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^(t - 1) but not β^t 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(9) is the Hamming code, which is a one-error-correcting perfect code. Gorenstein, 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 Gorenstein, Peterson, and Zierler [1], which says that B(d) is never quasi-perfect when d > 7.

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

We will need the following lemmas.

**Lemma 1:** If d = 2^r - 1, 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 9.4 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 p_d and t_d denote the covering radius and actual error correcting ability of B(d), respectively, and let 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

\[ p_d - t_d > \frac{2^{r-s} - 3}{2^{r-s} - 1} (t_d + 1) \]

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

\[ p_d - t_d > \frac{2^{(r-1)/2} - 1}{2^{(r-1)/2} - 1} (t_d + 1) \]

**Proof:**

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

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

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

\[ p_d > 2^r - 2^s - 1 \]

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

\[ t_d < 2^r - 1 - 2^s - 1 \]

Combining (1) and (2) we have

\[ p_d - t_d > 2^r - 1 - 2^s - 1 \]

which combined with (2) gives

\[ p_d - t_d > \frac{(t_d + 1)(2^{r-s} - 3)}{2^{r-s} - 1} \]

This proves i).

Corollary: If t_d ≥ 2^r - 2^s - 1, then p_d - t_d ≥ \frac{1}{2} (t_d + 1).

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

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

Note that we have

\[ p_d - t_d > \frac{1}{2} (t_d + 1) \]

when d belongs to case i).

Hence we always have p_d - t_d > 1 since t_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 t_d > 1 and t_d ≠ 2^r - 1, then p_d - t_d > \frac{1}{2} (t_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 the received symbol stream. A bound is given for 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 synchronization and thus data loss. To avoid this problem, alternates 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 stream occurs? The answer is found in the following theorem.

**Theorem 1:** The maximum length of a transition-free run in the output of a convolutional code is given by

\[ L = \frac{2^r - 1}{2^r - 1 - s} \]

where

\[ 3 < r < m - 1 \]

and

\[ s = [r], [r] + 1, ... \]

**Proof:**

The proof is given in the following corollary.

**Corollary:** The maximum length of a transition-free run in the output of a convolutional code is given by

\[ L = \frac{2^r - 1}{2^r - 1 - s} \]

where

\[ 3 < r < m - 1 \]

and

\[ s = [r], [r] + 1, ... \]

**REFERENCES**


---

0018-9448/79/0200-0362$00.75 ©1979 IEEE
then? That is, how long a run of alternating symbols
\[ \cdots 01010101 \cdots \] 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 refers to discrete time intervals. In practice each sequence "starts" at some finite time; i.e., there is an index \( t \) such that \( t < t_s \) 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 \( x = x_0 D^t + x_1 D^{t+1} + \cdots \). We also use certain algebraic properties of these formal power series, e.g., \( D^s + D^{s+1} + \cdots = D^s / (1 + D) \).

II. CONVOLUTIONAL CODES WITH AN INFINITE RUN OF ALTERNATING SYMBOLS

Theorem 1: 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 \( v = [v_1, \ldots, v_n] \) of the rows \( g_i \) of \( G \) such that

\[ [v_1, \ldots, v_n] \equiv [0,1,\ldots,0,1] \pmod{1+D} \] for even \( n \)

\[ [v_1, \ldots, v_n] \equiv [1,0,\ldots,1,0] \pmod{1+D} \] for odd \( n \).

Proof: (Sufficiency): When \( n \) is even, consider the code word produced by the inputs \( a_i / (1 + D) \) applied to the rows \( g_i \), where \( v = \Sigma a_i g_i \). Note that this same codeword is produced by applying \( 1 / (1 + D) = 1111 \cdots \) to each row of the equivalent encoder whose rows are \( a g_i \). Thus after an initial transient the output will be \( v_1(1), \ldots, v_2(1) \) and since \( v_2(1) \equiv v_1(1) \pmod{1+D} \) the result follows. For \( n \) odd note that \( v(D) \equiv D v \pmod{1+D} \) 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 \( v(1) \equiv 1 \pmod{1+D} \). Thus after an initial transient the input sequences \( a_i / (1 + D^2) \) 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 \( 10 \cdots 10 \) or \( 01 \cdots 01 \). For definiteness, assume the former occurs. Then, if a codeword of \( C \) contains such an infinite run, there exists a codeword \( v \) such that

\[ u = b + D^s [1,0,\ldots,1,0] / (1+D) \]

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 a polynomial and \( v(D) \equiv [0,1,\ldots,0,1] \pmod{1+D} \).

Similarly, for \( n \) odd, \( C \) contains

\[ w = h^t + D^{s-t} [1,0,\ldots,1,D,1] / (1 + D^2) \]

Define \( v(D) \) as \((1 + D^2) w(D) \). It follows as above that

\[ v(D) \equiv D^s [1,D,\ldots,D,1] \pmod{1+D^2} \]

and the proof is complete.

If a basic encoder \( G \) is known for \( C \) then only \( 2^k \) (respectively, \( 4^k \)) linear combinations \( v = \Sigma a_i g_i \) 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 \( f = [f_1, \ldots, f_n] \) generates the dual code, where \( \gcd(f_1, \ldots, f_n) = 1 \). Then there is an infinite run of alternating symbols in some codeword of \( C \) if and only if

\( n \) even

\[ \Sigma f_j \equiv 0 \pmod{1+D} \]

\( n \) odd

\[ \Sigma f_j + D \Sigma f_{j+1} \equiv 0 \pmod{1+D^2} \]

Proof: Since \( f_j \) are all codewords of the dual code are linear combinations of shifts of

\[ \cdots 0 f_{j_0} f_{j_0+1} f_{j_0+2} \cdots \]

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.

\( n \) even

\[ \cdots 0 1 0 1 0 1 0 1 \cdots \]

\( n \) odd

\[ \cdots 0 1 0 1 0 1 0 1 \cdots \]

(\( \alpha = 1 \))

\[ f_{j_0} f_{j_0+1} f_{j_0+2} \cdots \]

(\( \alpha = 0 \))

(\( \text{constant} \))

In both cases the necessity of the above conditions is immediate. (For \( n \) odd the coefficients referred to are \( a,b \) from \( \Sigma f_j + D \Sigma f_{j+1} \equiv 0 \pmod{1+D^2} \).

On the other hand, the above conditions obviously guarantee the existence of a codeword \( \cdots 01010101 \cdots \) 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 \( t_0 \), then, by time \( t_0 + \delta \), where \( \delta \) is the overall constraint length, the encoders shift registers will be set 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 \( n-k \) by \( n-k \) subdeterminants is 1, then, if \( [f_1, \ldots, f_n] \) is any row of \( F \) it follows that \( [f_1, \ldots, f_n] = 1 \).

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

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

and the maximum run of alternating symbols in any codeword of \( C \) has length \( l = \min I(C) \). (Corollary to Theorem 2 below)

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 $C$ 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 $L(C_i) = L(C) = \infty$ 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). In Section IV these bounds are applied to some specific examples.

Suppose $[f_i, \ldots, f_s]$ is a generator matrix for an $(n, 1)$ convolutional code $C$ over $\mathbb{GF}(2)$ with $d = \text{max} (\text{deg} f_i)$. Then

$$f_0 f_1 f_2 \cdots f_{s-1} f_s$$

is its associated bit pattern. Let $s$ be the number of symbols occurring between the first and last nonzero symbols $f_i$ inclusively. If $(f_1, \ldots, f_s) = 1$, $s$ 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 $(n, n-1)$ convolutional code over $\mathbb{GF}(2)$ with generator matrix for its dual code given by $[f_i, \ldots, f_s]$, where $(f_i, \ldots, f_s) = 1$. 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

$$h(D) = \sum_{i=1}^{s+1} f_i D^i \equiv 1 + D \text{ modulo } (1 + D^2).$$

If $n$ is odd and

$$h(D) = 1 \text{ or } D \text{ modulo } (1 + D^2),$$

the maximum run of alternating symbols has length $L = s + 2n - 2$.

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

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

**Proof:** Suppose $n$ is even. Then, from Theorem 2, $\Sigma f_i \equiv f_{i+1} \equiv 1 \text{ modulo } 1 + D$. If there were an alternating run of length $s > s + n - 1$ it would have $s$ consecutive symbols which would have inner product zero with the bit pattern of the $f$. This contradicts $\Sigma f_i \equiv 1$, so $L \leq s + n - 2$. 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 $1, \ldots, s$ and $n + 1, \ldots, n + s$. 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 an argument similar to that 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, $h(D) \equiv 0 \text{ modulo } 1 + D^2$. If $h(D) \equiv 1 + D$ the proof above applies, so $L = s + n - 2$. If $h(D) \equiv 1$ or $D$ 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 $s > s + 2n - 1$ 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 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 $n, \ldots, n + s - 1$ of this run have inner product zero with the bit pattern of the $f$.

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 $C_i$ and the obvious property

$$\bigcap_{i=1}^{n-k} C_i = \{0\} \text{ for } i \leq n$$

from which it follows that the maximum run of alternating symbols in any codeword of $C$ has length $L = L(C) \leq \min L(C_i)$. Suppose $L(C_i) = \infty$ 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 $\mathbb{GF}(2)$ is given with basic generator matrix $G$. Let $\mu$ be the maximum degree of the $k \times k$ subdeterminants of $G$. Then either

$$L(C) = \infty \text{ or } L(C) \leq \begin{cases} n(\mu+2) - 2, & n \text{ even} \\ n(\mu+3) - 2, & n \text{ odd}. \end{cases}$$

**Proof:** Under these conditions $C_i^\perp$ has a generator matrix $F$ (a so-called minimal encoder for $C_i$) all of whose entries are of degree $\leq \mu$. 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 codeword can be determined by replacing row $i$ of $F$ in turn by the sum of row $i$ and row $j$, for $j = 1, \ldots, n-k$ ($j \neq i$). 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. Examples

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

$$\begin{bmatrix} D^3 + D^2 + D + 1 \\ D^2 + D + 1 \\ D + 1 \end{bmatrix}.$$
+D^5 \cdot P^9, D + D^2 + D^3 + D^6 + D^7 + D^8 + D^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 \( L \leq 16 \) (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 \( n \mu + k + 1 \) which compares reasonably well with the bounds given by the Corollary. For example, consider the \((n,k)\) convolutional encoder

\[
G = \begin{bmatrix}
I & 0' \\
0 & 0 \\
0 & p q p q p q \\
\end{bmatrix}
\]

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

Here \( p = p(D) = 1 + D + D^2 + D^6 \) and, for \( n \) even, \( q = q(D) = 1 + D^2 + D^6 \) (\( \mu = 3 \)) while for \( n \) odd \( q(D) = 1 + D^3 + D^6 \) (\( \mu > 3 \)). \( G \) 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 \( x(0), \ldots, x(k) \) properly. For example, let \( n = 8, k = 4, \) and \( \mu = 3 \), then the bit pattern associated with the bottom row of \( G \) is

\[
00011111 00001001 00000100 00011111.
\]

So if \( x(0) = 1 + D^2 + D^3 \) \((-10110\ldots)\) and \( x(0) = 1 + D^2 + D^3 + D^4 \) \((-10110\ldots)\) the codeword generated by \( G \) is

\[
00011111 01010101 01010100 01010011 01011111.
\]

which, starting with its 8th symbol, has an alternating run of length 29 = 8 + 4 + 11 obviously \( x(0), \ldots, x(k-1) \) 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 \( x(k) \) is the critical one. For \( n \) even, \( k \) even, and \( \mu \) odd, \( x(k) = 1 + D^3 + D^4 + \cdots + D^{\mu-1} + D^a \). 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 \( G = \left[ g_1, g_2, g_3 \right] \) or \( \left[ g_1, g_2, g_3, g_4 \right] \) with \( g_1 = 1 + D^2 + D^4 + D^6 + D^8, g_2 = 1 + D^2 + D^4 + D^6 + D^8 \) and \( g_3 = 1 + D^2 + D^4 + D^6 + D^8 \). 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 \( L < 10.8 \) for \( n = 14 \) and \( k = 2 \). 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

\[
F = \left[ D + D^3 + D^5, D + D^3 + D^5, D + D^3 + D^5 \right], \quad h(D) \equiv 1 + D^2 + D^3 + D^4, \quad h(D) = 0.
\]

Apply Theorem 3 to the first row of \( F \). Here \( s = 11 \) so \( L < 14 \). A finite codeword with an alternating run of length 12 is generated from \( G \) by the input \( x(0) = 1 + D + D^2 + D^4 + D^7 \) \((-\ldots)\) 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 \((2,4)\) convolutional code whose generator matrix modulo 1 + D is

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

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

## A Note on Optimal Quantization

### JAMES A. BUCKLEW AND NEAL C. GALLAGHER, JR., MEMBER, IEEE

**Abstract**—For a general class of optimal quantizers the variance of the output is less than that of the input. Also the mean value is preserved by the quantizing operation.

### I. INTRODUCTION

J. Max [1] is generally credited with being the first to consider the problem of designing a quantizer to minimize a distortion measure given that the input statistics are known. Max derives necessary conditions for minimizing the mean square quantization error. These results are summarized in the following equations:

\[
y_j = \frac{1}{2} \left( x_j + x_{j+1} \right) = y_j
\]

where \( f(x) \) is the probability density of the variable to be quantized and \( P(x_{j-1} < x \leq x_j) \) is the probability that \( x \) lies in the interval \( (x_{j-1}, x_j) \). The \( y_j \) are output levels and the \( x_j \) are the break points where an input value between \( x_{j-1} \) and \( x_j \) is quantized to \( y_j \). Fleischer [2] later gave a sufficient condition for Max's equations to be the optimal set.

Typically, the above equations are intractable except for simple input densities, causing some researchers to derive approximate formulae for some common densities. Roe [3] derives an approximation for the input interval endpoints assuming that the widths of these intervals are small, i.e., the number of output levels is large. Wood [4] derives a result which states, in effect, that the variance of the output of a minimum mean-square error quantizer should be less than the input variance. He also states that the significance of his result is that the signal and noise are dependent and that no pseudo-independence of the sort considered by Widrow [4] is possible.

However, Wood's derivation assumes the input density to be five times differentiable and that the quantizer input intervals be very small in order to truncate various Taylor series expansions. Furthermore, the derived expression for the output variance is dependent upon the input interval lengths and the input probability density function evaluated at the midpoints of these intervals.

In this note we derive a generalization of Wood's results that eliminates a number of his approximations and generalizes the results to apply to more than just Max quantizers.

## REFERENCES