Fault-Tolerant Dot-Product Engines

Ron M. Roth
Computer Science Department, Technion, Haifa 3200003, Israel
ronny@cs.technion.ac.il

Abstract—Coding schemes are presented that provide the ability to correct and detect computational errors while using dot-product engines for integer vector–matrix multiplication. Both the $L_1$-metric and the Hamming metric are considered.

Index Terms—Analog arithmetic circuits, Berlekamp codes, Dot-product engines, In situ computing, Lee metric.

I. INTRODUCTION

We consider the following computational model. For an integer $q \geq 2$, let $\Sigma_q$ denote the subset $\{0, 1, \ldots, q-1\}$ of the integer set $\mathbb{Z}$. Also, let $\ell$ and $n$ be fixed positive integers. A dot-product engine (in short, DPE) is a device which accepts as input an $\ell \times n$ matrix $A = (a_{i,j})_{i \in [\ell], j \in [n]}$ over $\Sigma_q$ and a row vector $u = (u_i)_{i \in [\ell]} \in \Sigma_q^n$, and computes the vector–matrix product $c = uA$, with addition and multiplication carried out over $\mathbb{Z}$. Thus, $c = (c_j)_{j \in [n]}$ is an integer vector in $\mathbb{Z}^n$ (more specifically, over $\mathbb{Z}_{\ell(q-1)+1}$). In the applications of interest, the matrix $A$ is modified much less frequently than the input vector $u$ (in some applications, the matrix $A$ is determined once and then remains fixed, in which case only $u$ is seen as input). Typically, the alphabet size $q$ is a power of 2.

In recent proposals of nanoscale implementations of a DPE, the matrix $A$ is realized as a crossbar array consisting of $\ell$ row conductors, $n$ columns conductors, and programmable nanoscale resistors (e.g., memristors) at the junctions, with the resistor at the junction $(i, j)$ set to have conductance, $G_{i,j}$, that is proportional to $a_{i,j}$. Each entry $u_i$ of $u$ is fed into a digital-to-analog converter (DAC) to produce a voltage level that is proportional to $u_i$. The product, $uA$, is then computed by reading the currents at the (grounded) column conductors, after being fed into analog-to-digital converters (ADCs); see Figure 1. For early implementations and applications of DPE’s, as well as recent ones, see, for example, [3], [8], [11], and [14].

Inaccuracies while programming the resistors in the crossbar and noise while reading the currents are examples of factors that can affect the accuracy of the computation. Specifically, the actually-read row vector, $y = (y_i)_{i \in [n]} \in \mathbb{Z}^n$, may differ from the correct vector, $c = uA$. The error vector is defined as the following vector in $\mathbb{Z}^n$: $e = (e_j)_{j \in [n]} = y - uA$.

One can consider the broader problem where the matrix $A$ and the vector $u$ are over different integer alphabets. Yet, for the sake of simplicity, we will assume hereafter that those alphabets are the same. It is primarily the alphabet of the matrix that will affect the coding schemes that will be presented in this work.

In such context of errors, we find it natural to define the number of errors to be the $L_1$-norm of $e$: $\|e\|_1 = \sum_{j \in [n]} |e_j|$.

In our case—where $e$ is an integer—this norm is also referred to as the Manhattan weight of $e$, which equals the Manhattan distance between $uA$ and $y$.

Another source of computational errors could be junctions in the crossbar becoming shorted due to faults in the programming process. In this case, the current read in the shorted column will be above some compliance limit (“$\infty$”), which, in turn, will flag the respective entry in $y$ as “unavailable” or as an erasure. The $L_1$-metric has been studied quite extensively in the coding literature, along with its finite-field Lee-metric variant; see [1] Ch. 9] and [13, Ch. 10]) (and Subsection IV-A below).

In the other extreme, a junction in the array may become non-programmable or get stuck at an open state, in which cases these can be detected before the DPE is put into operation.

This work was done in part while visiting Hewlett Packard Laboratories, 1501 Page Mill Road, Palo Alto, CA 94304.
if this type of errors is predominant, one could consider the Hamming metric instead, whereby the figure of merit is the Hamming weight of $e$, equaling the number of positions in which $y$ differs from $uA$ (disregarding the extent at which the values of the respective entries actually differ). This Hamming metric is suitable for handling erasures as well.

In this work, we propose methods for using the DPE computational power to self-protect the computations against errors. The first $k$ $(<n)$ entries in $c = uA$ will carry the (ordinary) result of the computation of interest, while the remaining $n-k$ entries of $c$ will contain redundancy symbols, which can be used to detect or correct computational errors, assuming that the number of the latter (counted with respect to either the $L_1$-metric or the Hamming metric) is bounded from above by some design parameter. Specifically, the programmed $\ell \times n$ matrix $A$ will have the structure

$$A = (A' | A''),$$

where $A'$ is an $\ell \times k$ matrix over $\Sigma_q$ consisting of the first $k$ columns of $A$, and $A''$ consists of the remaining $n-k$ columns; the computed output row vector for an input vector $u \in \Sigma_q^\ell$ will then be $c = (c' | c'')$, where the $k$-prefix $c' = uA'$ $(\in \mathbb{Z}^k)$ represents the target computation while the $(n-k)$-suffix $c'' = uA''$ $(\in \mathbb{Z}^{n-k})$ is the redundancy part. In this setting, $A'$ and $u$ are the actual inputs, and $A''$ will need to be computed from $A'$, e.g., by a dedicated circuitry, prior to—or while—programming $A'$ and $A''$ into the crossbar array (yet recall that it is expected that $A'$ will be modified much less frequently than $u$). The error decoding mechanism will be implemented by dedicated circuitry too. Clearly, we will aim at minimizing $n-k$ given the desired error correction capability.

**Example 1.** Let us consider the simplest case where we would like to be able to only detect one $L_1$-metric error. In this case, we select $n = k+1$ and let the $\ell \times 1$ matrix $A'' = (a_{i,k})_{i \in [\ell]}$ be obtained from $A' = (a_{i,j})_{i \in [\ell], j \in [k]}$ by

$$a_{i,k} = \left( \sum_{j \in [k]} a_{i,j} \right) \text{ MOD } 2 , \quad i \in [\ell],$$

where “MOD” stands for (the binary operation of) remaindering; thus, the entries of $A''$ are in fact over $\Sigma_2$, and the sum of entries along each row of $A$ is even. It follows by linearity that the sum of entries of (an error-free) $c = (c_j)_{j \in [n]} = uA$ must be even. On the other hand, if $e \in \mathbb{Z}^n$ is an error vector with $\|e\| = 1$ then the sum of entries of $y = c + e$ will be odd.

Observe that the contents of $A''$ depends on $A'$, but should **not** depend on $u$. In particular, $A''$ should be set so that the specified error correction–detection capabilities hold when $u$ is taken to be a unit vector. Thus, for every row index $i$, the set of (at least) $q^k$ possible contents of row $i$ in $A$ must form a subset of $\Sigma_q^k$ that, by itself (and independently of the contents of the other rows in $A$), meets the correction–detection capabilities.

Secondly, note that a given computed $k$-prefix $c' = uA'$ can be associated with different $(n-k)$-suffixes (redundancy symbols) $c'' = uA''$, depending on $u$. This is different from the common coding theory setting, where the redundancy symbols are uniquely determined by the information symbols (the latter being the counterparts of the entries of $c'$ in our setting). For instance, if $A$ in Example II is

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

(where $q = 2$, $k = 3$, and $n = 4$), then, for $u = (0 \ 0 \ 1)$, $(0 \ 0 \ 1)A = (1 \ 1 \ 0 \ 0)$

while for $u = (1 \ 1 \ 0)$,

$$(1 \ 1 \ 0)A = (1 \ 1 \ 0 \ 2)$$

(in both cases, $c' = (1 \ 1 \ 0)$). Indeed, we will see in the sequel some coding schemes where we will be able to recover $c'$ correctly out of $y$ (which will suffice for our purposes), yet we will not necessarily recover $c''$. This means that we will need to present the error correction–detection specification of a DPE coding scheme slightly differently than usual; we do this in Section III below.

In Section III we present methods for single-error correction and double-error detection in the $L_1$-metric. Methods for multiple-error correction for that metric are then discussed in Section IV. Finally, the Hamming metric is considered in Section V. We will mainly focus on a regime where the number $r$ of correctable errors is fixed (i.e., small) while $n$ grows. Under these conditions, the required redundancy, $n-k$, of our methods will be of the order of $r \cdot \log_2 n$ in the case of the $L_1$-metric, and approximately twice that number in the case of the Hamming metric. Moreover, both the encoding and decoding can be efficiently implemented; in particular, the decoding requires a number of integer (or finite field) arithmetic operations which is proportional to $rn$ (and the implementation can be parallelized to a latency proportional to $r$), where the operands are of the order of $\log_2 n$ bits long.

**II. Definitions**

For integer vectors $x_1$ and $x_2$ of the same length, we denote by $d_C(x_1, x_2)$ the $L_1$-distance between them, namely, $d_C(x_1, x_2) = ||x_1 - x_2||_1$. The Manhattan sphere of radius $t$ centered at $y \in \mathbb{Z}^n$ is defined as the set of all vectors in $\mathbb{Z}^n$ at $L_1$-distance at most $t$ from $y$:

$$S_C(y, t) = \{ x \in \mathbb{Z}^n : d_C(x, y) \leq t \} .$$

The **volume** (size) of $S_C(y, t)$ is known to be $[5, 6]$

$$V_C(n, t) = \sum_{i=0}^{\min\{t, n\} 2^i \binom{n}{i} \binom{t}{i} .$$

Moreover, while systematic encoding is a matter of preference in ordinary coding applications, in our setting it is actually a necessity: the benefits of using the DPE would diminish if post-processing of its output were required even when the output were error-free.
In particular, $V_2(n,1) = 2n+1$, and for any fixed $t$ and sufficiently large $n$ we have $V_{2t}(n,t) = O(n)$, where the 
hidden constant depends on $t$.

Turning to the Hamming metric, we denote by $d_H(x_1, x_2)$ the Hamming distance between $x_1$ and $x_2$, and the Hamming 
sphere of radius $t$ centered at $y \in \mathbb{Z}^n$ is defined by

$$S_H(y, t) = \{ x \in \mathbb{Z}^n : d_H(x, y) \leq t \}$$

(which has infinite size when $t > 0$). In what follows, we will sometimes omit the identifier “$H$” or “$L$” from $d(\cdot, \cdot)$
and $S(\cdot, \cdot)$, if the text applies to both metrics.

Given $\Sigma_q$ and positive integers $t$, $n$, and $k < n$, a DPE coding scheme is a pair $(\mathcal{E}, \mathcal{D})$, where

- $\mathcal{E} : \Sigma_q^{\ell \times k} \to \Sigma_q^{\ell \times n}$ is an encoding mapping such that for every $A' \in \Sigma_q^{\ell \times k}$, the image $A = \mathcal{E}(A')$ has the form $(A' | A'')$ for some $A'' \in \Sigma_q^{(n-k)}$. The set

$$C = \{ u \mathcal{E}(A') : A' \in \Sigma_q^{\ell \times k}, u \in \Sigma_q^t \}$$

is the code induced by $\mathcal{E}$ and its members are called codewords. Thus, $C \subseteq \Sigma_q^m$, where $m = \ell(q-1)^2 + 1$.

- $\mathcal{D} : \Sigma_q^n \to \Sigma_q^{\ell} \cup \{\text{"e"}\}$ is a decoding mapping (the return value "e" will designate a decoding failure).

Note that in the above definition, the decoding mapping $\mathcal{D}$ is not a function of $A'$ (yet one could consider also a 
different setting where $A'$ is known to the decoder).

Borrowing (somewhat loosely) classical coding terms, we will refer to $n$ and $k$ as the length and dimension, respectively, of 
the coding scheme. In the context of a given coding scheme, the $k$-prefix (respectively, $(n-k)$-suffix) of a vector
$x \in \mathbb{Z}^n$ will be denoted hereafter by $x'$ (respectively, $x''$). This notational convention extends to $\ell \times n$ 
matrices over $\mathbb{Z}$, with $A'$ (respectively, $A''$) standing for the sub-matrix consisting of the first $k$ columns (respectively, $n - k$ columns) of an $\ell \times n$ matrix $A$ over $\mathbb{Z}$. Denoting row $i$ of a matrix $X$ by $X_i$, we
then have $(A'_i) = (A_i)'$ and $(A''_i) = (A_i)''$, for every $i \in [\ell]$.

Given nonnegative integers $\tau$ and $\sigma$, a coding scheme $(\mathcal{E}, \mathcal{D})$ is said to correct $\tau$ errors and detect $\tau + \sigma$ errors (in the $L_1$-metric or the Hamming metric, depending on the context) if the following conditions hold for every computed vector $c = uA \in C$ and the respective read vector $y \in \Sigma_Q^m$.

- (Correction condition) If $d(y, c) \leq \tau$, then $\mathcal{D}(y) = c'$.
- (Detection condition) Otherwise, if $d(y, c) \leq \tau + \sigma$, then $\mathcal{D}(y) \in \{c', \text{"e"}\}$.

That is, if the number of errors is $\tau$ or less, then the decoder must produce the correct result of the target computation; 
otherwise, if the number of errors is $\tau + \sigma$ or less, the decoder can flag decoding failure instead (but it cannot produce an 
incorrect result).

So, unlike the respective conditions for ordinary codes, the sphere $S(y, \tau)$ may contain multiple codewords of $C$, yet they
all must agree on their $k$-prefixes. Similarly, the sets $S(y, \tau)$ and $S(y, \tau + \sigma) \setminus S(y, \tau)$ may both contain codewords of $C$, yet these codewords must agree on their $k$-prefixes.

By properly defining the minimum distance of $C$, we can extend to our setting the well known relationship between
minimum distance and correction capability. Specifically, the minimum distance of $C$, denoted $d(C)$ (with an identifier of 
the particular metric used), is defined as the smallest distance between any two codewords in $C$ having distinct $k$-prefixes:

$$d(C) = \min_{e_1 \neq e_2 \subseteq C} d(e_1, e_2)$$

The following result then extends from the ordinary coding setting [13, p. 14, Prop. 1.5] (for completeness, we include a proof in
Appendix A).

**Proposition 1.** Let $\mathcal{E} : \Sigma_q^{\ell \times k} \to \Sigma_q^{\ell \times n}$ be an encoding mapping with an induced code $C$, and let $\tau$ and $\sigma$ be nonnegative integers such that

$$2\tau + \sigma < d(C)$$

Then there exists a decoding mapping $\mathcal{D} : \Sigma_q^n \to \Sigma_q^\ell \cup \{\text{"e"}\}$ such that the coding scheme $(\mathcal{E}, \mathcal{D})$ can correct $\tau$ errors and detect $\tau + \sigma$ errors.

For the special case of the Hamming metric, Proposition 1 can be generalized to handle erasures as well (see [13, p. 16, 
Prop. 1.7] and Appendix A).

**Proposition 2.** With $\mathcal{E}$ and $C$ as in Proposition 1 let $\tau$, $\sigma$, and $\rho$ be nonnegative integers such that

$$2\tau + \sigma + \rho < d_H(C)$$

Then there exists a decoding mapping $\mathcal{D} : \Sigma_q^n \to \Sigma_q^\ell \cup \{\text{"e"}\}$ such that the coding scheme $(\mathcal{E}, \mathcal{D})$ can correct $\tau$ errors and detect $\tau + \sigma$ errors, in the presence of up to $\rho$ erasures.

The coding schemes that we present in upcoming sections are based on known codes, in particular known schemes for the
Lee and Manhattan metrics—primarily Berlekamp codes [11, Ch. 9], [13, Ch. 10]. Yet certain adaptations are needed due to 
the fact that the computation of the redundancy symbols of the codewords in the induced code $C = \{c = u \mathcal{E}(A')\}$ has to 
be done only through the computation of $A' \to \mathcal{E}(A')$ (which is independent of $u$). Moreover, the alphabet, $\Sigma_Q$, of the entries 
of $\mathcal{E}(A')$ is smaller than the alphabet, $\Sigma_Q$, of the codewords in $C$. Our coding schemes will be separable, in the sense that 
for each row index $i \in [\ell]$, the contents $(\mathcal{E}(A'_i))$, will only be a function of $A'_i$ (and not of the rest of the rows in $A'$); in 
fact, that function will be the same for all $i$, and will not depend on $\ell$. It is expected, however, that the designed number 
of correctable errors, $\tau$, will tend to increase with $\ell$.

### III. Single Error Correction in the $L_1$-metric

In this section, we describe a DPE coding scheme, $(\mathcal{E}_1, \mathcal{D}_1)$, for correcting one $L_1$-metric error; this scheme 
will then be extended (in Subsection III-B) to also detect two errors.

#### A. The coding scheme

Given an alphabet size $q \geq 2$ and a code length $n$, we let $m = \lceil \log_q (2n+1) \rceil$ and $k = n - m$ (thus, $m$ will be 
the redundancy). Let

$$\alpha = (\alpha_0, \alpha_1, \ldots, \alpha_{n-1})$$

It is assumed hereafter that the entries of the received vector remain in the same alphabet, $\Sigma_Q$, as of the computed vector; while errors 
could push the entries to outside that range, they can always be coerced back into $\Sigma_Q$. 

be a vector in $\mathbb{Z}^n$ that satisfies the following properties.

(i) The entries of $\alpha$ are nonzero distinct elements in $[2n+1]$. 

(ii) For any two indexes $i, j \in [n]$, 
$$\alpha_i + \alpha_j \neq 2n + 1.$$ 

(iii) $\alpha_{k+j} = q^j$, for $j \in [m]$. 

We will refer to the entries of $\alpha$ as code locators. Code locators that satisfy conditions (i)–(iii) can be easily constructed for every $q \geq 2$: e.g., when $q^{m-1} \leq n$, we can take 
$$\{\alpha_j \}_{j \in [n]} = \{1, 2, 3, \ldots, n\},$$ 
otherwise, 
$$\{\alpha_j \}_{j \in [n]} = \{(1, 2, 3, \ldots, n) \setminus \{2n+1-q^{m-1}\} \cup \{q^{m-1}\}$$ 
(note that $q^{m-1} < 2n + 1$ and that $q^{m-2} \leq n$, yet $q^{m-1}$ may be larger than $n$; in fact, this will always be the case when $q = 2$).

The encoding mapping $E_1 : \Sigma_q^{\ell \times k} \rightarrow \Sigma_q^{\ell \times n}$ is defined as follows: for every $A' = (a_{i,j})_{i \in [\ell], j \in [k]}$, the last $m$ columns in $A = (A' | A'') = E_1(A')$ are set so that 
$$\sum_{j \in [m]} a_{i,k+j} (q^{k+j} \equiv (\sum_{j \in [k]} a_{i,j} \alpha_j)_{\text{MOD } (2n+1)}, \ i \in [\ell],$$
(2)

where the remainder (the result of the “MOD” operation) is taken to be in $[2n+1]$. Simply put, $a_{i,n-1}a_{i,n-2} \ldots a_{i,k+1}a_{i,k}$ is the representation to base $q$ (from the most-significant digit down to the least) of the remainder in $[2n+1]$ of the (nonpositive) integer $-\sum_{j \in [k]} a_{i,j} \alpha_j$, when divided by $2n+1$.

It follows from (2) that each $A = E_1(A')$ satisfies 
$$A \alpha^T \equiv 0 \pmod{(2n+1)},$$

where $(\cdot)^T$ denotes transposition and the congruence holds component-wise. Hence, for each codeword $c = uA$ in the induced code $C$ we have 
$$c \cdot \alpha^T = uA \alpha^T \equiv 0 \pmod{(2n+1)}. \quad (3)$$

This, in turn, implies that for every two distinct codewords $c_1, c_2 \in C$, 
$$(c_1 - c_2) \cdot \alpha^T \equiv 0 \pmod{(2n+1)},$$
and, therefore, by conditions (i)–(ii) we get that $d(C, c_2) = \|c_1 - c_2\| > 2$, namely, that $d(C) \geq 3$. We conclude from Proposition 1 that when using the encoding mapping defined by (2), to map $A'$ into $A = E_1(A')$, we should be able to correct one error; alternatively, we should be able to detect two errors. We demonstrate next a single-error-correcting decoding mapping.

Let $y = (y_j)_{j \in [n]} = c + e = uA + e$ be the read vector at the output of the DPE, where $e \in \mathbb{Z}^n$ is an error vector having at most one nonzero entry, equaling $\pm 1$. The decoding will start by computing the syndrome of $y$, which is defined by 
$$s = (y \cdot \alpha^T) \mod{(2n+1)}$$
$$= \left(\sum_{j \in [n]} y_j \alpha_j\right) \mod{(2n+1)}.$$

We then have, 
$$s \equiv uA\alpha^T + e \cdot \alpha^T \equiv e \cdot \alpha^T \pmod{(2n+1)}.$$

It follows that $s = 0$ when $e = 0$; otherwise, if $e$ has $\pm 1$ at position $j$ (and is zero otherwise), then 
$$s \equiv \pm \alpha_j \pmod{(2n+1)}.$$

Hence, due to conditions (i)–(iii), the syndrome $s$ identifies the error location $j$ and the error sign uniquely.

The encoding and decoding procedures for our single-error correction scheme are summarized in Figures 2 and 3.

---

**Fig. 2.** Encoding mapping $E_1 : A' \mapsto (A' | A'')$ for single-error correction (or double-error detection).

---

**Fig. 3.** Decoding mapping $D_1 : y \mapsto w$ for single-error correction.

---

**Example 2.** Let $q = 2$ and $n = 15$, in which case $m = 5$ and $k = 10$. Select $\ell = 3$ and 
$$\alpha = (3 \ 5 \ 6 \ 7 \ 9 \ 10 \ 11 \ 12 \ 13 \ 14 \ | \ 1 \ 2 \ 4 \ 8 \ 16)$$
(which satisfies conditions (i)–(iii)). Suppose that $A'$ is the following $3 \times 10$ matrix: 
$$A' = \begin{pmatrix}
1011010010 \\
0001011001 \\
0100010111
\end{pmatrix}.$$
For \( i = 0, 1, 2 \), the values at the right-hand side of (2) are given by
\[
- (3 + 6 + 7 + 10 + 13) \mod 31 = 23 \\
- (7 + 10 + 11 + 14) \mod 31 = 20 \\
- (5 + 10 + 12 + 13 + 14) \mod 31 = 8 ,
\]
and so,
\[
A = E_1(A') = \begin{pmatrix}
1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 \\
0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\
0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0
\end{pmatrix} .
\]
For \( u = (1 1 1) \), we get
\[
c = uA = (1 1 1 2 0 3 1 1 2 2 1 1 2 1 2) .
\]
Suppose that the read vector is
\[
y = (1 1 1 2 0 3 1 1 2 2 1 1 2 1 2) .
\]
The syndrome of \( y \) is given by
\[
s = (y \cdot \alpha^T) \mod 31 = (3 + 5 + 6 + 2 + 7 + 9 + 2 + 10 + 11 + 12 + 13 + 14 + 2 + 13 + 12 + 12 + 2 + 2 + 1 + 12 + 1 + 2 + 2 + 1 + 2 + 4 + 12 + 1) \mod 31 = 21 .
\]
Namely, \( s = 31 - 21 = 10 = \alpha_5 \), indicating that the error location is \( j = 5 \) (the sixth entry) and the error value is \(-1\) (corresponding to changing the value 3 into 2).

We end this subsection by demonstrating that a redundancy of \( n - k = \lceil \log_q(2n + 1) \rceil \) is within one symbol from the smallest possible for any coding scheme that corrects one error in the \( L_1 \)-metric. Recall that by taking \( u \) as a unit vector it follows that for any row index \( i \), the set of the \( q^k \) possible contents of \( A_i \), forms an (ordinary) code \( C_i \subseteq \Sigma_q^n \) that is capable of correcting one error. Hence, by a sphere-packing argument we conclude that for distinct \( c \in C_i \), the (truncated) spheres \( S(c, 1) \cap \Sigma_q^n \) must be disjoint subsets of \( \Sigma_q^n \). Yet \( |S(c, 1) \cap \Sigma_q^n| \geq n + 1 \), and, so, \( q^k = |C_i| \leq q^n/(n + 1) \), namely, we have the lower bound
\[
n - k \geq \lceil \log_q(n + 1) \rceil .
\]

**B. Allowing additional error detection**

The presented coding scheme can be easily enhanced so that the induced code has minimum distance 4; namely, the scheme can detect two errors on top of correcting a single error, or, alternatively, it can detect three errors with no attempt to correct any error. We do this by extending the code length by 1 and adding a parity bit to each row of \( A = E_1(A') \), as we did in Example 1 (with \( A \) playing the role of \( A' \) therein). This, in turn, allows the decoder to recover the parity of the number of errors (whether it was even or odd). An odd number is seen as one error, and the algorithm in Figure 3 is then applied. Otherwise, a zero syndrome will indicate no errors, while a nonzero syndrome indicates two errors (which will be flagged by “e”).

When \( q > 2 \), this extra error detection capability can sometimes be achieved without increasing the redundancy. To see this, consider first the case where \( q \) is odd: redefine \( m \) to be \( \lceil \log_q(4n + 2) \rceil \) (depending on \( n \), the value of \( m \) may remain unchanged by this redefinition), and modify condition (i)-(ii) as follows.

(i’) The entries of \( \alpha \) are odd distinct elements in \( [4n + 2] \).

(ii’) For any two indexes \( i, j \in [n] \),
\[
\alpha_i + \alpha_j \neq 4n + 2 .
\]
(Note that condition (iii), which remains unchanged, is consistent with condition (i’). Also, condition (ii’) disqualifies \( 2n + 1 \) to be an entry of \( \alpha \).) The encoding is similar to (2), except that the remainder at the right-hand side is now computed modulo \( 4n + 2 \). Accordingly, during decoding, the syndrome is redefined to
\[
s \leftarrow (y \cdot \alpha^T) \mod (4n + 2) ,
\]
and, so, the parity of the syndrome equals the parity of the number of errors. An odd syndrome indicates that one error has occurred, in which case the error location and sign can be recovered from the value of \( s \). A nonzero even syndrome \( s \) indicates that two errors have occurred.

Assume now that \( q \) is an even integer greater than 2. In this case, condition (i’) would contradict condition (iii), as the latter requires that the last \( m - 1 \) entries of \( \alpha \) be even. To overcome this impediment, we will modify the definition of \( m \) and rewrite condition (iii). Specifically, we let \( m \) be the smallest integer that satisfies \( \sum_{i=1}^{m} f_i(q) \geq 4n + 2 + (-1)^m \) where, for every nonnegative \( j \in \mathbb{Z} \),
\[
f_j(q) = \frac{q^{j + 1} + (-1)^j}{q + 1} .
\]
Note that \( f_0(q) = 1 \) and that for every \( j > 0 \),
\[
f_j(q) = (q - 1) \sum_{i=0}^{j} f_i(q) + \begin{cases} 1 & \text{if } j \text{ is even} \\ 0 & \text{otherwise} \end{cases} ,
\]
which means that every integer in \( [4n+2] \) has a representation of the form \( \sum_{j \in [m]} b_j f_j(q) \), for \( b_j \in \mathbb{Z} \). Moreover, the values \( f_j(q) \) are all odd. Hence, rewriting condition (iii) as follows will be consistent with condition (i’):

(iii’) \( \alpha_{6+j} = f_j(q) \), for \( j \in [m] \).

From this point onward, we proceed as in the case of odd \( q \). (We point out that since \( f_m(q) < q^m \) for \( m > 0 \), the inequality \( f_m(q) \geq 4n + 2 + (-1)^m \) is generally stronger than \( m \geq \log_q(4n + 2) \); however, the ratio \( f_m(q)/q^m \) does approach 1 as \( q \to \infty \).)

**Example 3.** Suppose that \( q = 8 \) and \( n = 13 \). Since \( f_1(8) = 7 \) and \( f_2(8) = 57 \), we can take \( m = 2 \) and
\[
\alpha = (3 5 9 11 13 15 17 19 21 23 25 1 7) ,
\]
resulting in a single-error-correcting double-error-detecting coding scheme. The redundancy \( n - k \) will be only 2 in

\[\text{Yet the coding scheme will work also when } (\alpha_{n-1} =) \ q^{n-1} = 2n+1, \text{in spite of violating condition (ii’).}\]

\[\text{The sequence } (f_j(q))_j \text{ can be seen as a generalization of the Jacobsthal sequence: see for instance [7].}\]

\[\text{The coding scheme will work also when } (\alpha_{n-1} =) \ f_{m-1}(q) = 2n+1, \text{in spite of violating condition (ii’).}\]
Thus, $n$ is the largest possible, for a given length term that depends on F-Lee-metric A. Lee-metric codes next subsection. Our coding schemes will make use of known construction for the least decoding up to syndrome, § over $\mathbb{Z}$ over $\mathbb{C}$ over $\mathbb{Z}$. It is easy to see that here, too, such a vector $\beta$ always exists. The respective Berlekamp code, $C_{\text{Ber}} = C_{\text{Ber}}(\beta, \tau)$, is defined as the set of all row vectors in $F^n$ in the right kernel of the following $\tau \times n$ parity-check matrix, $H_{\text{Ber}} = H_{\text{Ber}}(\beta, \tau)$, over $\Phi$: 

$$H_{\text{Ber}} = \begin{pmatrix} \beta_1 & \beta_2 & \ldots & \beta_\tau \\ \beta_1^2 & \beta_2^2 & \ldots & \beta_\tau^2 \\ \vdots & \vdots & \ddots & \vdots \\ \beta_1^{2\tau-1} & \beta_2^{2\tau-1} & \ldots & \beta_\tau^{2\tau-1} \end{pmatrix}.$$  

(5) 

That is, 

$$C_{\text{Ber}} = \left\{ c \in F^n : c \cdot H_{\text{Ber}}^T = 0 \right\}.$$ 

Thus, $C_{\text{Ber}}$ is a linear $[n, k]$ code over $F$ with a redundancy $n - k$ of at most $\tau h = \tau \lceil \log_p(2n + 1) \rceil$. 

The minimum Lee distance of $C_{\text{Ber}}$ is known to be at least $2\tau + 1$, and there are known efficient algorithms for decoding up to $\tau$ Lee-metric errors (see [11], Ch. 9) and [13, §10.6]). These decoders typically start with the syndrome, $s = y H_{\text{Ber}}^T$, of the received vector $y \in F^n$, and then implement a function $D_{\text{Ber}} : \Phi^r \rightarrow F^n$ which maps $s$ to the error vector $e = D_{\text{Ber}}(s)$. The condition $2\tau < p$ implies that (1) is also the volume of a Lee-metric sphere of radius $t$ in $F^n$. Hence, by sphere-packing arguments, the size of any Lee-metric $\tau$-error-correcting code in $F^n$ is bounded from above by $p^n/V(n, \tau)$ [13, p. 318]. Thus, up to an additive term that depends on $\tau$ (but not on $n$) the dimension of $C_{\text{Ber}}$ is the largest possible, for a given length $n$ and number $\tau$ of Lee-metric errors to be corrected.

There is a close relationship between the construction presented in Section III and Berlekamp codes. Specifically, when $n$ is taken so that $p = 2n + 1$ is a prime, then each row of $A = E_1(A')$ is a codeword of $C_{\text{Ber}}(\alpha, 1)$, assuming that the entries of $A$ and $\alpha$ are seen as elements of $\Phi = F = GF(2n + 1)$. Consequently, the induced code $C$ forms a subset of $C_{\text{Ber}}(\alpha, 1)$. With this relationship in mind, we will next present coding schemes whose induced codes have minimum $L_1$-distances 5 and above.

B. Double-error-correcting coding scheme

In this subsection, we present a DPE coding scheme for correcting two errors in the $L_1$-metric (namely, the induced code will have minimum $L_1$-distance at least 5). This scheme will then be extended to also detect three errors (minimum distance 6).

Given the alphabet $\Sigma_q$ and the number of rows $\ell$, let $p > 3$ be a prime, and define $n_1 = (p - 1)/2$, $m = \lceil \log_q p \rceil$, $n_2 = n_1 + m$, and $n = n_2 + 1$. The coding scheme will have dimension $k = n_1 - m$, length $n$ and, therefore, redundancy $n - k = 2m + 1$. The encoding mapping, $E_2 : \Sigma_q^{\ell} \times \Sigma^\tau_q \rightarrow \Sigma^\ell_q$, will take the form of a composition

$$E_2 = \varphi_2 \circ \varphi_2 \circ E_1,$$

where the component functions are defined in Figure 4.

- $E_1$ is the encoding mapping in Figure 2 with $n$ therein replaced by $n_1$ (in particular, the remainder in the right-hand side of (2) is computed modulo $p$).
- $\varphi_2 : \Sigma_q^{\ell} \times n_1 \rightarrow \Sigma_q^{\ell \times n_2}$ maps an $\ell \times n_1$ matrix $A' = (a_{i,j})_{i \in [\ell], j \in [n_1]}$ over $\Sigma_q$ to $A = (A' | A'')$, where the last $m$ columns in $A$ are set so that

$$\sum_{j \in [m]} a_{i,n_2+j}q^j = \left( \sum_{j \in [n_1]} a_{i,j}q^j \right) \mod p, \quad i \in [\ell].$$  

(6)

- $\varphi_2 : \Sigma_q^{\ell \times n_2} \rightarrow \Sigma_q^{\ell \times n}$ is the parity mapping defined on the last $m$ columns of the argument matrix; that is, an $\ell \times n_2$ matrix $A' = (a_{i,j})_{i \in [\ell], j \in [n_2]}$ over $\Sigma_q$ is mapped to $A = (A' | A'')$, where the entries of the last column in $A$ are given by

$$a_{i,n_2} = \left( \sum_{j \in [m]} a_{i,n_1+j} \right) \mod 2, \quad i \in [\ell].$$

Fig. 4. Component functions of a double-error-correcting encoding mapping $E_2 = \varphi_2 \circ \varphi_2 \circ E_1$.

Example 4. Let $q = 2$ and $p = 31$, in which case $n_1 = 15$, $m = 5$, $n_2 = 20$, $n = 21$, and $k = 10$. Select $\ell = 3$ and let $\alpha$ 9The seeming restriction on $n$ imposed by requiring that $2n_1 + 1$ is a prime can be lifted by code shortening.
and $A'$ be as in Example 2. For $i = 0, 1, 2$, the values at the right-hand side of (6) are

\[(3^3 + 6^3 + 7^3 + \cdots + 13^3 + 1^3 + 2^3 + 3^3 + 4^3 + 5^3 + 6^3) \mod 31 = 16 \]
\[(7^3 + 10^3 + 11^3 + 13^3 + 4^3 + 1^3 + 5^3 + 6^3) \mod 31 = 30 \]
\[(5^3 + 10^3 + 12^3 + 13^3 + 4^3 + 8^3) \mod 31 = 29 \]

and, so,

\[
\mathcal{E}_2(A') = \varphi_2(\varphi_2(\mathcal{E}_1(A'))) = \begin{pmatrix}
1011010010 & 11101000001 & 00010111000 & 10001100100 & 11001101101
\end{pmatrix}
\]

It follows from the definition of $\mathcal{E}_2$ that every codeword $c = uA$ in the code $\mathcal{C}_2$ induced by $\mathcal{E}_2$ satisfies the following congruences:

\[
\sum_{j \in [n_1]} c_j\alpha_j \equiv 0 \pmod{p} \tag{7}
\]

\[
\sum_{j \in [n_1]} c_j\alpha_j^3 = \sum_{j \in [m]} c_{n_1+j}q^j \pmod{p} \tag{8}
\]

\[
\sum_{j \in [m+1]} c_j \equiv 0 \pmod{2} \tag{9}
\]

**Proposition 3.** The induced code $\mathcal{C}_2$ satisfies

\[d(\mathcal{C}_2) \geq 5.\]

**Proof.** Let $y = c + e$ be the read vector at the DPE output, where $c \in \mathcal{C}$ and $\|e\| \leq 2$. Write $e = (e_1 \mid e_2)$ and $y = (y_1 \mid y_2)$, where $e_1$ (respectively, $y_1$) is the $n_1$-prefix of $e$ (respectively, $y$). We associate with $y$ the integer syndrome $(s_1, s_2, \hat{s}_2)$ computed as in Eqs. (10)–(12) in Figure 5.

We distinguish between the following cases.

**Case 1:** $s_1 = 0$. In this case, $e_1 = 0$ (i.e., $y_1$ is error-free), since $\|e_1\| \in \{1, 2\}$ implies $s_1 \neq 0$.

**Case 2:** $s_1 \neq 0$ and $\hat{s}_2 = 0$. In this case, $\|e_1\| \in \{1, 2\}$, and (by 9), $y_2$ contains an even number of errors, which means that $e_2 = 0$. Therefore,

\[
s_2 = \sum_{j \in [n_1]} y_j\alpha_j^3 - \sum_{j \in [m]} y_{n_1+j}q^j
\]

\[
\equiv \sum_{j \in [n_1]} y_j\alpha_j^3 - \sum_{j \in [m]} c_{n_1+j}q^j
\]

\[
\equiv \sum_{j \in [n_1]} (c_j + e_j)\alpha_j^3 - \sum_{j \in [m]} c_{n_1+j}q^j
\]

\[
\equiv \sum_{j \in [n_1]} e_j\alpha_j^3 \pmod{p}. \tag{10}
\]

On the other hand, from (7) we also have

\[
s_1 = \sum_{j \in [m]} e_j\alpha_j \pmod{p}. \tag{11}
\]

Hence, we can recover $e_1$ by applying a decoder $D_{Ber}$ for $C_{Ber}(\alpha, 2)$ to the syndrome $(s_1, s_2, \hat{s}_2)$.

**Case 3:** $s_1 \neq 0$ and $\hat{s}_2 = 1$. This is possible only if $\|e_1\| = \|e_2\| = 1$, which means that we are able to recover $e_1$ from $s_1$ (using the decoding mapping in Figure 5).

Note that in case 1 in the last proof, $y_2$ may contain one or two errors, yet we do not attempt to decode them; in fact, their decoding might not be unique. However, $y_1$ still decodes correctly.

**Theorem 3.** The decoding mapping $D_2$ shown in Figure 5.

**Input:** $y = (y_1 \mid y_2) \in \Sigma_q^n$.

**Output:** $w \in \Sigma_q^m$.

\[
\begin{array}{l}
// \; n_1 = \langle p - 1 \rangle / 2, \; m_1 = \lceil \log_2 p \rceil, \; n_2 = n_1 + m. \\
// \; n = n_2 + 1, \; k = n_1 - m. \\
// \; \alpha \text{ satisfies conditions (i)-(iii).} \\
// \; y_1 \text{ is the } n_1\text{-prefix of } y. \\
// \; D_{Ber}(\cdot) \text{ is a decoder for } C_{Ber}(\alpha, 2). \\
\end{array}
\]

Let

\[
s_1 \leftarrow \left( \sum_{j \in [n_1]} y_j\alpha_j^3 \right) \pmod{p} \tag{12}
\]

\[
s_2 \leftarrow \left( \sum_{j \in [n_1]} y_j\alpha_j^3 - \sum_{j \in [m]} y_{n_1+j}q^j \right) \pmod{p} \tag{13}
\]

\[
\hat{s}_2 \leftarrow \left( \sum_{j \in [m+1]} y_{n_1+j} \right) \pmod{2};
\]

If $s_1 = 0$ then \{ \\
Let \( w \leftarrow y' \); \; // \; y_1 \text{ is error-free} \\
\} \]

Else if $\hat{s}_2 = 0$ then \{ \\
Let $e_1 \leftarrow D_{Ber}(s_1, s_2)$; \\
Let $w \leftarrow y_1 - e_1$; \; // \; w \text{ is the } k\text{-prefix of } y_1 - e_1 \\
\} \]

Else \{ \\
Let $w \leftarrow D_1(y_1)$.
\}

**Fig. 5.** Decoding mapping $D_2 : y \mapsto w$ for double-error correction.

We include in Appendix B an example of an application of the decoder in Figure 5 for self-containment, that example also recalls the decoding principles of Berlekamp codes.

The coding scheme $(\mathcal{E}_2, D_2)$ can be extended to also detect three errors, by adding an overall parity bit to each row of the encoded matrix $A$, as was done in Subsection III-B. Moreover, the savings shown there when $q > 2$ carries over also to minimum distances 5 and 6.

Specifically, for odd $q$, we redefine $m$ to be $\lfloor \log_2 (2p) \rfloor = \lfloor \log_q (4n_1+2) \rfloor$, and require $\alpha$ to satisfy conditions (i')–(ii'). The encoding mapping $\mathcal{E}_2$ is redefined to just $\varphi_2 \circ \mathcal{E}_1$, with code length $n = n_2 = n_1 + m$ and redundancy $n - k = 2m$.

The component functions $\mathcal{E}_1$ and $\varphi_2$ are as in Figure 4 except that all remainders modulo $p$ are now computed modulo $2p$.

10When applying this decoder, we regard $\alpha$ and $(s_1, s_2)$ as vectors over $\Phi = \mathbb{F} = \mathbb{GF}(p)$. The decoder $D_{Ber}$ produces an error vector in $\mathbb{F}^{n_1}$, which is mapped back to $\mathbb{Z}^{n_1}$ by changing each given entry $z \in \mathbb{F}$ into $\pm|z|$, with the negative sign taken when $z$ is a “negative” element of $\mathbb{F}$.

11See Footnote 6 for the case where $q = p$. 
The function of the syndrome element \( s_2 \) in the decoding process is replaced by the parities of \( s_1 \) and \( s_2 \), when computed as in (10) and (11), except that the remainders are taken modulo 2\( p \) (yet \( \text{C}_{\text{Ber}} \) is still defined over \( F = GF(p) \)), so when its decoder is applied to \((s_1, s_2)\), the syndrome entries are reduced first modulo \( p \). Specifically, assuming that at most three errors have occurred, Table I presents the various combinations of parities of \( s_1 \) and \( s_2 \), and the corresponding \( L_1 \)-norms of \( e_1 \) and \( e_2 \). The first three rows in the table correspond, respectively, to the three cases in the proof of Proposition 3. The fourth row corresponds to three errors and therefore should result in a decoding failure. The last three rows correspond to three different combinations of numbers of errors: the distinction among them can be made by checking whether \( s_2 \equiv s_2^1 \mod p \), and, since \( \text{C}_{\text{Ber}}(\alpha, 2) \) has minimum Lee distance 5, this condition will be met only when one error has occurred (in which case it can be found using the decoding mapping in Figure 3).

When \( q \) is even and greater than 2, we will follow the same strategy as in Section III, namely, replacing the terms \( q^j \) with \( f_j(q) \) defined in (4), both in condition (iii) and in (4).

Example 5. Suppose that \( q = 4 \) and \( p = 101 \), corresponding to \( n_1 = 50 \). The values \( f_j(4) \) for \( j = 0, 1, 2, 3, 4 \) are 1, 3, 13, 51, and 205, respectively, so we can take \( m = 4 \) and \( \alpha = (5 \ 7 \ 9 \ 11 \ 15 \ 17 \ldots \ 47 \ 49 \ 53 \ 55 \ldots \ 97 \ 99 \ | \ 1 \ 3 \ 13 \ 51) \). The respective double-error-correcting triple-error-detecting coding scheme has length \( n = n_1 + m = 54 \) and redundancy \( n - k = 2m = 8 \) (and dimension \( k = 46 \)). An example of an image of the encoding mapping \( \mathcal{E}_2 = \varphi_2 \circ \mathcal{E}_1 \) is given by

\[
A = \mathcal{E}_2 \Bigl( \begin{bmatrix} 123012000 \cdots 021020132 \ 030123000 \cdots 033211223 \ 211320000 \cdots 023020202 \end{bmatrix} \Bigr)
\]

with

\[
c = uA = (4 \ 14 \ 7 \ 6 \ 10 \ 13 \ 0 \ 0 \ | \ 15 \ 14 \ 6 \ 9 \ | \ 5 \ 8 \ 12 \ 15)
\]

being an example of a codeword (which corresponds to the input vector \( u = (2 \ 3 \ 1) \)).

Given a read vector \( y = (y_j)_{j \in \{n\}} \in \Sigma_q^n \) (where \( Q = 28 \) and \( n = 54 \)), its syndrome is given by

\[
s_1 = \left( \sum_{j \in \{n_1\}} y_j \alpha_j \right) \mod (2p)
\]

\[
s_2 = \left( \sum_{j \in \{n_1\}} y_j \alpha_j^3 - \sum_{j \in \{m\}} y_{n_1+j} f_j(q) \right) \mod (2p)
\]

where \( q = 4 \), \( p = 101 \), \( n_1 = 50 \), and \( m = 4 \). Correction of two errors and detecting of three then proceeds by following Table I where \( y_1 \) and \( y'_1 \) are the prefixes of \( y \) of lengths \( n_1 = 50 \) and \( k = 46 \), respectively. \[\square\]

C. Recursive coding scheme

The construction in Subsection IV-B does not seem to generalize in a straightforward way to larger minimum \( L_1 \)-distances. However, with some redundancy increase (which will be relatively mild for code lengths sufficiently large), we can construct coding schemes for any prescribed number of correctable errors. We show this next.

Given the alphabet \( \Sigma_q \), number of rows \( \ell \), designed number of correctable errors \( \ell \tau \), let \( p > 2 \tau \) be a prime, and define

\[
n = (p - 1)/2 \quad \text{and} \quad m = \lceil \log_q p \rceil.
\]

Also, let \( \alpha = (\alpha_j)_{j \in \{n\}} \) be an integer vector that satisfies conditions (i)-(iii) in Section III.

Given a matrix \( A \in \Sigma_q^\ell \times m \) (which, at this point, is not assumed to be the result of any encoding), we can compute the following \( \ell \times \tau \) syndrome matrix of \( A \) over \( Z \):

\[
S = (s_{i,v})_{i \in \{\ell\}, v \in \{\tau\}} = AH_{\text{Ber}}^T \mod p,
\]

where \( H_{\text{Ber}} = H_{\text{Ber}}(\alpha, \tau) \) is the parity-check matrix defined in (5), now seen as a matrix over \( Z \), and the remainder computed entry-wise. For a vector \( u \in \Sigma_q^\ell \), the syndrome \( s = s(u) \) of \( c = c(u) = uA \) is then given by

\[
s = cH_{\text{Ber}}^T \mod p = uAH_{\text{Ber}}^T \mod p = uS \mod p.
\]

If the syndrome \( s \) is available to the decoder, then the decoder should be able to recover \( c = uA \) from an erroneous copy \( y = c + e \in Z^n \), provided that \( ||e|| \leq \tau \); this is simply because the syndrome \( s \) of \( e \) is computable from \( s \) and the syndrome of \( y \).

\[
\hat{s} = eH_{\text{Ber}}^T \mod p = (y - c)H_{\text{Ber}}^T \mod p = (yH_{\text{Ber}}^T - s) \mod p,
\]

and \( e \leftarrow \mathcal{D}_{\text{Ber}}(\hat{s}) \), where \( \mathcal{D}_{\text{Ber}}(\cdot) \) is a decoder for \( \text{C}_{\text{Ber}}(\alpha, \tau) \). Thus, our encoding mapping will be designed so that, inter alia, the decoder is able to reconstruct a copy of \( s \).

Each entry in \( S \), being an integer in \( [p] \), can be expanded to its base-\( q \) representation

\[
s_{i,v} = \sum_{j \in \{m\}} s_{i,v}^{(j)} q^j,
\]

where \( s_{i,v}^{(j)} \in \Sigma_q \). For \( j \in \{m\} \), let \( S^{(j)} \) be the \( \ell \times \tau \) matrix \( (s_{i,v}^{(j)})_{i \in \{\ell\}, v \in \{\tau\}} \) over \( \Sigma_q \). Clearly,

\[
S = \sum_{j \in \{m\}} q^j S^{(j)}
\]

12See Footnote 8 for the case where \( f_{m-1}(q) = p \).
and, so,
\[ s = uS \mod p = \left( \sum_{j \in [m]} q^j(uS^{(j)}) \right) \mod p \]
\[ = \left( \sum_{j \in [m]} q^j s^{(j)} \right) \mod p , \]
where \( s^{(j)} = s^{(j)}(u) = uS^{(j)} \) is a vector in \( \Sigma_q^m \). Consider an encoding mapping \( E : \Sigma_q^{\ell \times n} \rightarrow \Sigma_q^{\ell \times (n + \tau m)} \) defined by
\[ E : A \mapsto \left( A \mid S^{(0)} \mid S^{(1)} \mid \cdots \mid S^{(m-1)} \right) . \]
Then, for \( u \in \Sigma_q \) we have
\[ uE(A) = \left( c \mid s^{(0)} \mid s^{(1)} \mid \cdots \mid s^{(m-1)} \right) \]
where \( c = c(u) \) and \( s^{(j)} = s^{(j)}(u) \). If \( y = uE(A) + e \) where \( ||e|| \leq \tau \), then, based on our previous discussion, we will be able to recover \( c \), as long as the \( \tau m \)-suffix of \( y \) is error-free.

The latter assumption (of an error-free suffix) can be guaranteed by applying a (second) encoding mapping to the \( \ell \times \tau n \) matrix
\[ \left( S^{(0)} \mid S^{(1)} \mid \cdots \mid S^{(m-1)} \right) \]
(over \( \Sigma_q \)) so that \( \tau \) errors can be corrected. Note that the matrix now has \( \tilde{n} = \tau n \) columns (instead of \( n \)), so we can base our encoding on a Berlekamp code over \( GF(\tilde{p}) \), where \( \tilde{p} \) is the smallest prime which is at least \( 2\tilde{n} + 1 \). The size of the syndrome now will be \( \tau \tilde{n} \), where \( \tilde{n} = [\log_q \tilde{p}] \), namely, becoming doubly-logarithmic in \( n \).

We can continue this process recursively; by just applying one more recursion level with a simple repetition encoding mapping (which copies its input \( 2\tau + 1 \) times at the output), we obtain a total redundancy of
\[ \tau m + (2\tau + 1)\tau \tilde{n} = \tau[\log_q(2n + 1)] + O\left(\tau^2 \log_q(\tau \log_q n)\right) . \quad (13) \]
Hence, for \( n \) large compared to \( \tau \), most of the redundancy is due to the first encoding level. In fact, by extending the sphere-packing argument presented at the end of Subsection III it follows that the redundancy (13) is optimal, up to an additive term that depends on \( \tau \), but not on \( n \).

Decoding is carried out backwards, starting with recovering the codeword that corresponds to the last encoding level, which, in turn, serves as the syndrome of the previous encoding level.

Reflecting now back on our constructions in Sections III and [IV-B] if the matrix \( A (\in \Sigma_q^{\ell \times n}) \) is the output of the encoding scheme in Figure 2 then the first column of the syndrome matrix \( S \) is zero, and therefore so is the first column in each matrix \( S^{(j)} \) (and the first entry in each vector \( s^{(j)} \)). Hence, those zero columns can of course be removed. As for the second column, our construction in Subsection [IV-B] implies that it can be error-protected simply by a parity bit (or, when \( q \geq 2 \), by changing the modulus from \( p \) to \( 2p \) and selecting the entries of \( \alpha \) to be odd).

The approach of recursive encoding is not new, and has been used, for example, in the context of constrained coding (e.g., see [2], [4], [9], [10], [12]). In our setting, this approach allows us to use codes (namely, Berlekamp codes), which are originally defined over one alphabet of size \( p \), while the result of the encoding (namely, the contents of the rows of the DPE matrix) are restricted to belong to another alphabet of size \( q \) (the challenge is evident when \( q < p \)). In the next subsection, we consider a more straightforward application of Berlekamp codes to construct a coding scheme for the case where \( q \) is large enough; this scheme may sometimes have a smaller redundancy than (13).

**D. Coding scheme for large alphabets**

We consider here the case where the number of correctable errors \( \tau \) and the alphabet size \( q \) are such that there exists a prime \( p \) that satisfies
\[ 2\tau < p \leq q . \]
We will then assume that \( p \) is the largest prime that does not exceed \( q \), and we let \( F \) be the finite field \( GF(p) \).

We will use a systematic encoder \( E_{Ber} : F^k \rightarrow C_{Ber} \), where \( C_{Ber} = C_{Ber}(\beta, \tau) \) is a Berlekamp code of a prescribed length \( n \) over \( F \) and redundancy
\[ n - k \leq \tau \cdot \left[ \log_q(2n + 1) \right] = \tau \cdot \left[ \left( \log_p q \right) \cdot \log_q(2n + 1) \right] \]
(when \( n \) is sufficiently large compared to \( \tau \), the inequality is known to hold with equality). When \( q = p \), this redundancy is smaller than (13); otherwise, it will be larger for \( \tau \) (much) smaller than \( n \), due to the factor \( \log_p q \) (e.g., for \( q = 8 \), this factor is approximately 1.07).

Our encoding mapping \( E : \Sigma_q^{\ell \times k} \rightarrow \Sigma_q^{\ell \times n} \) takes each row in the pre-image matrix \( A' \in \Sigma_q^{\ell \times k} \), computes the remainder of each entry modulo \( p \), regards the result as a vector in \( F^k \), and applies to it the encoder \( E_{Ber} \) to produce a codeword \( c \in C_{Ber} \). The \((n-k)\)-suffix, \( c'' \), of \( c \) becomes the \((n-k)\)-suffix of the respective row in the image \( A = (A' \ | \ A'') = E(A') \) (see Figure 6).

**Input:** \( \ell \times k \) matrix \( A' = (a_{i,j})_{i \in [\ell], j \in [k]} \) over \( \Sigma_q \).

**Output:** \( \ell \times n \) matrix \( (A' \ | \ A'') = (a_{i,j})_{i \in [\ell], j \in [n]} \) over \( \Sigma_q \).

// \( F = GF(p) \), for a prime \( p \) such that \( 2\tau + 1 \leq p \leq q \).

// \( C_{Ber}(\beta, \tau) \) is a Berlekamp code of length \( n \) over \( F \).

// \( E_{Ber} : F^k \rightarrow C_{Ber}(\beta, \tau) \) is a systematic encoder.

For all \( i \in [\ell] \) do 
\[ \text{Let } c \leftarrow E_{Ber}(A'_i \mod p) ; \]
\[ \text{Let } A''_i \leftarrow c'' . \quad \text{// } F \text{ seen as a subset of } \Sigma_q \]

Fig. 6. Encoding mapping \( E : A' \mapsto (A' \ | \ A'') \) for large alphabets.

It follows from the construction that the codewords of the induced code \( C \), when reduced modulo \( p \), are codewords of \( C_{Ber} \). Thus, we obtain a coding scheme that can correct \( \tau \) errors.
V. CODING SCHEME FOR THE HAMMING METRIC

In this section, we present a coding scheme that handles errors in the Hamming metric; namely, the number of errors equals the number of positions in which the read vector \( y = (y_j)_{j \in [n]} \in \Sigma_Q^n \) differs from the correct computation \( c = (c_j)_{j \in [n]} = uA \).

For the purpose of the exposition, we will introduce yet another design parameter, \( \vartheta \), which will be an assumed upper bound on the absolute value of any error value, namely, on

\[
\max_{j \in [n]} |y_j - c_j|.
\]

Such an error model may be of independent interest in DPE applications, with the special case \( \vartheta = Q - 1 = \ell(q - 1)^2 \) being equivalent to the ordinary Hamming metric.

Given the alphabet \( \Sigma_q \), number of rows \( \ell \), number of columns \( n \), number of correctable errors\(^{14}\) \( \tau \), and an upper bound \( \vartheta \) on the error absolute value (where \( \vartheta = \ell(q - 1)^2 \) for unconstrained error values), let \( p \) be a prime greater than \( 2\vartheta \) and let \( m = [\log_q p] \). We select a respective linear \( \tau \)-error-correcting \([\tilde{n}, k]\) code \( C \) over \( F = GF(p) \) (in the Hamming metric), which is assumed to have an efficient bounded-distance decoder \( D : F^{\tilde{n}} \rightarrow F^n \); for a received word \( \tilde{y} \in F^n \), the decoder returns the true error vector \( \tilde{e} \in F^{\tilde{n}} \), provided that its Hamming weight \( w(\tilde{e}) \) was at most \( \tau \).

The parameters \( n \) and \( \tilde{n} \) are related by

\[
n = k + m(\tilde{n} - k).
\]

Figure 7 presents an encoding mapping \( E : A' \rightarrow (A' | A'') \), where each row of \( A' \), when reduced modulo \( p \), is first extended by the systematic encoder for \( C \) into a codeword \( \tilde{c} \) of \( C \), and then the \( \tilde{n} - k \) redundancy symbols (over \( F \)) in \( \tilde{c} \) are expanded to their base-\( q \) representations to form the respective row in \( A'' \). Specifically,

\[
A'' = \begin{pmatrix} A(0) | A(1) | \cdots | A(m-1) \end{pmatrix},
\]

where each block \( A(j) \) is an \( \ell \times (\tilde{n} - k) \) sub-matrix over \( \Sigma_q \), such that the rows of the \( \ell \times n \) matrix

\[
\tilde{A} = \begin{pmatrix} A' | \sum_{j \in [m]} q^j A(j) \end{pmatrix} \mod p
\]

form codewords of \( C \).

Let the mapping \( \lambda : \mathbb{Z}^n \rightarrow F^{\tilde{n}} \) be defined as follows: for a vector \( x = (x_v)_{v \in [n]} \in \mathbb{Z}^n \), the entries of the image \( \lambda(x) = \tilde{x} = (\tilde{x}_v)_{v \in [\tilde{n}]} \in F^{\tilde{n}} \) are given by

\[
\tilde{x}_v = x_v \mod p, \quad \text{for } v \in [k],
\]

and

\[
\tilde{x}_{k+v} = \left( \sum_{j \in [m]} x_{k+v+j(\tilde{n}-k)} q^j \right) \mod p, \quad \text{for } v \in [\tilde{n}-k].
\]

It is easy to see that each row in \( \tilde{A} \) is obtained by applying the mapping \( \lambda \) to the respective row in \( A = (A' | A'') \). Moreover,

\(^{14}\)We assume that \( \sigma = 0 \) (as in Subsection IV.C) and that there are no erasures.

**Input:** \( \ell \times k \) matrix \( A' = (a_{ij})_{i \in [\ell], j \in [k]} \) over \( \Sigma_q \).

**Output:** \( \ell \times n \) matrix \( (A' | A'') = (a_{ij})_{i \in [\ell], j \in [n]} \) over \( \Sigma_q \).

// \( F = GF(p) \), for a prime \( p > 2\vartheta \).

// \( m = [\log_q p] \) and \( n = k + m(\tilde{n} - k) \).

// \( C \) is a linear \( \tau \)-error-correcting \([\tilde{n}, k]\) code over \( F \).

// \( E : F^k \rightarrow C \) is a systematic encoder.

For all \( i \in [\ell] \) do

\[
\text{Let } \tilde{e}_i = (\tilde{e}_v)_{v \in [\tilde{n}]} \leftarrow E(A'_i \mod p);
\]

For each \( v \in [n-k] \) do

\[
\text{Set } ((A(0))_{i,v} (A(1))_{i,v} \cdots (A(m-1))_{i,v}) \text{ to be the base-}\, q\, \text{representation of } \tilde{e}_{k+v};
\]

\[
\text{Let } A'' \leftarrow (A(0) | A(1) | \cdots | A(m-1));
\]

**Fig. 7.** Encoding mapping \( E : A' \rightarrow (A' | A'') \) for the Hamming metric.

\( \lambda \) is a homomorphism in that it preserves vector addition and scalar multiplication: for every \( x_1, x_2 \in \mathbb{Z}^n \) and \( b_1, b_2 \in \mathbb{Z} \),

\[
\lambda(b_1 x_1 + b_2 x_2) = \overline{b_1} \lambda(x_1) + \overline{b_2} \lambda(x_2),
\]

where \( \overline{b_i} = b_i \mod p \) (seen as elements of \( F \)). Consequently, for every \( u \in \Sigma_q^k \),

\[
\lambda(\lambda(uA)) = uA \mod p \in C.
\]

The properties of \( \lambda(\cdot) \) immediately imply a decoding algorithm (shown in Figure 8). Given the read vector \( y = e + e' \), where \( w(e) \leq \tau \), an application of \( \lambda \) to \( y \) yields:

\[
\lambda(y) = \lambda(e) + \lambda(e'),
\]

where \( \lambda(e) \in C \) and \( w(\lambda(e)) \leq w(e) \leq \tau \). Hence, a decoder for \( C \), when applied to \( \lambda(y) \), will recover \( \lambda(e) \). By the definition of \( \lambda \), the vectors \( e \) and \( \lambda(e) \) coincide, modulo \( p \), on their \( k \)-prefix; and since the values of the entries of \( e \) are all within \( +\vartheta, \) the \( k \)-prefix of \( \lambda(e) \) uniquely determines the \( k \)-prefix of \( e \).

**Input:** \( y \in \Sigma_q^n \).

**Output:** \( w \in \Sigma_q^k \).

// Parameters are as defined in Figure 7.

// \( \lambda : \mathbb{Z}^n \rightarrow F^{\tilde{n}} \) is defined by (14)–(15).

// \( D : F^{\tilde{n}} \rightarrow F^n \) is a decoder for \( C \).

Let \( \tilde{e} = (\tilde{e}_j)_{j \in [\tilde{n}]} \leftarrow D(\lambda(y)); \)

Set \( e' = (e_j)_{j \in [k]} \) to

\[
e_j \left\{ \begin{array}{ll}
\tilde{e}_j & \text{if } \tilde{e}_j \text{ is “nonnegative” in } F \\
-\tilde{e}_j & \text{otherwise}
\end{array} \right.
\]

Let \( w \leftarrow y' - e' \).

**Fig. 8.** Decoding mapping \( D : y \rightarrow w \) for the Hamming metric.
Finally, we specialize to the case where $C$ is a (normalized and possibly shortened) BCH code. In this case,
\[
\hat{n} - k \leq \left[ 1 + \frac{p-1}{p} (2\tau-1) \right] \cdot \left[ \log_p \hat{n} \right]
\]
(see \[13\], p. 260, Problem 8.13), and so, the redundancy of our coding scheme is bounded from above by
\[
n - k \leq \left[ 1 + \frac{q-1}{q} (2\tau-1) \right] \cdot \left[ \log_q n \right] \approx \log_q n
\]
For reference, recall that for every row index $i \in [\ell]$, the possible contents of $A_i$ must form an (ordinary) code over $\Sigma_q$ of minimum Hamming distance at least $2\tau + 1$ (assuming that $\ell \geq q-1$). For $n$ sufficiently large compared to $\tau$, BCH codes over $\text{GF}(q)$ are the best codes currently known for all prime powers $q$ except 4 and 8. Hence, we should expect the redundancy of the coding scheme to be no less than
\[
\left[ 1 + \frac{q-1}{q} (2\tau-1) \right] \cdot \left[ \log_q n \right].
\]

**ACKNOWLEDGMENT**

The author would like to thank Dick Henze, Naveen Muralimanohar, and John Paul Strachan for introducing him to the problem, and for the many helpful discussions.

**APPENDIX A**

**Proofs**

*Proof of Proposition 6* For any two codewords $c_1, c_2 \in C$ and a vector $y \in \Sigma_q^n$, we have
\[
d(c_1, c_2) \leq d(y, c_1) + d(y, c_2).
\]
Hence, the inequalities
\[
d(y, c_1) \leq \tau \quad \text{and} \quad d(y, c_2) \leq \tau + \sigma
\]
can hold simultaneously, only if $d(c_1, c_2) \leq 2\tau + \sigma < d(C)$, namely, only when $c_1' = c_2'$. This, in turn, implies that the following decoding mapping is well-defined and satisfies the correction and detection conditions above: for every $y \in \Sigma_q^n$,
\[
D(y) = \begin{cases} 
  c' & \text{if there is } c \in C \text{ such that } d(y, c) \leq \tau \\
  \text{otherwise}
\end{cases}
\]

(It can be easily shown that the condition on $\tau$ and $\sigma$ in Proposition 6 is also necessary: if $2\tau + \sigma \geq d(C)$, then there can be no decoding mapping that corrects (any pattern of up to) $\tau$ errors and detects $\tau + \sigma$ errors.)

*Proof of 2* Let $c_1$ and $c_2$ be codewords in $C$ with distinct $k$-prefixes. Ignoring the coordinates that have been erased, these codewords will differ on at least $d_{H}(c_1, c_2) - \rho$ coordinates. Next apply Proposition 1 with $d(C)$ therein replaced by $d_{H}(C) - \rho$.

**APPENDIX B**

**Example**

We include here an example of an execution of the decoder in Figure 5.

*Example 6.* Continuing Example 4 for $u = (111)$, we get
\[
c = u.A = (1112031122|11212|201212).
\]
Suppose that the read vector is
\[
y = (1112021122|11222|201212).
\]
The syndrome of $y$ is computed by (10)-(12) to yield
\[
(s_1 s_2 \hat{s}_2) = (2980).
\]
A nonzero $s_1$ indicates that (one or two) errors have occurred in the $n_1$-prefix, $y_1$, of $y$, and a zero $\hat{s}_2$ then indicates that the remaining part of $y$ is error-free. Hence, we are in the scenario of case 2 in the proof of Proposition 3, i.e., $(s_1 s_2) = e_1 H T_{\text{Ber}} \mod p$ ($p = 31$), which allows us to find $e_1$ using a decoder for $C_{\text{Ber}}$. Next, we recall the principles of the decoding algorithm for $C_{\text{Ber}}$, by demonstrating them on our particular example.

We first observe that if $\|e_1\| = 1$ then necessarily $s_2 \equiv s^2_1 \mod 31$. Since this congruence does not hold in our case, we deduce that the two errors have occurred, say at positions $i, j \in [n_1]$. We have
\[
s_1 \equiv e_1 \alpha_i + e_j \alpha_j \quad (\mod p)
\]
\[
s_2 \equiv e_1 \alpha_i^3 + e_j \alpha_j^3 \quad (\mod p),
\]
where $e_i, e_j \in \{-1, 1\}$. Squaring both sides of (16) yields
\[
s^2_1 \equiv \alpha_i^2 + 2(e_i \alpha_i)(e_j \alpha_j) + \alpha_j^2 \quad (\mod p),
\]
while dividing each side of (17) by the respective side in (16) yields
\[
\frac{s^2_2}{s_1} \equiv \alpha_i^2 - (e_i \alpha_i)(e_j \alpha_j) + \alpha_j^2 \quad (\mod p),
\]
where $1/s_1$ stands for the inverse of $s_1$ modulo $p$. Subtracting each side of the last congruence from the respective sides of the previous congruence leads to
\[
s^2_1 - \frac{s^2_2}{s_1} \equiv 3(e_i \alpha_i)(e_j \alpha_j) \quad (\mod p).
\]
It follows from (16) and (18) that $e_i \alpha_i$ and $e_j \alpha_j$ are solutions of the following quadratic equation (in $F = \text{GF}(p)$):
\[
x^2 - s_1 x + \frac{1}{3} \left( s^2_1 - \frac{s^2_2}{s_1} \right) \equiv 0 \quad (\mod p).
\]
Specifically, in our case,
\[
\frac{1}{s_1} = \frac{1}{29} \equiv 15 \mod 31 \quad \text{and} \quad \frac{1}{3} \equiv 21 \mod 31,
\]
resulting in the quadratic equation
\[
x^2 + 2x + 13 \equiv 0 \mod 31.
\]
The two roots of this equation in $\text{GF}(31)$ are 8 and 21: the first points at an error with a value 1 at location 13 (since $\alpha_{13} = 8$), and the second points at an error with a value $-1$ at location 5 (since $\alpha_5 = 10 = -21 \mod 31$).
REFERENCES

[1] E.R. Berlekamp, *Algebraic Coding Theory*, Revised Edition, Aegean Park Press, Laguna Hills, California, 1984.

[2] W.G. Bliss, “Circuitry for performing error correction calculations on baseband encoded data to eliminate error propagation,” *IBM Tech. Discl. Bull.*, 23 (1981), 4633–4634.

[3] B.E. Boser, E. Sackinger, J. Bromley, Y. Le Cun, L.D. Jackel, “An Analog neural network processor with programmable topology,” *IEEE J. Solid-State Circuits*, 26 (1991), 2017-2025.

[4] J. Fan, R. Calderbank, “A modified concatenated coding scheme, with applications to magnetic data storage,” *IEEE Trans. Inf. Theory*, 44 (1998), 1565–1574.

[5] S.W. Golomb, L.R. Welch, “Algebraic coding and the Lee metric,” in *Error Correcting Codes*, H.B. Mann (Editor), Wiley, New York, 1968, pp. 175–194.

[6] S.W. Golomb, L.R. Welch, “Perfect codes in the Lee metric and the packing of polyominoes,” *SIAM J. Appl. Math.*, 18 (1970), 302–317.

[7] A.F. Hoaradam, “Jacobsthal representation numbers,” *Fibonacci Quart.*, 34 (1996), 40–54.

[8] M. Hu, J.P. Strachan, Z. Li, E.M. Grafals, N. Davila, C. Graves, S. Lam, N. Ge, J. Yang, R.S. Williams, “Dot-product engine for neuromorphic computing: Programming ITIM Crossbar to accelerate matrix-vector multiplication,” *Proc. 53rd Annual Design Automation Conference (DAC’16)*, Austin, Texas (2016), Article No. 19.

[9] K.A.S. Immink, “A practical method for approaching the channel capacity of constrained channels,” *IEEE Trans. Inf. Theory*, 43 (1997), 1389–1399.

[10] D.E. Knuth, “Efficient balanced codes,” *IEEE Trans. Inf. Theory*, 32 (1986), 51–53.

[11] F. Kub, K. Moon, I. Mack, F. Long, “Programmable analog vector–matrix multipliers,” *IEEE J. Solid-State Circuits*, 25 (1990), 207-214.

[12] M. Mansuripur, “Enumerative modulation coding with arbitrary constraints and post-modulation error correction coding and data storage systems,” *Proc. SPIE*, Vol. 1499 (1991), 72–86.

[13] R.M. Roth, *Introduction to Coding Theory*, Cambridge University Press, Cambridge, UK, 2006.

[14] A. Shafiee, A. Nag, N. Muralimanohar, R. Balasubramonian, J.P. Strachan, M. Hu, R.S. Williams, V. Srikumar, “ISAAC: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars,” *Proc. 43rd ACM/IEEE Int'l Symp. on Computer Architecture (ISCA 2016)*, Seoul, Korea (2016), pp. 14–26.