Prefixless q-ary Balanced Codes with ECC

Theo G. Swart
Dept. of Electrical and Electronic Engineering Science,
University of Johannesburg, South Africa
Email: tgswart@uj.ac.za

Kees A. S. Immink
Turing Machines Inc,Willemskade 15b-d,
3016 DK Rotterdam, The Netherlands
E-mail: immink@turing-machines.com

***Abstract*—We present a Knuth-like method for balancing q-ary codewords, which is characterized by the absence of a prefix that carries the information of the balancing index. Look-up tables for coding and decoding the prefix are avoided. We also show that this method can be extended to include error correction of single channel errors.**

I. Introduction

Balanced, sometimes called dc-free, q-ary sequences have found widespread application in popular optical recording devices such as Compact Disc, DVD and, Blu-Ray [1], cable communication, and recently in non-volatile (Flash) memories [2]. Prior art codes were presented by Capocelli *et al.* [3], Tallini and Vacaro [4], and Swart and Weber [5].

Let $\boldsymbol{x} = (x_1, \ldots, x_m)$ be a word of m q-ary symbols, q and m positive integers, taken from the q-ary alphabet $\mathcal{Q} = \{0, 1, \ldots, q-1\}$, $q \geq 2$. The *weight*, or unbalance, of $\boldsymbol{x}$, denoted by weight($\boldsymbol{x}$), is simply defined as the sum of the m q-ary symbols, that is,

$$\text{weight}(\boldsymbol{x}) = \sum_{i=1}^{m} x_i.$$

An m-symbol codeword $\boldsymbol{x}$ is said to be balanced if

$$\text{weight}(\boldsymbol{x}) = \frac{m(q-1)}{2}. \tag{1}$$

For certain practical applications, it is a desideratum to generate balanced q-ary sequences. Clearly, look-up tables can be used in case the sequences are not too long. Knuth [7] described a simple encoding technique for generating binary, $q = 2$, balanced codewords, which is capable of handling (very) long binary blocks. Swart and Weber [5] generalized this idea to q-ary sequences. In both cases, an additional prefix is necessary for decoding the sequence. In this paper we will make use of this simple encoding technique to construct balanced codes which do not require a prefix for decoding.

In Section II, we start with relevant results from the literature. Section III presents a new method for constructing prefixless q-ary balanced codes. In Section IV, we introduce error correction in addition to balancing and in Section V investigate the redundancy and performance of the new scheme. Finally, we conclude in Section VI.

II. Background

In Knuth's algorithm, a binary ($q = 2$) m-bit user word (pay load), m even, is forwarded to the encoder. The encoder splits the user word into a first segment consisting of the first v bits of the user word, and a second segment consisting of the remaining $m-v$ bits. The encoder adds (modulo 2) a '1' to the $m-v$ symbols in the second segment. The index v is chosen in such a way that the modified word is balanced. Knuth showed that such an index v can always be found. In the simplest case, the index v is represented by a balanced word, called *prefix*, of length p. The balanced p-bit prefix and the balanced m-bit user word are both transmitted, and the rate of the code is simply $m/(m+p)$. After observing the p-bit prefix, the receiver can easily undo the modifications made. Note that both encoder and decoder do not require look-up tables for the pay load, and we conclude that Knuth's algorithm is very attractive for constructing long balanced codewords. Note, however, that the scheme does require look-up tables for encoding and decoding the prefix. Modifications of the generic binary scheme have been discussed by Al-Bassam and Bose [8], Tallini *et al*. [9], and Weber and Immink [10]. Binary balancing schemes that include error correction has been presented by Al-Bassam and Bose [11] and Weber *et al*. [12].

Capocelli *et al*. [3], Tallini and Vacaro [4], and Swart and Weber [5] generalized Knuth's binary scheme to the balancing of q-ary codewords, $q > 2$. In [5], balancing is achieved, as in Knuth's scheme, by splitting the user word into a first and second segment of v and $m - v$ symbols, respectively. The encoder adds (modulo q) an integer $s \in \mathcal{Q}$ to the symbols in the first segment, and an integer $s + 1$ to the symbols in the second segment. Swart and Weber showed that there exists at least one pair of integers s and v such that the modified word is balanced. The integers s and v are represented by a balanced q-ary p-symbol prefix, which is appended to the balanced codeword, and subsequently transmitted to the receiver. The prefix must therefore be sufficiently long to be able of representing qm distinct pairs of integers s and v. As in the binary 'Knuth' case, look-up tables for encoding and decoding the prefix are required which may be prohibitively complex for large values of p. A further improvement of [5] was presented by Pelusi *et al*. [6].

III. Prefixless Balanced Codes

As before, let $\boldsymbol{x} = (x_1, \ldots, x_m)$ be a word of m, q-ary symbols, $x_i \in \mathcal{Q}$, where q and m are chosen such that $(q-1)m/2$ is an integer. The word $\boldsymbol{w} = (w_1, \ldots, w_m)$ is obtained by modulo q integration of $\boldsymbol{x}$, that is by the following

978-1-4673-0223-4/13/$31.00 ©2013 IEEE

operation

$$w_i = w_{i-1} \oplus_q x_i, \quad 1 \le i \le m,$$

where $w_0 = 0$, and the $\oplus_q$ sign indicates modulo q summation. The above operation, often called *precoding*, will be denoted by the shorthand notation $\boldsymbol{w} = I(\boldsymbol{x})$. Note that the original word $\boldsymbol{x}$ can be uniquely restored by modulo q *differentiation*:

$$x_i = w_i \ominus_q w_{i-1}, \quad 1 \le i \le m, \tag{2}$$

where $\ominus_q$ indicates modulo q subtraction. The above differentiation operation will be denoted by $\boldsymbol{x} = I^{-1}(\boldsymbol{w})$. Clearly, $I^{-1}(I(\boldsymbol{x})) = \boldsymbol{x}$.

Define the binary m-bit word $\boldsymbol{u}_v = (0^{v-1}10^{m-v})$ (that is, $\boldsymbol{u}_v$ consists of '0's except a single '1' at position v). We are now in the position to formulate Theorem 1.

Theorem 1 *There is at least one pair of integers, s and v, $s \in \mathcal{Q}$, $v \in \{1, \ldots, m\}$, such that $I(\boldsymbol{x} \oplus_q \boldsymbol{u}_v \oplus_q s\boldsymbol{u}_1)$ is balanced, that is weight$(I(\boldsymbol{x} \oplus_q \boldsymbol{u}_v \oplus_q s\boldsymbol{u}_1)) = m(q-1)/2$.* □

PROOF Trivial considering Theorem 1 from [5]. ■

Example 1 Let $q = 5$ and $m = 6$, and let the pay load be $\boldsymbol{x}$ = (4,2,1,0,0,0). After a search, we find $s = 2$ and $v = 5$. Adding $\boldsymbol{u}_v \oplus_q s\boldsymbol{u}_1$ = (2,0,0,0,1,0) to the pay load, yields $\boldsymbol{y}$ = (1,2,1,0,1,0). After precoding $\boldsymbol{y}$, we obtain $\boldsymbol{w} = I$(1,2,1,0,1,0) = (1,3,4,4,0,0). And we may verify that $\boldsymbol{w}$ is balanced since the sum of its entries equals $(q-1)m/2 = 12$. □

Using the above theorem, we will show that the precoding operation combined with error correction may lead to an efficient construction of a balanced code.

For the balancing of a q-ary word, we must find a pair of integers s and v, $s \in \mathcal{Q}$ and $v \in \{1, \ldots, m\}$. Note that in the binary case, $q = 2$, the search is restricted to finding the balancing index v. There is, except full search, no simple algorithm available for computing s and v.

The next encoding and decoding algorithm exploits Theorem 1 and we will show below that in conjunction with error correcting or detecting codes, it will be possible to efficiently balance q-ary words, and circumvent the encoding and decoding of the prefix in the prior art construction.

A. Encoding

The encoding procedure consists of four steps. We will make use of a q-ary $(m-1, k)$ linear block code of dimension k and length $m-1$. The encoding function is denoted by ϕ_q. Let $r' = m-1-k$ be the redundancy of the block code, and define the $r' \times (m-1)$ matrix $C_{q,r'}$ whose i-th column c_i is the q-ary representation of the integer i, $1 \le i \le m-1$, $m \le q^{r'}$. For example, for $q = 3$, $r' = 2$, and $m = 9$ we obtain

$$C_{3,2} = \begin{bmatrix} 0 & 0 & 1 & 1 & 1 & 2 & 2 & 2 \\ 1 & 2 & 0 & 1 & 2 & 0 & 1 & 2 \end{bmatrix}. \tag{3}$$

We call $C_{q,r'}$ a *check matrix*, for which we have an easy syndrome decoding available similar to that of binary *Hamming* codes [13]. This forms a single, magnitude-one error correction code, or a single error detection code. The maximum row length of the check matrix $C_{q,r'}$ is $q^{r'} - 1$, $r' > 1$. The coding function $\boldsymbol{x}' = \phi_q(\boldsymbol{a})$ is defined in such a way that $\boldsymbol{x}'$ satisfies $C_{q,r'}\boldsymbol{x}'^T = 0$.

Step 1: The k-symbol pay load, $\boldsymbol{a}$, is encoded into the codeword $\boldsymbol{x}' = \{x'_1, \ldots, x'_{m-1}\}$ using the q-ary $(m-1, k)$ linear block code, i.e. $\boldsymbol{x}' = \phi_q(\boldsymbol{a})$.

Step 2: The m-symbol word x is obtained by prefixing a redundant '0' to $\boldsymbol{x}'$, that is, $\boldsymbol{x} = \{0, x'_1, \ldots, x'_{m-1}\}$.

Step 3: Using Theorem 1, find a pair of integers, $s \in \mathcal{Q}$ and $v \in \{1, \ldots, m\}$, such that $\boldsymbol{w} = I(\boldsymbol{x} \oplus_q \boldsymbol{u}_v \oplus_q s\boldsymbol{u}_1)$, with weight($\boldsymbol{w}$) satisfying (1).

According to Theorem 1, such a pair of integers s and v can always be found.

Step 4: The balanced m-bit word $\boldsymbol{w}$ is transmitted.

B. Decoding

At the receiver side, the m-symbol word $\boldsymbol{y}$ is retrieved from the received by modulo q differentiation, i.e.

$$\boldsymbol{y} = I^{-1}(\boldsymbol{w}) = \boldsymbol{x} \oplus_q \boldsymbol{u}_v \oplus_q s\boldsymbol{u}_1.$$

We drop the first symbol, 's', of $\boldsymbol{y}$ obtaining the $(m-1)$-symbol $\boldsymbol{y}'$. Then $(m-1)$-symbol words $\boldsymbol{y}'$ and $\boldsymbol{x}'$ differ only at an unknown index position v. As $\boldsymbol{x}'$ satisfies $C_{q,r'}\boldsymbol{x}'^T = 0$, we have

$$C_{q,r'}\boldsymbol{y}'^T = \boldsymbol{c}_v,$$

where $\boldsymbol{c}_v$ is the v-th column of $C_{q,r'}$. Thus, we can uniquely retrieve the index v, and restore the original word by subtracting '1' from y'_v, that is, $\boldsymbol{x}' = \boldsymbol{y}' \ominus_q \boldsymbol{u}_v$.

By a straightforward reshuffling of the symbols, and removing the redundant symbols, we obtain the original k-symbol pay load $\boldsymbol{a}$.

IV. Adding Error Correction

From (2), any single channel error in, say w_j, will be transformed to a double adjacent error in x_j and x_{j+1}. This, together with the single "error" we introduced during balancing, means that we must be able to correct three errors. However, this would come at a price of much more redundancy. We can avoid this by extending our code used in the previous section and by introducing interleaving.

A. Encoding

We start with a similar code as in Section III-A, but extend it by adding a symbol that sums modulo q over all the previous symbols. We denote this extended code by C^*_{q,r^*}. Choose q to be an odd value and l such that $m - 1 = 2l$. The code then forms an (l, k) linear block code, with redundancy of $r^* = l - k$. In general, the check matrix C^*_{q,r^*} will be of size $r^* \times l$, with the i-th column $\boldsymbol{c}_i$ being the q-ary representation of the integer $(i + q^{r^*-1})$, $1 \le i \le l$. As an example, the check matrix in (3) becomes

$$C^*_{3,3} = \begin{bmatrix} 0 & 0 & 1 & 1 & 1 & 2 & 2 & 2 \\ 1 & 2 & 0 & 1 & 2 & 0 & 1 & 2 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

Lemma 1 *The (l, k) linear block code with check matrix $C^*_{q,r}$ can correct a single magnitude-one error, and it can detect a magnitude-one error and any other random error.* □

PROOF Let e_i and e_j represent the error magnitudes, where $e_i = 1$ is the magnitude-one error and $e_j \in \{1, 2, \ldots, q-1\}$ is the random error. The syndrome for the code is $\boldsymbol{s} = \{s_1, s_2, \ldots, s_r\} = e_i \boldsymbol{c}_i \oplus_q e_j \boldsymbol{c}_j$, and $s_r = 0$ for all e_j except when $e_j = q-1$. If $e_j = q-1$, then $s = 0$ only if $i = j$. Therefore, a single magnitude-one error and any random error can be detected. Further, $s_r = 1$ only if $e_j = 0$ (there is no random error), then with $\boldsymbol{s} = \boldsymbol{c}_i$ the magnitude-one error can be corrected. ■

We have two k-symbol payloads $\boldsymbol{a}$ and $\boldsymbol{a}'$ that are encoded into codewords $\boldsymbol{b} = \{b_1, b_2, \ldots, b_l\}$ and $\boldsymbol{b}' = \{b'_1, b'_2, \ldots, b'_l\}$ respectively, using the q-ary (l, k) code. Interleave these two codewords to a depth of two, to form

$$\boldsymbol{x}' = \{x'_1, x'_2, \ldots, x'_{m-1}\} = \{b_1, b'_1, b_2, b'_2, \ldots, b_l, b'_l\}.$$

The encoding now follows the same steps as in Construction 1 to add a redundant '0', to find the values s and v to balance the sequence and to encode it into $\boldsymbol{w}$. The final encoding step is to append symbols α and β to $\boldsymbol{w}$, where

$$\alpha = w_1 \oplus_q w_3 \oplus_q \cdots \oplus_q w_m \oplus_q \delta_{q,m}, \text{ and}$$
$$\beta = w_2 \oplus_q w_4 \oplus_q \cdots \oplus_q w_{m-1},$$

with $\delta_{q,m} \equiv 2(q-1)(2-m) \pmod q$. In essence, α and β are "parity check" symbols over the odd and even symbols respectively, and $\delta_{q,m}$ is added to ensure that α and β together are balanced.

Lemma 2 *If $\delta_{q,m} \equiv 2(q-1)(2-m) \pmod q$, then (α, β) will be balanced.* □

PROOF Let $\alpha' = w_1 + w_3 + \cdots + w_m$ and $\beta' = w_2 + w_4 + \cdots + w_{m-1}$, then since $\boldsymbol{w}$ is balanced, $\alpha' + \beta' = \frac{m(q-1)}{2}$. We also want (α, β) to be balanced, therefore it must hold that $\alpha + \beta = q - 1$.

Now, since $\alpha \equiv \alpha' + \delta_{q,m} \pmod q$ and $\beta \equiv \beta' \pmod q$, then

$$\alpha + \beta \equiv \alpha' + \delta_{q,m} + \beta' \pmod q$$
$$q - 1 \equiv \delta_{q,m} + \tfrac{m(q-1)}{2} \pmod q$$

which with some manipulation proves the lemma. ■

The sender then sends the balanced sequence $(w_1, w_2, \ldots, w_m, \alpha, \beta)$ to the receiver. The encoding process is summarized in Fig. 1.

Note that we described the algorithm for odd values of q. If q is even, then instead of using $2l = m - 1$ and $\boldsymbol{x} = \{0, x'_1, \ldots, x'_{m-1}\}$ during balancing, use $2l = m-2$ and $\boldsymbol{x} = \{0, 0, x'_1, \ldots, x'_{m-2}\}$, so that the length is even and balancing can be achieved.

B. Decoding

Let $\hat{\boldsymbol{w}}$ be the (possibly corrupted) received codeword of length $m + 2$. Let $\boldsymbol{s} = \{s_1, s_2, \ldots, s_r\}$ and $\boldsymbol{s}' = \{s'_1, s'_2, \ldots, s'_r\}$ be the respective syndromes that are calculated after multiplying the decoded and deinterleaved codewords with the parity check matrix $C_{q,r}$.

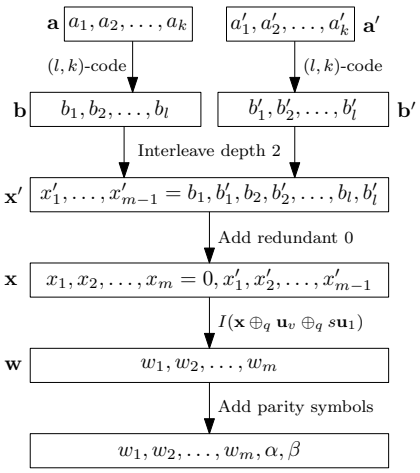

Fig. 1. Summary of encoding algorithm for q odd

The columns of $C_{q,r}$ were constructed in such a way that the decimal representation of the first $r-1$ q-ary entries will give us the position, say p, of an error provided that the error has a magnitude of one, i.e. if $s_r = 1$ or $s'_r = 1$. For simplicity, let $\boldsymbol{s} = \{p, \alpha\}$ and $\boldsymbol{s}' = \{p', \alpha'\}$, where p, p' are the decimal representation of the first $r-1$ q-ary entries and $\alpha = s_r$ and $\alpha' = s'_r$. Note that p, p' will only indicate the error position if a magnitude-one error occurred $(\alpha, \alpha' = 1)$.

If no channel errors occurred, none or only one of the deinterleaved codewords would contain a magnitude-one error that was introduced during balancing. Therefore if either (a) $\boldsymbol{s} = \{p, 1\}$, $\boldsymbol{s}' = \{0, 0\}$, (b) $\boldsymbol{s} = \{0, 0\}$, $\boldsymbol{s}' = \{p', 1\}$, or (c) $\boldsymbol{s} = \boldsymbol{s}' = \{0, 0\}$ is true (p, $p' > 0$), we can undo the $\boldsymbol{u}_v$ sequence that was added during encoding. If this is not the case, then possible channel errors occurred that first need to be corrected.

Decoding is done according to the following steps.

Step 1: Since the transmitted codeword was balanced, we can determine the error magnitude by looking at the imbalance in the received codeword. Let σ be the imbalance in the codeword and σ' be the imbalance in the "parity check" symbols, where:

$$\sigma = \sum_{i=1}^{m} \hat{w}_i - \tfrac{m(q-1)}{2} \text{ and } \sigma' = \hat{w}_{m+1} + \hat{w}_{m+2} - (q-1).$$

If $\sigma = 0$ (irrespective of σ''s value) then assume no error occurred in the codeword and go to Step 4. If $\sigma \neq 0$ and $\sigma' \neq 0$, or $|\sigma| > q-1$, then assume multiple errors occurred, declare a decoding failure and STOP.

Step 2: Check the "parity check" symbols. Let:

$$\gamma = \hat{w}_1 \oplus_q \hat{w}_3 \oplus_q \cdots \oplus_q \hat{w}_m \oplus_q \delta_{q,m} \oplus_q \hat{w}_{m+1},$$
$$\gamma' = \hat{w}_2 \oplus_q \hat{w}_4 \oplus_q \cdots \oplus_q \hat{w}_{m-1} \oplus_q \hat{w}_{m+2}.$$

If $\gamma \neq 0$ and $\gamma' = 0$, then an error occurred in an odd position and set $i = 1$. If $\gamma = 0$ and $\gamma' \neq 0$, then an error occurred in

an even position and set $i = 2$. If $\gamma = 0$ and $\gamma' = 0$, or $\gamma \neq 0$ and $\gamma' \neq 0$, then possible multiple errors occurred, declare a decoding failure and STOP.

Step 3: Subtract σ from the symbol in position i. (Note that in this case we are not doing modulo q subtraction.) If $\hat{w}_i - \sigma \in \{0, 1, \ldots, q-1\}$, proceed to the next step, otherwise let $i \leftarrow i + 2$ and repeat this step. If $i > m$, then declare a decoding failure and STOP.

Step 4: Perform modulo q differentiation with $\boldsymbol{y} = I^{-1}(\hat{\boldsymbol{w}} \ominus_q \sigma \boldsymbol{u}_i)$, drop the redundant first symbol to obtain $\boldsymbol{y}'$, deinterleave the codewords and determine the syndromes for both codewords. If the syndromes are not calculated as

$$\begin{aligned} &\boldsymbol{s} = \{p, 1\},\ \boldsymbol{s}' = \{0, 0\},\ p > 0, \text{ or} \\ &\boldsymbol{s} = \{0, 0\},\ \boldsymbol{s}' = \{p', 1\},\ p' > 0, \text{ or} \\ &\boldsymbol{s} = \{0, 0\},\ \boldsymbol{s}' = \{0, 0\}, \end{aligned} \tag{4}$$

then the channel error was not corrected, let $i \leftarrow i + 2$ and return to Step 3. If $\sigma = 0$ (coming from Step 1) and either statement is not true, then declare a decoding failure and STOP. If one of these statements is true, then subtract one (modulo q) from the corresponding codeword, according to p or p', and retrieve the information.

Theorem 2 *Using the encoding and decoding algorithm described above, a single channel error can be corrected.* □

PROOF Let a single channel error with magnitude e occur in position j in $\boldsymbol{w}$, resulting in $\hat{\boldsymbol{w}} = \boldsymbol{w} \oplus_q e\boldsymbol{u}_j$. If $j \in \{m+1, m+2\}$, then $\sigma = 0$ and one of the conditions in (4) are true, thus we can decode correctly. If $j \in \{1, 2, \ldots, m\}$, then $|\sigma| \in \{1, 2, \ldots, q-1\}$ with $\sigma \equiv e \pmod{q}$. Because of (2), $\boldsymbol{y} = I^{-1}(\hat{\boldsymbol{w}})$ will contain adjacent errors in y_j and y_{j+1}, along with a possible magnitude-one error in y_{v+1}. After deinterleaving, one codeword will have one error, and the other codeword will have one error (possibly two errors).

We try to correct the error by subtracting σ from the symbol in position i, $i \in \{1, 2, \ldots, m\}$, i.e. $\boldsymbol{y} = I^{-1}(\boldsymbol{w} \oplus_q e\boldsymbol{u}_j \ominus_q \sigma \boldsymbol{u}_i)$. If $i \neq j$, then we introduce two more errors in y_i and y_{i+1}, and after deinterleaving, one codeword will have two errors, and the other codeword will have two (possibly three) errors. According to Lemma 1, we can detect the two errors and none of the conditions in (4) will be true. If $i = j + 1$ or $i = j - 1$, then three adjacent errors occur in $\{y_{j-1}, y_j, y_{j+1}\}$ or $\{y_j, y_{j+1}, y_{j+2}\}$ respectively, and after deinterleaving it is possible to have one error in one codeword and three errors in the other codeword. Since detection of three errors is not guaranteed (the codeword can be valid and the syndrome will be zero), it is possible to have one of the conditions in (4) true. However, this situation is avoided by using γ and γ' to determine whether i should be even or odd.

Finally, if $i = j$ then $\boldsymbol{y} = I^{-1}(\boldsymbol{w})$, one of the conditions in (4) is true and we can decode correctly. ■

V. ANALYSIS

A. Redundancy

We first look at the redundancy of the balancing scheme in Section III. Let r denote the total number of redundant symbols of the balanced code, $r = r' + 1$. Since the maximum length of the check matrix $C_{q,r}$ equals $q^{r-1} - 1$, we conclude that the maximum length, $L_q(r)$, of the pay load is

$$L_q(r) = q^{r-1} - r, \quad q > 2.$$

For the binary case $q = 2$, since only the index v needs to be encoded and not the integer s, we find

$$L_2(r) = 2^r - r - 1,$$

which is the same value as presented by Knuth [7] using a construction with a prefix. Note that for $q = 2$ the check matrix $C_{2,r}$ defines a regular (binary) Hamming code with redundancy $r' = r$.

Swart and Weber's construction has a balanced prefix of length r, where each prefix uniquely represents the pair of integers s and v. Let $N_q(r)$ denote the number of distinct q-ary balanced prefixes of length r. Then for Swart and Weber's construction, we require that the length of the pay load, denoted by $L_q^{SW}(r)$, must satisfy

$$L_q^{SW}(r) \leq \left\lfloor \frac{N_q(r)}{q} \right\rfloor.$$

Using generating functions, we can straightforwardly compute the number of distinct q-ary prefixes, $N_q(r)$, of length r. Table I shows, for $q = 3$ and $q = 5$, $L_q(r)$ and $L_q^{SW}(r)$ as a function of r.

We conclude from the table that the redundancy of the new balanced q-ary code is significantly reduced with respect to Swart and Weber's method. Note that Capocelli *et al*. presented a code construction where the length of the pay load is less than

$$\frac{q^r - 1}{q - 1}. \tag{5}$$

For large alphabet q we conclude that the redundancy of the new method is approximately a factor of $q/(q-1)$ higher than that of the prior art construction by Capocelli *et al*. The construction of [4] introduces compression to the construction from [3], resulting in even lower redundancies.

TABLE I
$L_q^{SW}(r)$ AND $L_q(r)$ AS A FUNCTION OF r

q	r	$L_q^{SW}(r)$	$L_q(r)$	$L_q^{\text{ECC}}(r)$
3	4	6	23	–
3	5	17	76	–
3	6	47	237	–
3	7	131	722	–
3	8	369	2179	–
3	9	1046	6552	10
3	10	2984	19672	9
5	4	17	120	–
5	5	76	620	–
5	6	350	3119	–
5	7	1627	15618	4
5	8	7633	78117	3
5	9	36065	390616	42
5	10	171389	1953115	41

For the redundancy of the balancing scheme with error correction in Section IV, the total redundancy is $r = 2r^* + 3$. The maximum length of the check matrix for one (l, k) code is $l = q^{r^*-1} - 1$. Let $L_q^{\text{ECC}}(r)$ denote the maximum length of the pay load, then taking into account that we use two (l, k) codes, add one redundant symbol for balancing and two more redundant "parity" symbols, it can be shown that for q odd

$$L_q^{\text{ECC}}(r) = 2q^{\frac{r-5}{2}} - r + 1.$$

For q even, one more redundant symbol is added. Values for $L_q^{\text{ECC}}(r)$ are also shown in Table I.

Again, for the binary case $q = 2$, since only the index v needs to be encoded and not the integer s, we find

$$L_2^{\text{ECC}}(r) = 2^{\frac{r-2}{2}} - r - 1.$$

B. Performance

We look at the performance of two codes:

- $R = 10/19$ code with $q = 3$: an (8,5) linear block code is used, giving rate (16,10) after interleaving, and after balancing and adding the extra two "parity symbols" this becomes (19,10).
- $R = 4/11$ code with $q = 5$: a (4,2) linear block code is used, which after interleaving, balancing and adding "parity symbols" is (11,4).

Fig. 2 shows the symbol error rate after decoding for these two codes. If a decoding failure occurred, the information was discarded, and was not taken into account in the symbol error rate calculations. Typically in an ARQ system if decoding failure occurred, the information would be requested again.

Fig. 3 shows the probability that a decoding failure occurred. That is, for the imbalance detected, during decoding no position could be found such that the syndromes indicated either a magnitude-one error or no error, or the algorithm determined early on that possible multiple errors occurred.

VI. Conclusions

We have presented a simple method for balancing q-ary codewords, where look-up tables for coding and decoding the prefix can be avoided. The receiver only needs to know the check matrix of the linear code being used. We have compared the redundancy of the new construction with that of prior art constructions. The redundancy of the new construction is much less than that of Swart and Weber's construction, and slightly larger than that of the Capocelli *et al*. construction. The method was also extended to include error correction capabilities to correct single channel errors, by simply adding four redundant symbols and introducing interleaving.

Since all operations are linear, encoding and decoding can be simplified by setting up a generator matrix and parity check matrix that does interleaving/deinterleaving and integration/differentiation all in one step. This would alleviate the problem having to repeatedly decode for every attempt at correcting the error. We leave this as future work.

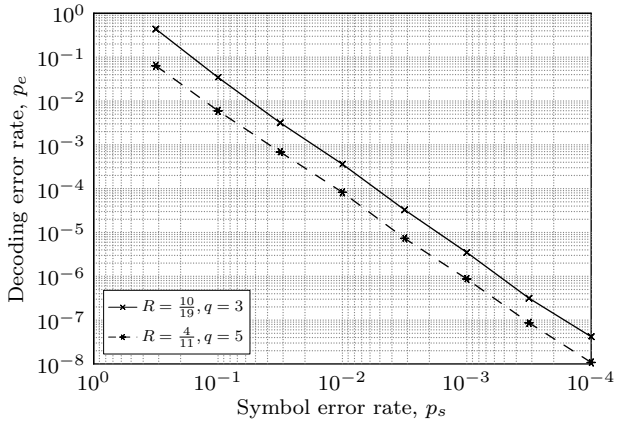

Fig. 2. Decoding error rate

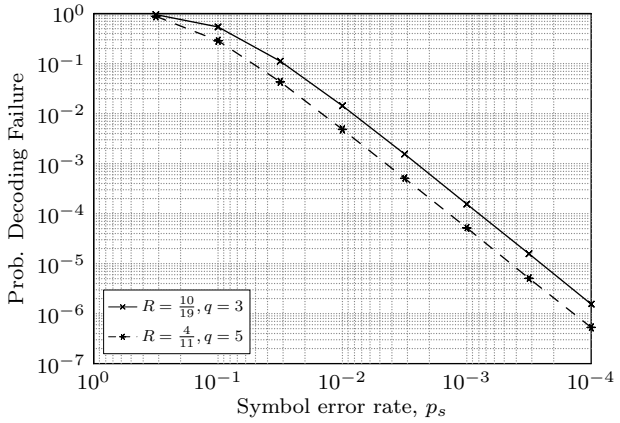

Fig. 3. Probability of decoding failure

References

[1] K. A. S. Immink, "Coding Methods for High-Density Optical Recording," *Philips J. Res.*, vol. 41, pp. 410–430, 1986.

[2] H. Zhou, A. Jiang, and J. Bruck, "Balanced Modulation for Nonvolatile Memories," submitted to *IEEE Trans. Inform. Theory*.

[3] R. M. Capocelli. L. Gargano, and U. Vaccaro, "Efficient q-ary immutable codes," *Discrete Appl. Math.*, vol. 33, pp. 25–41, 1991.

[4] L. G. Tallini and U. Vaccaro, "Efficient m-ary balanced codes", *Discrete Applied Mathematics*, vol. 92, pp. 17–56, 1999.

[5] T. G. Swart and J. H. Weber, "Efficient Balancing of q-ary Sequences with Parallel Decoding," in *Proc. IEEE Intl. Symp. Inform. Theory*, Seoul, South Korea, Jun. 29–Jul. 3, 2009, pp. 1564–1568.

[6] D. Pelusi, L. G. Tallini and B. Bose, "On m-ary balanced codes with parallel decoding," in *Proc. IEEE Intl. Symp. Inform. Theory*, Austin, Texas, Jun. 13–18, 2010, pp. 1305–1309.

[7] D. E. Knuth, "Efficient Balanced Codes," *IEEE Trans. Inform. Theory*, vol. 32, no. 1, pp. 51–53, Jan. 1986.

[8] S. Al-Bassam and B. Bose, "On Balanced Codes," *IEEE Trans. Inform. Theory*, vol. 36, no. 2, pp. 406–408, Mar. 1990.

[9] L. G. Tallini, R. M. Capocelli, and B. Bose, "Design of Some New Balanced Codes," *IEEE Trans. Inform. Theory*, vol. 42, pp. 790–802, May 1996.

[10] J. H. Weber and K. A. S. Immink, "Knuth's Balanced Codes Revisited," *IEEE Trans. Inform. Theory*, vol. 56, no. 4, pp. 1673–1679, Apr. 2010.

[11] S. Al-Bassam and B. Bose, "Design of Efficient Error-Correcting Balanced Codes," *IEEE Trans. Computers*, vol. 42, no. 10, pp. 1261–1266, Oct. 1993.

[12] J. H. Weber, K. A. S. Immink, and H. C. Ferreira, "Error-Correcting Balanced Knuth Codes," *IEEE Trans. Inform. Theory*, vol. 58, pp. 82–89, Jan. 2012.

[13] R. W. Hamming, *Coding and Information Theory*, Prentice-Hall, Englewood Cliffs, 1986.