On the Problem of Finding Zero-Concurring Codewords
Alexander Vardy and Yair Be'ery

Abstract—Zero-concurring codewords disclose a certain structure of the code that may be employed for efficient soft-decision decoding and for designing de-free codes. New methods for constructing sets of zero-concurring codewords are presented for several families of codes. For the general case an algorithmic solution of the problem is offered. A table of results obtained using the proposed techniques is supplied for all the primitive narrow-sense binary BCH codes of length up to 127.

Index Terms—Zero-concurring codewords, soft-decision decoding, de-free codes, BCH codes.

1. INTRODUCTION

The concept of zero-concurring codewords was first defined by Be'ery and Snyders [1]. We propose a slightly more general definition.

Definition 1: Let C be a block code of length n over a finite field GF(q). A set of j independent codewords of C is called zero-concurring if in all the n positions at most one of the j codewords has a nonzero entry.

The idea of using some specific (symmetric) structure of the code in order to simplify its decoding may be found in a few works, for example [1]–[7]. The decoding algorithms employed in each case seem to be very different in their nature. It appears, however, that zero-concurring codewords disclose a certain inherent structure of the code, which can be exploited for its efficient soft decoding. Thus, Be'ery and Snyders [1] demonstrated that a considerable computational gain may be obtained by employing zero-concurring codewords in soft decision maximum likelihood decoding of low and mid-rate binary linear codes. Conway and Sloane [3] and Forney [4] provide similar results. The computational gain increases at least exponentially with the number J of zero-concurring codewords found in the code. An even greater computational gain may be achieved by introducing the so-called λ-concurring codewords (see [1], [6, 7]). However, the problem of search for such codewords is out of the scope of this correspondence and will be dealt with elsewhere [18].

Yet another application of zero-concurring codewords for designing de-free and run length constrained error-correcting codes was recently presented by Deng and Herro in [8]. These authors introduce a class of de-free codes derived by finding a set of zero-concurring codewords in a binary linear block code. They also propose a construction method applicable to primitive BCH codes of composite block length n and designed distance d, provided that d is a factor of n. This paper significantly extends the results of Deng and Herro by exhibiting zero-con-

IEEE Log Number 9038864.

The term “zero-concurring” stems from the word concurring, encountered in the literature on majority logic decoding [13]. Some of the subcodes of the Golay code discussed by Conway and Sloane [3] and the coset codes of Forney [4] are spanned by sets of zero-concurring codewords, though neither Conway nor Sloane nor Forney provide a specific definition.

REFERENCES

curing codewords in all the primitive BCH codes of length up to 127, several nonprimitive BCH codes and other binary codes. It should be pointed out that the problems of constructing a zero-concuring subset for efficient soft decoding and constructing a zero-concuring subset for the design of de-free codes are slightly different. In the former case the main objective is to find a zero-concuring set with maximum possible cardinality while in the latter case we need a zero-concuring set covering all the length of the code but not necessarily of maximal cardinality. Nevertheless, the results derived in the sequel have immediate applications for both problems and possibly for other problems in coding theory.

Given a linear block code $C(n, k, d)$ of length $n$, dimension $k$, and minimum Hamming distance $d$, let $J$ denote the cardinality of a maximal set of zero-concuring codewords of $C$. Evidently, $J \leq \left\lfloor \frac{n}{d} \right\rfloor$ (1)

where $\lfloor x \rfloor$ denotes the largest integer less than or equal to $x$. Although this bound is not always tight (see Section IV), it will be shown in the correspondence that the number of zero-concuring codewords in many codes meets the bound. In the next section we derive methods for constructing bound reaching zero-concuring sets in several frequently encountered families of codes. Various combinatorial algorithms for the search of zero-concuring codewords are presented in Section III. The efficiency of the proposed algorithms is demonstrated by the results obtained that are summarized in Table I of Section IV.

II. CONSTRUCTIONS

Cotsworth and Snyders [1] consider several methods for constructing zero-concuring codewords.

1) Let $RM(r, m)$ be the binary Reed–Muller code of order $r$ and length $n = 2^m$. A set of $J = \lfloor n/d \rfloor = 2^r$ zero-concuring codewords of $RM$ may be obtained by performing elementary row operations on the $2^r \times 2^r$ matrix consisting of all the codewords $a(x)$, such that

$$a_i(x) = (1 + x)^{2^j - 1} + x^{2^j - 1},$$

for $j = 0, 1, \cdots, 2^r - 1$ (2)

where $a(x)$ stands for

$$\sum_{i=1}^{n} a_i x^{i-1}$$

and $(a_1, a_2, \cdots, a_n) \in RM(r, m)$.

2) Let $C(n, k)$ be a binary cyclic code and let $\delta \geq 2$ be a divisor of $n$, such that $x^{\delta} + 1$ is a divisor of the parity check polynomial of $C$. Then the set

$$\{ x^j a_j(x) : j = 0, 1, \cdots, \delta - 1 \}$$

where

$$a_j(x) = 1 + x^\delta + x^{2\delta} + \cdots + x^{n-\delta} = x^{\delta - 1} + x^{n-1}$$

is a zero-concuring subset of $C$.

New constructions are presented in the following.

A. Lower Bounds

A simple constructive lower bound on the number of zero-concuring codewords in a linear code may be formulated as follows.

**Theorem 1:** Any linear code $C(n, k, d)$ contains a set of $\lfloor n/(k + 1) \rfloor$ zero-concuring codewords.

**Proof:** Let $G = [g_j]$, $1 \leq j \leq n$, $1 \leq j \leq k$, be a generator matrix of $C$. Evidently, $G$ may be always brought into the form of a band matrix, such that $g_j \neq 0$ only if $i \leq j \leq n_k + k$. Now let $g_j$, $1 \leq j \leq k$, denote the rows of $G$. Then the set

$$\{ g_1, g_2, \cdots, g_n, g_{n-k+1}, \cdots \}$$

is a zero-concuring subset of $C$.

\hfill \Box

For high-rate codes the above construction provides large, though not bound reaching, sets of zero-concuring codewords. However, if the code is maximum distance separable (MDS), i.e., such that $d = n - k + 1$, the lower bound coincides with the upper bound of (1).

If the minimum distance of the code is known, the lower bound of Theorem 1.1 may be slightly improved. As shown in Section III, the set of all the codewords of $C$ that are nonconcur to a specific codeword of Hamming weight $d$ may be regarded as a linear code $C(n', k'; d')$ where $n' = n - d$ and $k' = k - d$. Hence, applying Theorem 1.1 to $C'$ we get

$$J = \left\lfloor \frac{n'}{n' - k'} + 1 \right\rfloor \geq 1 + \left\lfloor \frac{n - d}{n - k + 1} \right\rfloor.$$

Now let $d_1$ be an upper bound on the minimum distance of a code of length $(n - d)$ and dimension $(k - d)$. Then clearly $d' \geq d_1$, and using a similar argument, we obtain

$$J \geq 2 + \left\lfloor \frac{n - d - d}{n - k + 1} \right\rfloor$$

provided that $k' > 0$. Proceeding in this manner we arrive at the following lower bound on $J$.

**Theorem 1.2:** Denote by $D(n; k)$ the maximum distance of a linear code of length $n$ and dimension $k$ over $GF(q)$. Let $d_0 = d$ and let $d_i = D(n - d_0 - d_1 - \cdots - d_{i-1}, k - d_0 - d_1 - \cdots - d_i)$ for $i = 1, 2, \cdots, n+1$, where $n$ is the largest integer such that $k - d_0 - d_1 - \cdots - d_i > 0$. Then $J \geq \frac{n}{2} + 2$.

Theorem 1.2 implies that various lower bounds on $J$ may be derived using the known upper bounds on the minimum distance of linear codes.

B. Cyclic Codes with Degenerate Subcodes

A cyclic code which consists of several repetitions of a code of smaller block length is said to be degenerate. In the following we derive a simple procedure for identifying a degenerate cyclic subcode within a given cyclic code, which leads to a powerful construction method.

**Lemma 2:** A linear cyclic code $C$ of length $n$ with parity check polynomial $h(x)$ contains a nontrivial degenerate cyclic subcode if and only if $\gcd(x^n - 1, h(x)) = 1$ for some $\delta < n$.

**Proof:** Suppose $\gcd(x^n - 1, h(x)) \neq 1$. Let $C_{\delta}$ be the cyclic code with parity check polynomial $h_{\delta}(x) = \gcd(x^n - 1, h(x))$. Since $h_{\delta}(x)$ divides $h(x)$, $C_{\delta}$ is a cyclic subcode of $C$. As $h_{\delta}(x)$ also divides $x^n - 1$, every codeword of $C_{\delta}$ is of the form

$$f(x) x^{n-1} h_{\delta}(x) = f(x) x^{n-1} h_{\delta}(x) (1 + x^\delta + x^{2\delta} + \cdots + x^{n-\delta}),$$

where $\deg f(x) < \deg h_{\delta}(x)$. Therefore $C_{\delta}$ is degenerate. Conversely, if $C_{\delta}$ is a nontrivial degenerate cyclic subcode of $C$ with parity check polynomial $h_{\delta}(x)$, then $h_{\delta}(x)|h(x)$ and $h_{\delta}(x)|x^n - 1$ for some $\delta < n$. Therefore, $\gcd(x^n - 1, h(x)) = 1$.

The condition of Lemma 2 may be also formulated in terms of the zeros of the cyclic code $C$. Obviously $\gcd(x^n - 1, h(x)) = 1 \iff (x^n - 1)$ and $h(x)$ have common roots. This is only possible if $\delta$
is a divisor of $n$ and therefore we may define $m = n/\delta$, where $m$ is an integer. Now let $\alpha$ be a primitive $m$th root of unity. Then the roots of $(x^m - 1)$ are $1, \alpha, \alpha^2, \ldots, \alpha^{m-1}$. Thus we have proved the following corollary.

**Corollary 2:** A linear cyclic code $C$ of length $n$ contains a nontrivial degenerate cyclic code if and only if some $m$, a factor of $n$, at least one of the elements of the set $(1, \alpha, \alpha^2, \ldots, \alpha^{m-1})$ is not a zero of $C$.

Comparing the following special cases of Lemma 2. If $gcd(x^k - 1, h(x)) = h(x)$, then the whole code is degenerate (see [9, p. 224]). On the other hand if $gcd(x^k - 1, h(x)) = x^k - 1$, i.e., $x^k - 1|b(x)$, then the degenerate code $C_d$ is isomorphic to $GF(q)^k$, where $GF(q)$ is the ground field of $C$. Indeed, the generator polynomial of $C_d$ is $1 + x^d + x^{2d} + \cdots + x^{d^t-1}$ and therefore $C_d$ consists of all the vectors of length $n$ and of the form $[u_1, u_2, \ldots, u_d]$, where $u \in GF(q)^d$. Choosing $d$ unit vectors from $GF(q)^d$ we obtain a binary zero-concuring subset of $C$ with $d$ elements. For $q = 2$ this reduces to (3). Hence, Lemma 2 may be viewed as a generalization of the second construction method of Beery and Snyders [1]. Also, if $C$ is a BCH code with zeros at $\alpha, \alpha^2, \ldots, \alpha^m$ and their cyclotomic conjugates, then $[8]$ for any $n/\delta = m \geq 2r+1$ none of the elements in the set $(1, \alpha, \alpha^2, \ldots, \alpha^{m-1})$ is a zero of $C$ and therefore $gcd(x^k - 1, h(x)) = x^k - 1$. Thus Corollary 2.1 may be regarded as a generalization of Theorem 2.3 of [8].

In general $C_d$ is a much smaller code than $C$. It is therefore easier to find a maximal set of zero-concuring codewords in $C_d$ rather than in $C$. In many cases this is so the maximal zero-concuring subset of the original code $C$. Thus, for example, for the $(65,40,10)$ binary BCH code with a generator polynomial given by $s(x) = 1 + x + x^2 + x^5 + x^6 + x^9 + x^19 + x^{22} + x^{26} + x^{31}$, we have $h(x) = gcd(x^k - 1, h(x)) = 1 + x + x^2 + x^5 + x^9 + x^{19}$. Hence $C_d$ is the $(65,12,2)$ degenerate cyclic code by the Kroncker product of the $(13,12,2)$ binary parity code and the $(5,1,5)$ repetition code. A maximal set of 6 zero-concuring codewords of $C_d$ is readily found to be

$$\begin{align*}
x^r(1 + x + x^{12} + x^{14} + x^{20} + x^{22} + x^{30} + x^{40} + x^{52} + x^{53});
\end{align*}$$

with $j = 0, 2, 4, 6, 8, 10$,

which is a maximal zero-concuring subset of the $(65,40,10)$ BCH code as well. Bound reaching zero-concuring subsets of many other BCH codes (for instance entries 2, 3, 10, 12, 13, 17–19, 39, and 44 of Table 1) may be obtained using this technique.

C. **Primitive BCH Codes with Distance One less than a Power of 2**

This construction is due to E. R. Berlekamp.

**Theorem 3:** Any primitive binary BCH code of length $n = 2^m - 1$ and minimum distance $d = 2^r - 1$ contains a set of $(n/d)$ zero-concuring codewords, each of weight $d$ or $d + 1$.

**Proof:** For any $i$, the $2^m - 1$ nonzero $m$-dimensional binary vectors can be partitioned into $[(2^m - 1)/(2^r - 1)]$ disjoint sets, such that each set is either the $2^r$ vectors in an $i$-dimensional affine subspace or the $2^r - 1$ nonzero vectors in an $i$-dimensional subspace of $GF(2)^m$ (the existence of such partition is proved in the Appendix). It is known [10, p. 363] that codewords of weight $2^r - 1 = m + r$, in the binary Reed–Muller code of order $r$ and length $n = 2^m$ are given by

$$c(x) = \sum_{i \geq A} a^i$$

where $A$ is an $i$-dimensional (affine) subspace of $GF(2^m)$ and $a_i = 1$ iff $a^i \in A$. The singly punctured Reed–Muller codes $RM^*(r, m)$ have one parity check (at location $m$) removed. Hence, the $[(2^m - 1)/(2^r - 1)]$ disjoint sets indicated in the foregoing theorem correspond to $[n/d]$ zero-concuring codewords in $RM^*(r, m)$, each of weight $d = 2^r - 1$ or $d + 1 = 2^r$. All primitive binary BCH codes whose designed distances are one less than a power of 2 are supercodes of singly punctured Reed–Muller codes of the same minimum distance, which proves the theorem. □

This construction was explicitly used to obtain bound reaching sets of zero-concuring codewords in entries 21, 23, and 27 of Table 1.

D. **Direct Product and Concatenated Codes**

Let $A$ and $B$ be respectively $(n_1 k_1, d_1)$ and $(n_2 k_2, d_2)$ linear codes over $GF(q)$. The direct product code $C = A \circ B$ is the $(n_1 n_2, k_1 k_2, d_1 d_2)$ code whose codewords consist of all $n_1 \times n_2$ arrays in which the columns belong to $A$ and the rows to $B$.

**Theorem 4a:** Let $J_1$ and $J_2$ denote the number of elements in a maximal zero-concuring subset of $A$ and $B$, respectively. Then $C$ contains a set of $J_1 J_2$ zero-concuring codewords.

**Proof:** The generator matrix of $C$ is given by $G = G_A \times G_B$, where $G_A$ and $G_B$ are the generator matrices of respectively $A$ and $B$, and $\times$ stands for the Kronecker product over $GF(q)$. Choose $G_A$ and $G_B$, such that the top $J_1$ and $J_2$ rows of $G_A$ and $G_B$, respectively, are zero-concuring. The Kronecker product of these rows produces the following set of $J_1 J_2$ zero-concuring codewords of $C$

$$\{g_{i j} : jk_1 + 1 \leq i \leq jk_2 + 1, \quad j \in [0, 1, \ldots, J_1 - 1]\}$$

where $g_{i j}$ denotes the $i j$th row of $G$.

Now let $C = A \circ B$ be an $(N, Kk, d^* \geq Dd)$ concatenated code over $GF(q)$, where the outer code $A$ is $(N, K, D)$ code over $GF(q^k)$ and the inner code $B$ is an $(n, k, d)$ code over $GF(q)$. □

**Theorem 4b:** Let $J_1$ and $J_2$ denote the number of elements in a maximal zero-concuring subset of $A$ and $B$, respectively, and let $A$ also contain a subset of $J_1$ zero-concuring codewords, such that all of its elements belong to $GF(q^k)$. Then there exists a zero-concuring subset of $C$ with $\max(J_1, J_2)$ elements.

**Proof:** The generator matrix of $C$ is again given by $G = G_A \times G_B$. Note that $\alpha G_B$, where $\alpha \in GF(q^k)$, is understood here as a product of a $k \times k$ matrix representation of $\alpha$ over $GF(q)$ and a $k \times n$ generator matrix of $B$. Now let $G_B$ be such that its top $J_2$ rows are zero-concuring. Taking $G_A$ either such that its top $J_1$ rows are zero-concuring or such that its top $J_1$ rows are zero-concuring and belong to $GF(q^k)$, proves the theorem. □

The foregoing theorem may be generalized as follows. Let $\Omega = \{a_1, a_2, \ldots, a_r\}$ be a basis of $GF(q^r)$ over $GF(q)$. When the inner encoder is presented at its input with a $q$-ary $k$-tuple corresponding to one of the elements of $\Omega$, it will produce at its output a scalar multiple of one of the rows of $G_B$. Now let $G_B$ have $J_2$ zero-concuring rows and let $F$ be some fixed subset of $\Omega$ of cardinality $J_2$; then the inner encoder may be chosen such that all the elements of $F$ are mapped onto zero-concuring $n$-tuples. Let us call two nonbinary codewords zero-concuring in general sense (z.c.g.) if they coincide only in positions containing zeros. Evidently, if the inner encoder is presented with z.c.g. vectors, such that all their entries belong to $F$, it will produce zero-concuring strings at its output. Thus we have proved the following.

**Theorem 4c:** Let $J_1$ denote the number of codewords in a maximal z.c.g. subset of $A$, such that all its elements belong to
$F^N$. Then there exists a zero-concurring subset of $C$ with $\max(J_1, J_2)$ elements.

The proposition of Theorem 4b is seen to be a special case of Theorem 4c if we construct a set of $J_1 = 1, J_2, \ldots$ vectors in $A$ by taking the $J_1$ zero-concurring codewords that belong to $G_{PA}^{Q_0}$ and all the multiples thereof by elements of $F$.

It would be interesting to investigate the case of general concatenated codes (GCC). For the definition of such codes and notation see for instance [9, p. 590]. Obviously, if GCC contains a usual concatenated code then Theorems 4b and 4c may be applied. If this is not the case other constructions should be used. As a simple example consider two outer codes: $A_1$, a code over $G_{PA}^{Q_0}$ and a binary code $A_2$. Thus the inner code $B_0$ is the union of $2^N - 1$ cosets of the code consisting of two words, say $(00 \cdots 0)$ and $(11 \cdots 1)$. Now let $A_1$ and $A_2$ both contain $J_1$ zero-concurring codewords of equal Hamming weights and let these codewords be such that their nonzero coordinates coincide. Then the number of zero-concurring codewords in GCC is at least $2J_1$.

It is noteworthy that if a Reed–Solomon code is used as the outer code in the concatenation the set of $J_1, J_2$ zero-concurring codewords of $C$ would often be bound reaching. Consider, for example, the $(120, 44, 20)$ binary code obtained by concatenating the $(15, 11, 5)$ RS code over $G_{PA}^{Q_0}$ with the $(8, 4, 4)$ extended binary Hamming code. In this case we have $J_1 = 2$ and $J_2 = 3$. Thus, a bound reaching set of $J_1, J_2$, zero-concurring codewords of the concatenated code may be constructed using Theorem 4b. As the Reed–Solomon codes are MDS, a maximal set of $[N/D]$ zero-concurring codewords of the RS code is given by (4). Hence, another direct consequence of Theorem 4b is that a $(2N, K, 2D)$ Justesen code obtained from an $(N, K, D)$ RS code contains a bound reaching set of $J_1 - [2N/2D]$ zero-concurring codewords. In general, since many cyclic codes are representable as either direct product or concatenated codes, the foregoing results suggest that an efficient method of search for zero-concurring codewords in a long cyclic code may be based on its factorization.

E. Constructing Zero-Concurring Codewords from Concurring Codewords

A set of independent codewords of $C$, a code over $G_{PA}^{Q_0}$, is called concurring if in some $f$ coordinates $i_1, i_2, \ldots, i_f$, all the codewords are identical (up to multiplication by a constant) and in all the other coordinates at most one of the codewords has a nonzero entry. Concurring codewords were extensively studied in the context of majority logic decoding. Blahut [13] provides an upper bound on the number of concurring codewords in a linear code of length $n$ and minimum distance $d$

$$J^* \leq \left \lfloor \frac{2n}{d} \right \rfloor - 1$$

(7)

Let $S$ be a set of $J^*$ concurring codewords of $C$. For any pair of codewords $e_i, e_j \in S$ there exists a linear combination $e_i = \alpha e_j + \beta e_j$, where $\alpha, \beta \in G_{PA}^{Q_0}$, such that $e_i$ contains zeros in coordinates $i_1, i_2, \ldots, i_f$. Thus if vectors of $S$ are taken together in pairs, each vector of $S$ occurring in at most one pair, then the resulting set $\{e_i : i = 1, 2, \ldots, [J^*/2]\}$ is a zero-concurring subset of $C$. If $J^*$ is odd, the remaining codeword (i.e., the one with no pair) may be appended to the foregoing zero-concurring subset. Hence, for any linear code

$$J \geq \left \lceil \frac{J^* + 1}{2} \right \rceil$$

(8)

Comparing (7), (8), and (1), a linear code that contains a bound reaching set of concurring codewords also contains a bound reaching zero-concurring subset.

It is known that the generalized Reed–Muller (GRM) codes are in many cases majority logic decodable. The dual of a GRM code over $G_{PA}^{Q_0}$ is also a GRM code over $G_{PA}^{Q_0}$. Thus, the GRM codes that attain the upper bound of (7) contain $[n/d]$ zero-concurring codewords. It may be shown [6] that the binary $(24, 12, 8)$ Golay code and the ternary $(12, 6, 6)$ Golay code have $[n/d] - 1$ concurring codewords. Hence, using the foregoing construction, they also contain bound reaching sets of zero-concurring codewords.

III. Algorithms

Each of the constructions derived in the previous section applies solely to a certain specific family of codes. Apparently in the case of a general linear code one has to adopt a computer search by some efficient algorithm. In fact, the table of maximal zero-concurring subsets of several binary linear codes, presented in [1], was compiled with the help of a "brute force" search. In such a brute force search all possible sets of distinct codewords are checked for being zero-concurring. Therefore, if checking whether two given codewords are zero-concurring is regarded as a single operation, then the number of operations required by the brute force search is of the order of

$$2 \cdot \left(\frac{q^k}{2}\right) + 3 \cdot \left(\frac{q^k}{3}\right) + \cdots + \min\{J + 1, \lfloor n/d \rfloor\} \cdot \min\{J + 1, \lfloor n/d \rfloor\}$$

which obviously makes such search prohibitive even for moderately large codes. In this section we derive several general algorithms for the search for zero-concurring codewords, with computational complexity substantially lower than that of the brute force search. The efficiency of these algorithms is manifested by the fact that we were able to obtain subsets with $[n/d]$ or $[n/d] - 1$ zero-concurring codewords for 44 codes in Table 1 in a typical running time of a few minutes for a code of length 127.

The formulation of our algorithms demands a certain elaboration of the well-known technique of shortening (or taking the cross-section of) linear codes. As before, $G$ is a generator matrix of a linear code $C(n, k, d)$. The cross-section weight of $e \in C$, $w_C(e)$, is the rank of the submatrix $A_e$ of $G$ consisting of the columns that correspond to the nonzero coordinates of $e$.

The foregoing definition is unambiguous, in the sense that the cross-section weight of a codeword does not depend on the choice of $G$, since the rank of a matrix is invariant under elementary row operations. Furthermore, it is easily verified that if $C$ is binary the cross-section weight generates a norm in the linear space $C$ in a way similar to the Hamming weight, which can also be viewed as a norm in this space.

Lemma 3: The set $C$ of all the codewords of $C$ that are nonconcurring to a specific codeword $e \in C$ is a linear code $C(n, k', d')$ obtained by means of shortening the original code $C$ by all the nonzero coordinates of $e$, where $k' = k - w_C(e)$.

Proof: Let $L = \{i_1, i_2, \ldots, i_k\}$ be the set of nonzero coordinates of $e$. As the rank of $A_e$, by definition $w_C(e)$, there exists a nonsingular $k \times k$ matrix $E$, such that the top $w_C(e)$ rows of $EA_e$ are linearly independent and all the other rows are zero. Since $E$ is nonsingular, $EG$ is also a generator matrix of $C$. Let $G'$ be a matrix obtained by deleting the top $w_C(e)$ rows for $EG$. Evidently $G'$ is a row submatrix of $EG$, such that the columns corresponding to the nonzero coordinates of $e$ contain only zero entries. Therefore $G'$ generates a shortened subcode of $C$, given
by

$$C' = \{ e = (c_1, c_2, \ldots, c_L) : e \in C \text{ and } c_i = 0 \text{ for all } i \in L \}$$

which is the required set of all the codewords of \( C \) that are nonconcurring to \( e \). The dimension of \( C' \) is by construction

$$k' = \text{rank}(G) = k - w_0(e).$$

Alternatively, the dimension of \( C' \) may be expressed as \( k' = k - w_0(e) + a \), where \( w_0(e) \) is the Hamming weight of \( e \), and \( a \) is the dimension of the code that consists of all the codewords of the dual code of \( C \) whose nonzero coordinates are confined to \( \{i_1, i_2, \ldots, i_s\} \). Given \( G \) and \( e \), a generator matrix of \( C' \) may be obtained by performing elementary row operations on \( G \). We refer to this as performing cross-section on \( G \) by \( e \). Now let \( Z \) be a given zero-concurring subset of \( C \). We say that \( Z \) is complete if it is not contained in any larger zero-concurring subset of \( C \). It follows from Lemma 5 that \( Z \) is complete if and only if \( w_0(e) = k \), where \( e \) is the sum of all the elements of \( Z \). This implies the following recursive algorithm.

**Algorithm A**

1) Select a nonzero codeword \( e \in C \). Let \( C' \) be the subcode of \( C \) resulting from performing cross-section on \( G \) by \( e \).

2) If \( k' = 0 \), set \( Z_e = \{ e \} \); otherwise use Algorithm A to find a maximal zero-concurring subset of \( C' \), say \( Z_{e'} \), and set \( Z_e = Z_{e'} \cup \{ e \} \).

3) Repeat Steps 1 and 2 for all the nonzero codewords of \( C \), unless at some execution of Step 2 a bound reaching set of zero-concurring codewords is found.

A set \( Z^*_e \) such that for all \( e \in (C - \{\emptyset\}) : |Z^*_e| \geq |Z_e| \), with respect to a set denotes its cardinality, is a maximal zero-concurring subset of the code \( C \).

The main idea of the foregoing algorithm is to consider only complete zero-concurring subsets of \( C \) instead of all the possible subsets of \( C \) as in the brute force approach. This results in a substantial computational gain. Let \( M \) and \( M' \) be the number of codewords in \( C \) and \( C' \), respectively. It follows from Lemma 5 that

$$M' = M / q^{wd}$$

where \( d^* \) is the minimum cross-section weight of \( C \). Thus, \( M / q^w \) is an upper bound on \( M' \); yet it does not provide sufficient insight. It will be shown in the sequel (see Lemma 6) that \( d^* \leq d \), with strict equality for about half of the linear codes. Therefore, for the purpose of estimating the computational complexity of Algorithm A, we approximate the average value of \( M' \) by

$$M' \approx M / q^{w}.$$

This estimate has been experimentally verified for many codes, yielding in all cases an upper bound on the actual average value of \( M' \). We assume that performing cross-section requires, on the average, about \( nk / 2 \) elementary row operations. Then substituting \( q^w \) for \( M \) gives the following expression for the computational complexity of Algorithm A in terms of the total number of elementary row operations required by the algorithm

$$N_k = \frac{nk}{2} \sum_{j=0}^{k-1} \left( \frac{q^w}{j} \right)^j \sum_{\gamma \in GF(q)} \gamma^j,$$

$$= \frac{nk}{2} \left( q^w + q^{2w-d} + q^{3w-3d} + q^{4w-4d} + q^{5w-5d} + \cdots \right).$$

The complexity of Algorithm A may be considerably reduced if we remove the necessity to repeat Steps 1 and 2) of the algorithm for all the nonzero codewords of \( C \). This calls for some selection criterion that could be used to distinguish between the elements of a maximal zero-concurring subset of the code and all other codewords of \( C \). Let \( Z_e \) be one of the \( q^w - 1 \) zero-concurring subsets of \( C \) generated by Algorithm A, and let \( Z_e \) be a maximal zero-concurring subset of \( C \), eventually resulting from the algorithm. We define the average cross-section weight of a set \( Z_e \) as

$$w_e(Z_e) = \frac{1}{|Z_e|} \sum_{e \in Z_e} w_0(e),$$

where \( w_0(e) \) is the cross-section weight of a codeword \( e \in Z_e \) in the code obtained at stage \( j \) of recursion in Algorithm A. By Lemma 5,

$$\sum_{e \in Z_e} w_0(e) = \sum_{e \in Z_{e'}} w_0(e) + k,$$

and since \( |Z_e| \geq |Z_{e'}| \), we have \( w_e(Z_e) \leq w_{e'}(Z_{e'}). \) Consequently, if one of the \( q^w - 1 \) sets \( Z_e \) could be chosen, such that its average cross-section weight would be the smallest—this set would be a maximal zero-concurring subset of \( C \). Hence, we suggest the minimum cross-section weight selection criterion to be employed at Step 1 of the following algorithm.

**Algorithm B**

1) Select a codeword \( v \in C \), such that for all \( e \in (C - \{\emptyset\}): w_v(e) < w_e(e), \) and obtain \( C' \) by performing cross-section on \( G \) by \( v \).

2) If \( k' = 0 \) set \( Z_v = \{ v \} \); otherwise use Algorithm B to find a zero-concurring subset of \( C' \), say \( Z_{v'} \), and set \( Z_v = Z_{v'} \cup \{ v \} \).

It is conjectured that the set \( Z_v \) is a maximal zero-concurring subset of \( C \). In fact, no example to the contrary has been found. If a minimum cross-section weight codeword of \( C \) is selected with the help of an exhaustive search through the code then the number of elementary row operations required by Algorithm B is given by

$$N_v = \frac{1}{2} \left( nkq^k + n_1(k - d^*) q^{d^* - d} \right) + n_2 \left( k - d^* - d^* \right) q^{d^* - d} + \cdots \right) \approx \frac{nk}{2} q^{k^w},$$

where \( d^* \) and \( n_1 \) are the minimum cross-section weight and the block length of the code obtained at stage \( j \) of recursion in Algorithm B. This may be cut down further if instead of an exhaustive search a more efficient method is employed to find a minimum cross-section weight codeword of the code. Let \( G \) be a systematic generator matrix of \( C \) and let \( e = sG \), where \( s \in GF(q)^k \) is a source vector. Then evidently \( w_v(e) \geq w_0(s) \). Hence, we may consider source vectors in order of nondecreasing Hamming weight and terminate on \( w_v(s) = d^* \). Note that \( d^* \) need not be a priori known. The worst case complexity of a algorithm utilizing this property of a systematic generator matrix is given by

$$N = \frac{nk}{2} \sum_{i=1}^{d^*} \left( \frac{k}{q-1} \right)^i,$$

where \( N \) stands for the number of elementary row operations required. Obviously \( (10) \) is always lower than the complexity of exhaustive search given by \( (9) \). In addition, for a given \( k \) this is a rapidly decreasing function of \( d^* \) thus making the algorithm efficiently applicable to long high-rate codes where \( d^* \) is small. If this is not the case but \( d^* \) is a priori known, the algorithm may be terminated on \( w_0(e) = d^* \). The problem is that \( d^* \) is seldom known and, hence, the complexity of Algorithm B remains essentially exponential. A suboptimal polynomial time algorithm may be derived with the help of the following result.

**Lemma 6:** Let \( C = (n, n - k, d^*) \) be the dual code of \( (n, k, d^*) \). Then the minimum cross-section weight of \( C \) is bounded by

$$\min \{d^* - 1, d\} \leq d^* \leq d.$$
exist, such that \( w_{p}(c) \leq d - 1 \). For these codewords \( \Lambda_{c} \) contains \( w_{p}(c) \) linearly independent columns, so that \( w_{p}(c) = \text{rank}(\Lambda_{c}) = w_{p}(c) \geq d \). □

It follows from the foregoing lemma and its proof that whenever \( d \leq d' \leq d' \geq d \) and a minimum Hamming weight codeword of the code is also a minimum cross-section weight codeword. This justifies, to some extent, substitution of the minimum cross-section weight selection criterion in Algorithm B by the minimum Hamming weight criterion. We shall refer to the modified algorithm as Algorithm C. Algorithm C is suboptimal, in the sense that it does not necessarily provide a maximal set of zero-concurring codewords. Consider, for example, a binary code given by the following generator matrix,

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

It can be readily verified that the maximal zero-concurring subset of the code is

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

Yet, applying Algorithm C gives

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

Apparently many other such codes exist. However, in all but one of the 45 codes in Table 1 zero-concurring subsets with \( n/d \) or \( n/d - 1 \) elements have been found with the help of Algorithm C. The complexity of Algorithm C is given by

\[ N_{j} = \sum_{n_{cunt}(j, k, d, j)} \]

where \( j \) is upper bounded by \( n/d \); \( n_{cunt}(j, k, d, j) \) is the computational complexity of the algorithm employed for the search of a minimum Hamming weight codeword in the code. As the minimum Hamming distance of many codes is known we may use any kind of search and terminate on \( w_{p}(c) = d \). Alternatively, since Algorithm C makes no claim on optimality, one may employ a suboptimal polynomial time algorithm to find minimum weight codewords (such an algorithm was proposed by Harari [14]). In this case the computational complexity of Algorithm C would be polynomial.

A different approach to reducing the complexity of proposed algorithms may be based on employing the automorphism group of the code. Let \( \pi \) be a permutation on the set \( \{1, 2, \ldots, n\} \) sending each \( i \) to \( \sigma(i) \) and each \( c = (c_{1}, c_{2}, \ldots, c_{n}) \) into \( c^{\pi} = (c_{\sigma(1)}, c_{\sigma(2)}, \ldots, c_{\sigma(n)}) \). We denote by \( C_{\pi} \) and \( C_{\pi}^{\pi} \) the set of all the codewords of \( C \) that are nonconcurring to, respectively, \( c \) and \( c^{\pi} \). Evidently, if \( \sigma \in \text{Aut}(C) \), where \( \text{Aut}(C) \) is the automorphism group of \( C \), the codes \( C_{\pi} \) and \( C_{\pi}^{\pi} \) are equivalent and, hence, contain the same number of zero-concurring codewords. Now let the set \( \Omega_{\pi} \) be defined by \( \Omega_{\pi} = \{a \in C : a = c^{\pi} \} \) for some \( \pi \in \text{Aut}(C) \); then \( \Omega_{\pi} \cap \Omega_{\pi} = \phi \) if and only if \( c_{\pi} \neq c_{\pi} \). Otherwise, any linear code \( C \) may be partitioned into \( t \) disjoint cosets \( \Omega_{1}, \Omega_{2}, \ldots, \Omega_{t} \), where the codewords \( c_{1}, c_{2}, \ldots, c_{t} \) are automorphic representatives of \( C \). It follows that when searching for a minimum cross-section weight codeword (Algorithm B) or for a maximal set of zero-concurring codewords (Algorithm A) it is sufficient to consider the set of automorphic representatives of \( C \) instead of the entire code. This implies computational gain of approximately \( q^{t}/t \) at each stage of recursion, provided that the automorphism group of the code and its automorphic representatives can be efficiently found. A polynomial time algorithm for computing automorphic representatives of linear codes appears in [15]. The problem of finding automorphic representatives is, to the best of our knowledge, still open. However, in the special case of cyclic permutations the situation is different. A way to find cyclic representatives of a minimal cyclic code is described in [9] and [16]. If the code is not minimal one may employ other techniques [17].

IV. Results

All the algorithms derived in the previous section are general, in the sense that they apply to any linear code. We arbitrarily chose the family of primitive binary BCH codes as a "benchmark" for the proposed algorithms. Some results obtained for these codes and also for several nonprimitive BCH codes are listed in Table 1. Entries 1-4, 7-9, 19, 20, 37-40 were obtained with the help of Algorithm A. Entries 5, 6, 11, 17, 18, 22, 41 follow by applying Algorithm B. Entries 10, 12, 13, 41 and 44 were derived by using Lemma 2. Entries 21, 23, and 27 are due specifically to the construction of Theorem 3. All other
entries in the table were obtained by means of Algorithm C. The
(127,99,9) BCH code is the only code in the table for which
Algorithm C provides less than \([n/d]−1\) zero-concurring code-
words. The (63,18,21) BCH code is peculiar in that the maximal
set of zero-concurring codewords, namely \(\{x1 + x^3 + x^7 + \cdots + x^{15}\} \oplus \{x2 + x^8 + x^{12} + x^{14} + x^{15} + x^{20} + x^{22} + x^{24}\}\) (1 = 0, 1, 2), is unique (it is invariant under the
automorphism group of the code; any other zero-concurring set
contains at most 2 codewords).

The upper bound is everywhere \([n/d]\), with two exceptions
where this bound may be tightened. Short proofs of the tighter
bound follow.

1) Consider the (41,20,10) binary BCH code with generator
polynomial \(g(x) = 1 + x^3 + x^5 + x^7 + x^8 + x^{13} + x^{14} + x^{15} + x^{16} + x^{18} + x^{19} + x^{21}\). Assume that there exists a subset of \([n/d] = 4\) zero-concurring codewords. Since \((x + 1)\) in \([n/d]\) is a factor of \(g(x)\), the sum of all the 4 codewords
must have weight 40. It follows that the augmented code
generated by \(g(x)/x+1\) has to contain a codeword of
weight 1. Yet, by the BCH bound the minimum distance of the
augmented code is at least 9, which contradicts the
initial assumption. Consequently, the (41,20,10) BCH code
contains no more than 3 zero-concurring codewords.

2) Consider the (21,12,5) BCH code. It is precisely the
(24,12,8) Golay code punctured in three coordinates (cf.
[11]). If the punctured code contained two codewords of
weight 5 that were nonconcurring, then the code should
coincide with two Golay codewords of weight 8 that occurred
in the 3 punctured coordinates. However, it is well known that
the (24,12,8) Golay code contains only codewords of
weights divisible by 4 and no codewords of weight 10.
Hence, it does not contain any pair of codewords of weight
8 that intersect in three coordinates. Therefore, any set of
zero-concurring codewords of the (21,12,5) code contains,
at most, one codeword of weight 5. From this it follows
that there can be no more than 3 codewords in such a set.

The remaining seven codes for which the upper and lower
bounds do not agree are marked with an asterisk.

A nonnegative integer \(k\) is said to be the contraction index
of a linear \((n, k)\) code \(C\), if a maximal set of pairwise linearly
independent columns of a generator matrix of \(C\) contains
exactly \(k\) columns. Given this notation, finding a maximal
zero-concurring subset of \(C\) is equivalent to finding the highest
dimension subcode of \(C\) with contraction index zero. Thus the
problem discussed herein may be viewed as a special case of a
more general problem of finding large subcodes with small
contraction index in a given code. Algorithms employing such
subcodes for efficient soft-decision decoding appear in [6], while
bounds on the dimension of codes and subcodes with prescribed
contraction index are presented in [18].

ACKNOWLEDGMENT

The authors are greatly indebted to Elwyn R. Berlekamp for the
contribution of Theorem 3 and valuable comments. They are also
grateful to G. David Forney, Jr. for preprints of his papers,
to Jacko Snoj for helpful discussions and to the referees for
many constructive remarks. Alexander Vardy wishes to thank
Hagit Itzkowitz for her invaluable help.

APPENDIX

This Appendix shows the existence of partition used in the
proof of Theorem 3 [12]. For each fixed \(j\), \(0 \leq j \leq t\), we describe
a construction which, by induction, covers \(m = j + i, j + 2i, j + 3i\)

\[
\begin{align*}
& s_1 \quad z \\
& s_2 \quad z \\
& s_{2^i} \quad z \\
& s_{2^i-1} \quad z
\end{align*}
\]

where \((s_1, s_2, \ldots, s_{2^i-1}, 0)\) constitutes an \((m+i)\)-dimensional subspace.

We then include in an \((m+i)\)-dimensional partition one set of the
form

\[
\begin{align*}
& s_1 \quad z \\
& s_2 \quad z \\
& \quad \vdots \\
& s_{2^i-1} \quad z \\
& \quad \vdots
\end{align*}
\]

and \(2^{i+1}\) sets of the form

\[
\begin{align*}
& (a^1 s_1) \quad z \\
& (a^1 s_2) \quad z \\
& \quad \vdots \\
& (a^1 s_{2^i-1}) \quad z
\end{align*}
\]

where \(a\) is a primitive element of GF(\(2^{i+1}\)) and \(a^x\) stands for \(0\).

The \(i\)-tuples \(v_1, v_2, \ldots, v_{2^i-1}\) are the \(2^i-1\) nonzero binary
\(i\)-tuples in some fixed order, such that \(v_i + v_j = v_k\) whenever
\(s_i + s_j = s_k\). It is easily verified that each of the above sets
together with \(0\) is a subspace of GF(\(2^{m+i}\)). Finally, if \((b_1, b_2, \ldots, b_{2^i-1})\)
is a subspace in the \((m+i)\)-dimensional
partition and if it is ordered lexicographically, i.e., so that
\(b_1 < b_2 < \cdots < b_{2^i-1}\), then we include in our partition for \(m+i\) dimensions
all \(2^i\) sets of the form

\[
\begin{align*}
& b_1 \quad (a^1 v_1) \\
& b_2 \quad (a^1 v_2) \\
& \quad \vdots \\
& b_{2^i-1} \quad (a^1 v_{2^i-1})
\end{align*}
\]

where \(v_1, v_2, \ldots, v_{2^i-1}\) are the nonzero elements of GF(\(2^i\)), in
lexicographic order, and \(a\) is a primitive element of GF(\(2^i\)).

Altogether we have

\[
(2^{i+1} + 1 + 2^{i+1} + 2^{i+1}/2^{i+1} - 1) = \frac{2^{m+i} - 1}{2^i - 1}
\]

disjoint sets. This completes the construction.

REFERENCES


More on the Minimum Distance of Cyclic Codes
P. J. N. de Rooij and J. H. van Lint

Abstract—It was recently shown that the so-called Jensen bound is generally weaker than the product method and the shifting method introduced by van Lint and Wilson. We show that the minimum distance of the two cyclic codes of length 65 for which it is known that the product method does not produce the desired result can be proved using Jensen's method with some adaptations.

Index Terms—Minimum distance, 2-D-cyclic code, concatenated code, shifting.

I. INTRODUCTION

In 1986 a new method for calculating the minimum distance of cyclic codes was developed by J. H. van Lint and R. M. Wilson [4]. Their paper contained two related methods: a maximal product method and a method called "shifting." Previous bounds, such as the BCH bound, the Hartman–Tzeng bound and the method developed by Roos are all special cases of this method. It turned out that the minimum distance of all cyclic codes of length less than 63 (all codes in that paper are binary codes) with two exceptions can be determined using this method. The number of cyclic codes of length 63 is exceedingly large, and it is still not clear how many of them can be handled by this method. For the codes of length 65, it was shown by M. H. M. Smit [7] that again all but two of these codes can be handled by the product method. The first purpose of this correspondence is to determine the minimum distance of these two exceptional codes (for which presently only computer searches have established the minimum distance).

In 1985 J. M. Jensen [3] developed another method for calculating the minimum distance of cyclic codes based on the idea of Berlekamp and Justesen of representing these codes as two-dimensional cyclic codes. Jensen's method was recently analyzed by the first author in his master's thesis [6] with the rather disappointing result that the method is usually weaker than shifting. (However, the amount of computation required for shifting is often quite large.) The second purpose of this correspondence is to show that, with some extra work, Jensen's method is strong enough to handle the two cyclic codes of length 65 that could not be done by the product method. Clearly, it is not of great importance to consider two isolated examples of length 65, but the method of this correspondence can be used in many other situations e.g., for Block–Zyablov codes [2]. Hence, explaining the methods that we use in our examples in Section III is our main goal.

In the following, we shall use terminology, notation, and results from the paper by Jensen on the structure of cyclic codes. We assume that the reader is familiar with that paper and also with the product method. In Section II we only briefly review what we shall need in the sequel.

II. DEFINITIONS

Let G be an Abelian group of order nN that is the direct product of two cyclic subgroups G and G, of order resp. N, that is, G = G × G, contains (w.l.o.g.) the elements (x1y1) ≜ i ≤ i + j ≤ N), (x1 = y1 = 1). Furthermore let g be a prime power and gcd(nN, g) = 1.

Definition 2.1: The group algebra FgG is the ring (with unity) consisting of all (formal) polynomials

\[ f(x, y) = \sum_{i=0}^{n-1} \sum_{j=0}^{N-1} c_{ij} x^i y^j, \]

where c_{ij} \in F_g.

Definition 2.2: A 2-D cyclic code of size n by N over the alphabet F_g is an ideal in F_gG.

We represent a codeword by the corresponding polynomial or by the n×N matrix [c_{ij}]. We shall not distinguish between these notations.

If gcd(nN, 1) = 1, then the Chinese remainder theorem shows that every element x^i y^j is a power of Z = xy; so, Z is a generator of G. Thus G is cyclic.

Using this, the following can be derived (cf. [1]).

Theorem 2.3: If gcd(nN, 1) and G and g are as above, then a 2-D cyclic code G' in F_gG is cyclic.

The converse is true as well.

Theorem 2.4: A cyclic code of length nN, with gcd(nN, 1) is 2-D cyclic.

In the following, \theta will denote a minimal cyclic code and we shall use the symbol \theta for the idempotent of this code. It is well

More on the Minimum Distance of Cyclic Codes
P. J. N. de Rooij and J. H. van Lint

Abstract—It was recently shown that the so-called Jensen bound is generally weaker than the product method and the shifting method introduced by van Lint and Wilson. We show that the minimum distance of the two cyclic codes of length 65 for which it is known that the product method does not produce the desired result can be proved using Jensen's method with some adaptations.

Index Terms—Minimum distance, 2-D-cyclic code, concatenated code, shifting.

I. INTRODUCTION

In 1986 a new method for calculating the minimum distance of cyclic codes was developed by J. H. van Lint and R. M. Wilson [4]. Their paper contained two related methods: a maximal product method and a method called "shifting." Previous bounds, such as the BCH bound, the Hartman–Tzeng bound and the method developed by Roos are all special cases of this method. It turned out that the minimum distance of all cyclic codes of length less than 63 (all codes in that paper are binary codes) with two exceptions can be determined using this method. The number of cyclic codes of length 63 is exceedingly large, and it is still not clear how many of them can be handled by this method. For the codes of length 65, it was shown by M. H. M. Smit [7] that again all but two of these codes can be handled by the product method. The first purpose of this correspondence is to determine the minimum distance of these two exceptional codes (for which presently only computer searches have established the minimum distance).

In 1985 J. M. Jensen [3] developed another method for calculating the minimum distance of cyclic codes based on the idea of Berlekamp and Justesen of representing these codes as two-dimensional cyclic codes. Jensen's method was recently analyzed by the first author in his master's thesis [6] with the rather disappointing result that the method is usually weaker than shifting. (However, the amount of computation required for shifting is often quite large.) The second purpose of this correspondence is to show that, with some extra work, Jensen's method is strong enough to handle the two cyclic codes of length 65 that could not be done by the product method. Clearly, it is not of great importance to consider two isolated examples of length 65, but the method of this correspondence can be used in many other situations e.g., for Block–Zyablov codes [2]. Hence, explaining the methods that we use in our examples in Section III is our main goal.

In the following, we shall use terminology, notation, and results from the paper by Jensen on the structure of cyclic codes. We assume that the reader is familiar with that paper and also with the product method. In Section II we only briefly review what we shall need in the sequel.

II. DEFINITIONS

Let G be an Abelian group of order nN that is the direct product of two cyclic subgroups G and G, of order resp. N, that is, G = G × G, contains (w.l.o.g.) the elements (x1y1) ≜ i ≤ i + j ≤ N), (x1 = y1 = 1). Furthermore let g be a prime power and gcd(nN, g) = 1.

Definition 2.1: The group algebra FgG is the ring (with unity) consisting of all (formal) polynomials

\[ f(x, y) = \sum_{i=0}^{n-1} \sum_{j=0}^{N-1} c_{ij} x^i y^j, \]

where c_{ij} \in F_g.

Definition 2.2: A 2-D cyclic code of size n by N over the alphabet F_g is an ideal in F_gG.

We represent a codeword by the corresponding polynomial or by the n×N matrix [c_{ij}]. We shall not distinguish between these notations.

If gcd(nN, 1) = 1, then the Chinese remainder theorem shows that every element x^i y^j is a power of Z = xy; so, Z is a generator of G. Thus G is cyclic.

Using this, the following can be derived (cf. [1]).

Theorem 2.3: If gcd(nN, 1) and G and g are as above, then a 2-D cyclic code G' in F_gG is cyclic.

The converse is true as well.

Theorem 2.4: A cyclic code of length nN, with gcd(nN, 1) is 2-D cyclic.