Insertion/Deletion Correction with Spectral Nulls

H. C. Ferreira, *Member, IEEE*, W. A. Clarke, A. S. J. Helberg, K. A. S. Abdel-Ghaffar, and A. J. Han Vinck

***Abstract*— Levenshtein proposed a class of single insertion/deletion correcting codes, based on the number-theoretic construction due to Varshamov and Tenengolt's. We present several interesting results on the binary structure of these codes, and their relation to constrained codes with nulls in the power spectral density function. One surprising result is that the higher order spectral null codes of Immink and Beenker are subcodes of balanced Levenshtein codes. Other spectral null subcodes with similar coding rates, may also be constructed. We furthermore present some coding schemes and spectral shaping markers which alleviate the fundamental restriction on Levenshtein's codes that the boundaries of each codeword should be known before insertion/deletion correction can be effected.**

***Index Terms*—Insertion/deletion correction, spectral nulls, constrained codes, balanced codes.**

I. Introduction

In 1965, Varshamov and Tenengolt's [1] proposed the following code construction to correct a single *asymmetrical* error on a channel where the probability of the symbol *one* turning into a *zero* is considerably less than the probability of a *zero* turning into a *one*, or *vice versa*. Let $\boldsymbol{x} = (x_1, x_2, \cdots, x_n)$, $x_i \in \{0, 1\}$ denote a binary codeword and C the codebook. Varshamov and Tenengolt's required that

$$\boldsymbol{x} \in C \Longleftrightarrow \sum_{i=1}^{n} i\, x_i \equiv a \pmod m \tag{1}$$

for some fixed integers a and m, where $m \geq n+1$. Note that in (1), the n-dimensional vector space is partitioned into m different codebooks, all having the desirable error correction capability and we shall denote each of these codebooks with $C(n, m, a)$. We shall refer to the construction in (1) and other similar constructions as *number-theoretic* code constructions.

In 1966, Levenshtein [2] described two classes of codes capable of correcting a single insertion or deletion error in a codeword. Such an error results in the deletion of a bit in a random position, or the insertion of a random bit in a random position, and it, respectively, changes the length of the received word to $n-1$ or $n+1$.

Briefly, Levenshtein noted that by setting $m \geq n+1$ in (1), a single insertion/deletion error can be corrected under the assumption that the boundaries of each codeword, i.e., the location of bits x_1 and x_n, are known. This single insertion/deletion correction capability is denoted with the parameter $s = 1$. Furthermore, it is shown in [2] that for $m \geq 2n$, either a single insertion/deletion error or a single *reversal* error (i.e., modulo 2 *additive* error) can be corrected, thus $s = 1$ or $t = 1$, if parameter t denotes the reversal error-correction capability. Levenshtein furthermore showed that the cardinality of the first class of codes is lower-bounded by

$$|C(n, m, a)| \geq \frac{2^n}{n+1}. \tag{2}$$

Subsequently, Ginzberg [3] proved that the code cardinality can be maximized by setting $a = 0$ and minimized with $a = 1$.

The concept of a subword obtained when deleting bits from a codeword also plays an important role when investigating the correction of deletions/insertions [2]. If a code corrects s insertions/deletions, the following restriction is imposed on the length of the largest common subword $\delta(\boldsymbol{x}, \boldsymbol{y})$ obtained from codewords $\boldsymbol{x}$ and $\boldsymbol{y}$:

$$|\delta(x, y)| < n - s. \tag{3}$$

For the purpose of this correspondence, let us return to (1). We note here that first the series $\sum i x_i$, which sums all indices i where $x_i = 1$, represents the *codeword moment*, also more precisely referred to as the *first-order moment* [4]. By considering the integer codeword moment $\sum i x_i$, *before* it is reduced modulo m, as is done in (1) and in most investigations, we present some new insight in the binary structure of Levenshtein's codes in Section II. For related work refer to [5]–[9].

II. The Binary Structure of Levenshtein's Insertion/Deletion Correcting Codes

In the rest of this correspondence we investigate Levenshtein's first class of codes in [2]. Unless otherwise indicated, we set $m = n+1$ and $a = 0$ to maximize the code cardinality. Using the notation introduced in Section I, the results in this section thus mainly pertain to the structure of Levenshtein's $C(n, n+1, 0)$ class of codes, although some generalizations to $C(n, m, a)$, $m \neq n+1$, $a \neq 0$, are also presented. In later sections, we show that constant-weight w subcodes of the $C(n, n+1, 0)$ codes, which we shall denote by $C(n, n+1, 0, w)$, are also important for other coding applications, such as the creation of spectral nulls. We shall furthermore apply the results of this section in new code constructions and coding schemes.

Let

$$(x_1, x_2, \cdots, x_n) \in C(n, n+1, a) \tag{4}$$

and

$$\sigma = \sum_{i=1}^{n} i x_i\,. \tag{5}$$

Then

$$0 \leq \sigma \leq \frac{n\,(n+1)}{2} \tag{6}$$

and since

$$\sigma \equiv a \bmod (n+1) \tag{7}$$

σ belongs to a finite set of at most $\lfloor n/2 \rfloor + 1$ integers that differ by at least $n+1$.

Note that if $m = n+1$ and $a = 0$, the all-*zeros* codeword is included in the codebook. If, furthermore, n is even, the all-*ones* codeword is also included in $C(n, n+1, 0)$.

Manuscript received December 26, 1994; revised July 25, 1996. This work was supported in part by the South African Foundation for Research Development, by the Deutscher Akademischer Austauschdienst, and the National Science Foundation under Grant NCR-9115423. The material in this correspondence was presented in part at the 6th Joint Swedish–Russian International workshop on Information Theory, Mölle, Sweden, August 23–27, 1993; at the 1994 IEEE International Symposium on Information Theory, Trondheim, Norway, June 27–July 1, 1994; and at the 1996 International symposium on Information Theory and Its Applications, Victoria, Canada, September 17–20, 1996.

H. C. Ferreira, W. A. Clarke, and A. S. J. Helberg are with the Department of Electrical and Electronic Engineering, Rand Afrikaans University, P.O. Box 524, Aucklandpark, 2006, South Africa.

K. A. S. Abdel-Ghaffar is with the Department of Electrical and Computer Engineering, University of California, Davis, CA 95616 USA.

A. J. Han Vinck, is with Institut für Experimentelle Mathematik, Universität GHS Essen, 45326, Essen, Germany.

Publisher Item Identifier S 0018-9448(97)01287-X.

First, we are interested in the Hamming weight structure of Levenshtein's first class of codes. Using the theory of integer partitions [10], a simple generating function can be set up to evaluate the number of codewords of weight w. This follows from the observation that each nonzero codeword can be associated with a partition with no repeated parts, and no part greater than n, of the integer σ, which represents the unreduced codeword moment. Consequently, letting t^i represent the integer part, or codeword index i, and u one unit of weight, we can set up the following generating function for determining the weight spectrum:

$$f(u, t) = \prod_{i=1}^{n} (1 + ut^i). \tag{8}$$

The number of codewords $N(w)$ of weight w can be determined by summation of the coefficient(s) of $u^w t^\sigma$, $0 \le \sigma \le n(n+1)/2$, in (8).

Several previous workers investigated the cardinality and weight spectrum of Varshamov–Tenengolt's codes: see e.g., [3], [5], [7]. It is also interesting to note that Dickson [11, pp. 87–88] presents a formula which can be used to compute the cardinality of $C(n, n+1, a, w)$, attributed to R. D. von Sterneck (1902). For $a = 0$, it yields

$$|C(n, n+1, 0, w)| = \frac{(-1)^w}{n+1} \sum_{d|n+1} \phi(d)(-1)^{\lfloor w/d \rfloor} \binom{(n+1)/d - 1}{\lfloor w/d \rfloor} \tag{9}$$

where ϕ is Euler's function.

For certain weights w, we are also able to obtain $N(w)$ in explicit form, such as for $w = 2$.

Proposition 1: Let $\boldsymbol{x} \in C(n, n+1, 0, 2)$. The two binary ones in each $\boldsymbol{x}$ occur in a symmetrically spaced pair at indices i and $n-i+1$ and $N(2) = \lfloor n/2 \rfloor$.

Proof: If there are two ones in any binary word of length n bits, the moment is bounded by

$$3 \le \sum_{i=1}^{n} i\, x_i \le 2n - 1.$$

Consequently, the moments of all codewords of $C(n, n+1, 0, 2)$ have to satisfy

$$\sum_{i=1}^{n} i x_i = n + 1.$$

This is only possible if the ones in a codeword occur in a symmetrically spaced pair at indices i and $n - i + 1$. Consequently, there are $\lfloor n/2 \rfloor$ codewords with $w = 2$. □

We next investigate when the binary complement of codeword $\boldsymbol{x}$, i.e., $\overline{\boldsymbol{x}}$ with $\overline{x}_i = x_i + 1 (\mathrm{mod}\, 2)$, is included in the codebook. This insight also enables us to further evaluate the weight spectrum.

Proposition 2: Let $\boldsymbol{x} \in C(n, n+1, a)$ and let n be even. Then $\overline{\boldsymbol{x}} \in C(n, n+1, -a)$.

Proof:

$$\sum_{i=1}^{n} i x_i + \sum_{i=1}^{n} i \overline{x}_i = \frac{n(n+1)}{2} \equiv 0 \bmod (n+1).$$

Thus

$$\sum_{i=1}^{n} i \overline{x}_i \equiv - \sum_{i=1}^{n} i x_i \equiv -a \bmod (n+1). \qquad \square$$

Corollary 1: Let $\boldsymbol{x} \in C(n, n+1, 0)$ and let n be even. Then $\overline{\boldsymbol{x}} \in C(n, n+1, 0)$. □

The codewords thus occur in complementary pairs. This leads to the following corollary:

Corollary 2: For $C(n, n+1, 0)$ with n even, the weight spectrum is symmetrical, i.e.,

$$N(n-i) = N(i), \qquad \text{for } 0 \le i \le n/2 - 1. \qquad \square$$

Corollary 3: If n is even and $a \not\equiv 0 \bmod (n+1)$, then $\boldsymbol{x}$ and its binary complement $\overline{\boldsymbol{x}}$ are in different codebooks. □

The previous results can be generalized as follows:

Proposition 3: If $\boldsymbol{x} \in C(n, m, a)$, then $\overline{\boldsymbol{x}} \in C(n, m, e-a)$ where $n(n+1)/2 \equiv e \,(\mathrm{mod}\, m)$.

Proof:

$$\sum_{i=1}^{n} i x_i + \sum_{i=1}^{n} i \overline{x}_i = \frac{n(n+1)}{2} \equiv e \,(\mathrm{mod}\, m).$$

Since

$$\sum_{i=1}^{n} i x_i \equiv a \,(\mathrm{mod}\, m)$$

it follows that

$$\sum_{i=1}^{n} i \overline{x}_i \equiv (e-a) \,(\mathrm{mod}\, m). \qquad \square$$

We now investigate the Hamming distance structure. First, we present some results which apply to all $s = 1$ insertion/deletion correcting codes.

Proposition 4: Any code correcting $s = 1$ insertions/deletions has $d_{\min} > 1$.

Proof: If codewords $\boldsymbol{x}$ and $\boldsymbol{y}$ have Hamming distance $d = 1$, a common subword of length $|\delta(\boldsymbol{x}, \boldsymbol{y})| = n - 1$ can be obtained by deleting the distance building bit from each word. However, this contradicts $|\delta(\boldsymbol{x}, \boldsymbol{y})| < n - 1$, as required in (3). □

Since the *all-zero* word is included in the $C(n, n+1, 0)$ codebook, we have the following corollary:

Corollary 4: There can be no codewords of weight $w = 1$ in the $C(n, n+1, 0)$ codebook, i.e., $N(1) = 0$. □

The following result again applies to any $s = 1$ insertion/deletion correcting code.

Theorem 1: Any code correcting $s = 1$ insertion/deletion has the following property: two codewords with weight j and $j+1$ have $d \ge 3$.

Proof: Two codewords with weights j and $j+1$ have odd Hamming distance, d. Since $d > 1$ (Proposition 4), $d \ge 3$. □

The following result is known to hold for the Varshamov–Tenengolt's construction (see, e.g., [5], [9]):

Theorem 2: Let $C(n, m \ge n, a)$ be a Varshamov–Tenengolt's code. Two equal-weight codewords have $d \ge 4$. □

Corollary 5: The Hamming distance properties in Theorems 1 and 2 also holds for Levenshtein's codes. □

We now investigate the image $\hat{\boldsymbol{x}}$ of codeword $\boldsymbol{x}$. We define the image of a codeword $\boldsymbol{x}$ to be the word $\hat{\boldsymbol{x}}$, found by writing the bits in $\boldsymbol{x}$ in reverse order, i.e.,

$$\hat{x}_j = x_{n+1-j}, \qquad 1 \le j \le n.$$

The image word $\hat{\boldsymbol{x}}$ will play a role in coding schemes presented in later sections. The proof of the following proposition is straightforward:

Proposition 5: Let $\boldsymbol{x} \in C(n, n+1, a)$. Then it holds for the image $\hat{\boldsymbol{x}}$ of $\boldsymbol{x}$, that $\hat{\boldsymbol{x}} \in C(n, n+1, -a)$. □

Corollary 6: If $a = 0$, it follows that $\boldsymbol{x}, \hat{\boldsymbol{x}} \in C(n, n+1, 0)$. □

Note that some codewords of length n, n even, may be their own images, i.e.,

$$x_j = x_{n+1-j}, \qquad 1 \le j \le n.$$

We shall call these codewords symmetrical.

Cyclic shifts of codewords play an important role in coding theory and in some synchronization recovery schemes. Of particular interest is if a cyclic shift of $\boldsymbol{x} \in C(n, m, a)$ will also be in the codebook. The proof of the following proposition is again straightforward:

Proposition 6: By cyclic shifting a weight w codeword $\boldsymbol{x} \in C(n, m, a)$ to the right (left), the resulting word $\boldsymbol{x}^r(x^l)$ falls into a new codebook as follows:

$$\begin{aligned} 1)\quad & x_n = 0\colon \boldsymbol{x}^r \in C(n, m, a+w)\\ & x_1 = 0\colon \boldsymbol{x}^l \in C(n, m, a-w)\\ 2)\quad & x_n = 1\colon \boldsymbol{x}^r \in C(n, m, a+w-n)\\ & x_1 = 1\colon \boldsymbol{x}^l \in C(n, m, a-w+n). \end{aligned} \qquad \square$$

Corollary 7: For $\boldsymbol{x} \in C(n, n, 0, n/2)$, a codeword in the same codebook is obtained after two cyclic shifts in the same direction. □

Corollary 8: For any $\boldsymbol{x} \in C(n, n+1, 0, n/2)$, shifting $\boldsymbol{x}$ in the same direction, results in a codeword which belongs to one of two other codebooks, as determined by x_n for right shifts and x_1 for left shifts. □

III. Some Constant-Weight Subcodes of Levenshtein's Insertion/Deletion Correcting Codes

In this section, we consider constant-weight codes, including the balanced codes which have a spectral null at dc.

A. Bounds on the Cardinality of Constant-Weight Codes

Constant-weight codes have been investigated extensively in the past. The cardinalities of these codes are usually denoted by $A(n, d, w)$, where n is the length of the binary block code with a minimum Hamming distance of $d_{\min}$ (denoted here with $d = d_{\min}$) and weight w. Upper bounds on the cardinalities of these codes appear, e.g., in [6], [12], [13].

The Varshamov–Tenengolt's construction has previously been employed to construct good constant-weight codes with $d_{\min} = 4$ and $d_{\min} = 6$ (see, e.g., [5], [6].) The rate efficiency of the constant-weight Levenshtein codes *per se* can be appreciated if the cardinalities are compared with the latest results on $A(n, d, w)$ in [12]. From Corollary 5 it can be seen that the constant-weight Levenshtein subcodes will have minimum Hamming distance $d_{\min} = 4$.

By evaluating the generating function in (8), it can be seen that the cardinality of the class of $C(n, n+1, 0, w)$ codes compares favorably to the upper bounds. In fact, the achievable code rates of interest for implementation are often the same, i.e.,

$$R = \frac{k}{n} = \frac{\lfloor \log_2 |C(n, n+1, 0, w)| \rfloor}{n} = \frac{\lfloor \log_2 A(n, 4, w) \rfloor}{n} \tag{10}$$

or differ at the most with one unit in the numerator. It is thus often possible to construct an optimal rate $R = k/n$, $d_{\min} = 4$ constant-weight code, and also be able to correct insertions/deletions.

B. Balanced Block Codes with $n = 16$, $d_{\min} = 4$

In [14], we presented the construction of an $(n, k) = (16, 8)$ balanced or *dc-free* block code with $d_{\min} = 4$. Blaum improved on this result by constructing an $(n, k) = (16, 9)$ balanced code in [15], while the bounds in, e.g., [12] indicates the existence of an $(n, k) = (16, 10)$ code. Immink and Beenker [4] proposed an $n = 16$ dc^2-constrained code with cardinality 526 for this purpose. By evaluating (8) for $n = 16$ and $w = 8$, it can be seen that an $(n, k) = (16, 9)$ balanced code with $d_{\min} = 4$ can be constructed using $C(16, 17, 0, 8)$. This code improves on the results in [14] and [15], since it can correct either a single insertion/deletion in error or the single reversal error considered in [14] and [15]. Furthermore, it represents a simpler construction and a simple decoder exists, as can be seen in Section V. In the next section, we shall show that Immink and Beenker's code is in fact one possible subcode of this code.

Using a computer to trim selected code words from $C(16, 17, 0, 8)$ which has cardinality 758, we can reduce the maximum runlength of the $(n, k) = (16, 9)$ code, which needs a codebook with 512 codewords, to a maximum runlength of 5 bits (in comparison to 6 for Immink and Beenker's code). The new code has a running digital sum bounded by

$$-5 \le z_j \triangleq \sum_{i=1}^{j} x_i \le 5, \qquad 1 \le j \le n,\ x_i \in \{-1, +1\}$$

where we assume a mapping of $x_i \in \{0, 1\}$ onto $x_i \in \{-1, +1\}$.

IV. Some Spectral Null Subcodes of Levenshtein's Insertion/Deletion Correcting Codes

The balanced codes in the previous section have first-order spectral nulls at dc. In this section, we consider block codes with either higher order spectral nulls, or spectral nulls at other frequencies. Note that once we turn to the topic of spectral nulls, the polar representation of the binary symbols, i.e., $x_i \in \{-1, +1\}$ must be used. In contrast, when codes with error-correcting capabilities are investigated, the $x_i \in \{0, 1\}$ representation is often used. By using the straightforward mapping of $\{0, 1\}$ onto $\{-1, +1\}$, the results of the previous sections, can be applied to this section.

A. Subcodes with Higher Order Spectral Zeros at Zero Frequency

Immink and Beenker [4] described a subclass of balanced codes, sometimes called higher order dc-constrained codes, which suppresses low-frequency components in the code's power spectral density function. Briefly, they investigated codewords $\boldsymbol{x}, x_i \in \{-1, +1\}$, of Kth-order zero disparity, i.e., with the first $K + 1$ codeword moments at dc all zero, or

$$u_k = \sum_{i=1}^{n} i^k x_i = 0 \qquad k \in \{0, 1, \cdots, K\}. \tag{11}$$

This class of codes then has the property that the first $2K + 1$ derivatives of the power spectral density vanish at dc or $\omega = 0$, and they show that codewords have minimum Hamming distance

$$d_{\min} \ge 2(K + 1). \tag{12}$$

Furthermore, it is shown in [4] that n should be divisible by 4, and that

$$\boldsymbol{x} \in C \Leftrightarrow \sum_{x_i=+1} i x_i = -\sum_{x_i=-1} i x_i = n(n+1)/4. \tag{13}$$

If we assume a straightforward mapping of $\{-1, +1\}$ onto $\{0, 1\}$, set $n = 4e$, and consider the first-order spectral null, it follows that for any one of the Kth-order zero disparity codes

$$\boldsymbol{x} \in C \Leftrightarrow \sum_{i=1}^{n} i x_i = e(n+1) = 0 \bmod (n+1). \tag{14}$$

Consequently, the moment of any higher order dc-constrained code is $\sigma = e(n+1)$ and the code is a subcode of $C(n, n+1, 0, n/2)$, which may also contain codewords with other moments.

We thus conclude that the class of higher order dc-constrained codes are capable of correcting a single insertion/deletion error, a property which seems to be hitherto unknown.

We now present a result on the binary structure of higher order *dc-constrained* codes.

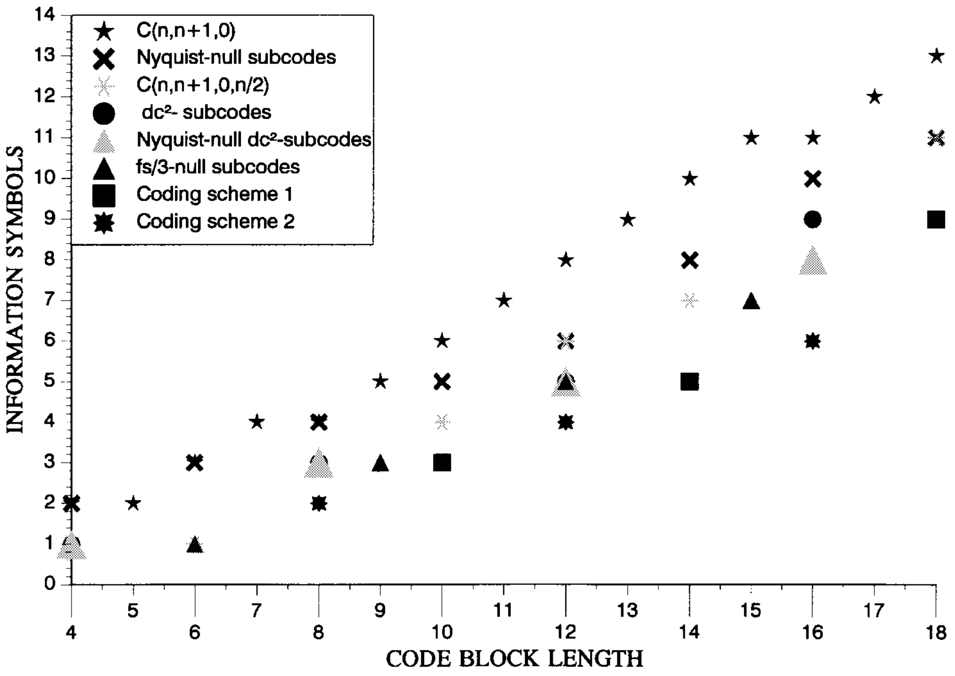

Fig. 1. The number of information symbols versus codeword length of some Levenshtein subcodes.

Proposition 7: Let $\boldsymbol{x}$ be dc^2-constrained. Then $\overline{\boldsymbol{x}}$ is also dc^2-constrained.

Proof:

$$\sum_{i=1}^{n} ix_i + \sum_{i=1}^{n} i\overline{x}_i = \frac{n(n+1)}{2}$$

$$\therefore \sum_{i=1}^{n} i\overline{x}_i = \frac{n(n+1)}{2} - \frac{n(n+1)}{4} = \frac{n(n+1)}{4}$$

which indicates that $\overline{\boldsymbol{x}}$ is also dc^2-constrained. □

B. Nyquist Null dc^2-Constrained Subcodes

It is possible to further constrain the dc^2-constrained codes in the previous section, in order to obtain codes with a null at the Nyquist frequency f_s. This class of codes is sometimes called minimum-bandwidth codes. The Nyquist null dc^2-constrained subcodes of $C(n, n+1, 0)$ were first described in [16].

Following Kim [17], who investigated a zeroth-order null at f_s, we require in addition to (11) and (13), that the running alternate sum for each codeword is zero, or

$$RAS = \sum_{i=1}^{n} (-1)^i x_i = 0, \qquad x_i \in \{-1, +1\}. \tag{15}$$

By applying the theory of integer partitions, the structure of codewords can be analyzed and a generating function for enumerating codewords can be set up. Refer back to (13) and consider the indices p_i where $x_i = +1$. Since $w = n/2$, there are $n/2$ of these indices which partition the integer $n(n+1)/4$ such that there are exactly $n/2$ parts, with no repeated parts and no part larger than n. The same structure applies to the indices n_i where $x_i = -1$.

A moment's reflection reveals that (15) can be written in terms of these indices, as

$$RAS = N(p^e) - N(p^o) + N(n^o) - N(n^e) = 0 \tag{16}$$

where the superscript e denotes even and o odd.

Since each codeword is balanced, we may also write

$$RDS = N(p^e) + N(p^o) - N(n^e) - N(n^o) = 0. \tag{17}$$

Furthermore, since the codeword length n is even, there are equal numbers of even and odd indices, thus

$$N(n^e) + N(p^e) = N(n^o) + N(p^o). \tag{18}$$

There are n indices in total, or

$$N(p^e) + N(p^o) + N(n^e) + N(n^o) = n. \tag{19}$$

Solving (16)–(19) yields

$$N(p^e) = N(p^o) = N(n^e) = N(n^o) = n/4. \tag{20}$$

The Nyquist null dc^2-constrained codes are thus defined by

$$\boldsymbol{x} \in \boldsymbol{C} \Leftrightarrow \sum_{i=1}^{n/4} p_i^o + \sum_{i=1}^{n/4} p_i^e = \sum_{i=1}^{n/4} n_i^e + \sum_{i=1}^{n/4} n_i^o = n(n+1)/4. \tag{21}$$

Referring back to (8), and assuming a mapping of $\{-1, +1\}$ onto $\{0, 1\}$, the generating function for this class of codes can be set up. Let t account for one unit of weight, u^i for index i and v for p^o

$$f(u, t, v) = (1+utv)(1+u^2t)(1+u^3tv)\cdots(1+u^nt) = \prod_{i=1}^{n} (1 + u^i t v^{i-2\lfloor i/2 \rfloor}). \tag{22}$$

Here the coefficient of $u^{n(n+1)/4}t^{n/2}v^{n/4}$ should be evaluated to enumerate this class of codes.

The code rate is depicted in Fig. 1, from which it can be seen that the rate loss is not significant when further constraining the dc^2-constrained codes. (For $n \le 12$, there is no rate loss.) Note that every $\boldsymbol{x} \in C(8, 9, 0, 4)$ satisfies both the dc^2 and Nyquist null properties.

We next present some propositions on the binary structure of Nyquist null dc^2-constrained codes.

Proposition 8: If n divides by 4, all symmetrical codewords $\boldsymbol{x} \in C(n, n+1, 0, n/2)$ with $x_i = x_{n+1-i}$, $1 \le i \le n$, are included in the dc^2-code of length n.

Proof: Let each codeword $\boldsymbol{x}$ constitute of two equal-length subwords, $(x_1, \cdots, x_{n/2})$ and $(x_{n/2+1}, \cdots x_n)$. Since $\boldsymbol{x}$ is symmetrical, each subword contains $n/4$ *ones*. Thus there are $n/4$ pairs of *ones* in $\boldsymbol{x}$, which each contribute $i + (n+1-i) = n+1$ to the moment of $\boldsymbol{x}$. Consequently the total moment is

$$\sum_{i=1}^{n} ix_i = n(n+1)/4$$

which is the moment of a dc^2 codeword [4]. Hence, $\boldsymbol{x}$ is dc^2-constrained. □

In the previous proposition, we divided each codeword into two equal-length subwords. We now investigate another class of even-length codewords where the second subword is the inverse of the first subword, i.e.,

$$x_{n/2+i} = \overline{x}_i\,. \tag{23}$$

Proposition 9: If n divides by 4, all codewords $\boldsymbol{x} \in C(n, n+1, 0, n/2)$ which have the inverse property, i.e., $x_{n/2+i} = \overline{x}_i$, are included in the dc^2-code of length n.

Proof:

$$\begin{aligned}\sum_{i=1}^{n} ix_i &= \sum_{i=1}^{n/2} ix_i + \sum_{j=n/2+1}^{n} jx_j \\ &= \sum_{i=1}^{n/2} ix_i + \sum_{i=1}^{n/2} \left(i + \frac{n}{2}\right)\overline{x}_i \\ &= \sum_{i=1}^{n/2} i(x_i + \overline{x}_i) + \sum_{i=1}^{n/2} \frac{n}{2}\overline{x}_i \\ &= \sum_{i=1}^{n/2} i + \frac{n}{4}\cdot\frac{n}{2} \\ &= \frac{n/2(n/2+1)}{2} + \frac{n}{4}\cdot\frac{n}{2} \\ &= \frac{n(n+1)}{4}\end{aligned}$$

which is the moment of a dc^2 codeword. Hence $\boldsymbol{x}$ is dc^2-constrained. □

Proposition 10: All symmetrical balanced codewords $\boldsymbol{x} \in C(n, n+1, 0, n/2)$, have a null at the Nyquist frequency.

Proof:

$$\begin{aligned}\sum_{i=1}^{n}(-1)^i x_i &= \sum_{i=1}^{n/2}(-1)^i x_i + \sum_{i=n/2+1}^{n}(-1)^i x_i \\ &= \sum_{i=1}^{n/2}(-1)^i x_i + \sum_{j=1}^{n/2}(-1)^{n+1-j} x_{n+1-j} \\ &= \sum_{i=1}^{n/2}(-1)^i x_i - \sum_{j=1}^{n/2}(-1)^{j} x_{n+1-j} \\ &= \sum_{i=1}^{n/2}(-1)^i x_i - \sum_{j=1}^{n/2}(-1)^{j} x_{j} \\ &= 0.\end{aligned}$$

□

Proposition 11: If n divides by 4, all codewords $\boldsymbol{x} \in C(n, n+1, 0, n/2)$ which have the inverse property, i.e., $x_{n/2+i} = \overline{x}_i$, $1 \le i \le n/2$, have a null at the Nyquist frequency.

Proof:

$$\sum_{i=1}^{n}(-1)^i x_i = \sum_{i=1}^{n/2}(-1)^i x_i + \sum_{i=n/2+1}^{n}(-1)^i x_i.$$

Substitute $i = j + n/2$ in the last term. Thus

$$\begin{aligned}\sum_{i=1}^{n}(-1)^i x_i &= \sum_{i=1}^{n/2}(-1)^i x_i + \sum_{j=1}^{n/2}(-1)^{n/2+j} x_{n/2+j} \\ &= \sum_{i=1}^{n/2}(-1)^i x_i + \sum_{j=1}^{n/2}(-1)^{j} x_{n/2+j} \\ &= \sum_{i=1}^{n/2}(-1)^i x_i + \sum_{i=1}^{n/2}(-1)^{i} x_{n/2+1} \\ &= \sum_{i=1}^{n/2}(-1)^i [x_i + x_{n/2+1}] \\ &= \sum_{i=1}^{n/2}(-1)^i, \qquad \text{since } x_i + x_{n/2+i} = 1 \\ &= 0.\end{aligned}$$

□

Corollary 9: Both the symmetrical and inverse subcodes of $C(n, n+1, 0, n/2)$ are included in the Nyquist null dc^{2-}-constrained code of length n. □

C. Spectral Null rf_s/N Subcodes

As shown in [18], spectral nulls at rational submultiples r/N of the symbol frequency f_s can be achieved if we restrict N to be a prime number which divides n, i.e.,

$$n = Nz \tag{24}$$

and constrain every codeword to have

$$A_1 = A_2 = \cdots = A_N \tag{25}$$

where

$$A_i = \sum_{\lambda=0}^{z-1} x_{\lambda N+1}, \qquad i = 1, \cdots, N. \tag{26}$$

A spectral null at frequency f_s/N furthermore implies spectral nulls at frequencies $rf_s/N \,\gcd(r, N) = 1$, .

We can now set up a generating function to enumerate subcodes of $C(n, n+1, 0)$, which also conform with (25). Note that (25) can only be satisfied if the set of bits which contribute to each A_i, have the same weight h, where $0 \le h \le z$. In the following generating function, x^i represents bit x_i in (26) and u_i represents a contribution to A_i, while t represents one unit of weight as before

$$\begin{aligned}&f(u_1, \cdots, u_k, t, x) \\ &\quad = \prod_{i=0}^{n/N-1} (1 + tu_1 x^{1+Ni})(1 + tu_2 x^{2+Ni}) \cdots (1 + tu_n x^{N+Ni}).\end{aligned} \tag{27}$$

We thus need to evaluate the coefficients of $u_1^h u_2^h \cdots u_k^h x^\sigma$, $0 \le h \le n/N$, $\sigma \equiv 0 \bmod (n+1)$, to obtain the cardinality of the spectral null rf_s/N subcode of $C(n, n+1, 0)$, or the coefficients of $u_1^h u_2^h \cdots u_k^h t^w x^\sigma$ if we want the cardinality of a weight w subcode of $C(n, n+1, 0)$.

Theorem 3: A spectral null $A_1 = A_2 = \cdots = A_N$ subcode of $C(n, n+1, 0)$ has $d_{\min} \ge \min\{4, N\}$.

Proof: Since $A_1 = A_2 = \cdots = A_N$, it follows that each of the N subsets of code bits $\{x_i, x_{i+N}, \cdots, x_{i+n-N}\}$, $i = 1, \cdots, N$, should have the same weight h, $0 \le h \le z$. Consequently, any codeword $(x_1, x_2, \cdots, x_n)$ has weight $w = Nh$. Thus any two codewords of different weights have $d = N$. Furthermore, any two codewords of the same weight have $d = 4$. It follows that $d_{\min} = \min\{4, N\}$. □

Code rates of spectral null $rf_s/3$ subcodes are also depicted in Fig. 1.

V. Decoding the Constant-Weight and Spectral Shaping Subcodes

The constant weight and spectral shaping subcodes $C(n, n+1, a)$ are highly structured and the decoding procedures are thus simple. We propose a decoding process that consists of three phases and now briefly discuss these three phases. Note that all the codes under consideration here correct either $s = 1$ insertion/deletion error, or $t = 1$ reversal error in each codeword.

A. Error Detection

The decoder functions under the assumption that the beginning and end, and thus also the length of each received word, is known. This is the assumption also made by Levenshtein when the $C(n, m, a)$ class of codes was proposed, and it may be achieved in practice by inserting periodic markers or synchronization words in the transmitted sequence.

Both the nature of the error and the restored value $\phi \in \{0, 1\}$ of the affected bit can be determined uniquely from the length n' and weight w' of the received word x' as follows:

$$\begin{aligned} \text{a)}\quad & n' = n \\ & w' = w + 1(w-1) \qquad &&\text{: reversal error } 0(1) \to 1(0) \\ \text{b)}\quad & n' = n-1 \\ & w' = w(w-1) &&\text{: deletion error } 0(1) \to \wedge \end{aligned}$$

and

$$\begin{aligned} \text{c)}\quad & n' = n+1, \\ & w' = w + 1(w) \qquad &&\text{: insertion error } \wedge \to 1(0). \end{aligned}$$

Here $\wedge$ denotes the empty word and the weight w is known for constant-weight w codes, or for higher order spectral null subcodes $(w = n/2)$, while for the spectral null rf_s/N subcodes, we use

$$w = \min\{|w' - Nh|\}, \qquad 0 \le h \le z. \tag{28}$$

B. Restoration of the Transmitted Codeword

In order to determine the index of the affected bit, we proceed as follows.

For higher order spectral null subcodes, the restored codeword can have only one moment, $n(n+1)/4$, and the following syndrome function can always be used:

$$\boldsymbol{S} = \left| n(n+1)/4 - \sum_{i=1}^{n'} i x_i' \right|. \tag{29}$$

For constant-weight subcodes, or for spectral null rf_s/N subcodes, the moment of the restored codeword is determined by the nature of the error and the restored value ϕ of the affected bit as follows:

If a reversal error with $\phi = 0$, or an insertion is detected, select σ to minimize the syndrome function

$$\boldsymbol{S} = \sum_{i=1}^{n'} i x_i' - \sigma, \qquad \boldsymbol{S} > 0 \tag{30}$$

thus

$$\boldsymbol{S} = \sum_{i=1}^{n'} i x_i' \,(\operatorname{mod} n+1). \tag{31}$$

If a reversal error with $\phi = 1$, or a deletion is detected, select σ to minimize the syndrome function

$$\boldsymbol{S} = \sigma - \sum_{i=1}^{n'} i x_i', \qquad \boldsymbol{S} > 0 \tag{32}$$

thus

$$\boldsymbol{S} = (n+1) - \sum_{i=1}^{n} i x_i' \,(\operatorname{mod} n+1). \tag{33}$$

The value of S in one of (29)–(33), as dictated by the class of codes and ϕ, is then used to determine the index of the corrected bit as follows:

a) For a reversal error, invert the bit at index S, since this is the only way to restore the moment to σ.

b) For insertions/deletions we may apply the decoding algorithm proposed by Levenshtein [2]. However, some simplification is possible if we make use of the fact that each codeword's complement is also included in the codebook and satisfies (1). (Refer back to Corollary 1.) For insertion/deletion of a zero, we delete/insert a zero to the right of the Sth one from the left in $(x'_{\max}, \cdots, x'_1)$, this is again the only way to restore the moment to σ. For insertion/deletion of a one, we first complement $\boldsymbol{x}'$, then compute S and proceed as above for insertion/deletion of a zero, and again complement the restored word.

C. Mapping of the Restored Codeword Onto Information Bits

From Fig. 1 can be seen that the code rates of all the balanced $w = n/2$ subcodes which we investigated eventually exceeds rate $R = 1/2$. Consequently, it is not possible to always construct a systematic code: for example the all zeros information word will require $w < n/2$ and the all ones information word $w > n/2$. However, for rates $R \le 1/2$, it may be possible to find a systematic mapping by inspection [16]. Alternatively, a lookup table should be used.

VI. Insertion/Deletion Correcting Coding Schemes

The practical application of the class of $C(n, m, a)$ Levenshtein codes has been restricted by the requirement that the boundaries of each received word be known before insertion/deletion correction can be affected and framing of words be maintained. In this section we propose two coding schemes which determine the boundaries of each codeword as long as $s = 1$ insertion/deletion per codeword is not exceeded, and if every word with an error is followed by an error-free word. Though the rate of the second scheme is somewhat less than the rate of the first scheme, it also offers correction of $t = 1$ reversal errors.

A. Coding Scheme 1

In this coding scheme, we use a subcode of $C(n, m, a, w)$, $0 < w < n$, such that each codeword has a moment from a set $\{\sigma\}$, where $|\sigma_1 - \sigma_2| \ge 2n + 1$ for any distinct σ_1, σ_2 in $\{\sigma\}$.

Note that it is possible to use a dc^2-constrained code, since each codeword has $w = n/2$ and fixed $\sigma_l = n(n+1)/4$. We furthermore need to insert two buffer bits, $b_1 = b_2 = 0$, between every two codewords. In the following description, it is assumed that a dc^2-constrained code is used. The transmitted sequence is thus constituted as follows:

$$\cdots x_n^{j-1}\, b_1^{j-1}\, b_2^{j-1}\, x_1^j \cdots x_n^j\, b_1^j\, b_2^j\, x_1^{j+1} \cdots x_n^{j+1}\, b_1^{j+1}\, b_2^{j+1} \cdots \tag{34}$$

where $\boldsymbol{x}^j$ is the jth transmitted codeword.

For each codeword, the receiver establishes a frame of $n+1$ bits and computes the moment

$$\sigma = \sum_{i=1}^{n+1} i y_i \tag{35}$$

where $(y_1, \cdots, y_{n+1})$ is the received word corresponding to $(x_1, \cdots, x_n, b)$. When decoding $\boldsymbol{x}^j$, we assume that framing up to $\boldsymbol{x}^{j-1}$ has proceeded correctly. Without any insertion/deletions the decoder observes

$$\sum_{i=1}^{n+1} i y_i = n(n+1)/4 \tag{36}$$

and proceeds to $\boldsymbol{x}^{j+1}$.

If a bit is deleted in $\boldsymbol{x}^j$, $\boldsymbol{b}_2^j$ replaces $\boldsymbol{b}_1^j$ in (35) and the decoder observes

$$\sum_{i=1}^{n+1} i y_i \le n(n+1)/4. \tag{37}$$

If a bit is inserted in $\boldsymbol{x}^j$, x_n^j replaces $\boldsymbol{b}_1^j$ in (35) and the receiver observes

$$\sum_{i=1}^{n+1} i y_i \ge n(n+1)/4. \tag{38}$$

Note that equality in (37) and (38) is only achieved if there are only zeros to the right of the inserted/deleted bit and if the inserted/deleted bit is a zero. If this is the case, the decoder observes a word which resembles $\boldsymbol{x}^j$ in the first n bits of the frame and need not take any action. However, this event will then be reflected as an insertion/deletion at index $i = 1$ in $\boldsymbol{x}^{j+1}$ when the decoder incorrectly establishes the next frame, and it is corrected if $\boldsymbol{x}^{j+1}$ is received error-free.

If inequality is observed in (37) and (38), the decoder detects and discriminates between an insertion/deletion, and uses the observed weight w' to determine the restored value ϕ of the affected bit, as in Section V.

If one of the buffer bits $\boldsymbol{b}^{j-1}$ is deleted, this is perceived at the decoder as the deletion of x_1^j, while an insertion preceding $\boldsymbol{b}_i^{j-1}$, is perceived as the insertion of a 0 at $i = 1$ in $\boldsymbol{x}^j$.

Restoration of the transmitted codeword proceeds as in Section V and the receiver adjusts the framing if necessary. Note that the decoder is unable to discriminate between insertions/deletions and reversal errors, consequently, the capability to do reversal error correction is lost.

In case the set of moments $\{\sigma\}$ contains more than one integer, we see that the decoder needs to discriminate between a deletion from a codeword with moment σ_1 or an insertion in a codeword with moment σ_2, where $\sigma_1 > \sigma_2$, hence in general, for $C\,(n, m, a, w)$ we require

$$|\sigma_1 - \sigma_2| \ge 2n+1 \tag{39}$$

for any distinct σ_1, σ_2 in $\{\sigma\}$. The coding rate of this scheme, when implemented with dc^2-constrained codes, is $R = k/(n+2)$, where k and n can be obtained from Fig. 1 for the dc^2-constrained codes. The overall rate is also depicted in Fig. 1. For example, $R = 1/2$ is achieved when using the $(n, k) = (16, 9)$ dc^2-constrained code.

B. Coding Scheme 2

Alternatively, we can achieve insertion/deletion correction, by making use of a symmetrical subcode of $C\,(n, n+1, 0)$, which has $x_j = x_{n+1-j}$, $1 \le j \le n$, and which was investigated in Propositions 5, 8, and 10.

If we are furthermore interested in correcting reversal errors, we require $d_{\min} = 4$, hence we need a symmetrical constant weight subcode of $C\,(n, n+1, 0)$, which then achieves the highest cardinality for $w = n/2$. From Propositions 8 and 10 it thus follows that we shall make use of a subcode of the Nyquist null, dc^2-constrained codes discussed in Section IV.

No buffer bits need to be transmitted. The decoder in this coding scheme assumes that framing up to $\boldsymbol{x}^{(j-1)'}$ has proceeded correctly, establishes a frame of n bits, inputs $\boldsymbol{x}^{j'}$, and checks for the symmetry. If a deviation from symmetry is observed, it inputs $\boldsymbol{x}^{(j+1)'}$. It performs reversal error correction on x^j if $x^{(j+1)'}$ is symmetrical. If $\boldsymbol{x}^{(j+1)'}$ is not symmetrical, it attempts to restore its symmetry by means of one bit left shift or one bit right shift. If it is necessary to perform a right shift on $\boldsymbol{x}^{(j+1)'}$, a deletion occurred in $\boldsymbol{x}^{j'}$, while a left shift indicates an insertion.

The decoder thus sets up the correct frame for $\boldsymbol{x}^{j'}$, and uses the observed weight w' within this frame, to determine the restored value ϕ of the affected bit. Restoration of the transmitted codeword proceeds as in Section V.

By careful consideration of candidate symmetrical codewords, it can be shown that there are four codewords, which may achieve symmetry in $\boldsymbol{x}^{(j+1)'}$ again, after two like shifts, which will then lead to an incorrect decision about the nature of the deletion/insertion in $\boldsymbol{x}^{j'}$. These are the all-*zeros* and all-*ones* codewords, and the two complementary codewords which each constitute only of runs of length 2 like symbols, except for x_1 and x_n, which each constitute a run of length 1. These words have to be omitted from the codebook.

The cardinality of the codes in this scheme can be determined by noting that once the first subword $(x_1, \cdots x_{n/2})$ of $\boldsymbol{x}$ is specified, the second subword is fixed. Furthermore, any pair of ones (x_j, x_{n+1-j}) contributes $(n+1)$ to the codeword moment, hence

$$\sum_{i=1}^{n} i x_i \equiv 0 \,(\operatorname{mod} n+1)$$

and for any symmetrical word $\boldsymbol{x}$, $\boldsymbol{x} \in C(n, n+1, 0)$.

For only insertion/deletion correction, we allow the first subword to have any weight and then omit the four words discussed

$$|\boldsymbol{C}_1| = 2^{n/2} - 4\,. \tag{40}$$

To afford reversal correction in addition we need a constant-weight w subcode and allow the first subword to contain $w/2 = n/4$ ones. Thus the all-zeros and all-ones codewords are excluded, and

$$|\boldsymbol{C}_2| = \binom{n/2}{n/4} - 2\,. \tag{41}$$

For both these classes of codes, the asymptotic code rate is

$$R_\infty = 1/2. \tag{42}$$

Code rates for finite n are depicted in Fig. 1. Although these rates appear low, the minimum-bandwidth property of the second class of codes, together with the rate, should be considered to evaluate the efficiency on bandwidth-limited channels.

VII. Spectral Null Markers for the Detection of Deletion/Insertion Errors

A. Preliminaries

In this section the use of a marker (or synchronization sequence) is proposed to separate each codeword in the channel. Markers are predetermined sequences used for the alignment of transmitted sequences. We present markers for use with either unconstrained or spectral null $C(n, n+1, a)$ codes. Sellers [19] introduced the idea of a marker for deletion/insertion error control. No data is usually mapped onto a marker and therefore the overall rate is reduced. Sellers' method needed to wait for two markers before any detection could be done. In this section, a merging of Sellers' method and the Levenshtein codes is proposed. The goal of the markers introduced in this section is to afford spectral nulls when necessary, as well as to detect errors. The correction of errors must be done by the decoding procedures for the insertion/deletion error correcting code.

The idea of a marker is better understood by first introducing the concept of a *synchronization error correcting sequence*. A synchronization error correcting sequence consists of an insertion/deletion correcting codeword (e.g., a Levenshtein codeword) $\boldsymbol{x} = (x_n, \cdots, x_1)$ of length n followed by a marker $\boldsymbol{b} = b_{M_s} \cdots b_2 b_1$ of length M_s. The insertion/deletion correcting codeword $\boldsymbol{x}$ will be called the *information segment* because this is the part of the sequence carrying the information. The total length of the synchronization error-correcting sequence is thus

$$n + M_s. \tag{43}$$

Sellers [19] showed that the minimum length of a marker $\boldsymbol{b}$ must be

$$M_s \geq 2s + 1, \ s \geq 1 \tag{44}$$

where s is the insertion/deletion correcting ability of the code. For an $s = 1$ code the minimum length of the marker must be 3. Before constructing the markers, it is necessary to investigate the function of a marker. Similarly as for the coding schemes in the previous section, the assumption is made that any synchronization error-correcting sequence with error should be followed by one without error.

A marker must enable the receiver to detect the occurrence of an error, be that an insertion/deletion or a reversal error. The order of detection is also important. An insertion/deletion shifts the rest of the symbols in a codeword up or down, depending on the error. Up to the point of the error, the symbols will be correct. From the insertion/deletion error position i onwards all the symbols may be wrong. If the receiver first checks for reversal errors, the incorrectly framed received word $\boldsymbol{x}'$ may appear to contain a maximum of $n - i + 1$ reversal errors. If the reversal error-correcting ability of the code is t and $t < n - i + 1$, then the receiver will be unable to correct or even detect the errors. The word may even look like another valid codeword, only with t or fewer additive errors and the receiver may then map it onto a wrong codeword. If, however, the receiver first checks for insertion/deletion errors by evaluating a marker, it will first detect and correct the insertion/deletion error if it exists. The next codeword will then again be correctly framed.

B. Rules for the Construction of Markers

It is assumed that when codewords are transmitted, then x_1 is sent first and x_n last. (Refer to (34).) Let B be a marker codebook and $\hat{\boldsymbol{b}} \in B$ be a marker of length M_s, $M_s \geq 3$, and

$$\boldsymbol{b} = b_1 b_2 \cdots b_{M_s} \tag{45}$$

with $b_i \in \{0, 1\}$ for a binary marker. The resulting marker after a single deletion in the $previous$ codeword is

$$\boldsymbol{b}^d = b_2 b_3 \cdots b_{M_s} x_1 \tag{46}$$

where x_1 is the first symbol from the next codeword. The first symbol of $\boldsymbol{b}$, b_1 is shifted out and becomes the last symbol of the previous information segment. The last symbol of $\boldsymbol{b}^d$ now contains the first symbol of the following information segment. Note that x_1 may assume any binary value. The word $\boldsymbol{b}^d$, which will be called a *deletion indicator*, is the resulting marker $\boldsymbol{b}$ due to a deletion in the preceding information segment. Similarly, the resulting marker after an insertion in the previous information segment is

$$\boldsymbol{b}^i = x_n b_1 b_2 \cdots b_{M_s - 1} \tag{47}$$

where x_n is the last symbol from the preceding information segment. The word $\boldsymbol{b}^i$ will be called an *insertion indicator*. For a sequence $\boldsymbol{b}$ to be considered as a marker, the following rules must be valid:

1) $\boldsymbol{b}^i \neq \boldsymbol{b}^d$
2) $\boldsymbol{b}^i \neq \boldsymbol{b}, \boldsymbol{b}^d \neq \boldsymbol{b}$.

Rule 1) ensures that the decoder can *differentiate* between insertion and deletion errors. Rule 2) enables the receiver to *detect* the occurrence of a insertion/deletion.

If more than one marker is to be used together in the same transmission, the detection and differentiation rules must still be valid. Let $\boldsymbol{b}_1$ and $\boldsymbol{b}_2$ be two markers of equal length. For $\boldsymbol{b}_1$ and $\boldsymbol{b}_2$ to be valid markers in the same marker codebook B, they must first comply with Rules 1) and 2) and with the following additional rules, where $\boldsymbol{b}_j^i$ is the insertion indicator for marker j, $\boldsymbol{b}_j^d$ is the deletion indicator of marker j and $\boldsymbol{b} \in \{\boldsymbol{b}_1, \boldsymbol{b}_2\}$:

3) $\boldsymbol{b}_1^i \neq \boldsymbol{b}_2^d, \boldsymbol{b}_1^d \neq \boldsymbol{b}_2^i$
4) $\boldsymbol{b}_1^i \neq \boldsymbol{b}, \boldsymbol{b}_1^d \neq \boldsymbol{b}, \boldsymbol{b}_2^i \neq \boldsymbol{b}, \boldsymbol{b}_2^d \neq \boldsymbol{b}$.

Again, Rule 3) ensures that the receiver will be able to differentiate between insertion and deletion errors. Rule 4) enables the receiver to detect synchronization errors.

For a 3-bit marker code book, only four codewords comply with both Rules 1) and 2), i.e., $001, 100, 011,$ and 110. The markers 001 and 100 are the ones which Sellers [19] proposed in his insertion/deletion correcting codes. The marker 011 was the one Ullman [20] used in his codes. None of these valid markers however can be used together in a marker codebook because any two markers violate Rules 3) and 4).

Until now, valid markers only enabled the decoder to detect and differentiate insertion/deletion errors in the previous information segment. It is also possible for an insertion/deletion or reversal error to occur in the marker itself. To combat this situation, it is first necessary for the marker codebook to have a minimum Hamming distance of $d_{\min} \geq 2$. This will guarantee that a reversal error in a marker codeword will be detected. To combat insertion/deletion errors, the marker codebook must furthermore have insertion/deletion-*detecting* capabilities. It is proposed to use markers from a Levenshtein code. Note that these markers will then also have minimum Hamming distance of $d_{\min} \geq 2$.

C. Example of Some 4-Bit Marker Codebooks

Table I presents eight 4-bit marker codebooks. These marker codebooks comply with the marker construction rules (1)–(4) given in the previous section. The markers may be used to detect as well as identify the type of synchronization error in the *previous* information segment *exclusively*.

Table II lists all the valid 4-bit marker codebooks which furthermore has an $s = 1$ insertion/deletion correcting ability (and thus also $d_{\min} \geq 2$). Note when used with the unconstrained Levenshtein code, the rate of the coding scheme with marker can be increased by mapping one information bit onto a set of two marker words chosen from Table I or II.

D. Decoding Procedure

The decoding of the synchronization error-correcting sequence consists of two steps. A lookup table is necessary for the marker

TABLE I
VALID BINARY 4-BIT MARKER CODEBOOKS

CODE BOOK	MARKER WORDS		
1	0001	1001	1011
2	0001	1001	1101
3	0010	0110	0111
4	0010	0110	1110
5	0100	0110	1110
6	0100	0110	0111
7	1000	1001	1011
8	1000	1001	1101

TABLE II
VALID 4-BIT MARKER CODEBOOKS

BOOK #	MARKER #1	MARKER #2
1	0001	1011
2	0001	1101
3	0001	1110
4	0010	0111
5	0010	1101
6	0010	1110
7	0100	0111
8	0100	1011
9	0100	1110
10	1000	1011
11	1000	1101
12	1000	0111

codebook and its error indicators. When a synchronization error-correcting sequence is received, the decoder separately checks if there are any errors in the information segment or in the marker. By applying (1) to the information segment, the receiver can detect whether an error has occurred or not. If the received information segment does not comply with (1), the receiver knows that an error has occurred in that information segment. Let r_i be the output of the information segment check and r_m the output of the marker check and $r_i, r_m \in \{0, 1\}$. The values or r_i and r_m are *zero* when there are no errors and *one* if there are any errors. Table III lists the result of the variables r_i and r_m and also the meanings and remedies. It assumed that the information segment is represented by a codeword from $C(n, m \geq 2n, a)$, or from $C(n, n+1, 0, w)$, hence reversal error correction can be achieved.

The deletion/insertion error correction for the information segment may be done by the procedures given by Levenshtein [2].

If a marker indicates that a deletion occurred, the n bits of the information segment preceding the beginning of the marker are taken to be the information segment in error. For the correction, however, the last symbol of the information segment must be left out, because this will be the first symbol from the original marker. The total length of the information sequence passed on to the correcting procedure will therefore be $n-1$. The framing must then be adjusted accordingly.

For an insertion the same procedure is followed. The n symbols of the information segment preceding the beginning of the marker, together with the first symbol of the marker, are taken as the received word to be corrected by the insertion correcting procedure. The first

TABLE III
DECODING SYNCHRONIZATION ERROR CORRESPONDING SEQUENCES

r_i	r_m	MEANING AND REMEDY
0	0	No error - continue.
0	1	Error in marker - leave as is and continue. If it was an deletion/insertion error, it will be corrected when decoding the next synchronization error correcting sequence.
1	0	Reversal error in information segment - correct and continue.
1	1	Error in information and marker segment - assume a synchronization error and compare marker with error indicators. Use the appropriate correction procedure and adjust the framing.

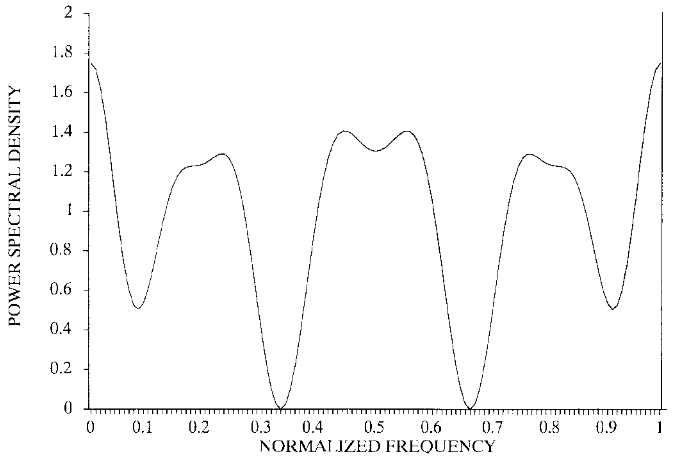

Fig. 2. Power spectral density of a 9-bit Levenshtein subcodes with spectral zeros at multiples of $f_s/3$.

symbol of the marker must be included because it is the last symbol of the original information segment. The total length of the word to be corrected is thus $n+1$ symbols. The decoder framing must be adjusted to correct the framing of subsequent codewords.

If the error is a reversal error in the information segment, the n symbols preceding the beginning of the marker are taken to be the word to be corrected by the reversal error-correcting procedure.

Sometimes it may happen that a reversal or insertion/deletion error in the marker appears to be the same as an insertion or deletion indicator. In these cases, the decoder simply ignores the indicator and continues with the next codeword. If the error was an insertion/deletion, the next codeword will be corrupted because of this insertion/deletion. If the next synchronization error-correcting sequence is received error-free, the marker in this sequence will detect the error and the decoder will be able to correct the error.

If more than one insertion/deletion occur per synchronization error-correcting sequence, the decoder will be unable to correct these errors and the framing will be lost. If more than one reversal error occur per sequence, or if two consecutive sequences with errors are received, the decoder will be unable to correct the errors, but word framing will still be intact.

E. Spectral Shaping Parameters of Markers

The question may arise why a marker *codebook* is needed, when traditionally only one marker was used. If one uses an arbitrary single marker, the spectral properties of the channel sequence may be altered. The marker, or set of markers, must therefore be chosen carefully to match the spectral properties of the insertion/deletion error correcting code. Table IV lists the values of the running digital sum, second-order moment at dc, running alternate sum and second-

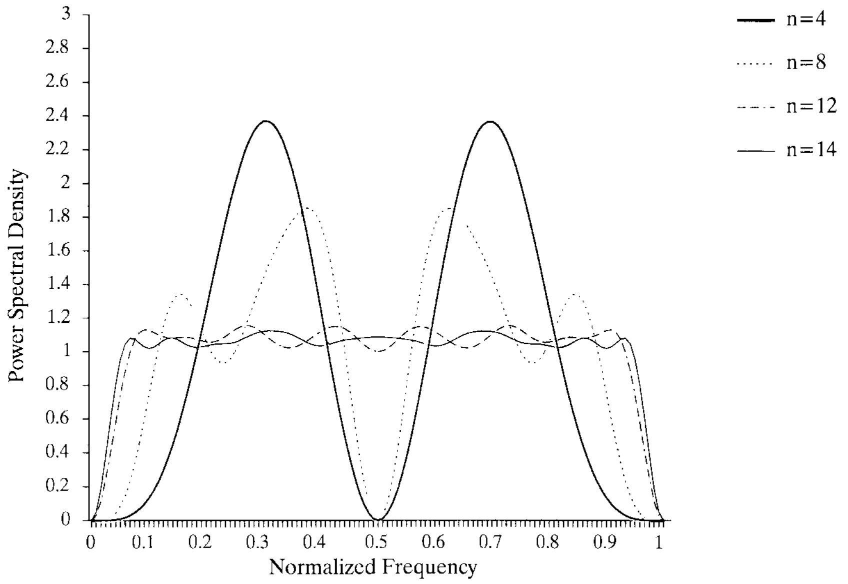

Fig. 3. Some spectra of dc-free Levenshtein subcodes.

TABLE IV
SPECTRAL PROPERTIES OF MARKERS

MARKER	$\sum_{i=1}^{n} x_i$	$\sum_{i=1}^{n} ix_i$	$\sum_{i=1}^{n} (-1)x_i$	$\sum_{i=1}^{n} (-1)ix_i$
0001	-2	-2	-2	-6
0010	-2	-4	2	8
0011	0	4	0	0
0100	-2	-6	-2	-2
0110	0	0	0	4
0111	2	8	-2	-4
1000	-2	-8	2	4
1001	0	0	0	-4
1011	2	6	2	2
1100	0	-4	0	0
1101	2	4	-2	-8
1110	2	2	2	6

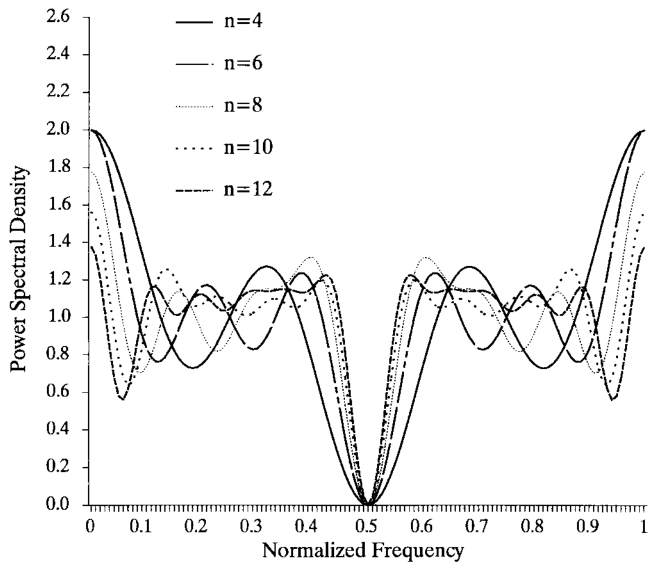

Fig. 4. Spectra of minimum-bandwidth Levenshtein subcodes.

order moment at the Nyquist frequency for every candidate marker. Note that the binary symbols 0 and 1 are mapped onto -1 and 1, respectively. The markers for which these spectral shaping parameters have value *zero*, are of particular interest, for using as a single marker in conjunction with the subcodes in this paper.

VIII. CONCLUSIONS

In this correspondence we have presented some new insight into the structure of a class of Varshamov–Tenengolt's codes, as well as several important subclasses. We have shown that there is a relation between Levenshtein's codes for correcting deletions/insertions and some classes of codes with nulls in the power spectral density function. Some of these power spectral density functions are depicted in Figs. 2–4.

From Fig. 1 we see that the rates of the Nyquist-null subcodes are indeed higher than those of the dc-free subcodes. Another observation is that the null notch width decreases as the codeword length increases in both the dc-free and Nyquist-null Levenshtein subcodes. This can be explained by the fact that the digital sum variation increases.

Although we have presented some results to this effect, future work can focus on improving the rates of coding schemes which alleviate the restriction on Levenshtein's codes that the boundaries of each codeword should be known.

REFERENCES

[1] R. R. Varshamov and G. M. Tenengolt's, "Correction code for single asymmetrical errors," *Avtom. Telemekh.*, vol. 26, no. 2, pp. 288–292, Feb. 1965.

[2] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions and reversals," *Sov. Phys.–Dokl.*, vol. 10, no. 8, pp. 707–710, Feb. 1966.

[3] B. D. Ginzburg, "A number-theoretic function with an application in the theory of coding," *Probl. Cybern.*, vol. 19, pp. 249–252, 1967, in Russian.

[4] K. A. S. Immink and G. F. M. Beenker, "Binary transmission codes with higher order spectral zeros at zero frequency," *IEEE Trans. Inform. Theory*, vol. IT-33, no. 3, pp. 452–454, May 1987.

[5] P. H. Delsarte and P. H. Piret, "Spectral enumerators for certain additive-error-correcting codes over integer alphabets," *Inform. Contr.*, vol. 48, pp. 193–210, 1981.
[6] R. L. Graham and N. J. A. Sloane, "Lower bounds for constant weight codes," *IEEE Trans. Inform. Theory*, vol. IT-26, no. 1, pp. 37–41, Jan. 1980.
[7] T. Helleseth and T. Kløve, "On group-theoretic codes for asymmetric channels," *Inform. Contr.*, vol. 49, pp. 1–9, 1981.
[8] L. E. Mazur, "Correcting codes for asymmetric errors," *Probl. Pered. Inform.*, vol. 10, no. 4, pp. 40–46, Oct.–Dec. 1974.
[9] W. H. Kautz and B. Elspas, "Single-error-correcting codes for constant-weight data words," *IEEE Trans. Inform. Theory*, vol. IT-11, no. 1, pp. 132–141, Jan. 1965.
[10] J. Riordan, *An Introduction to Combinatorial Analysis.* Princeton, NJ: Princeton Univ. Press, 1980.
[11] L. E. Dickson, *History of the Theory of Numbers*, vol. 2. New York: Chelsea, 1952.
[12] A. E. Brouwer, J. B. Shearer, N. J. A. Sloane, and W. D. Smith, "A new table of constant weight codes," *IEEE Trans. Inform. Theory*, vol. 36, no. 6, pp. 1334–1380, Nov. 1990.
[13] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes.* New York: North-Holland, 1988.
[14] H. C. Ferreira, "Lower bounds on the minimum Hamming distance achievable with runlength constrained or dc-free block codes and the synthesis of a $(16, 8)$ $d_{\min} = 4$ dc-free block code," *IEEE Trans. Magn.*, vol. MAG-20, no. 5, pp. 881–883, Sept. 1984.
[15] M. Blaum, "A $(16, 9, 6, 5, 4)$ error-correcting dc free block code," *IEEE Trans. Inform. Theory*, vol. 34, no. 1, pp. 138–141, Jan. 1988.
[16] A. S. J. Helberg, W. A. Clarke, H. C. Ferreira, and A. S. J. Vinck, "A class of dc free synchronization error correcting codes," *IEEE Trans. Magn.*, vol. 29, no. 6, pp. 4048–4049, 1993.
[17] D. J. Kim and J. Kim, "A condition for stable minimum bandwidth line codes," *IEEE Trans. Commun.*, vol. COM-33, no. 2, pp. 152–157, 1985.
[18] K. A. S. Immink, *Coding Techniques for Digital Recorders.* Englewood Cliffs, NJ: Prentice-Hall, 1991.
[19] F. F. Sellers, Jr, "Bit loss and gain correction code," *IRE Trans. Inform. Theory*, vol. IT-8, no. 1, pp. 35–38, Jan. 1962.
[20] J. D. Ullman, "Near-optimal, single-synchronization-error-correcting code," *IEEE Trans. Inform. Theory*, vol. IT-12, no. 4, pp. 418–424, Oct. 1966.

A Note on the q-ary Image of a q^m-ary Repeated-Root Cyclic Code

Li-zhong Tang, Cheong Boon Soh, and Erry Gunawan

***Abstract*—For $(n, q) = p^s$, where $p = \mathrm{ch}\,(F_q)$, $s \geq 1$, V a q^m-ary repeated-root cyclic code of length n with generator polynomial $g(x)$, we give a partial answer about whether the q-ary image of V is cyclic or not with respect to a certain basis for F_{q^m} over F_q.**

***Index Terms*— Cyclic code, q-ary image, ideal, rings, repeated-root cyclic code.**

I. Introduction

Let m and n be two positive integers, and let F_q be a q-ary finite field of characteristic p. Then we know that q is a power of p. Now let $\underline{\alpha} = (\alpha_0, \alpha_1, \cdots, \alpha_{m-1})$ be a basis (ordered) for F_{q^m} over F_q and define the mapping

$$d_{(\underline{\alpha}, m, n)}: F_{q^m}[z]/(z^n - 1) \longrightarrow F_q[z]/(z^{mn} - 1)$$

as follows:

$$d_{(\underline{\alpha}, m, n)}(a(z)) = \sum_{i=0}^{m-1}\sum_{j=0}^{n-1} a_{i,j} z^{mj+i}$$

where

$$a(z) = \sum_{j=0}^{n-1} a_j z^j$$

and

$$a_j = \sum_{i=0}^{m-1} a_{i,j}\alpha_i, \quad a_{i,j} \in F_q.$$

Then $d_{(\underline{\alpha}, m, n)}$ has the following properties:

i) It is a bijective map (injective and surjective).
ii) It is q-ary linear (i.e., linear over F_q).
iii)

$$d_{(\underline{\alpha}, m, n)}(g(z)a(z)) = g(z^m) d_{(\underline{\alpha}, m, n)}(a(z))$$

for any $g(z) \in F_q[z]$ and $a(z) \in F_{q^m}[z]$.

If V is a q^m-ary $[n, k]$ cyclic code, then its q-ary image with respect to the basis $\underline{\alpha}$ is $d_{(\underline{\alpha}, m, n)}(V)$ where

$$d_{(\underline{\alpha}, m, n)}(V) = \{d_{(\underline{\alpha}, m, n)}(a(z)) \,|\, a(z) \in V\}.$$

It follows that $d_{(\underline{\alpha}, m, n)}(V)$ is a q-ary $[mn, km]$ linear code invariant under multiplication by z^m; hence $d_{(\underline{\alpha}, m, n)}(V)$ is a q-ary quasicyclic code [1], [2]. Then the general problem is as follows:

For which pair $(\underline{\alpha}, V)$, where V—a q^m-ary cyclic code and $\underline{\alpha}$ a basis for F_{q^m} over F_q, is $d_{(\underline{\alpha}, m, n)}(V)$ a cyclic code?

Several authors attacked this problem [3]–[11]. Especially, in [11], under the only restriction $(n, q) = 1$, Séguin gave a very simple characterization of all the cyclic codes V for which there exists a basis $\underline{\alpha}$ such that $d_{(\underline{\alpha}, m, n)}(V)$ is cyclic. His main result is quoted as follows:

Lemma 1.1 ([11, Theorem 10]): Let $(n, q) = 1$ and let V be a q^m-ary cyclic code of length n with generator polynomial $g(z)$. Then there exists a basis $\underline{\alpha}$ for F_{q^m} over F_q for which $d_{(\underline{\alpha}, m, n)}(V)$ is cyclic if and only if:

i) $g(z) \in F_q[z]$, in which case $d_{(\underline{\alpha}, m, n)}(V)$ is cyclic for every basis $\underline{\alpha}$ and the generator polynomial of $d_{(\underline{\alpha}, m, n)}(V)$ is $g(y^m)$; or
ii) $g(z) = g_0(z)(z - \gamma^{-q^v}), g_0(z) \in F_q[z], F_q \neq F_{q^k} = F_q(\gamma) \subset F_{q^m}, v \in Z_k$, and $\omega^m - \gamma$ has a divisor over F_{q^k} of degree $e = m/k$. In this case, $d_{(\underline{\alpha}, m, n)}(V)$ is cyclic if and only if $\alpha_{m-1}, \alpha_{m-2}, \cdots, \alpha_{m-e}$ are F_{q^k}-independent and

$$\alpha_j = \sum_{i=1}^{e} a_i^{q^v}\alpha_{j+i}, \quad 0 \leq j < m - e$$

and

$$a(\omega) = \omega^e - a_1\omega^{e-1} - \cdots - a_e \in F_{q^k}[\omega]$$

divides $\omega^m - \gamma$. Moreover, the generator polynomial of $d_{(\underline{\alpha}, m, n)}(V)$ is $\overline{a_{-1}}(y)g_0(y^m)$, where $\overline{a_{-1}}(y)$ is the reciprocal of $\overline{a(y)}$; or

Manuscript received February 16, 1996; revised August 5, 1996.

The authors are with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798.

Publisher Item Identifier S 0018-9448(97)00629-9.