate $\frac{1}{n}$ convolutional code, the order j column distance, d_j, is given by

$$d_j = \min_{i_0 \neq i_0'} d_H(\underline{i}_j G_j, \; \underline{i}_j' G_j) \tag{7}$$

for $j = 0,1,2,\ldots$.

Clearly,

$$d_{min} = d_m . \tag{8}$$

We now proceed to collect some simple properties of the column distance. A trivial modification of the proof of (6) yields:

Property 1. $d_j = \min_{i_0 \neq 0} W_H(\underline{i}_j G_j)$.

Property 2. $d_j \leq W_H(\underline{g}_j)$.

Proof: Taking $i_o = 1$ and $i_1 = i_2 = \ldots = i_j = 0$, we have $\underline{i}_j G_j = \underline{g}_j$.

Hence by property 1, $d_j \leq W_H(\underline{g}_j)$.

Since $\underline{i}_j G_j$ is always the leading portion of $[\underline{i}_j, i_{j+1}] G_{j+1} = \underline{i}_{j+1} G_{j+1}$, it follows immediately from property 1 that:

Property 3. $d_j \leq d_{j+1}$, $j = 0,1,2,\ldots$.

It is well known [1] that i_o can be correctly decoded by a minimum distance decoder operating on the first n_A received digits if $\frac{d_{min}-1}{2}$ or fewer errors occur over the constraint length. Once i_o has been decoded by operating on the received digits in subblocks 0 through m, its effect can be removed from the received sequence and i_1 can be decoded in exactly the same fashion by operating on the received digits in subblocks 1 through m+1. This procedure can be continued indefinitely to decode all the information digits. Such a decoding technique has been termed feedback decoding by Robinson [2].

This discussion suggests that an appropriate "criterion of goodness" for convolutional codes is a high minimum distance to constraint length ratio, $\frac{d_{min}}{n_A}$. Indeed the commonly accepted "criterion of goodness" is the asymptotic Gilbert bound [1], i.e. a convolutional code of memory order m is said to be a "good" code if $H(\frac{d_{min}}{n_A}) \geq 1-R$, where $H(x) = -x \log_2 x - (1-x) \log_2(1-x)$ is the binary entropy function. Also it is obviously desirable that the complexity of the encoder be kept as small as possible. The usual encoding circuit for an $R = \frac{1}{n}$ systematic convolutional code is shown is figure 1, and it is noted that the number of two-input modulo-two adders required to implement this encoder is exactly $W_H(\underline{g}) - n$. Thus minimizing $W_H(\underline{g})$ for a given d_{min} and n_A minimizes the number of modulo-two adders in the encoder. All the codes presented in this paper will exhibit this property, i.e. for a given d_{min} and n_A, $W_H(\underline{g})$ will be the minimum possible value.

II. An Algorithm for Finding "Good" $R = \frac{1}{2}$ Convolutional Codes

In this section, a simple algorithm will be given which will be shown to produce "good" $R = \frac{1}{2}$ convolutional codes for all $m \leq 71$. First a statement of the algorithm is given and then several interesting properties of the codes produced are shown.

Algoritim A1.

(0) Set $g_o = 1$, $d_o = 2$, and $j = 1$.

(1) Set $g_j = 1$.

(2) Compute d_j. If $d_j > d_{j-1}$, go to (4).

(3) Set $g_j = 0$.

(4) If $j = m$, stop. Otherwise set $j = j + 1$ and go to (1).

Property A1-1. $W_H(\underline{g}_j) = d_j$ for $j = 0,1,\ldots,m$.

Proof: $W_H(\underline{g}_j) \geq d_j$ by property 2 of the column distance. Since g_j is permanently set to 1, i.e. $W_H(\underline{g}_j)$ is increased by one, if and only if $d_j > d_{j-1}$, $W_H(\underline{g}_j) \leq d_j$. Therefore $W_H(\underline{g}_j) = d_j$.

Since property 2 of the column distance requires that $W_H(\underline{g}_m) \geq d_m = d_{min}$, property A1-1 ensures that $W_H(\underline{g}_m)$ is minimal and hence the resultant code requires the minimum number of modulo-two adders in its encoding circuits.

Property A1-2. In the computation of step (2), if $d_j > d_{j-1}$, then $d_j = d_{j-1}+1$. If g_j is set to 0 in step (3), then $d_j = d_{j-1}$.

Proof: Property A1-2 follows directly from algorithm A1 and from property A1-1.

Property A1-3. The codes obtained from algorithm A1 exhibit the "nested" property, i.e. for $m_1 < m_2$, $\underline{g}_{m2} = [\underline{g}_{m_1}, 0, g_{m_{1+1}}, 0, g_{m_{1+2}}, \ldots$ $\ldots, 0, g_{m_2}]$.

Proof: Property A1-3 follows directly from algorithm A1.

Property A1-4. If $g_j = 1$, then $g_{j+1} = 0$, $j = 1,2,\ldots,m$.

Proof: Assume $g_j = 1$, $j \geq 1$. Then set $g_{j+1} = 1$. The information sequence $i_o = i_j = 1$, $i_1 = i_2 = \ldots = i_{j-1} = i_{j+1} = 0$ always produces a codeword with $d_{j+1} = d_j$. Therefore algorithm A1 will set $g_{j+1} = 0$. Property A1-4 allows us automatically to add a "zero" to g_j after adding each "one" beyond g_o. This permits a shortcut to reduce the number of times steps 1 and 2 must be applied to reach a given length code.

Property A1-5. (optimality property). Let $\underline{g}_m$ be the generator obtained by using algorithm A1. Let $\underline{g}_m' \neq \underline{g}_m$ be any other generator of the same length such that $W_H(\underline{g}_j') = d_j'$, $j = 0, 1, \ldots, m$, i.e. such that each "one" in the generator increases the column distance by one. Then there exists a $j_o, 0 \leq j_o \leq m$ such that $d_{j_o} > d_{j_o}'$ and $d_i = d_i'$ for $i = 0, 1, 2, \ldots, j_o-1$.

Proof: Assume the first point at which the two generators disagree, j_o, $0 \leq j_o \leq m$, has $g_{j_o} = 0$, $g_{j_o}' = 1$. Then $d_{j_o}' = d_{j_o}+1 > d_{j_o}$. But this is impossible, since if the column distance can increase at j_o algorithm A1 would make $g_{j_o} = 1$. Therefore the first point at which the two generators disagree must have $g_{j_o} = 1$, $g_{j_o}' = 0$, and hence $d_{j_o} > d_{j_o}'$.

The optimality property shows that any other algorithm for generating convolutional codes which increases the column distance by one each time a "one" is added to the generator differs from algorithm A1 in that such "ones" are not always added at the first opportunity.

Algoritim A1 was programmed on the Univac 1107 computer at the University Computer Center. The most difficult part of algorithm A1 to program is the computation of d_j in step (2). This was done by using a sequential-decoding-like algorithm suggested by Forney [3]. The program took approximately one and one-half hours to reach m = 71.

The codes obtained from algorithm A1 are compared with Bussgang's [4] optimal codes and Lin and Lyne's [5] near-optimal codes in Table I. Bussgang's computer search for optimal codes reached m = 15 before the amount of computation became too large. Lin and Lyne were able to carry their near-optimal search out to m = 20. Algorithm A1 is sufficiently simple to allow hand computation out to m = 22 and it was extended to m = 71 by computer. Table I also compares the codes obtained with the Gilbert Bound [4,5], and it can be seen that the codes remain "good" out to m = 71. An interesting, but as yet unsolved, question is whether algorithm A1 will continue to produce "good" codes, i.e. codes whose distance increases linearly with j as j becomes arbitrarily large. The amount of computation required by algorithm A1, because of the calculation of d_j in step (2), appears to increase exponentially with increasing m, as it does in all known search techniques for finding codes. However, for a given m, algorithm A1 requires substantially less computation than that suggested by Lin and Lyne.

III. Algorithms for Generating "Good" $R = \frac{1}{3}$ and $R = \frac{1}{4}$ Codes

For rates $R = \frac{1}{n}$, $n > 2$, we again seek an algorithm for generating codes such that $d_j = W_H(\underline{g}_j)$ for $j = 0, 1, 2, \ldots, m$ and "ones" are added to the generator at the first opportunity consistant with this constraint. Since there are now $n - 1$ digits, viz. $g_j^{(2)}, g_j^{(3)}, \ldots, g_j^{(n)}$, to be specified in each subblock, there will not be a unique algorithm with the above property for $n > 2$. For example, for $n = 3$ the three following algorithms each result in a code such that $d_j = W_H(\underline{g}_j)$ and "ones" are added to $\underline{g}_m$ at the earliest opportunity. For $n = 3$, it is well known [5] that $d_j \leq d_{j-1}+1$ so that it is unnecessary to test the choice $g_j^{(2)} = g_j^{(3)} = 1$ since the column distance can never increase by 2.

Algorithm A2.

(0) Set $g_o^{(2)} = g_o^{(3)} = 1$, $d_o = 3$, and $j = 1$.

(1) Set $g_j^{(2)} = 1$, $g_j^{(3)} = 0$.

(2) Compute d_j. If $d_j > d_{j-1}$, go to (6).

(3) Set $g_j^{(2)} = 0$, $g_j^{(3)} = 1$.

(4) Compute d_j. If $d_j > d_{j-1}$, go to (6).

(5) Set $g_j^{(2)} = g_j^{(3)} = 0$.

(6) If $j = m$, stop. Otherwise set $j = j+1$ and go to (1).

Algorithm A3.

Steps (0) through (5) are the same as in algorithm A2.

(6) If $j = m$, stop. Otherwise, interchange steps (1) and (3), set $j = j+1$, and go to (1).

Algorithm A4.

Steps (0) through (5) are the same as in algorithm A2.

(6) If $j = m$, stop. Otherwise, if d_j increased during step (2),

interchange steps (1) and (3), set $j = j + 1$, and go to (1). If d_j increased during step (4) or remained the same, set $j = j + 1$ and go to (1).

The codes obtained from algorithms A2, A3, and A4 are shown in Table II and are compared to Bussgang's codes, Lin and Lyne's codes, and to the Gilbert bound. Each algorithm was carried out to $m = 35$ by computer in a few minutes. Again the codes were quite "good" and are considerably longer than other known "good" $R = \frac{1}{3}$ codes. Note that the codes obtained from Algorithms A2, A3, and A4 exhibit about the same distance properties. Indeed it seems the many variations of the algorithm available for $R = \frac{1}{3}$ will have little effect on the distance properties of the resulting codes.

Note that at $m = 7$, the code obtained from algorithm A2 is actually better than Lin and Lyne's near optimal code. It can be shown that this code meets the Plotkin upper bound [6] on minimum distance at $m = 7$.

To generate $R = \frac{1}{4}$ codes, $g_j^{(2)}$, $g_j^{(3)}$, and $g_j^{(4)}$ must be specified for each j, and it must be recognized that an increase of either one or two in the column distance for each j is possible. Only one algorithm will be given for generating $R = \frac{1}{4}$ codes with the property that $d_j = W_H(\underline{g}_j)$ and "ones" are added to the generator at the earliest opportunity.

<u>Algorithm A5.</u>

(0) Set $g_o^{(2)} = g_o^{(3)} = g_o^{(4)} = 1$, $d_o = 4$, and $j = 1$.

(1) Set $g_j^{(2)} = g_j^{(3)} = 1$, $g_j^{(4)} = 0$, $i = 1$, and go to (8).

(2) Set $g_j^{(3)} = 0$, $g_j^{(4)} = 1$, $i = 2$, and go to (8).

(3) Set $g_j^{(2)} = 0$, $g_j^{(3)} = 1$, $i = 3$, and go to (8).

(4) Set $g_j^{(3)} = 0$, $i = 4$, and go to (8).

(5) Set $g_j^{(3)} = 1$, $g_j^{(4)} = 0$, $i = 5$, and go to (8).

(6) Set $g_j^{(2)} = 1$, $g_j^{(3)} = 0$, $i = 6$, and go to (8).

(7) Set $g_j^{(2)} = 0$ and to to (9).

(8) Compute d_j. If $d_j = d_{j-1}$, go to (i+1).

(9) If $j = m$ stop. Otherwise, set $j = j+1$ and go to (1).

Table III compares the $R = \frac{1}{4}$ codes generated by algorithm A5, Lin and Lyne's codes, and the Gilbert bound. Algorithm A5 was carried out to $m = 35$ by computer in about 10 minutes and again "good" codes were found.

IV. Free Distance and Its Implications

For certain decoding schemes, such as sequential decoding, the decoder is not constrained to consider only one constraint length of received digits while attempting to decode a particular transmitted digit, but may search over several constraint lengths. In such cases, the conventional minimum distance loses much of its meaning. Massey [7] has suggested defining a new distance measure, called the free distance, appropriate for an hypothetical decoder which makes its decoding decisions on the basis of the entire received sequence.

Definition 2. $d_{free} = d_\infty$ i.e. the free distance is equal to the column distance at $j = \infty$.

Some properties of d_{free} can readily be derived.

Property 1. $d_j \leq d_{free} \leq W_H(\underline{g})$ for all finite j, in particular $d_m = d_{min} \leq d_{free} \leq W_H(\underline{g})$.

Proof: $d_j \leq d_\infty = d_{free}$ follows directly from property 3 of the column distance. $d_{free} = d_\infty \leq W_H(\underline{g})$ follows directly from property 2 of the column distance.

Property 2. For all the codes obtained from algorithms A1, A2, A3, A4, and A5, $d_{free} = d_{min} = d_m$.

Proof: $d_{min} = d_m = W_H(\underline{g}_m) = W_H(\underline{g})$ is a property of the codes obtained from algorithms A1, A2, A3, A4, and A5.

Property 3. For $R = \frac{1}{n}$ canonic systematic convolutional codes, $d_{free} =$ $= d_{(n-1)(m+1)m}$.

Proof: We need not consider any information sequence with a string of m or more "zeros" in it since additional "ones" in the information sequence can only add weight to the codeword. Property 1 implies that d_{free} can never be more than $(n-1)(m+1)+1$, the maximum number of "ones" in any $R = \frac{1}{n}$ canonic systematic generator. Since we are considering only information sequences which have at least one "one" every m digits, all codewords with $i_o \neq 0$ have at least one "one" in the 0^{th} subblock and at least one "one" in every succeeding set of m subblocks. Therefore all such codewords with $i_o \neq 0$ must reach a weight of $(n-1)(m+1)+1$ in $(n-1)(m+1)m$ subblocks. Hence $d_{free} = d_{(n-1)(m+1)m}$.

Property 3 indicates that d_{free} can always be found by computing the column distance over a finite number of subblocks. It is conjectured that the result of property 3 can be strengthened considerably by more detailed arguments, and probably that $d_{free} = d_{2m}$. If true, this would substantially simplify the calculation of d_{free} for a given code.

Free distance is an appropriate distance measure, not only for an unconstrained hypothetical decoder, but for a practical decoder which decodes in "frames" of perhaps 10 constraint lengths. Suppose $d_{free} = d_{10m}$ and for simplicity assume that d_{free} is an odd integer. Then a decoder confined to one constraint length will make a decoding error for at least one pattern of $\frac{d_{min+1}}{2}$ errors in a "frame" whereas a decoder looking over 10 constraint lengths cannot make a decoding error unless $\frac{d_{free+1}}{2}$ errors occur in the "frame". Since for small enough p(digit error probability), the decoding error probability is a function only of the minimum number of errors in a "frame" that can cause a decoding error, d_{free} is an important parameter for a practical decoder. Since a sequential decoder scans several constraint lengths before making a decision, d_{free} is a more appropriate distance measure than d_{min} for codes used with sequential decoding.

Clearly, it is of considerable interest to find codes with known d_{free}, especially codes for which $d_{free} > d_{min}$. A slight modification of the preceeding algorithms can be used for this purpose. Algorithm A6 indicates the necessary modification of algorithm A1.

Algorithm A6 (assume $L \geq m$).

(0) Set $g_o = 1$, $D_o = 2$, and $j = 1$.

(1) Set $g_j = 1$.

(2) Compute d_L. If $d_L > D_{j-1}$, set $D_j = d_L$ and go to (4).

(3) Set $g_j = 0$ and $D_j = D_{j-1}$.

(4) If $j = m$, stop. Otherwise, set $j = j+1$ and go to (1).

The following properties of the codes resulting from algorithm A6 will be presented without proof.

Property A6-1. $W_H(\underline{g}_j) = D_j$ for all j.

Property A6-2. In the computation of step (2), if $d_L > D_{j-1}$, then $D_j = D_{j-1}+1$.

Property A6-3. The codes obtained from algorithm A6 exhibit the "nested property". (See property A1-3.).

Theorem A6-1. $W_H(\underline{g}_j) = D_j = d_{free}$ for all j, where d_{free} is the free distance of the truncated code with memory order j.

Proof: $W_H(\underline{g}_j) = D_j \leq d_{free}$ by property A6-1 and property 1 of free distance. $d_{free} \leq W_H(\underline{g}_j)$ by property 1 of free distance. Therefore $d_{free} = W_H(\underline{g}_j)$ for all j.

In general algorithm A6 will result in generators with greater weight than those obtained from algorithm A1. Therefore, d_{free} for the codes obtained from algorithm A6 will be larger than d_{min} for the same length codes obtained from algorithm A1.

Table IV shows the results of applying algorithm A6 to the construction of a $R = \frac{1}{2}$ canonic systematic code with m = 35 and L = 71. It is interesting to note that algorithm A1 produced a code with m = 35 and $d_{min} = d_{free} = 13$. Algorithm A6 resulted in a code with m = 35 and $d_{free} = 17$. d_{min} was checked for this code and found to be 13. Therefore, algorithm A6 gave us a code with the same length and the same d_{min}, but with a larger d_{free}. Clearly, although the two codes have the same d_{min}, the code obtained from algorithm A6 would exhibit a lower probability of error when used with sequential decoding. Since for this code, $d_{free} = d_{71} = d_{2m+1}$, it would seem most appropriate for use with a decoder which searched over approximately two constraint lengths on the average.

V. Summary and Conclusions

Simple and efficient algorithms were given for constructing convolutional codes of $R = \frac{1}{n}$ which yielded codes with d_{min} considerably better than the Gilbert bound out to m = 71 for $R = \frac{1}{2}$, and m = 35 for $R = \frac{1}{3}$ and $R = \frac{1}{4}$. The algorithms always retained the property of minimizing the number of modulo-2 adders needed in the encoding circuit for codes of a given length and minimum distance.

A definition was given for a new distance measure for convolutional codes, called the free distance. It was indicated that d_{free} is a more important distance measure for convolutional codes used with sequential decoding than d_{min}, since a sequential decoding search is not limited to a constraint length. Specifically, d_{free} is more closely related to the probability of error for sequential decoding than d_{min}. Finally, a slightly modified algorithm was shown to produce codes with known d_{free}.

Acknowledgement

A great deal of credit for the results presented in this paper belongs to Professor J. L. Massey. His constant suggestions and guidance made this work possible.

References

[1] J. M. Wozencraft and B. Reiffen, Sequential Decoding, M.I.T. Press and Wiley, 1961.

[2] J. P. Robinson, "Error Propagation and Definite Decoding of Convolutional Codes", IEEE Trans. on Inf. Th., IT-14, pp. 121-128, January, 1968.

[3] G. D. Forney, "Final Report on a Study of a Simple Sequential Decoder", U. S. Army Satellite Communication Agency Contract DAAB07-68-C-0093, Appendix A, Codex Corporation, Watertown, Mass., April 15, 1968.

[4] J. J. Bussgang, "Some Properties of Binary Convolutional Code Generators", IEEE Trans. on Inf. Th., IT-11, pp.90-100, January,1965.

[5] S. Lin and H. Lyne, "Some Results on Binary Convolutional Code Generators", IEEE Trans. on Inf. Th., IT-13, pp. 134-139, January, 1967.

[6] J. L. Massey, "Some Algebraic and Distance Properties of Convolutional Codes", Proceedings, Symposium on Error Correcting Codes, sponsored by the Mathematics Research Center, United States Army at the University of Wisconsin, May 6-8, 1968.

[7] J. L. Massey, Private Communication, University of Notre Dame, Notre Dame, Indiana, 1967.

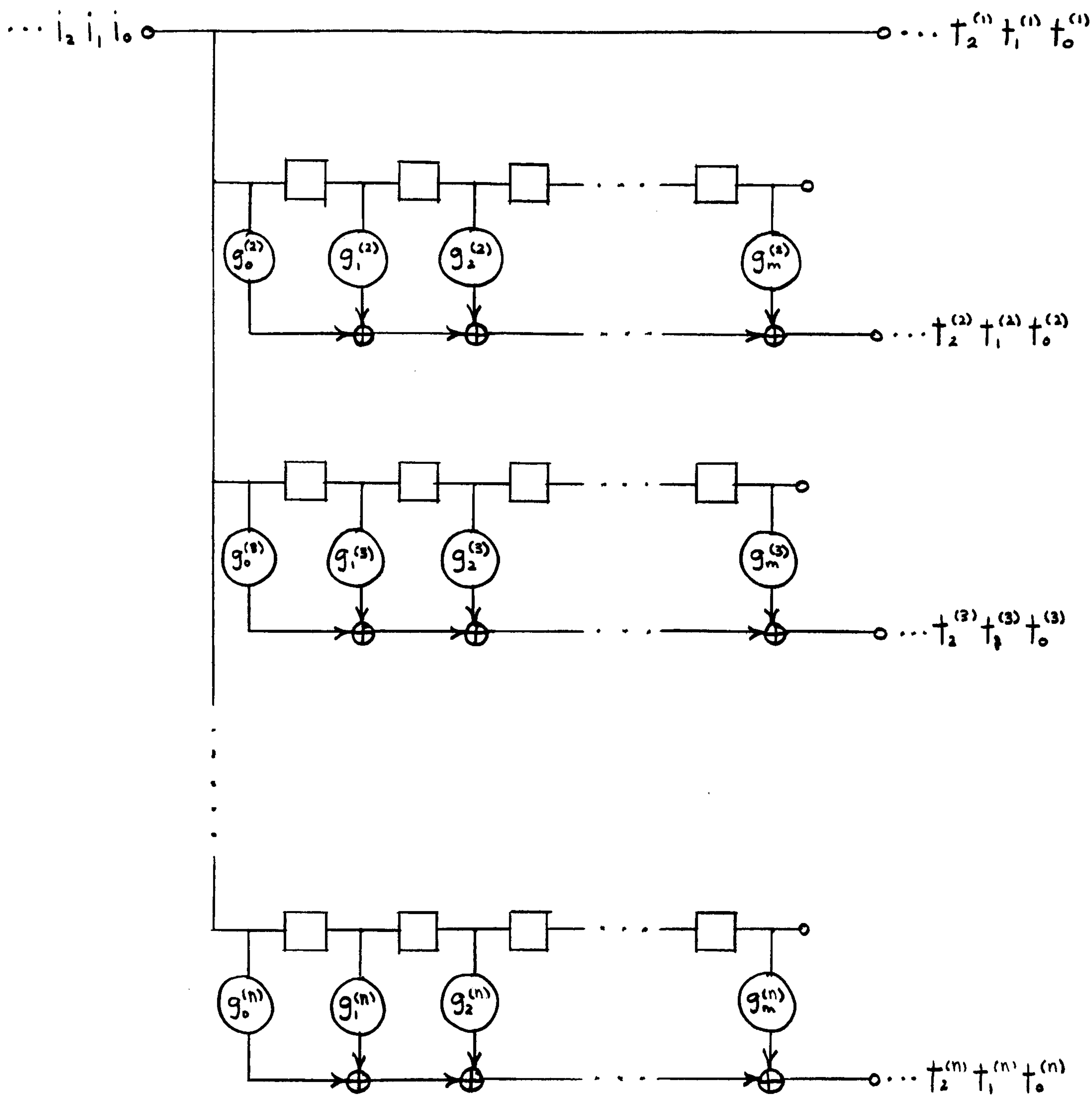


Figure 1. An encoding circuit for an $R = \frac{1}{n}$ canonic systematic convolutional code, $g_i^{(j)} \varepsilon GF(2)$, $i = 0,1,\ldots,m$, $j = 2,3,\ldots,n$.

TABLE I.

j	g_j	d_j	d_G	d_B	d_{LL}	j	g_j	d_j	d_G
0	1	2	2	2	2	36	0	13	11
1	1	3	3	3	3	37	0	13	12
2	0	3	3	3	3	38	0	13	12
3	1	4	4	4	4	39	0	13	12
4	0	4	4	4	4	40	1	14	12
5	1	5	4	5	5	41	0	14	13
6	0	5	4	5	5	42	0	14	13
7	0	5	5	6	5	43	1	15	13
8	1	6	5	6	6	44	0	15	13
9	0	6	5	6	6	45	0	15	13
10	0	6	5	7	7	46	0	15	14
11	1	7	6	7	7	47	0	15	14
12	0	7	6	8	7	48	1	16	14
13	0	7	6	8	8	49	0	16	14
14	0	7	6	8	8	50	0	16	15
15	0	7	7	9	9	51	0	16	15
16	1	8	7		9	52	0	16	15
17	0	8	7		9	53	1	17	15
18	0	8	7		9	54	0	17	15
19	0	8	8		9	55	0	17	16
20	1	9	8		10	56	1	18	16
21	0	9	8			57	0	18	16
22	0	9	8			58	0	18	16
23	0	9	9			59	0	18	16
24	1	10	9			60	0	18	17
25	0	10	9			61	0	18	17
26	0	10	9			62	1	19	17
27	1	11	9			63	0	19	17
28	0	11	10			64	0	19	18
29	0	11	10			65	1	20	18
30	0	11	10			66	0	20	18
31	1	12	10			67	0	20	18
32	0	12	11			68	0	20	18
33	0	12	11			69	0	20	19
34	0	12	11			70	0	20	19
35	1	13	11			71	1	21	19

d_G = $d_{\text{GILBERT BOUND}}$ d_B = d_{BUSSGANG} d_{LL} = $d_{\text{LIN and LYNE}}$

TABLE II.

				Algorithm A2			Algorithm A3			Algorithm A4		
j	d_G	d_B	d_{LL}	$g_j^{(2)}$	$g_j^{(3)}$	d_j	$g_j^{(2)}$	$g_j^{(3)}$	d_j	$g_j^{(2)}$	$g_j^{(3)}$	d_j
0	3	3	3	1	1	3	1	1	3	1	1	3
1	4	4	4	1	0	4	1	0	4	1	0	4
2	5	5	5	1	0	5	0	1	5	0	1	5
3	6	6	6	0	1	6	1	0	6	1	0	6
4	6	7	7	1	0	7	1	0	7	1	0	7
5	7	8	8	0	1	8	1	0	8	1	0	8
6	8	9	9	0	1	9	0	0	8	0	0	8
7	8		9	0	1	10	1	0	9	0	1	9
8	9		10	0	0	10	0	1	10	1	0	10
9	9		11	1	0	11	0	0	10	0	0	10
10	10		12	1	0	12	1	0	11	0	1	11
11	10		12	0	0	12	0	1	12	0	1	12
12	11		13	1	0	13	0	0	12	1	0	13
13	11		14	0	0	13	0	0	12	0	0	13
14	12		15	1	0	14	0	1	13	1	0	14
15	12		15	1	0	15	1	0	14	0	0	14
16	13		16	0	0	15	1	0	15	0	1	15
17	14		16	1	0	16	0	0	15	0	0	15
18				0	1	17	0	1	16	1	0	16
19				0	0	17	1	0	17	1	0	17
20				1	0	18	0	0	17	1	0	18
21				0	0	18	0	1	18	1	0	19
22				1	0	19	0	0	18	0	0	19
23				0	1	20	1	0	19	0	0	19
24				0	0	20	0	0	19	0	1	20
25				0	0	20	1	0	20	1	0	21
26				1	0	21	0	1	21	0	0	21
27				1	0	22	0	0	21	0	0	21
28				0	1	23	0	1	22	0	1	22
29				0	0	23	0	1	23	0	0	22
30				0	0	23	0	0	23	1	0	23
31				0	1	24	0	0	23	1	0	24
32				0	0	24	1	0	24	0	0	24
33				1	0	25	1	0	25	1	0	25
34				0	0	25	0	0	25	0	1	26
35				1	0	26	1	0	26	0	0	26

$d_G = d_{\text{GILBERT BOUND}}$ $\quad$ $d_B = d_{\text{BUSSGANG}}$ $\quad$ $d_{LL} = d_{\text{LIN and LYNE}}$

TABLE III.

j	$g_j^{(2)}$	$g_j^{(3)}$	$g_j^{(4)}$	d_j	d_G	d_{LL}
0	1	1	1	4	4	4
1	1	1	0	6	6	6
2	1	0	1	8	7	8
3	0	0	1	9	8	9
4	0	1	0	10	9	10
5	0	0	1	11	10	11
6	0	0	1	12	11	13
7	1	0	1	14	12	14
8	0	0	1	15	13	15
9	0	1	0	16	13	16
10	0	1	0	17	14	17
11	0	0	0	17	15	18
12	1	1	0	19	16	19
13	0	0	1	20		21
14	0	1	0	21		22
15	0	0	1	22		23
16	0	0	1	23		
17	0	0	1	24		
18	0	1	0	25		
19	0	1	0	26		
20	1	0	0	27		
21	0	0	1	28		
22	0	0	1	29		
23	0	0	1	30		
24	0	1	0	31		
25	0	0	1	32		
26	0	1	0	33		
27	0	1	0	34		
28	0	0	0	34		
29	0	0	1	35		
30	0	1	0	36		
31	0	1	0	37		
32	0	0	0	37		
33	1	1	0	39		
34	0	0	0	39		
35	0	0	1	40		

TABLE IV.

j	g_j	d_{free}
0	1	2
1	1	3
2	1	4
3	0	4
4	1	5
5	1	6
6	0	6
7	1	7
8	0	7
9	1	8
10	0	8
11	0	8
12	1	9
13	1	10
14	0	10
15	0	10
16	0	10
17	0	10
18	1	11
19	1	12
20	1	13
21	0	13
22	0	13
23	0	13
24	0	13
25	1	14
26	1	15
27	1	16
28	0	16
29	0	16
30	0	16
31	0	16
32	0	16
33	0	16
34	0	16
35	1	17

$d_G = d_{\text{GILBERT BOUND}}$ $\quad$ $d_{LL} = d_{\text{LIN and LYNE}}$