Decoding of Matrix-Product Codes*

Fernando Hernando†
San Diego State University
fhernando@mail.sdsu.edu

Diego Ruano‡
Aalborg University
diego@math.aau.dk

Abstract

We propose a decoding algorithm for the $(u \mid u+v)$-construction that decodes up to half of the minimum distance of the linear code. We extend this algorithm for a class of matrix-product codes in two different ways. In some cases, one can decode beyond the error correction capability of the code.

1 Introduction

Matrix-product codes, $[C_1 \cdots C_s] \cdot A$, were introduced by Blackmore and Norton in [1]. They may also be seen as a generalization of the $(u \mid u+v)$-construction. Advantages of this method are, first, that long codes can be created from old ones and, second, that the parameters or the codes are known under some conditions [1, 2, 5]. Other generalizations include [3] and [6].

In [2], a decoding algorithm for matrix-product codes with $C_1 \supset \cdots \supset C_s$ was presented. In this work, we present an alternative to that algorithm, where we do not need to assume that the codes $C_1, \ldots, C_s$ are nested. In section 3, we present the new algorithm for $s = l = 2$, $(u \mid u+v)$-construction, the main assumption that we should consider is $d_2 \geq 2d_1$, where d_i is the minimum distance of C_i, $d_i = d(C_i)$. The new algorithm decodes up to half of the minimum distance. Furthermore, if d_1 is odd and $d_2 > 2d_1$, we are able to decode beyond this bound, obtaining just a codeword with a high probability.

From the algorithm in section 3 we derive two extensions for matrix-product codes defined using a matrix A, of arbitrary size $s \times l$, which verifies a certain property called non-singular by columns. The main difference between these two algorithms resides in the following fact: the algorithm is section 4 requires stronger assumptions ($d_i \geq l d_1$, for all i) than the one in section 5 ($d_i \geq i d_1$, for all i), but it is computationally less intense. Both algorithms decode up to half of the designed minimum distance of the code [5], that is known to be sharp in several cases [1, 2] (for intance if $C_1 \supset \cdots \supset C_s$). If d_1 odd and l even, we

*Published in Journal of Algebra and its Applications, Volume 12, Issue 4, Article ID 1250185, 15 pages (2013). Keywords: Linear Codes, Matrix-Product Codes, Decoding Algorithm, Minimum Distance. Subclass: 94B05; 94B35.

†Supported in part by Spanish MEC MTM2007-64704.

‡Supported in part by the Danish National Research Foundation and the National Science Foundation of China (Grant No.11061130539) for the Danish-Chinese Center for Applications of Algebraic Geometry in Coding Theory and Cryptography, and Spanish MEC MTM2007-64704.

can decode beyond this bound obtaining a list of codewords that will contain just one codeword with a high probability. The algorithm in section 4 does not become computationally intense for large s, l.

2 Matrix-Product Codes

A matrix-product code is a construction of a code from old ones.

Definition 2.1. Let $C_1, \dots, C_s \subset \mathbb{F}_q^m$ be linear codes of length m and a matrix $A = (a_{i,j}) \in \mathcal{M}(\mathbb{F}_q, s \times l)$, with $s \leq l$. The matrix-product code $C = [C_1 \cdots C_s] \cdot A$ is the set of all matrix-products $[c_1 \cdots c_s] \cdot A$ where $c_i \in C_i$ is an $m \times 1$ column vector $c_i = (c_{1,i}, \dots, c_{m,i})^T$ for $i = 1, \dots, s$. Therefore, a typical codeword p is

$$p = \begin{pmatrix} c_{1,1}a_{1,1} + \cdots + c_{1,s}a_{s,1} & \cdots & c_{1,1}a_{1,l} + \cdots + c_{1,s}a_{s,l} \\ \vdots & \ddots & \vdots \\ c_{m,1}a_{1,1} + \cdots + c_{m,s}a_{s,1} & \cdots & c_{m,1}a_{1,l} + \cdots + c_{m,s}a_{s,l} \end{pmatrix}. \qquad (1)$$

The i-th column of any codeword is an element of the form $\sum_{j=1}^{s} a_{j,i}c_j \in \mathbb{F}_q^m$, therefore reading the entries of the $m \times l$-matrix above in column-major order, the codewords can be viewed as vectors of length ml,

$$p = \left(\sum_{j=1}^{s} a_{j,1}c_j, \dots, \sum_{j=1}^{s} a_{j,l}c_j \right) \in \mathbb{F}_q^{ml}. \qquad (2)$$

If C_i is an $[m, k_i, d_i]$ code then one has that $[C_1 \cdots C_s] \cdot A$ is a linear code over $\mathbb{F}_q$ with length lm and dimension $k = k_1 + \cdots + k_s$ if the matrix A has full rank and $k < k_1 + \cdots + k_s$ otherwise.

Let us denote by $R_i = (a_{i,1}, \dots, a_{i,l})$ the element of $\mathbb{F}_q^l$ consisting of the i-th row of A, for $i = 1, \dots, s$. We denote by D_i the minimum distance of the code C_{R_i} generated by $\langle R_1, \dots, R_i \rangle$ in $\mathbb{F}_q^l$. In [5] the following lower bound for the minimum distance of the matrix-product code C is obtained,

$$d(C) \geq d_C = \min\{d_1 D_1, d_2 D_2, \dots, d_s D_s\}, \qquad (3)$$

where d_i is the minimum distance of C_i. If $C_1, \dots, C_s$ are nested codes, $C_1 \supset \cdots \supset C_s$, the previous bound is sharp [2].

In [1], the following condition for the matrix A is introduced.

Definition 2.2. [1] Let A be a $s \times l$ matrix and A_t be the matrix consisting of the first t rows of A. For $1 \leq j_1 < \cdots < j_t \leq l$, we denote by $A(j_1, \dots, j_t)$ the $t \times t$ matrix consisting of the columns $j_1, \dots, j_t$ of A_t.

A matrix A is non-singular by columns if $A(j_1, \dots, j_t)$ is non-singular for each $1 \leq t \leq s$ and $1 \leq j_1 < \cdots < j_t \leq l$. In particular, a non-singular by columns matrix A has full rank.

Moreover, if A is non-singular by columns, the bound d_C in (3) is

$$d(C) \geq d_C = \min\{l d_1, (l-1)d_2, \dots, (l-s+1)d_s\}$$

and it is known to be sharp in several cases: it was shown in [1] that if A is non-singular by columns and triangular, (i.e. it is a column permutation of an upper triangular matrix), then the bound (3) for the minimum distance is sharp. Furthermore, if A is non-singular by columns and the codes $C_1 \dots C_s$ are nested, then this bound (3) is also sharp.

A decoding algorithm for the matrix-product code $C = [C_1 \cdots C_s] \cdot A \subset \mathbb{F}_q^{ml}$, with A non-singular by columns and $C_1 \supset \cdots \supset C_s$ was presented [2], assuming that we have a decoding algorithm for C_i, for $i = 1, \dots, s$. We present in next section another decoding algorithm for a matrix-product code with $s = l = 2$.

3 A decoding algorithm for the $(u \mid u+v)$-construction

We consider a decoding algorithm for the $(u \mid u+v)$-construction, that is, a matrix-product code with $s = l = 2$, $C = [C_1 C_2] \cdot A$ with $d_2 \geq 2d_1$ and $d_1 \geq 3$, where $d_i = d(C_i)$ is the minimum distance of C_i. Let

$$A = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}.$$

Note that C is the $(u|u+v)$-construction and that an equivalent code will be obtained with any matrix of rank 2.

Let t_1 be the error-correction capability of C_1, $t_1 = \lfloor \frac{d_1-1}{2} \rfloor \geq 1$, that is $d_1 = 2t_1 + 1$ if d_1 is odd and $d_1 = 2t_1 + 2$ if d_1 is even. The minimum distance of C is $d(C) = \min\{2d_1, d_2\} = 2d_1$ [4]. Thus the error correction capability of the code C is

$$t = \left\lfloor \frac{2d_1 - 1}{2} \right\rfloor = \begin{cases} 2t_1 & \text{if } d_1 \text{ is odd} \\ 2t_1 + 1 & \text{if } d_1 \text{ is even} \end{cases}$$

We provide a decoding algorithm for the matrix-product code C, assuming that we have a decoding algorithm DC_i for C_i which decodes up to t_i errors, for $i = 1, 2$. Let $r = p + e$ be a received word where $p \in C$ and the error vector e verifies $wt(e) \leq t$. Note that a typical word $p \in C$ is $[c_1 c_2] \cdot A = (c_1, c_1 + c_2)$, namely a received word r is $r = (r_1, r_2) = (c_1 + e_1, c_1 + c_2 + e_2)$.

Consider $r_2 - r_1 = c_1 + c_2 + e_2 - c_1 - e_1 = c_2 + (e_2 - e_1)$. We may decode $r_2 - r_1$ using the decoding algorithm DC_2 to obtain c_2, since $c_2 \in C_2$ and $wt(e_2 - e_1) < d_2/2$ because

$$wt(e_2 - e_1) \leq wt(e_1) + wt(e_2) = wt(e) \leq t < d_1 \leq \frac{d_2}{2}.$$

Since we know c_2 we may consider $r_2^{2)} = r_2 - c_2 = c_1 + e_2$ and let $r_1^{2)} = r_1 = c_1 + e_1$. We claim that there exists $i_1 \in \{1, 2\}$ such that $wt(e_{i_1}) \leq t_1$: assume that such an i does not exist, then

$$wt(e) = wt(e_1) + wt(e_2) \geq 2t_1 + 2,$$

a contradiction. Let $wt(e_{i_1}) \leq t_1$, then we can obtain c_1 by decoding $r_{i_1}^{2)}$ with the decoding algorithm DC_1. A priori, we do not know which index i_1 is, however we will be able to detect it by checking that we have not corrected more than $\lfloor (d(C) - 1)/2 \rfloor$ errors in total. That is, for $p = (c_1, c_1 + c_2)$ and $p' = (c'_1, c'_1 + c_2)$, we check whether $d(r, p) \leq t$ and $d(r, p') \leq t$.

Remark 3.1. Let us compare this decoding algorithm to the algorithm in [2]. In the algorithm in [2], we assume that $C_1 \supset C_2$ and for this algorithm we assume that $d_2 \geq 2d_1$. Comparing the complexity of the algorithms: In the algorithm in [2], we should run DC_1 and DC_2 twice, in the worst case situation. For this algorithm, we run DC_1 twice and DC_2 once. Both algorithms decode up to the error-correction capability of the code.

For d_1 odd and $d_2 > 2d_1$, the previous algorithm can also be used for correcting $t + 1 = 2t_1 + 1$ errors, that is, one more error than the error-correction capability of C. The algorithm outputs a list with one or two codewords, containing the sent word. Let us assume now that $wt(e) \leq t + 1$, again we may obtain c_2 by decoding $r_2 - r_1$ since $wt(e_2 - e_1) \leq t_2$ because

$$wt(e_2 - e_1) \leq wt(e_1) + wt(e_2) = wt(e) \leq t + 1 = 2t_1 + 1 = d_1 < \frac{d_2}{2}.$$

Again there will be an index $i_1 \in \{1, 2\}$ such that $wt(e_{i_1}) \leq t_1$ because otherwise $wt(e) \geq 2t_1 + 2 > 2t_1 + 1$. Hence, we also decode $r_{i_2}^{2)}$ using DC_1. Let $p = (c_1, c_1 + c_2)$ and $p' = (c'_1, c'_1 + c_2)$ as above, $d(p, r) \leq t + 1$ and $d(p', r) \geq t + 1$.

- $d(r, p) = wt((c_1 + e_1, c_1 + c_2 + e_2) - (c_1, c_1 + c_2)) = wt(e) \leq t + 1$.
- $d(r, p') = wt((c_1 + e_1, c_1 + c_2 + e_2) - (c'_1, c'_1 + c_2)) = wt(c_1 - c'_1 + e_1, c_1 - c'_1 + e_2) \geq 2d_1 - wt(e) \geq 2(2t_1 + 1) - (2t_1 + 1) = 2t_1 + 1 = t + 1$.

If we have that $d(p, r), d(p', r) \leq t + 1$ we output both codewords, in other case we output only p. Note that the probability of having two codewords in the output list is negligible, since $d(r, p') = t + 1$ if and only if $d(c_1, c'_1) = d_1$ and for every $e_{j,i} \neq 0$, with $j = 1, \ldots, m$, $i = 1, 2$, one has that $e_{j,i} = -(c_{j,i} - c'_{j,i})$.

We will consider in this article two different extensions of this algorithm for any s and l, with $s \leq l$. Namely, for the particular case where $s = l = 2$, both extensions are equal.

4 A decoding algorithm for Matrix-Product codes, first extension

In this section we propose an extension of the algorithm in the previous section for matrix-product codes with any $s \leq l$, the algorithm in this section is less computationally intense than the algorithm in [2] for large s, l. In the following section we will propose another extension. Let $C = [C_1 \cdots C_s] \cdot A$ be a matrix-product code, with $d_i \geq ld_1$, for $i = 2, \ldots, s$, and $d_1 \geq 3$, where $d_i = d(C_i)$ is the minimum distance of C_i. We also require that A is non-singular by columns.

The error-correction capability of C_i is $t_i = \lfloor \frac{d_i - 1}{2} \rfloor \geq 1$. From (3), one has that the designed minimum distance of C is $d(C) \geq d_C = \min\{ld_1, (l - 1)d_2, \ldots, (l - s + 1)d_l\} = ld_1$. Hence, the designed error correction capability of the code C is

$$t = \left\lfloor \frac{ld_1 - 1}{2} \right\rfloor = \begin{cases} lt_1 + \lfloor \frac{l-1}{2} \rfloor & \text{if } d_1 \text{ is odd} \\ lt_1 + l - 1 & \text{if } d_1 \text{ is even,} \end{cases}$$

since $d_1 = 2t_1 + 1$ if d_1 is odd and $d_1 = 2t_1 + 2$ if d_1 is even.

We provide a decoding algorithm for the matrix-product code C that decodes up to half of its designed minimum distance, assuming that we have a decoding algorithm DC_i for C_i which decodes up to t_i errors, for $i = 1, \ldots, s$. A codeword in C is an $m \times l$ matrix which has the form $p = [c_1, \ldots, c_s] \cdot A = (\sum_{j=1}^{s} a_{j,1} c_j, \ldots, \sum_{j=1}^{s} a_{j,l} c_j)$, where $c_j \in C_j$, for all j. We denote by $p_i = \sum_{j=1}^{s} a_{j,i} c_j \in \mathbb{F}_q^m$ the i-th block of p, for $i = 1, \ldots, l$. Suppose that p is sent and that we receive $r = p + e$, where $e = (e_1, e_2, \ldots, e_l)$ is an error vector, an $m \times l$ matrix, with weight $wt(e) \leq t$.

Let B be a matrix in $\mathcal{M}(\mathbb{F}_q, l \times s)$, such that AB is the $s \times s$-identity matrix. Such a matrix exists because A has rank s and it can be obtained by solving a linear system, but it is not unique if $s < l$. Let $w_i = (0, \ldots, 0, 1, 0, \ldots, 0)^T \in \mathbb{F}_q^s$ be the vector that has all coordinates equal to zero, excepting the i-th coordinate that is equal to 1. For $i \in \{2, \ldots, s\}$, consider $v_i = (v_{1,i}, \ldots, v_{l,i})^T \in \mathbb{F}_q^l$ equal to $v_i = Bw_i$. One has that $pv_i = \sum_{j=1}^{l} v_{j,i} p_j = c_i$, since $pv_i = pBw_i = [c_1, \ldots, c_s] w_i = c_i$. Therefore

$$rv_i = \sum_{j=1}^{l} v_{j,i} r_j = \sum_{j=1}^{l} v_{j,i} p_j + \sum_{j=1}^{l} v_{j,i} e_j = c_i + \sum_{j=1}^{l} v_{j,i} e_j.$$

For $i = 2, \ldots, s$, we can decode rv_i with the decoding algorithm DC_i to obtain c_i, since $c_i \in C_i$ and

$$wt\left(\sum_{j=1}^{l} v_{j,i} e_j\right) \leq \sum_{j=1}^{l} wt(e_j) = wt(e) \leq t = \left\lfloor \frac{ld_1 - 1}{2} \right\rfloor \leq \left\lfloor \frac{d_i - 1}{2} \right\rfloor = t_i$$

As we have already computed $c_2, \ldots, c_s$ we may consider now $r_i' = r_i - \sum_{j=2}^{s} a_{i,j} c_j = a_{1,i} c_1 + e_i$, for $i = 1, \ldots, l$. We claim that there exists $i \in \{1, \ldots, l\}$ such that $w(e_i) \leq t_1$ because if $wt(e_i) > t_1$ for all i then $wt(e) \geq lt_1 + l > t$. Therefore, we correct $r_1'/a_{1,1}, \ldots, r_l'/a_{1,l}$, with DC_1 and at least one of them gives c_1 as output. Note that $a_{1,i} \neq 0$, for $i = 1, \ldots, l$ since A is non-singular by columns. We have l candidates for c_1, $c_1^{i)} = DC_1(r_j'/a_{1,i})$, for $i = 1, \ldots, l$, we can detect which candidate is equal to c_1 by checking that we have not corrected more than $\lfloor (d-1)/2 \rfloor$ errors in total, that is, we check whether $d(r - [c_1^{i)}, c_2 \ldots, c_s] \cdot A) \leq \lfloor (d-1)/2 \rfloor$, for $i = 1, \ldots, l$.

The algorithm is outlined as a whole in procedural form in Algorithm 1.

Remark 4.1. Let us compare this decoding algorithm to the algorithm in [2]. In both algorithms we assume that A is non-singular by columns. For the algorithm in this section, we assume that $ld_1 < d_i$ for all $i = 2, \ldots, s$. In the algorithm in [2], we assume that $C_1 \supset \cdots \supset C_s$, therefore the bound in (3) is sharp. Hence, if $C_1, \ldots, C_s$ are nested, both algorithms decode up to half of the minimum distance of the matrix-product code. In the algorithm in [2], we run DC_i $\binom{l}{s}$ times, for $i = 1, \ldots, s$, in the worst-case. However, in the algorithm presented in this section, we only run DC_i once, for $i = 2, \ldots, s$ and we run DC_1 l times, thus it will have polynomial complexity if the algorithms DC_i have polynomial complexity, for $i = 1, \ldots, s$ (since B is computed by solving a linear system). Hence the algorithm in [2] becomes computationally intense for large values of s, l but this algorithm does not.

Algorithm 1 A DECODING ALGORITHM FOR $C = [C_1 \cdots C_s] \cdot A$, FIRST EXTENSION

Input: Received word $r = p + e$ with $c \in C$ and $wt(e) < d(C)/2$, where $d_i = d(C_i)$ with $ld_1 < d_i$ and A full rank. Decoder DC_i for code C_i, $i = 1, \ldots, s$.

Output: p.

1: $r' = r$;
2: Find B, a right inverse of A ($AB = Id$);
3: **for** $i = 2, \ldots, s$ **do**
4:     $v = Be_i$;
5:     $c_i = DC_i(rv)$;
6: **end for**
7: $r = (r_1 - \sum_{j=2}^{s} a_{j,1}c_j, \ldots, r_l - \sum_{j=2}^{s} a_{j,l}c_j)$;
8: **for** $i = 1, \ldots, l$ **do**
9:     $c_1 = DC_1(r_i/a_{1,i})$;
10:     **if** $c_1 =$ *"failure"* **then**
11:         *Break* the loop and consider next i in line 8;
12:     **end if**
13:     $p = [c_1 \cdots c_s] \cdot A$;
14:     **if** $p \in C$ and $wt(r' - p) \leq \lfloor (d(C) - 1)/2 \rfloor$ **then**
15:         **return** p;
16:     **end if**
17: **end for**

We can also consider this algorithm for correcting beyond the designed error-correction capability of C, if l is even, d_1 is odd and $d_i > ld_1$, for $i = 2, \ldots, s$. Namely, the designed error correction capability of C is $lt_1 + \lfloor \frac{l-1}{2} \rfloor = lt_1 + (l-2)/2$ and we consider now an error vector with $wt(e) < lt_1 + l/2$, that is, we are correcting 1 error beyond the designed error correcting capability of C. We should just modify line 14 in Algorithm 1 to accept codewords p with $wt(r' - p) \leq lt_1 + l/2$ and create a list with all the output codewords.

Again, we can decode rv_i with the decoding algorithm DC_i to obtain c_i, since

$$wt\left(\sum_{j=1}^{l} v_{j,i}e_j\right) \leq wt(e) \leq lt_1 + \frac{l}{2} = \frac{l}{2}(2t_1 + 1) \leq \frac{l}{2}d_1 < \frac{d_i}{2}.$$

Moreover, there exists $i \in \{1, \ldots, l\}$ such that $w(e_i) \leq t_1$ as well because if $wt(e_i) > t_1$ for all i then $wt(e) \geq lt_1 + l > lt_1 + l/2$. As before, we have l candidates for c_1 and at least one of them is c_1, however now we cannot uniquely determine it: let $p = [c_1, \ldots, c_s] \cdot A$ and $p' = [c_1', c_2 \ldots, c_s] \cdot A$ with $c_1 \neq c_1'$, one has that $d(p, r) \leq lt_1 + l/2$ and $d(p', r) \geq lt_1 + l/2$.

- $d(r, p) = wt(e) \leq lt_1 + l/2$.
- $d(r, p') = wt(a_{1,1}(c_1 - c_1') + e_1, \ldots, a_{1,l}(c_1 - c_1') + e_l) \geq ld_1 - wt(e) \geq l(2t_1 + 1) - (lt_1 l/2) = lt_1 + l/2$.

The algorithm outputs p and all the other codewords -obtained from the other $l - 1$ candidates- that are at distance at most $lt_1 + l - 1$ from r. As with

$s = l = 2$, the probability of having more than one codeword in the output list is negligible, since $d(r, p') = lt_1 + l/2$ if and only if the bound in (3) is sharp, $d(c_1, c'_1) = d_1$ and for every $j = 1, \dots, m$, $i = 1, \dots, l$, with $e_{j,i} \neq 0$, one has that $e_{j,i} = -a_{1,i}(c_{j,1} - c'_{j,1})$.

Example 4.2. Consider the following linear codes over $\mathbb{F}_3$,

- C_1 the $[26, 20, 4]$ cyclic code generated by $f_1 = x^6 + x^5 + 2x^4 + 2x^3 + x^2 + x + 2$.
- C_2 the $[26, 7, 14]$ cyclic code generated by $f_2 = x^{19} + x^{18} + x^{17} + x^{15} + 2x^{14} + x^{13} + 2x^{12} + x^{11} + 2x^8 + 2x^7 + x^6 + x^4 + x^3 + 2$.
- C_3 the $[26, 3, 18]$ cyclic code generated by $f_3 = x^{23} + 2x^{22} + x^{21} + 2x^{19} + 2x^{18} + x^{17} + x^{16} + x^{15} + x^{13} + x^{10} + 2x^9 + x^8 + 2x^6 + 2x^5 + x^4 + x^3 + x^2 + 1$.

Let $C = [C_1 C_2 C_3] \cdot A$, where A is the non-singular by columns matrix

$$A = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{pmatrix}.$$

We use decoder DC_i for C_i, which decodes up to half the minimum distance, i.e., DC_1, DC_2, DC_3 decode up to $t_1 = 1$, $t_2 = 6$ and $t_3 = 8$ errors, respectively. We have that $d_C = 3d_1 = 12$ and since A is triangular we have that the minimum distance of C is $d(C) = d_C = 12$ and we may correct up to $t = 5$ errors in a codeword of C. Note that $12 = 3d_1 \leq d_2, d_3$.

We consider now polynomial notation for codewords of C_i, for all i. Hence the codewords of length 23 in C_i are polynomials in $\mathbb{F}_q[x]/(x^{23} - 1)$ and the words in C are elements in $(\mathbb{F}_q[x]/(x^{23} - 1))^3$. Note that C is a quasi-cyclic code. Let $r = p + e$ be the received word, with codeword $p = (0, 0, 0)$ and the error vector of weight $t = 5$

$$e = (e_1, e_2, e_3) = (1 + x, 2x^2 + x^7, 2x^{11}).$$

The matrix

$$B = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}$$

verifies that $AB = I_3$. Then v_2 and v_3 are the second and third columns of B respectively. Therefore $rv_2 = c_2 + 2e_1 + e_2 = 2 + 2x + 2x^2 + x^7$ and $rv_3 = c_3 + e_1 + e_2 + e_3 = 1 + x + 2x^2 + x^7 + 2x^{11}$.

- We decode rv_3 with DC_3 and we obtain $c_3 = 0$ because $wt(e_1 + e_2 + e_3) \leq wt(e) = 5 < t_2 = 6$.
- We decode rv_2 with DC_2 and we obtain $c_2 = 0$ because $wt(2e_1 + e_2) \leq wt(e) = 5 < t_2 = 6$.
- Subtracting c_2 and c_3 from $r = (c_1 + e_1, c_1 + c_2 + e_2, c_1 + 2c_2 + c_3 + e_3)$ we get $r' = (c_1 + e_1, c_1 + e_2, c_1 + e_3)$. Moreover we know that either $r'_1 = c_1 + e_1$ or $r'_2 = c_1 + e_2$ or $r'_3 = c_1 + e_3$ can be decoded with DC_1, so we should decode these three words. The weight of r'_3 is 1, since the

minimum distance of C_1 is 4 there is only one codeword at distance 1 of the zero-codeword, and thus $c_3 = 0$. In the other two cases (r'_1 and r'_2) the weight is 2, thus the output of the decoding algorithm DC_1 in both cases is either zero if it is the only codeword at distance 2 (from r'_1 and r'_2 respectively) or "failure" if there is more than one codeword at distance 2.

5 A decoding algorithm for Matrix-Product codes, second extension

In this section, we consider another extension of the algorithm in section 3 for arbitrary $s \leq l$. This algorithm imposes softer conditions (than the one in previous section) on the minimum distance of the constituent codes, however it can become computationally intense for large s or l. Let $C = [C_1 \cdots C_s] \cdot A$ be a matrix-product code, we shall assume that A is non-singular by columns and that $d_i \geq id_1$, for $i = 2, \ldots, s$, where $d_i = d(C_i)$ is the minimum distance of C_i.

The error-correction capability of C_i is $t_i = \lfloor \frac{d_i - 1}{2} \rfloor$. From (3), one has that the designed minimum distance of C is given by $d(C) \geq d_C = \min\{ld_1, (l-1)d_2, \ldots, (l-s+1)d_s\}$ and it is computed in the following lemma.

Lemma 5.1. *Let $C = [C_1 \cdots C_s] \cdot A$ be a matrix-product code, with A non-singular by columns and $d_i \geq id_1$, for $i = 2, \ldots, s$. The designed minimum distance of C is $d_C = ld_1$.*

Proof. We claim that $ld_1 \leq (l - i + 1)d_i$, for $i = 2, \ldots, s$. Since $id_1 \leq d_i$, we have that $i(l-i+1)d_1 < (l-i+1)d_i$. Hence, $ld_1 \leq i(l-i+1)d_1 < (l-i+1)d_i$ if and only if $l \leq i(l - i + 1)$. One has that

$$l \leq i(l-i+1) \Longleftrightarrow l(i-1) \geq i^2 - i \Longleftrightarrow l \geq \frac{i^2 - i}{i-1} = i$$

Thus, the claim holds since $i \leq s \leq l$.

Finally, we have that

$$d_C = \min\{ld_1, (l-1)d_2, \ldots, (l-s+1)d_s\} = ld_1.$$

□

Hence, the designed error correction capability of the code C is

$$t = \left\lfloor \frac{ld_1 - 1}{2} \right\rfloor = \begin{cases} lt_1 + \lfloor \frac{l-1}{2} \rfloor & \text{if } d_1 \text{ is odd} \\ lt_1 + l - 1 & \text{if } d_1 \text{ is even,} \end{cases}$$

because $d_1 = 2t_1 + 1$ if d_1 is odd and $d_1 = 2t_1 + 2$ if d_1 is even.

As in previous sections, we provide a decoding algorithm for the matrix-product code C, that decodes up to half of its designed minimum distance, assuming that we have a decoding algorithm DC_i for C_i which decodes up to t_i errors, for $i = 1, \ldots, s$. A codeword in C is an $m \times l$ matrix which has the form $p = [c_1 \cdots c_s] \cdot A = (\sum_{j=1}^{s} a_{j,1} c_j, \ldots, \sum_{j=1}^{s} a_{j,l} c_j)$, where $c_j \in C_j$, for all j. Suppose that p is sent and that we receive $r = p + e$, where $e = (e_1, e_2, \ldots, e_l)$ is an error vector, an $m \times l$ matrix, with weight $wt(e) \leq t$.

In order to decode r, we compute c_i, for $i = s, s-1, \ldots, 1$, inductively. Then, after s iterations we compute p by $p = [c_1 \cdots c_s] \cdot A$. We will now show how c_i is obtained, assuming that we have already obtained $c_s, c_{s-1}, \ldots, c_{i+1}$ (for $i = s$, we do not assume anything). Let $r^{i)} = (\sum_{j=1}^{i} a_{j,1} c_j + e_1, \ldots, \sum_{j=1}^{i} a_{j,l} c_j + e_l)$. We can obtain $r^{i)}$ from r and $c_s, c_{s-1}, \ldots, c_{i+1}$, since $r^{s)} = r$ and $r^{i)} = (r_1^{i+1)} - a_{i+1,1} c_{i+1}, \ldots, r_l^{i+1)} - a_{i+1,l} c_{i+1})$ for $i = s-1, \ldots, 1$.

Let A_i be the submatrix of A consisting of the first i rows of A. Note that $A_s = A$ and A_i is an $i \times l$-matrix that is non-singular by columns. Let $v^{i)} \in \mathbb{F}_q^l$ such that $Av^{i)} = w_i = (0, \ldots, 0, 1)^T \in \mathbb{F}_q^i$. Such a $v^{i)}$ is not unique in general (it is only unique if $i = s = l$). For the sake of simplicity we will denote the coordinates of $v^{i)}$ by $v^{i)} = (v_1, \ldots, v_l)$. Note that $v^{i)}$ is a solution of the corresponding linear system

$$A_i x = w_i \tag{4}$$

Since $Av^{i)} = w_i$, we have that $[c_1 \cdots c_i] \cdot A_i v^{i)} = [c_1 \cdots c_i] w_i = c_i$. Hence, $r^{i)} v^{i)} = c_i + \sum_{j=1}^{l} v_j e_j$, in particular for $i = s$, we have $rv^{i)} = c_s + \sum_{j=1}^{l} v_j e_j$. We may decode $r^{i)} v^{i)}$ with DC_i to obtain c_i if $wt(\sum_{j=1}^{l} v_j e_j) < d_i/2$. Therefore, it is wise to consider a vector $v^{i)}$ with low weight, that is with many coordinates equal to zero.

We will consider a vector $v^{i)}$ with at least $l-i$ coordinates equal to zero, i.e. of weight $wt(v^{i)}) \leq l-(l-i) = i$. Let $J = \{j_1, \ldots, j_i\} \subset \{1, \ldots, l\}$ with $\#J = i$, we claim that we can compute $v^{i)}$, a solution of (4), such that $v_j = 0$ for $j \notin J$. Let A_J be the $i \times i$-submatrix of A_i given by $A_J = (a_{k,j})_{k \in \{1,\ldots,i\}, j \in J}$. Since A is non-singular by columns, one has that A_J is a full rank squared matrix. Let us consider the linear system

$$A_J x = w_i, \tag{5}$$

where $x \in \mathbb{F}_q^i$. The linear system (5) has a unique solution. Let $v_J^{i)} = (v_1 \ldots, v_l)$, where $v_{j_k} = x_k$, for $k = 1, \ldots, i$, and $v_j = 0$ otherwise. Then, $v_J^{i)}$ is a solution of (4) of weight lower than or equal to i, and the claim holds.

There are several choices for the set $J \subset \{1, \ldots, l\}$. We will prove in Theorem 5.2 that at least for one choice of J, we will obtain c_i by decoding $r^{i)} v_J^{i)}$ with DC_i. Therefore, in practice, we should consider $\binom{l}{i}$ vectors $\{v_J^{i)}\}_{J \in \mathcal{J}}$, with $\mathcal{J} = \{J \subset \{1, \ldots, l\} : \#J = i\}$ and decode $r^{i)} v_J^{i)}$ with DC_i. We will have, at most, $\binom{l}{i}$ different candidates for c_i and at least one of them will give c_i as output.

In order to obtain c_{i-1} we should iterate this process for every candidate obtained for c_i. After considering the previous computations for $i = s, s-1, \ldots, 1$, we may have several candidates for $[c_1, \ldots, c_s]$. We can detect which candidate is equal to p by checking that we have not corrected more than $\lfloor (d(C)-1)/2 \rfloor$ errors in total, that is, we check if $d(r - [c_1 \ldots, c_s] \cdot A) \leq \lfloor (d(C)-1)/2 \rfloor$. The algorithm can be seen in procedural form in Algorithm 2. However, it remains to prove that, at least for one choice of the set $J \subset \{1, \ldots, l\}$, one will obtain c_i.

Theorem 5.2.

Let e with $wt(e) \leq t$. There exists $J \subset \{1, \ldots, l\}$, with $\#J = i$, such that $\sum_{j \in J} wt(e_j) < d_i/2$, for $i = 1, \ldots, s$.

Proof. Let $v_J^{i)} = (v_1, \ldots, v_l)$ as before. We have that,

$$wt\left(\sum_{j=1}^{l} v_j e_j\right) \leq wt\left(\sum_{j\in J} e_j\right) \leq \sum_{j\in J} wt(e_j).$$

The result claims that there exists $J \subset \{1, \ldots, l\}$, with $\#J = i \in \{2, \ldots, s\}$, such that $\sum_{j\in J} wt(e_j) < d_i/2$. Let $\mathcal{J} = \{J \subset \{1, \ldots, l\} : \#J = i\}$, and let us assume that the claim does not hold. We consider every $\binom{l}{i}$ possible subset $J \subset \{1, \ldots, l\}$ with i elements, then

$$\sum_{J\in\mathcal{J}} \sum_{j\in J} wt(e_j) \geq \binom{l}{i} \frac{d_i}{2}.$$

Moreover, since $\binom{l-1}{i-1}$ sets of $\mathcal{J}$ contain j, for $j \in \{1, \ldots, l\}$, we have

$$\sum_{J\in\mathcal{J}} \sum_{j\in J} wt(e_j) = \sum_{j=1}^{l} \binom{l-1}{i-1} wt(e_j) = \binom{l-1}{i-1} wt(e) < \binom{l-1}{i-1} \frac{ld_1}{2}.$$

Which implies that

$$\binom{l}{i} d_i < \binom{l-1}{i-1} ld_1,$$

therefore $id_1 > d_i$, contradiction.

For $i = 1$, we have that $\mathcal{J} = \{\{1\}, \ldots, \{l\}\}$.Hence, we have that $r^{1)} v_J = c_j + e_j$, for $J = \{j\}$. The result claims that there exists $j \in \{1, \ldots, l\}$, such that $wt(e_j) < d_1/2$. Otherwise, $wt(e) \geq lt_1 + l > t$, which is a contradiction. □

Remark 5.3. Let us compare this decoding algorithm to the algorithm in [2]. In the algorithm in [2], we assume that $C_1 \supset \cdots \supset C_s$, A is non-singular by columns and in the worst-case we run DC_i $\binom{l}{s}$ times, for $i = 1, \ldots, s$. For the algorithm in this section, we assume that $d_i \geq id_1$ for all $i = 2, \ldots, s$, A is also non-singular but in the worst-case we run $DC_i \prod_{j=i}^{s} \binom{l}{j}$ times. Thus, the algorithm presented in this section can become computationally intense for large values of s, l. If $C_1, \ldots, C_s$ are nested, both algorithms decode up to half of the minimum distance of the code, since the bound in (3) is sharp.

As in previous sections, one can also consider this algorithm for correcting beyond the designed error-correction capability of C, if l is even, d_1 is odd and $d_i > id_1$. Namely, the designed error correction capability of C is $lt_1 + \lfloor \frac{l-1}{2} \rfloor = lt_1 + (l-2)/2$ and we consider now an error vector with $wt(e) < lt_1 + l/2$, that is, we are correcting 1 error beyond the error correcting capability of C. We should just modify line 24 of Algorithm 2 to accept codewords p with $wt(r' - p) \leq lt_1 + l/2$ and create a list with all the output codewords.

We shall prove that, at least for one choice of the set $J \subset \{1, \ldots, l\}$, one will again obtain c_i.

Theorem 5.4.

Let e with $wt(e) \leq lt_1 + l/2$, with d_1 odd, l even and $d_i > id_1$. There exists $J \subset \{1, \ldots, l\}$, with $\#J = i$, such that $\sum_{j\in J} wt(e_j) < d_i/2$, for $i = 1, \ldots, s$.

Algorithm 2 A DECODING ALGORITHM FOR $C = [C_1 \cdots C_s] \cdot A$, SECOND EXTENSION

Input: Received word $r = p + e$ with $c \in C$ and $wt(e) < d(C)/2$. Where $d_i = d(C_i)$ with $id_1 < d_i$ and A a non-singular by columns matrix. Decoder DC_i for code C_i, $i = 1, \ldots, s$.

Output: p.

1: $r' = r$;
2: $Candidates' = \{[0 \cdots 0]\}$ $(0 \in \mathbb{F}_q^m)$;
3: **for** $i = s, s-1, \ldots, 2, 1$ **do**
4:     $Candidates = Candidates'$;
5:     $Candidates' = \{\}$;
6:     **for** $c = (c_1, \ldots, c_s)$ in $Candidates$ **do**
7:         $r = (r'_1 - \sum_{j=i+1}^{s} a_{j,1} c_j, \ldots, r'_l - \sum_{j=i+1}^{s} a_{j,l} c_j)$;
8:         **for** $J \subset \{1, \ldots, l\}$ with $\#J = i$ **do**
9:             Solve linear system $A_J x = w_i$;
10:             $v = (0, \ldots, 0)$;
11:             **for** $k = 1 \ldots, i$ **do**
12:                 $v_{j_k} = x_k$;
13:             **end for**
14:             $b_i = DC_i(rv)$;
15:             **if** $b_i =$ *"failure"* **then**
16:                 *Break* the loop and consider another J in line 8;
17:             **end if**
18:             $Candidates' = Candidates' \cup \{[0 \cdots 0 b_i c_{i+1} \cdots c_s]\}$;
19:         **end for**
20:     **end for**
21: **end for**
22: **for** c in $Candidates'$ **do**
23:     $p = [c_1 \cdots c_s] \cdot A$;
24:     **if** $p \in C$ and $wt(r - p) \leq \lfloor (d(C) - 1)/2 \rfloor$ **then**
25:         RETURN: p;
26:     **end if**
27: **end for**

Proof. Let $v_J^{i)} = (v_1, \ldots, v_l)$ as before. We have that,

$$wt\left(\sum_{j=1}^{l} v_j e_j\right) \leq wt\left(\sum_{j \in J} e_j\right) \leq \sum_{j \in J} wt(e_j).$$

The result claims that there exists $J \subset \{1, \ldots, l\}$, with $\#J = i \in \{2, \ldots, s\}$, such that $\sum_{j \in J} wt(e_j) < d_i/2$. Let $\mathcal{J} = \{J \subset \{1, \ldots, l\} : \#J = i\}$, and let us assume that the claim does not hold. We consider every $\binom{l}{i}$ possible subsets $J \subset \{1, \ldots, l\}$ with i elements, then

$$\sum_{J \in \mathcal{J}} \sum_{j \in J} wt(e_j) \geq \binom{l}{i} \frac{d_i}{2}.$$

Moreover, since $\binom{l-1}{i-1}$ sets of $\mathcal{J}$ contain j, for $j \in \{1,\ldots,l\}$, we have

$$\sum_{J\in\mathcal{J}}\sum_{j\in J} wt(e_j) = \sum_{j=1}^{l}\binom{l-1}{i-1} wt(e_j) = \binom{l-1}{i-1} wt(e) \leq \binom{l-1}{i-1}(lt_1 + \frac{l}{2}).$$

Which implies that,

$$\binom{l}{i}\frac{d_i}{2} \leq \binom{l-1}{i-1}\frac{l}{2}(2t_1+1),$$

therefore $d_i \leq i(2t_1+1) = id_1$, contradiction.

For $i = 1$, we have that $\mathcal{J} = \{\{1\},\ldots,\{l\}\}$. Therefore, we have that $r^{1)}v_J = c_j + e_j$, for $J = \{j\}$. The result claims that there exist $j \in \{1,\ldots,l\}$, such that $wt(e_j) < d_1/2$. Otherwise, $wt(e) \geq lt_1 + l > lt_1 + l/2$, which is a contradiction. □

This algorithm will output a list containing the sent word, however it cannot be uniquely determined: let $p = [c_1,\ldots,c_s]\cdot A$ and $p' = [c_1', c_2\ldots,c_s]\cdot A$ with $c_1 \neq c_1'$, we claim that $d(p,r) \leq lt_1 + l/2$ and $d(p',r) \geq lt_1 + l/2$.

- $d(r,p) = wt(e) \leq lt_1 + l/2$.
- $d(r,p') = wt(a_{1,1}(c_1 - c_1') + e_1,\ldots,a_{1,l}(c_1 - c_1') + e_l) \geq ld_1 - wt(e) \geq l(2t_1+1) - (lt_1 l/2) = lt_1 + l/2$.

The algorithm outputs p and all the other codewords -obtained from the other candidates- that are at distance at most lt_1+l-1 from r. As with $s = l = 2$, the probability of having more than one codeword in the output list is negligible, since $d(r,p') = lt_1 + l/2$ if and only if the bound in (3) is sharp, $d(c_1,c_1') = d_1$ and for every $j = 1,\ldots,m$, $i = 1,\ldots,l$, with $e_{j,i} \neq 0$, one has that $e_{j,i} = -a_{1,i}(c_{j,1} - c_{j,1}')$.

Example 5.5. Consider the following linear codes over $\mathbb{F}_3$,

- C_1 the $[26,16,6]$ cyclic code generated by $f_1 = x^{10} + 2x^7 + 2x^4 + x^3 + 2x^2 + x + 2$.
- C_2 the $[26,7,14]$ cyclic code generated by $f_2 = x^{19} + x^{18} + x^{17} + x^{15} + 2x^{14} + x^{13} + 2x^{12} + x^{11} + 2x^8 + 2x^7 + x^6 + x^4 + x^3 + 2$.
- C_3 the $[26,3,18]$ cyclic code generated by $f_3 = x^{23} + 2x^{22} + x^{21} + 2x^{19} + 2x^{18} + x^{17} + x^{16} + x^{15} + x^{13} + x^{10} + 2x^9 + x^8 + 2x^6 + 2x^5 + x^4 + x^3 + x^2 + 1$.

Let $C = [C_1C_2C_3]\cdot A$, where A is the non-singular by columns matrix

$$A = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 2 \\ 1 & 0 & 1 \end{pmatrix},$$

we consider again polynomial notation for C (see example 4.2). We use decoder DC_i for C_i, which decodes up to half the minimum distance, i.e., DC_1, DC_2, DC_3 decode up to $t_1 = 2$, $t_2 = 6$ and $t_3 = 8$ errors, respectively. Note that $2d_1 = 12 \leq 14 = d_2$ and $3d_1 = 18 \leq 18 = d_3$. We have that $d(C) = d_C = 3d_1 = 18$. Therefore we may correct up to $t = 8$ errors in a codeword of C.

Let $r = p+e$ be the received word, with codeword $p = (0, 0, 0)$ and the error vector of weight $t = 8$

$$e = (e_1, e_2, e_3) = (1 + x + x^2, 1 + 2x^2 + x^7, x^5 + 2x^{11}).$$

We solve the system

$$\begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 2 \\ 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}.$$

which has solution $(2, 2, 2)^T$. Set $r^{3)} = r$ and $v^{3)}_{\{1,2,3\}} = (2, 2, 2)$. Therefore $r^{3)} v^{3)}_{\{1,2,3\}} = c_3 + 2e_1 + 2e_2 + 2e_3$. Since DC_3 can correct up to 8 errors and $wt(-e_1 - e_2 - e_3) = 8$, we have

$$DC_3(r^{3)} v^{3)}_{\{1,2,3\}}) = c_3 = 0$$

Removing c_3 in $r^{3)}$, we obtain $r^{2)} = (r^{3)}_1 - c_3, \ldots, r^{3)}_l - c_3) = r$. Since there are 3 possible sets, $\{1, 2\}, \{1, 3\}, \{2, 3\} \subset \{1, 2, 3\}$, with 2 elements, we solve the corresponding systems of equations give by (5):

$$\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix},$$

$$\begin{pmatrix} 1 & 1 \\ 0 & 2 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix},$$

$$\begin{pmatrix} 1 & 1 \\ 1 & 2 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

These systems have solution $(0, 1)^T$, $(0, 2)^T$ and $(2, 1)^T$ respectively. Therefore, $v^{2)}_{1,2} = (0, 1, 0)$, $v^{2)}_{1,3} = (0, 0, 2)$ and $v^{2)}_{2,3} = (0, 2, 1)$. Thus $r^{2)} v^{2)}_{\{1,2\}} = c_2 + e_2$, $r^{2)} v^{2)}_{\{1,3\}} = c_2 + 2e_3$ and $r^{2)} v^{2)}_{\{2,3\}} = c_2 + 2e_2 + e_3$. Since $t_2 = 6$ and $wt(e_2) = 3 \leq 6$, $wt(2e_3) = 2 \leq 6$ and $wt(2e_2 + e_3) \leq wt(e_2) + wt(e_3) = 5 \leq 6$, we have

$$DC_2(r^{3)} v^{3)}_J) = c_2 = 0, \text{for } J = \{1, 2\}, \{1, 3\}, \{2, 3\}.$$

Therefore, we only have one candidate for c_2. Removing c_2 in $r^{2)}$, we obtain $r^{1)} = (r^{2)}_1 - c_2, \ldots, r^{2)}_l - c_2) = r$. Since there are 3 possible sets, $\{1\}, \{2\}, \{3\} \subset \{1, 2, 3\}$, with 1 element, we solve the corresponding systems of equations give by (5). In this case the 3 systems of equations are

$$\begin{pmatrix} 1 \end{pmatrix} \begin{pmatrix} x \end{pmatrix} = \begin{pmatrix} 1 \end{pmatrix}.$$

Thus, the solution is (1) and $v^{1)}_{\{1\}} = (1, 0, 0)$, $v^{1)}_{\{2\}} = (0, 1, 0)$ and $v^{1)}_{\{3\}} = (0, 0, 1)$. Thus $r^{1)} v^{1)}_{\{1\}} = c_1 + e_1$, $r^{1)} v^{1)}_{\{2\}} = c_1 + e_2$ and $r^{1)} v^{1)}_{\{3\}} = c_1 + e_3$. We consider $DC_1(r^{3)} v^{3)}_J)$: we obtain "failure" for $DC_1(r^{3)} v^{3)}_{\{1\}})$ and $DC_1(r^{3)} v^{3)}_{\{2\}})$ since e_1 and e_2 have weight 3 and there is no codeword at distance 2 because C_1 has minimum distance 6. One has that $wt(e_3) = 2 \leq t_1$, therefore

$$DC_1(r^{3)} v^{3)}_{\{3\}}) = c_1 = 0$$

Finally we get $p = [c_1 c_2 c_3] \cdot A = (0, 0, 0)$.

Acknowledgements

This paper was written in part during a visit of the second author to the Mathematics department of San Diego State University. He wishes to thank this institution and the first author for hospitality.

References

[1] Tim Blackmore and Graham H. Norton. Matrix-product codes over $\mathbb{F}_q$. *Appl. Algebra Engrg. Comm. Comput.*, 12(6):477–500, 2001.

[2] Fernando Hernando, Kristine Lally, and Diego Ruano. Construction and decoding of matrix-product codes from nested codes. *Appl. Algebra Engrg. Comm. Comput.*, 20(5-6):497–507, 2009.

[3] Fernando Hernando and Diego Ruano. New linear codes from matrix-product codes with polynomial units. *Adv. Math. Commun.*, 4(3):363–367, 2010.

[4] F.J. Macwilliams and N.J.A. Sloane. *The Theory of Error-Correcting Codes*, volume 16 of *North-Holland mathematical library.* North-Holland, 1977.

[5] Ferruh Özbudak and Henning Stichtenoth. Note on Niederreiter-Xing's propagation rule for linear codes. *Appl. Algebra Engrg. Comm. Comput.*, 13(1):53–56, 2002.

[6] Bram van Asch. Matrix-product codes over finite chain rings. *Appl. Algebra Engrg. Comm. Comput.*, 19(1):39–49, 2008.