Dynamic Threshold Schemes for Multi-Level Non-Volatile Memories

Frederic Sala, *Student Member, IEEE*, Ryan Gabrys, *Student Member, IEEE*, and Lara Dolecek, *Senior Member, IEEE*

Abstract—**In non-volatile memories, reading stored data is typically done through the use of predetermined fixed thresholds. However, due to problems commonly affecting such memories, including voltage drift, overwriting, and inter-cell coupling, fixed threshold usage often results in significant asymmetric errors. To combat these problems, Zhou, Jiang, and Bruck recently introduced the notion of dynamic thresholds and applied them to the reading of binary sequences.**

In this paper, we explore the use of dynamic thresholds for multi-level cell (MLC) memories. We provide a general scheme to compute and apply dynamic thresholds and derive performance bounds. We show that the proposed scheme compares favorably with the optimal thresholding scheme. Finally, we develop limited-magnitude error-correcting codes tailored to take advantage of dynamic thresholds.

Index Terms—**Non-binary error correction codes, non-volatile memory, dynamic reading thresholds, sequences, permutations.**

I. Introduction

RECENTLY, non-volatile memories (NVM) including Flash, and alternatives such as electrically erasable programmable read-only memory (EEPROM) and phase-change memories (PCM) have become among the most popular and promising data storage technologies. NVMs are commonly used in every type of device and, in particular, are ubiquitous in mobile storage. The defining feature of NVMs is their ability to retain data without requiring a power supply. NVMs have many additional advantages over traditional magnetic storage technologies. NVMs experience low failure rates due to their lack of mechanical parts. They also exhibit very fast seek times, often an order of magnitude faster in comparison to those of older storage technologies [1].

NVMs are made up of large arrays of cells, divided into blocks of size between 2^{18} and 2^{20} cells [1, Chapter 3]. Each cell stores one or more bits of information, represented by the amount of charge trapped in the cell. In the industry standard single-level cell (SLC) technology, one bit of information is stored per cell. In order to increase the capacity of NVMs, multi-level cell (MLC) memories storing multiple bits per cell have recently been introduced. For example, triple-level cell (TLC) Flash memories store three bits per cell.

To read the value of a cell, its voltage is measured and this value is typically compared to predetermined fixed thresholds. In SLC memories, there is a single threshold v so that a cell value is read as a 0 if its voltage is determined to be below v and as a 1 otherwise[1]. Unfortunately, this approach is prone to errors due to the physical properties of NVMs. Over time, voltage distributions tend to widen, expanding beyond the intervals between thresholds. As a result, the distributions overlap and form a potential source of error. This effect, known as charge leakage, is particularly problematic with dense MLCs, where intervals between thresholds are increasingly small. Other errors, such as those caused by overwriting cell values or parasitic inter-cell coupling, also negatively impact NVM technologies [2].

Additionally, NVMs exhibit asymmetry in the writing process. It is possible to write (that is, write a 1 to) a single cell by injecting electrons into it. However, in order to delete (write a 0 to) a cell, an entire block of cells must be deleted [1]. Furthermore, the lifetime of an NVM allows for only a limited number of erase cycles, typically around 10^5 [3].

Most coding solutions have focused on this inherent writing asymmetry in NVM cells. In particular, the *rank modulation* scheme was developed to alleviate this problem. In rank modulation, information is not represented by the charge level in a single cell, but rather by the relative charge levels of many cells [4]. Recently, the problem of creating codes for rank modulation has been studied in [5], [6], and [7].

To resolve other NVM problems such as charge leakage and inter-cell coupling, numerous coding solutions have been proposed. In [8], codes were developed to correct asymmetric limited-magnitude errors, which affect NVMs. General asymmetric error correction constructions were provided in [9], [10], [11], [12], and [13]. Constrained codes for Flash memories, studied in [14], were shown to be effective against inter-cell coupling. Additional NVM code constructions were provided in [15], [16], and [17]. Application of LDPC codes to Flash memories were explored in [18] and [19]. Solutions focused specifically on PCMs were studied in [20].

An entirely different approach to combating NVM problems (particularly charge leakage) was introduced in [21]. In this work, Zhou, Jiang, and Bruck introduced the notion of *dynamic thresholds*. With this approach, there are no

Manuscript received September 27, 2012; revised February 19, 2013. The editor coordinating the review of this paper and approving it for publication was D. Declercq.

Research supported in part by the SMART Scholarship, NSF grants CCF-1029030, CCF-1150212, and the NSF GRFP. This paper has been presented in part at the IEEE Asilomar Conference on Signals, Systems and Computers, Pacific Grove, CA, Nov. 2012.

The authors are with the Department of Electrical Engineering, University of California Los Angeles, Los Angeles, CA 90095, USA (e-mail: {fredsala, rgabrys}@ucla.edu, dolecek@ee.ucla.edu).

Digital Object Identifier 10.1109/TCOMM.2013.053013.120733

[1]Note that in practical implementations, a 0 is read if the voltage is *above* the threshold, and a 1 is read otherwise.

predetermined thresholds used for comparison when reading cells. Instead, a new set of thresholds is generated each time a cell is read, based on specific information known to the cell decoder. This technique was shown to be highly effective against errors caused by voltage drift.

The results in [21] were derived for binary sequences used in the single-level cell (SLC) case. In this work, we apply the notion of dynamic thresholds to non-binary sequences, making the technique applicable to MLC memories. We demonstrate that the advantages of dynamic thresholding are preserved in the MLC case. We propose specific implementations of dynamic thresholding for MLC NVMs. We then proceed to explore a more general problem: combining dynamic thresholding with error-correcting codes in order to guarantee the correction of very common errors.

The remainder of this paper is organized as follows. We introduce the necessary notation and preliminaries in Section II. We analyze the performance of our choice of dynamic thresholding scheme in Section III, where we show that our scheme compares favorably to an ideal (optimal) scheme. Section IV covers dynamic threshold implementation methods. We focus on two approaches: the use of metadata and the notion of balanced codes. In Section V, we construct codes that, when combined with dynamic thresholding, are capable of correcting up to t errors of limited magnitude l. We develop constructions for both the metadata and balanced code implementations. In Section VI, we construct dynamic thresholding codes that correct any number of limited-magnitude errors. We summarize our contributions in Section VII.

II. Preliminaries

We first introduce the necessary notation. Assume that each cell can take on q levels from the set $F = \{0, 1, 2, \dots, q-1\}$. Let $\mathbf{x} = (x_1, x_2, \dots, x_n) \in F^n$ be the word written into a block of cells of length n, and let $\mathbf{v}(\mathbf{x}) = (v_1, v_2, \dots, v_n)$ represent cell levels after writing $\mathbf{x}$. Define $\mathbf{k}(\mathbf{x}) = (k_0, k_1, \dots, k_{q-1})$ to be a vector such that k_a of the components of $\mathbf{x}$ are at level a for $0 \le a \le q-1$. It is clear that $\sum_{a=0}^{q-1} k_a = n$.

We define a **threshold vector** $\mathbf{t} = (t_1, t_2, \dots, t_{q-1})$ in the following way: Read the word $\mathbf{y}(\mathbf{t}, \mathbf{v}(\mathbf{x})) = (y_1, y_2, \dots, y_n)$ such that $y_i = m$ if $t_m \le v_i < t_{m+1}$, for $0 \le m \le q-1$ and $1 \le i \le n$, where $t_0 = -\infty$ and $t_q = +\infty$. When it is clear which $\mathbf{x}$ and $\mathbf{v}(\mathbf{x})$ we are referring to from the context, we remove them from the expression for $\mathbf{y}$ and write $\mathbf{y}(\mathbf{t})$ or simply $\mathbf{y}$.

Let $N(\mathbf{x}, \mathbf{y})$ be the Hamming distance between $\mathbf{x}$ and $\mathbf{y}$. In other words, $N(\mathbf{x}, \mathbf{y})$ is the number of positions i, $1 \le i \le n$, such that $y_i \ne x_i$. For example, if $\mathbf{x} = (1, 0, 2, 2)$ and $\mathbf{y} = (1, 1, 2, 0)$, $N(\mathbf{x}, \mathbf{y}) = 2$. We often compare the performance of different $\mathbf{t}$'s for a particular block $\mathbf{x}$ and corresponding $\mathbf{y}$. When we do so, we drop the $\mathbf{x}$ and $\mathbf{y}$ from the expression for distance and write $N(\mathbf{x}, \mathbf{y})$ as $N(\mathbf{t})$.

A **dynamic threshold** $\mathbf{t}^d$ is any threshold vector such that $\mathbf{k}(\mathbf{x}) = \mathbf{k}(\mathbf{y}(\mathbf{t}^d))$. That is, $\mathbf{t}^d$ is such that the number of components at each level is preserved from $\mathbf{x}$ to $\mathbf{y}(\mathbf{t}^d)$. Finally, if there exists an entry i in $\mathbf{x}$ such that $x_i = a$ and $y_i = b$ with $a \ne b$, we say that an $a \to b$ error has occurred. The number of entries i where $a \to b$ errors occur is denoted $N^{a,b}(\mathbf{t})$, so that $\sum_{a \ne b} N^{a,b}(\mathbf{t}) = N(\mathbf{t})$.

We write $[a, b]$ for the set $\{a, a+1, \dots, b\}$ and $[n]$ for the set $\{1, 2, \dots, n\}$. Finally, we will use the words "transmission" and "storage" interchangeably.

We illustrate these notions with an example. Take $n = 5$, $q = 3$, and $\mathbf{x} = (1, 0, 2, 2, 0)$. After some time, the cell voltages are given by $\mathbf{v}(\mathbf{x}) = (1.6, 0.3, 2.3, 1.7, 0.7)$. We have that $\mathbf{k}(\mathbf{x}) = (2, 1, 2)$.

Consider the threshold vectors $\mathbf{t}_1 = (0.5, 1.5)$ and $\mathbf{t}_2 = (0.8, 1.65)$. Reading $\mathbf{v}$ with $\mathbf{t}_1$ we have $\mathbf{y}_1 = (2, 0, 2, 2, 1)$ and $\mathbf{k}(\mathbf{y}_1) = (1, 1, 3)$. Then $\mathbf{k}(\mathbf{y}_1) \ne \mathbf{k}(\mathbf{x})$ and $\mathbf{t}_1$ is *not* a dynamic threshold. However, $\mathbf{y}_2(\mathbf{t}_2) = (1, 0, 2, 2, 0)$, and $\mathbf{k}(\mathbf{y}_2) = (2, 1, 2) = \mathbf{k}(\mathbf{x})$ so $\mathbf{t}_2$ is a dynamic threshold.

III. Performance of Dynamic Thresholds

In [21], it was shown that for binary sequences, dynamic thresholds are particularly effective (when compared to fixed thresholds) when the cell levels, modeled by Gaussian distributions, have variance increasing with time. The same result holds in the non-binary case.

A simple example of this scenario is depicted in Fig. 1. Here, a block contains only two cells x_1 and x_2 with true values (that is, the values originally written to x_1 and x_2) a and $a+1$, respectively. The distributions v_1 and v_2 of these two cells are shown immediately after writing and after some time has passed. The distributions are modeled as independent Gaussians. These Gaussians have the same variance initially, but after a period of time, the variance of v_1 increases, and its mean is shifted. The fixed threshold t_1 cannot adapt to these changes, leading to a source of error. On the other hand, t_2 is shifted to minimize the error probability. This observation provides the fundamental motivation for our use of dynamic thresholding schemes.

We illustrate the strength of dynamic thresholds with the following sample analysis. For q-level cells, we consider the fixed threshold $\mathbf{t} = (\frac{1}{2}, \frac{3}{2}, \dots, \frac{2q-3}{2})$. The voltage written into cell h, v_h $(1 \le h \le n)$, is read as a if $a - \frac{1}{2} \le v_h < a + \frac{1}{2}$. For simplicity, in this example we model the quantities v_h as (independent) Gaussians with identical variance: $v_h \sim \mathcal{N}(a, \sigma^2)$, where a is the original value written to x_h.

In order to keep our analysis tractable, we take $n = 2$, so that the block $\mathbf{x}$ has only two cells. Let x_1 and x_2 have values a and $a+1$, respectively, so that $0 < a < q-2$. If we use the threshold $\mathbf{t}$ to read the word, errors occur when the voltage values v_1 and v_2 fall outside of the ath and $(a+1)$st threshold intervals, respectively. That is, errors occur when $v_1 < a - \frac{1}{2}$, $v_1 \ge a + \frac{1}{2}$, $v_2 < (a+1) - \frac{1}{2}$, or $v_2 \ge (a+1) + \frac{1}{2}$. Thus,

$$\begin{aligned}\Pr\{\text{error}|\mathbf{t}\} = \; & \Pr\left\{ v_1 < a - \frac{1}{2} \cup v_1 \ge a + \frac{1}{2} \cup \right. \\ & \left. v_2 < (a+1) - \frac{1}{2} \cup v_2 \ge (a+1) + \frac{1}{2} \right\} \\ = \; & 2\left(1 - \Phi\left(\frac{1/2}{\sigma}\right) + \Phi\left(\frac{-1/2}{\sigma}\right)\right),\end{aligned}$$

where $\Phi(x)$ is the Gaussian c.d.f. $\frac{1}{\sqrt{2\pi\sigma^2}} \int_{-\infty}^{x} \exp(\frac{(z-\mu)^2}{-2\sigma^2}) dz$.

Instead, if we employ a dynamic threshold $\mathbf{t}^d$, errors take place when $v_1 \ge v_2$, since if $v_1 < v_2$, the ath component of

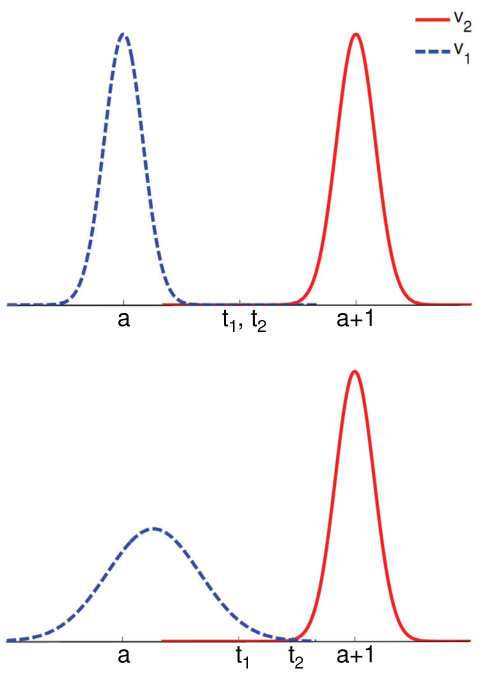


Fig. 1. Distributions v_1 and v_2 of cells x_1 and x_2 with true values a and $a+1$, shown immediately after writing and after some time. Initially, $v_1 \sim \mathcal{N}(a, \sigma^2)$ and $v_2 \sim \mathcal{N}(a+1, \sigma^2)$. After a period of time, v_1 has expanded and its mean has shifted. Note the fixed threshold t_1 and the dynamic threshold t_2. Here, t_1 cannot react to the change in v_1, but t_2 does.

the threshold vector $\mathbf{t}^d$, t_a, will be placed between v_1 and v_2, and x_1 and x_2 will be read correctly. Then,

$$\Pr\left\{\text{error}|\mathbf{t}^d\right\} = \Pr\left\{v_i \geq v_j\right\}$$
$$= \int_{-\infty}^{\infty} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(z-(a-1/2))^2}{2\sigma^2}} \Phi\left(\frac{z-(a+1/2)}{\sigma}\right) dz.$$

This quantity is much smaller than the fixed threshold probability. For example, if $\sigma = 0.25$, the fixed threshold error probability is ≈ 0.09 and the dynamic threshold error probability is ≈ 0.0023. The error probability as a function of the deviation is shown in Fig. 2.

Next, we compare our choice of thresholding scheme with the optimal scheme. Define $\mathbf{t}^* = \arg\min_{\mathbf{t}} N(\mathbf{t})$. We note that computing $\mathbf{t}^*$ requires the knowledge of the initial word $\mathbf{x}$, so the decoder cannot use the optimal threshold. Even so, the dynamic threshold $\mathbf{t}^d$ performs worse than the optimal threshold at most by a fixed factor (which depends on the number of cell levels q and the maximum error magnitude.) In [21], it was shown that this factor is 2 when $q = 2$, such that $N(\mathbf{t}^d) \leq 2N(\mathbf{t}^*)$. We derive general bounds for multi-level cells.

We define l to be the maximum error magnitude under the dynamic thresholding approach. In general, magnitude limitations model physical limits on the deviation of the v_i's. However, to keep our analysis tractable, we impose the limitation on the decoded output: for the limited magnitude l, $|y_i - x_i| \leq l$ for $1 \leq i \leq n$. This is a reasonable choice: in order for cell i with true value a to experience an error of magnitude k under dynamic thresholding, its voltage v_i must be larger than the voltages of all cells with values a, $a+1, \ldots,$ $a+k-1$ or smaller than the voltages of all cells with values

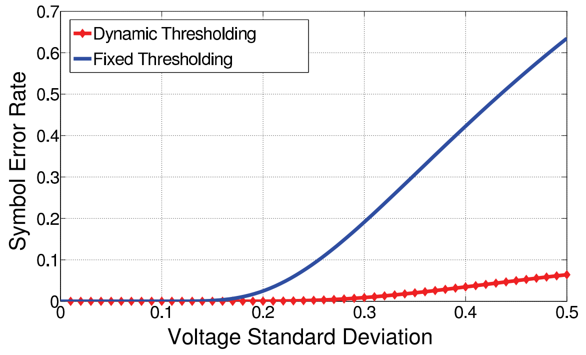


Fig. 2. Symbol error rate for a pair of cells with adjacent values a and $a+1$ as a function of σ for fixed threshold reading versus dynamic threshold reading.

$a, a-1, \ldots, a-k+1$. Thus k must be very small. We begin our analysis with the case $l = 1$, which is of practical interest due to the frequency of such errors in non-volatile memories.

Our approach is to find error patterns that cause a fixed number of errors when using dynamic thresholding and to lower bound the number of errors caused by these patterns when using the optimal threshold. Recall that the number of errors from cell value a to cell value b when reading using threshold $\mathbf{t}$ is denoted by $N^{a,b}(\mathbf{t})$. When $l = 1$, we note that $N^{a,b}(\mathbf{t}^d) = N^{b,a}(\mathbf{t}^d)$. This is easy to establish through an inductive argument: $N^{a,b}(\mathbf{t}^d) = 0$ if $|a-b| > 1$, so it is enough to show that $N^{a,a+1}(\mathbf{t}^d) = N^{a+1,a}(\mathbf{t}^d)$. Take $a = 0$ for the base case. It is clear that (under dynamic thresholding) $N^{0,1}(\mathbf{t}^d) = N^{1,0}(\mathbf{t}^d)$, since the only possible errors involving level 0 are $0 \to 1$ and $1 \to 0$. Assume that $N^{k+1,k}(\mathbf{t}^d) = N^{k,k+1}(\mathbf{t}^d)$. Now, the number of cells at level $k+1$ must not change, so the number of cells transitioning away from $k+1$ must be equal to the number of cells transitioning to $k+1$: $N^{k+1,k}(\mathbf{t}^d) + N^{k+1,k+2}(\mathbf{t}^d) = N^{k,k+1}(\mathbf{t}^d) + N^{k+2,k+1}(\mathbf{t}^d)$. Subtracting the induction hypothesis, we have that $N^{k+1,k+2}(\mathbf{t}^d) = N^{k+2,k+1}(\mathbf{t}^d)$, as desired. This fact leads us to the following theorem:

Theorem 1. *For any cell levels $v_1, v_2, \ldots, v_n$, with errors of magnitude limited to $l = 1$, and any dynamic threshold $\mathbf{t}^d$ and optimal threshold $\mathbf{t}^*$,*

$$N(\mathbf{t}^d) \leq 2N(\mathbf{t}^*).$$

Proof: From the above result, when using dynamic thresholds, for every error $a \to a+1$, there is a corresponding $a+1 \to a$ error. For such errors, there exists a cell $x_i = a$ with $v_i \geq t_a$ and a cell $x_j = a+1$ with $v_j < t_a$. (Here, t_a refers to the ath component of $\mathbf{t}^d$.) Thus, $v_j < v_i$. We have the following (disjoint) possibilities for t_a^* :

$$\begin{aligned} t_a^* &\leq v_j < v_i, \text{ or} \\ v_j &< t_a^* \leq v_i, \text{ or} \\ v_j &< v_i < t_a^*. \end{aligned}$$

The first case will result in an $a \to a+1$ error, the second in both $a \to a+1$ and $a+1 \to a$ errors, and the third in an $a+1 \to a$ error. Thus, for each pair of errors in the scheme using $\mathbf{t}^d$, there is at least one error when using $\mathbf{t}^*$, so that $N(\mathbf{t}^d) \leq 2N(\mathbf{t}^*)$. ■

At worst, the use of dynamic thresholds results in twice as many errors as the optimal threshold. We see that (when $l = 1$) the dynamic threshold scheme works as well for $q > 2$ as in the binary case. Notice the following argument: if there exists a pair of cells with levels a_1, a_2 with $a_1 < a_2$ and they suffer errors $a_1 \to b_1$ and $a_2 \to b_2$ where $b_1 > b_2$, the optimal threshold scheme will result in at least one error. The case $a_2 = a_1 + 1$ gives the above proof. We exploit this notion to derive a similar bound in the more general case where $1 < l \leq q$.

By definition, dynamic thresholds do not change the distribution of the values of cells in the stored word. The received sequence is then a multiset permutation of the original word. Every multiset permutation can be expressed as a product of cycles [22]. Then, any error $a \to b$ is part of a cycle of k errors $a_1 \to a_2 \to a_3 \to \ldots \to a_k \to a_1$, where $2 \leq k \leq q$. That is, there exist cells at positions $s_1, s_2, \ldots, s_k$ with levels $a_1, a_2, \ldots, a_k$, respectively, such that cell x_{s_i} suffers an error $a_i \to a_{i+1}$ if $1 \leq i < k$ and cell x_{s_k} suffers an error $a_k \to a_1$.

We illustrate this idea with an example for $n = 3$, $q = 4$, and $l = 2$. Let $(2, 1, 3)$ be written into the cells (x_1, x_2, x_3). After some time, the voltages representing the cells are determined to be $(v_1, v_2, v_3) = (2.4, 1.9, 1.8)$. The decoder knows that there was one cell at level 1, one at level 2, and one at level 3. To preserve the number of cells at these levels, it must pick dynamic thresholds satisfying $1.8 \leq t_1 < 1.9$, $1.9 \leq t_2 < 2.4$, and $2.4 \leq t_3$. Using these thresholds to read the v_i's results in $\mathbf{y} = (3, 2, 1)$, a permutation of the original sequence $(2, 1, 3)$. The errors are given by the cycle $1 \to 2 \to 3 \to 1$.

We are now ready to prove the general bound:

Theorem 2. *For any cell levels $v_1, v_2, \ldots, v_n$, with errors of magnitude limited to $l > 1$,*

$$N(\mathbf{t}^d) \leq (l + 1)N(\mathbf{t}^*).$$

Proof: Any error must be part of a cycle. Consider the cycle $a_1 \to a_2 \to a_3 \to \ldots \to a_k \to a_1$ of length $l < k \leq q$. Let $a_{\min} = \min\{a_1, a_2, \ldots, a_k\}$ and $a_{\max} = \max\{a_1, a_2, \ldots, a_k\}$. Since the a_i's are distinct, $a_{\max} \geq a_{\min} + (k-1)$. Now, split up the interval $[a_{\min}, a_{\max}]$ into $\lceil \frac{k-1}{l} \rceil + 1$ intervals (where each interval, other than the first, has length l) in the following way:

$$[a_{\min}, a_{\min}], [a_{\min} + 1, a_{\min} + l], [a_{\min} + l + 1, a_{\min} + 2l], \\ \ldots, \left[a_{\min} + \left(\left\lfloor \frac{k-2}{l} \right\rfloor\right) l + 1, a_{\max}\right].$$

Note that there are $\lfloor \frac{k-2}{l} \rfloor + 1$ intervals of length l, and, including the first interval $[a_{\min}, a_{\min}]$, there are a total of $\lfloor \frac{k-2}{l} \rfloor + 2 = \lceil \frac{k-1}{l} \rceil + 1$ intervals, as desired.

Since there is a path from $a_{\min}$ to $a_{\max}$, there exists a_i in the fth interval and a_j in the $(f+1)$st interval such that $a_i \to a_j$. If this was not the case and some interval was skipped, an error would be larger than the interval length of l, which is not possible. Similarly, as there is a path from $a_{\max}$ to $a_{\min}$, the cycle contains corresponding descending errors. That is, there exists $a_{i'}$ in the $(f + 1)$st interval and $a_{j'}$ in the fth interval such that $a_{i'} \to a_{j'}$. Now we use the observation derived in the earlier result. We have that initially there exist cells with values a_i and $a_{i'}$, respectively, such that $a_i < a_{i'}$. These cells are affected by errors of type $a_i \to a_j$ and $a_{i'} \to a_{j'}$ with $a_j > a_{j'}$. The optimal threshold results in one error for each such occurrence, as shown in the proof of Theorem 1.

The cells producing the error above were in the fth and $(f + 1)$st interval. There is at least one such error for each of the $\lceil \frac{k-1}{l} \rceil$ pairs of adjacent intervals. Thus, for an error cycle $\mathbf{h}$ of length k, usage of the optimal threshold results in at least $\lceil \frac{k-1}{l} \rceil$ errors. Call the number of errors caused by this cycle $N^{\mathbf{h}}(\mathbf{t})$. Then, $N^{\mathbf{h}}(\mathbf{t}^*) \geq \lceil \frac{k-1}{l} \rceil$, while $N^{\mathbf{h}}(\mathbf{t}^d) = k$, so that $N^{\mathbf{h}}(\mathbf{t}^d) \leq \frac{k}{\lceil \frac{k-1}{l} \rceil} N^{\mathbf{h}}(\mathbf{t}^*)$.

Now, for $l \geq 1$, we have that

$$\frac{k}{\lceil \frac{k-1}{l} \rceil} \leq l + 1,$$

and summing over all error cycles $\mathbf{h}$, we have $N(\mathbf{t}^d) = \sum_{\mathbf{h}} N^{\mathbf{h}}(\mathbf{t}^d) \leq \sum_{\mathbf{h}} \frac{k}{\lceil \frac{k-1}{l} \rceil} N^{\mathbf{h}}(\mathbf{t}^*) \leq (l + 1) \sum_{\mathbf{h}} N^{\mathbf{h}}(\mathbf{t}^*) = (l + 1)N(\mathbf{t}^*)$.

In the above analysis, we assumed that the cycle $\mathbf{h}$ has length $k > l$. If instead $k \leq l$, there will still be at least two cells whose values are exchanged relative to a threshold, as in our standard argument. Then, as before, using the optimal threshold will result in at least one error. Dynamic thresholds result in k errors for a cycle of length k, so $N^{\mathbf{h}}(\mathbf{t}^d) \leq kN^{\mathbf{h}}(\mathbf{t}^*) \leq lN^{\mathbf{h}}(\mathbf{t}^*) \leq (l + 1)N^{\mathbf{h}}(\mathbf{t}^*)$, and our result still holds. ■

We conclude that, even in the general case, the performance of our scheme is comparable to that of the optimal scheme, which cannot be used without the decoder already knowing the original codeword.

IV. Implementation

In order to find the dynamic thresholds for each codeword, the decoder needs to know the number of cells at each level, given by $\mathbf{k} = (k_0, k_1, \ldots, k_{q-1})$. We give several methods of accomplishing this task. These were introduced by Zhou *et al.* in [21] for the SLC case. We generalize them for MLC memories and suggest some improvements.

First, we may have the encoder append the base-q representation of each of the k_i's to a transmitted codeword of length n. It is sufficient to include k_i for $i < q-1$. Then, as $0 \leq k_i \leq n$, the size of this metadata is $(q - 1)\lceil \log_q(n + 1) \rceil$. The metadata will be used to generate the dynamic thresholds, so it must be decoded using a traditional fixed threshold. Since it is crucial that the k_i's are reproduced exactly, the metadata should be protected by a strong error-correcting code.

To compute a dynamic threshold for cell levels $\widetilde{v}_1, \widetilde{v}_2, \ldots, \widetilde{v}_n$, we begin by sorting the $\widetilde{v}_i$'s. Let $v_1 \leq v_2 \leq \ldots \leq v_n$ be the sorted sequence. There must be k_0 cells at level 0, so the threshold t_0 must be larger than the first k_0 values and smaller than the $(k_0 + 1)$th value. That is, t_0 must satisfy $v_{k_0} < t_0 < v_{k_0+1}$. Similarly, there are k_1 cells at level 1, so that t_1 must be larger than the first $k_0 + k_1$ values, and $v_{k_0+k_1} < t_1 < v_{k_0+k_1+1}$. Thus, it is natural to select the threshold

$$\mathbf{t}^d = \left(\frac{1}{2}\left(v_{k_0} + v_{k_0+1}\right), \frac{1}{2}\left(v_{k_0+k_1} + v_{k_0+k_1+1}\right), \\ \ldots, \frac{1}{2}\left(v_{k_0+k_1+\ldots+k_{q-2}} + v_{k_0+k_1+\ldots+k_{q-2}+1}\right)\right).$$

This sorting procedure has complexity $O(n \log n)$ in both the average-case and the worst-case. The process does not depend on q, as the sequence only needs to be sorted a single time.

As seen above, it is in fact only necessary to select the k_0th, (k_0+1)st, (k_0+k_1)st, ..., $(k_0+k_1+\ldots+k_{q-2}+1)$st smallest elements, for a total of $2q-2$ selections. There exist algorithms to find the kth order statistic with average- and worst-case runtime $O(n)$, such as the BFPRT algorithm in [23]. Thus, we can find a threshold $\mathbf{t}^d$ with worst-case complexity $O(2qn)$. If q is small compared to $\log n$, this algorithm is more efficient than sorting the list. If $q = 2$, this procedure is better than the proposed algorithm in [21], which has worst-case complexity $O(\gamma n)$ for a constant γ.

Above, we allowed any possible distribution of levels in our codewords, so that this distribution had to be included as metadata for each transmitted codeword. However, there is another approach. If we *fix* the distribution of levels beforehand, and restrict transmission to codewords having this distribution, we forego the need for metadata. The decoder will simply be informed beforehand of the chosen distribution. In order to maximize the size of our codebook, we should pick a "balanced" distribution, where the numbers of cells at each level are as close as possible. For example, if we have q levels and codewords of length $n = eq$, we should have e cells at levels $0, 1, \ldots, q-1$, respectively. This approach is referred to as a **balanced code.**

Since we need not send metadata, we may save $(q-1)\lceil \log_q(n+1) \rceil$ digits. The downside of this approach is that the codebook is smaller and that it is more difficult to construct error-correcting codes over the space of balanced codewords. There is also added complexity in encoding and decoding balanced codes. The binary case of this problem has been studied extensively. It was introduced by Knuth in his seminal paper [24]. Less is known about the non-binary case; generalizations of balanced codes were recently studied in [25], while more general constructions for q-ary constant weight codes were provided in [26].

Though dynamic thresholds reduce error probabilities, we may also wish to guarantee the correction of common errors. In the following two sections, we develop codes that take advantage of the properties of the dynamic thresholding scheme. When we rely on the metadata implementation approach, for the sake of simplicity, we abstract the metadata by assuming that it has been made available to the decoder with no error.

V. (t,l) Dynamic Thresholding ECCs

In general, we could combine dynamic thresholding with existing codes. However, such codes are not specifically tailored to the conditions which occur under dynamic thresholding. We seek codes that fully take advantage of such conditions. We begin by exploring codes that correct t errors of magnitude limited to l, where $1 \leq l \leq q$, $1 \leq t \leq \rho$, and ρ is a constant. We refer to such errors as (t,l)**-dynamic thresholding errors**, or (t,l)-DT errors.

We will see that the construction of (t,l)-DT error-correction codes is connected to several known coding theoretic problems. If our dynamic thresholding implementation uses the metadata approach, the resulting problem is similar to location correction. On the other hand, if we select the balanced codes implementation, the problem resembles coding for rank modulation.

A. Decomposability Distance

As described above, we will find that correcting (t,l)-DT errors is closely related to the problem of location correction, studied by Roth and Seroussi in [27]. In the location correction problem, the decoder is provided with the magnitudes of the errors, but does not know their locations. Our problem differs: the decoder knows the received sequence is a permutation of the original codeword. This fact only provides some information about the error magnitudes. Despite this, it will be shown the two problems are nearly equivalent.

Roth and Seroussi introduced the notion of *decomposability distance* [27]. We present a variant of it: A vector $\mathbf{x} \in F^n$ is (t,l)**-decomposable** if x_i has magnitude limited to l[1] for $1 \leq i \leq n$ and there are two vectors $\mathbf{y}$ and $\mathbf{z}$ in F^n with the same multiset of (up to) t nonzero values such that $\mathbf{x} = \mathbf{y} - \mathbf{z}$. (All operations are taken over the ring of integers modulo q.) We define the unit vector $\mathbf{u}_i$ as the vector with a 1 in the ith position and zeros elsewhere. If $i \neq j$, we let $\mathbf{u}_{i,j} = \mathbf{u}_i - \mathbf{u}_j$. Denote by $W_F(n)$ the set of all vectors $\mathbf{u}_{i,j} \in F^n$. The following lemma is equivalent to a result in [27]:

Lemma 1. *A vector $\mathbf{x} \in F^n$ is (t,l)-decomposable if and only if its components are limited to magnitude l and it can be written as a linear combination of t elements of $W_F(n)$. That is, there exist t elements $E_k \in F^n$ (with $1 \leq k \leq t$) and respective vectors $\mathbf{u}_{i_k,j_k} \in W_F(n)$ such that $\mathbf{x} = \sum_{k=1}^{t} E_k \mathbf{u}_{i_k,j_k}$.*

For fixed l, the **decomposability weight** of $\mathbf{x} \in F^n$ is the smallest non-negative integer t (if any such t exists) such that $\mathbf{x}$ is (t,l)-decomposable. If no such t exists, the decomposability weight is defined to be $n+1$. The **decomposability distance** between $\mathbf{x}$ and $\mathbf{y}$ in F^n is defined as the decomposability weight of $\mathbf{x} - \mathbf{y}$.

We illustrate these concepts with an example. If we take $\mathbf{x} = (1,2,3,4)$, $\mathbf{y} = (1,4,3,2)$, and $l = 2$, then $\mathbf{x} - \mathbf{y} = (0,-2,0,2) = -2\mathbf{u}_{2,4}$, so that $\mathbf{x} - \mathbf{y}$ is $(1,2)$-decomposable. Thus $\mathbf{x} - \mathbf{y}$ has decomposability weight 1, and $\mathbf{x}$ and $\mathbf{y}$ are at decomposability distance 1. On the other hand, if we had taken $l = 1$, $\mathbf{x} - \mathbf{y}$ is not $(t,1)$-decomposable for any t, so that the decomposability weight of $\mathbf{x} - \mathbf{y}$ is $n+1 = 5$, and $\mathbf{x}$ and $\mathbf{y}$ are at decomposability distance 5.

In our problem, the received vector $\mathbf{y}$ is a multiset permutation of the input $\mathbf{x}$ with at least $n-t$ fixed points, since there are at most t errors. As a consequence, the following lemma holds:

Lemma 2. *Under dynamic thresholding, the (t,l) error vector $\mathbf{e} = \mathbf{y} - \mathbf{x}$ is $(t-1,l)$-decomposable.*

Proof: Consider the cells at positions $s_1, s_2, \ldots, s_k$ with values $a_1, a_2, \ldots, a_k$, respectively. Say that the values are permuted by the cycle $\mathbf{h}$ of length k such that $a_1 \to a_2 \to \ldots \to$

[1]Although we are working in the ring of integers modulo q, we will often interpret the element $a > \frac{q}{2}$ as the negative number $-(q-a)$. In this context, magnitude is defined as $\min\{a, (q-a)\}$. For example, the magnitude of $-1 = q-1 \in F$ is $|q-1| = |-1| = |1| = 1$.

$a_{k-1} \to a_k \to a_1$. Then, these components in the error vector $\mathbf{e}$ will have values $a_2-a_1, a_3-a_2, \ldots, a_k-a_{k-1}, a_1-a_k$. This contribution can be expressed as $\mathbf{h} = (a_2-a_1)\mathbf{u}_{s_1,s_k} + (a_3-a_2)\mathbf{u}_{s_2,s_k} + \ldots + (a_k-a_{k-1})\mathbf{u}_{s_{k-1},s_k}$. The claim is clearly true for the first $k-1$ components. The last component (at position s_k) is correct, since $-(a_2-a_1+a_3-a_2+\ldots+a_k-a_{k-1}) = a_1-a_k$, as desired. Lemma 1 implies that $\mathbf{h}$ is $(k-1,l)$-decomposable (as, of course, $|a_{i+1}-a_i| \le l$.)

The error vector $\mathbf{e}$ is the sum of cycles $\mathbf{h_i}$ of length k_i where $\sum_i k_i = t$. Since each of the $\mathbf{h_i}$'s is (k_i-1,l)-decomposable, and $\sum_i (k_i-1) \le t-1$, $\mathbf{e}$ is $(t-1,l)$ decomposable. ■

We note some properties of error vectors based on this result. First, if the cycles involved in the multiset are all of length 2, that is, transpositions, then the contribution of that cycle to the error vector is a single term of the form $(a_2-a_1)\mathbf{u}_{s_1,s_2}$, for a transposition of the values a_1 and a_2 at positions s_1 and s_2, respectively. This contribution is $(1,l)$-decomposable, and there are at most $t/2$ such transpositions, so an error vector made of transpositions is $(t/2,l)$-decomposable.

It is clear that this is the "tightest" decomposability level - it is not possible for an error vector to be (e,l)-decomposable for $e < \frac{t}{2}$. More generally, we see that if the largest cycle in the permutation is of length $k \ge 2$, then the error vector is $(k-1,l)$-decomposable, and not (e,l)-decomposable for any $e < k-1$.

Recall from the proof of Theorem 1 that in the case where $l = 1$ (the error magnitude is limited to 1), the maximum error cycle length is 2. That is, all error vectors are made up of transpositions of adjacent values. Then, in this case, the error vectors are $(\frac{t}{2},1)$-decomposable. We rely on this fact in the next section.

B. Constructions

Now, consider two codewords $\mathbf{c_1}, \mathbf{c_2}$ and their respective (t,l) error vectors $\mathbf{e_1}, \mathbf{e_2}$ under dynamic thresholding. It is possible to produce the same received sequence if $\mathbf{c_1}+\mathbf{e_1} = \mathbf{c_2}+\mathbf{e_2}$, or $\mathbf{c_1}-\mathbf{c_2} = \mathbf{e_2}-\mathbf{e_1}$. Since $\mathbf{e_1}, \mathbf{e_2}$ are $(t-1,l)$–decomposable, their difference is $(2t-2,2l)$-decomposable, and the decomposability distance between $\mathbf{c_1}$ and $\mathbf{c_2}$ is at most $2t-2$. We have the following result:

Lemma 3. *A code C over F is capable of correcting all (t,l)-DT errors if and only if it has minimum decomposability distance $2t-1$ with respect to $2l$.*

We refer to such a code as a (t,l)**-dynamic thresholding error-correction code**, or a (t,l)-DTEC code.

With the above, we have shown the equivalence of our problem and location correction. We may now apply Roth and Seroussi's constructions from [27] to the dynamic thresholding problem. These constructions were used to build location-correcting codes (LCCs), but they also correct (t,l)-DT errors. We begin with a construction for $t=2$ and $l=q-1$. This is the smallest positive value of t, as a single error is not possible. From Lemma 2, an error vector must be $(1,l)$-decomposable. Recall that a B_2 set, also known as a *Sidon* set, is a set S with the property that for any four distinct elements $a,b,c,d \in S$, $a+b \ne c+d$ [28]. Then, we have:

Construction 1: Let C be a $[n,n-2]$ linear block code (of length n and dimension $n-2$) over F with the parity-check matrix

$$H = \begin{bmatrix} a_1 & a_2 & \cdots & a_n \\ a_1^2 & a_2^2 & \cdots & a_n^2 \end{bmatrix},$$

where $S = \{a_1, a_2, \ldots, a_n\}$ is a subset of distinct elements of F. Then, C is a $(2,q-1)$-DTEC code if S is a Sidon set.

Proof: If two codewords can be confused, then $\mathbf{e_1}H^T = \mathbf{e_2}H^T$ with $\mathbf{e_1} \ne \mathbf{e_2}$. Since $\mathbf{e_1}$ and $\mathbf{e_2}$ are $(1,l)$-decomposable, $\mathbf{e_1} = K\mathbf{u}_{b,c}$ and $\mathbf{e_2} = J\mathbf{u}_{d,f}$, with $K, J \in F$. Then,

$$\begin{aligned} K(a_b - a_c) &= J(a_d - a_f), \text{and} \\ K(a_b^2 - a_c^2) &= J(a_d^2 - a_f^2), \end{aligned}$$

and dividing the second equation by the first, we have that $a_b + a_c = a_d + a_f$. ■

Construction 1 provides a $(2,q-1)$-DTEC code. As we have done before, we examine the common case $l=1$. Then, we may define a $[n,n-1]$ $(2,1)$-DTEC code by the parity-check matrix

$$H = \begin{bmatrix} a_1 & a_2 & \ldots & a_n \end{bmatrix},$$

where $S = \{a_1, a_2, \ldots, a_n\}$ is a Sidon set. The proof of this claim follows along the same lines as that of Construction 1, but as $l=1$, we have that $K=J=1$, and we immediately get the equation $a_b + a_c = a_d + a_f$.

So far, our constructions allow for any error magnitude $(l=q-1)$ or an error magnitude of 1. If the error magnitude is limited to $1 < l < q-1$, we can take advantage of this fact by applying a technique introduced by Cassuto *et al.* in [8]. The proof is very similar to that in [8]:

Construction 2: Let C_2 be a (t,l)-DTEC code over an alphabet of size $q' = 2l+1$. Then, the code C_1 over F defined as

$$C_1 = \{\mathbf{x} \in F^n | \ \mathbf{x} \bmod q' \in C_2\}$$

is a (t,l)-DTEC code over an alphabet of size $q > q'$. Here, the notation $\mathbf{x}$ mod q' represents $(x_1 \bmod q', \ldots, x_n \bmod q')$.

Proof: Due to Lemma 3, it is enough to show that any two codewords $\mathbf{x}, \mathbf{z} \in C_1$ are at decomposability distance of at least $2t-1$. That is, we must show that the decomposability weight of $\mathbf{x}-\mathbf{z}$ (with respect to $2l$) is at least $2t-1$, or, $\mathbf{x}-\mathbf{z}$ is $(k,2l)$-decomposable only for $k \ge 2t-1$.

If there exists $i \in [n]$ such that $|x_i - z_i| > 2l$, $\mathbf{x}-\mathbf{z}$ has decomposability weight $n+1$, and we are done. If there is no such i, $\mathbf{x}-\mathbf{z} = \mathbf{x}-\mathbf{z} \bmod q'$. That is, the differences $|x_i - z_i|$ are identical in the fields of order q' and q. As C_2 is a (t,l)-DTEC code, $\mathbf{x}-\mathbf{z}$ mod q' has decomposability weight at least $2t-1$, so $\mathbf{x}-\mathbf{z}$ must also have decomposability weight at least $2t-1$, as desired. ■

Taking Construction 1 as the inner code C_2 in Construction 2, with alphabet size $2l+1$, we have a $(2,l)$-DTEC code C_1 over an alphabet of size q. Let $\mathbf{w}$ be a codeword in C_2. Then, if $x_i \equiv w_i \bmod (2l+1)$ for $1 \le i \le n$, $\mathbf{x}$ will be a codeword in C_1. There are between $\lfloor \frac{q}{2l+1} \rfloor$ and $\lceil \frac{q}{2l+1} \rceil$ choices for x_i for each $\mathbf{w} \in C_2$. Then, we can derive some bounds on the number of codewords $|C_1|$:

$$\left\lfloor \frac{q}{2l+1} \right\rfloor^n |C_2| \le |C_1| \le \left\lceil \frac{q}{2l+1} \right\rceil^n |C_2|.$$

For example, if $l = 11$, we have the Sidon set $S = \{0,1,2,4,7\}$ for a [5,3] inner code C_2 of size 8. Then, for

$q = 64$, C_1 is a $(2, 11)$-DTEC code with codebook size satisfying $256 \leq |C_1| \leq 1944$.

The length of our codes depends on the size of Sidon sets. The bound $|S|(|S|-3)+1 \leq q$ for $S \subset F$, along with several others are proved in [27]. This upper bound is problematic for small l in Construction 2. However, one can take the inner code C_2 over a larger field to get a longer code at the expense of additional redundancy.

Next we seek to derive (t, l)-DTEC codes for $t > 2$. We do so by relying on generalizations of Sidon sets called $B_t(S)$ sets, studied in [28]. We quote a famous number-theoretic result from [29]:

Theorem 3. *If $s = p^u$ (where p is a prime) and $m = (s^{v+1} - 1)/(s-1)$, we can find $s+1$ non-negative integers less than m*

$$d_0 = 0, d_1 = 1, d_2, \ldots, d_s,$$

such that the sums

$$d_{i_1} + d_{i_2} + \ldots + d_{i_v},$$

$(0 \leq i_1 \leq i_2 \leq \ldots \leq i_v \leq m)$, are all different modulo m.

This theorem is frequently cited in the coding literature. It is particularly useful when constructing codes where codewords must meet some *checksum* constraint, such as $\sum_{i=1}^{n} ix_i \equiv a \bmod m$. Such checksum-based codes are capable of correcting "unconventional" errors, such as asymmetric errors, insertion/deletion errors, and repetition errors. They were initially introduced by Varshamov and Tenengolts to correct a single asymmetric error [30]. We will use such a construction to develop l-DTEC codes based on the balanced codes approach in the next section.

Theorem 3 will enable us to build codes for the case $t > 2$. We give an example for the $l = 1$ case. Recall that the error vectors in this case are $(t/2, 1)$-decomposable, and thus have error value $E = 1$. Then, if we generate a set of $s+1$ integers $a_0 = 0, a_1, a_2, \ldots, a_s$ using the above theorem, the parity-check matrix

$$H = \left[\begin{array}{cccc} a_1 & a_2 & \ldots & a_s \end{array} \right],$$

defines an $[s, s-1]$ $(t/2, 1)$-DTEC code.

Of course, one can develop general (t, l)-DTEC codes with the approach of Construction 2, which does not depend on t. Decoders for codes similar to Construction 2 are presented in [8]. The optimality of codes equivalent to LCCs, such as Construction 1, is shown in [27].

C. Balanced Codes

We now examine the other implementation approach, based on balanced codes. For the sake of simplicity, we let $n = eq$ for some e. Then, in a balanced code, the space of possible codewords is

$$D = \{\mathbf{c} \in F^n \mid \mathbf{k}(\mathbf{c}) = (e, e, \ldots, e)\}.$$

That is, the space D is the set of permutations of the multiset

$$\{\underbrace{0, 0, \ldots, 0}_{e\ 0\text{s}}, \underbrace{1, 1, \ldots, 1}_{e\ 1\text{s}}, \ldots, \underbrace{q-1, q-1, \ldots, q-1}_{e\ (q-1)\text{s}}\}.$$

Each codeword is a multiset permutation that contains each symbol $0, 1, \ldots, q-1$ exactly e times. Note that the size of D is $\frac{n!}{(e!)^q}$. We seek to construct error-correction codes over the space D. Our approach is inspired by codes over the symmetric group S_n, such as *rank modulation* codes. In [6], [7], and [31] error correction codes for rank modulation are developed. We will see that a similar approach can be used to construct (t, l)-DTEC codes for our balanced codes dynamic thresholding implementation.

We note the fundamental differences between S_n and D. S_n is a group under permutation multiplication, defined as function composition. D cannot be defined as a group in this way. Although it is possible to introduce a multiplication operation for elements in D, inverses will not be unique. Due to these differences, some care must be taken when generalizing results originally derived for S_n.

First, we introduce the **Kendall tau distance** $d_\tau(\mathbf{c_1}, \mathbf{c_2})$, where $\mathbf{c_1}, \mathbf{c_2} \in D$. This measure of distance is typically applied to elements in S_n, but it is equally applicable for codewords in D, that is, multiset permutations. We define

$$d_\tau(\mathbf{c_1}, \mathbf{c_2}) = |\{(i,j) \in [n]^2 | i \neq j, c_1(i) < c_1(j), c_2(i) > c_2(j)\}|.$$

(Recall that $[n] = \{1, 2, \ldots, n\}$.) That is, the distance counts pairs of indices where the entries in the two permutations are oppositely ordered. For example, for $q = 3$ and $e = 2$, let $\mathbf{c_1} = (2, 0, 0, 1, 2, 1)$ and $\mathbf{c_2} = (1, 2, 0, 0, 2, 1)$. Then, the inversions are at index pairs $(2, 1), (2, 4)$, and $(2, 6)$ and $d_\tau(\mathbf{c_1}, \mathbf{c_2}) = 3$. Additionally, $d_\tau(\mathbf{c_1}, \mathbf{c_2})$ is at least the smallest number of transpositions of adjacent elements required to transform $\mathbf{c_1}$ into $\mathbf{c_2}$.

Although D is not a group, we will still refer to the permutation $\mathbf{e} = (0, 0, \ldots, 0, 1, \ldots, 1, \ldots, q-1)$ as the identity multiset permutation. The largest possible d_τ distance is $e^2\binom{q}{2}$. This occurs when the order of the elements in the identity $\mathbf{e}$ is exactly reversed, such as $(0, 0, 1, 1, 2, 2)$ and $(2, 2, 1, 1, 0, 0)$, which have distance 12.

We define an inversion in a multiset permutation $\mathbf{c}$ as a pair of indices $(i, j) \in [n]^2$ such that $i < j$ and $c(i) > c(j)$. For example $\mathbf{c} = (1, 0, 2, 0, 2, 1)$ has 5 inversions at index pairs $(1, 2), (1, 4), (3, 4), (3, 6)$, and $(5, 6)$.

Next, we introduce the notion of the **inversion vector**. Again, this is a concept typically applied to the elements of S_n which can be easily extended to multiset permutations. The inversion vector essentially keeps track of the number of inversions involving the various elements of a multiset permutation. There are several possible definitions. We generalize the definition in [32]. For $\mathbf{c} \in D$, define the inversion vector $\mathbf{x_c}$ such that

$$x_{\mathbf{c}}((i-1)e + a) = |\{(j, b), j \in [q-1], b \in [e]| \\ j < i+1, c_b^{-1}(j) > c_a^{-1}(i+1)\}|,$$

for $i \in [q-1]$ and $a \in [e]$. Here, $c_a^{-1}(j)$ refers to the position of the ath entry with value j in $\mathbf{c}$, so that for $\mathbf{c} = (1, 0, 2, 0, 2, 1)$, $c_1^{-1}(2) = 3$ and $c_2^{-1}(2) = 5$.

In words, the entry $x_{\mathbf{c}}((i-1)e + a)$ is the number of inversions in $\mathbf{c}$ in which the ath i value is the first entry $(1 \leq i \leq q-1)$. We will illustrate this idea with an example. Take $\mathbf{c} = (3, 2, 0, 0, 1, 2, 3, 1)$. There are no inversions beginning with the first 1 (at position 5), so the first element of $\mathbf{x_c}$ is 0. The second 1 (at position 8) also is not the first element in any inversion, so the second element of $\mathbf{x_c}$ is also 0. There are 4 inversions starting with the first 2 in $\mathbf{c}$: the inversions at positions $(2,3),(2,4),(2,5)$, and $(2,8)$, so $x_{\mathbf{c}}((2-1)\times 2+1) = x_{\mathbf{c}}(3) = 4$, and so on. We have that $\mathbf{x_c} = (0,0,4,1,6,1)$.

Note that the length of $\mathbf{x_c}$ is $(q-1)e = n-e$. Also, there are at most $e(i-1)$ elements smaller than i located to the right of it in the permutation $\mathbf{c}$, so $x_{\mathbf{c}}((i-1)e+a) \in [0, e(i-1)]$ so that

$$\mathbf{x_c} \in \underbrace{[0,e]\times\ldots\times[0,e]}_{e \text{ times}}\times\ldots\times \underbrace{[0,(q-2)e]\times\ldots\times[0,(q-2)e]}_{e \text{ times}}.$$

We refer to this space as G_n.

It is well known that there is a bijection between our codewords, elements in D, and inversion vectors. It is possible to recover the original permutation $\mathbf{c}$ from its inversion vector $\mathbf{x_c}$. Furthermore, we may introduce the inversion vector distance $d(\mathbf{x_{c_1}}, \mathbf{x_{c_2}}) = \sum_{i=1}^{n-e} |x_{\mathbf{c_1}}(i) - x_{\mathbf{c_2}}(i)|$. We have the following lemma, which relates the Kendall tau distance d_τ and the inversion vector distance:

Lemma 4. *If $\mathbf{c_1}, \mathbf{c_2} \in D$ are codewords, and $\mathbf{x_{c_1}}, \mathbf{x_{c_2}}$ are the respective inversion vectors,*

$$d_\tau(\mathbf{c_1}, \mathbf{c_2}) \geq d(\mathbf{x_{c_1}}, \mathbf{x_{c_2}}).$$

Proof: We begin with the permutation $\hat{\mathbf{c}}_1$ which has a single transposition of adjacent elements but is otherwise the same as $\mathbf{c_1}$. Then, $d_\tau(\mathbf{c_1}, \hat{\mathbf{c}}_1) = 1$. We show that $d(\mathbf{x_{c_1}}, \mathbf{x}_{\hat{\mathbf{c}}_1}) = 1$.

Let the tranposition be at locations i and $i+1$. Let $c_1(i) = \alpha$ and $c_1(i+1) = \beta$. We only consider the case $\alpha < \beta$, as, by definition, $\alpha \neq \beta$, and the $\alpha > \beta$ case is symmetric. When forming $\mathbf{x}_{\hat{\mathbf{c}}_1}$, no entries other than the ones corresponding to α and β can change. The number of inversions in which α is the first element does not change. Only the entry corresponding to β at location $i+1$ can be increased by 1 (from the additional inversion formed by the pair (β,α) now at positions $(i,i+1)$). Thus the distance $d(\mathbf{x_{c_1}}, \mathbf{x}_{\hat{\mathbf{c}}_1}) = 1$.

On the other hand, $d(\mathbf{x_{c_1}}, \mathbf{x}_{\hat{\mathbf{c}}_1}) = 1$ only implies that $d_\tau(\mathbf{c_1}, \hat{\mathbf{c}}_1) \geq 1$. Since d_τ is at least the number of transpositions of adjacent elements changing one permutation to another, we may repeatedly apply the single transposition case above to conclude that $d_\tau(\mathbf{c_1}, \mathbf{c_2}) \geq d(\mathbf{x_{c_1}}, \mathbf{x_{c_2}})$. ■

Thus, a code over the space of inversion vectors that corrects t additive errors of weight l will also correct the corresponding errors over the space D. In [7], Barg and Mazumdar demonstrated how such a code over the space of inversion vectors can be found. We quote the following theorem, which relies on Theorem 3 from the previous section. Denote the set of integers modulo n by $\mathbb{Z}_n$.

Theorem 4. *Let $m = \frac{s^{v+1}-1}{s-1}$ and set v to be t, the maximum number of errors. Then, generate $s+1$ integers $d_0 = 0, d_1, \ldots d_s$ according to Theorem 3. Take $h_i = d_{i-1} + \frac{t-1}{2}m$ for $i \in [s+1]$ if t is odd and $h_i = d_{i-1} + \frac{t}{2}m$ for $i \in [s+1]$ if t is even. Let $m_t = t(t+1)m$ if t is odd and $m_t = t(t+2)m$ if t is even.*

Then, for any error vector $\mathbf{e}$ in $\mathbb{Z}^{s+1}$ with t or fewer errors, the sums $\sum_{i=1}^{s+1} e_i h_i$ are all distinct and non-zero modulo m_t.

Now we may define the code C by

$$C = \left\{ \mathbf{x} \in \mathbb{Z}^{s+1} | \sum_{i=1}^{s+1} h_i x_i \equiv 0 \bmod m_t \right\}.$$

From Theorem 4, C corrects t additive errors of weight l over $Z_{m_t}^{s+1}$. Our initial goal was to construct a code over the space of inversion vectors G_n that corrects t additive errors. Let $s+1 = n-e$. Following [7], we note that G_n is a subset of $\mathbb{Z}_n^{n-e}$, which is itself a subset of $\mathbb{Z}_{m_t}^{n-e}$. Furthermore, since C is a group code with respect to addition modulo m_t, we may take an appropriate coset to form our desired additive error-correction code over G_n.

Thus, we see that there exist (t,l)-DTEC codes using the balanced codeword approach. We conclude that it is possible to develop dynamic thresholding coding constructions that correct (a fixed number of) limited-magnitude errors in either implementation. Further details regarding codes based on Theorem 4 above are discussed in [32].

VI. L-DYNAMIC THRESHOLDING ECCS

Finally, we wish to correct any number of magnitude l errors (the case where $t \leq n$). Codes that do so have been studied in [8], [9], and [33]. We develop such a code to be used in conjunction with dynamic thresholds. We refer to this code as an **l-dynamic thresholding error-correction code**, or an l-DTEC code.

The l-DTEC constructions will be based on the metadata implementation approach. Again, we assume that the metadata has been made available to the decoder with no error.

We begin by defining an auxilliary distance function which will be useful for defining our construction. The *permutation distance* $d_\pi(\mathbf{c_1}, \mathbf{c_2})$ is a measure of difference between codewords that are permutations of each other. If a cycle $(a_1, a_2, \ldots, a_k)$ in a permutation is such that $|a_i - a_{i+1}| \leq l$ for all $i < k$ and $|a_k - a_1| \leq l$, it will be called an l-cycle. A permutation that is the product of l-cycles is referred to as an l-permutation. Let the function g be defined so that $g(\pi)$ is the smallest nonnegative integer l such that π is an l-permutation. For $\mathbf{c_1}, \mathbf{c_2} \in F$, the permutation distance d_π is given by

$$d_\pi(\mathbf{c_1}, \mathbf{c_2}) = \begin{cases} \infty & \text{if } \mathbf{c_2} \text{ is not a permutation of } \mathbf{c_1}, \\ g(\pi) & \text{if } \mathbf{c_2} = \pi(\mathbf{c_1}) \text{ is a permutation of } \mathbf{c_1}. \end{cases}$$

A code of minimum d_π distance of $l+1$ corrects any number of errors limited to magnitude l. The following construction achieves this minimum d_π distance by selecting a single codeword from each set of l-permutations:

Construction 3: Let C be defined as

$$C = \{\mathbf{x} \in F^n \mid 0 < x_i - x_j \leq l \implies i > j\}.$$

Then, C is a l-DTEC code.

Note that this approach is effectively the dual of that of the previous section. When using balanced codes, *all* codewords are permutations of one another. In our l-DTEC constructions, *no* codewords are permutations of one another. This is a consequence of choosing between limited or unlimited number of errors in the channel model.

We introduce a natural decoding technique, which we refer to as τ-decoding. First, we comment on the name of the scheme. Although we defined Construction 3 in terms of the distance d_π, which measures the size of the largest cycle required to turn one permutation into another, the τ-decoder will recover the original codeword transposition by transposition. That is, the decoder will, by stages, reduce the Kendall tau distance between the received permutation and the original codeword.

The τ-decoder operates in the following way: if the decoder finds y_i and y_j in the received sequence $\mathbf{y}$ with $i < j$ and $0 < y_i - y_j \leq l$, the values of y_i and y_j will be exchanged. The decoder repeats this process as long as such pairs can be found. For example, consider the codeword 12345, where $n = 5$, $q = 6$, and $l = 2$. Assume the errors are $1 \to 3$, $2 \to 1$, $3 \to 5$, $4 \to 2$, and $5 \to 4$. The received codeword is 31524. Repeatedly applying the decoding rule, we have the progression $31524 \to 13524 \to 13425 \to 13245 \to 12345$.

As described above, τ-decoding reduces the Kendall tau distance d_τ between the original codeword and the received codeword. Recall that the distance d_τ is defined as the number of inversions among two permutations π and σ, where an inversion is a pair (i, j) with $i < j$, with either $\pi(i) < \pi(j)$ and $\sigma(i) > \sigma(j)$, or, $\pi(i) > \pi(j)$ and $\sigma(i) < \sigma(j)$ ([34]). We have that:

Theorem 5. *When errors are limited to magnitude l and dynamic thresholds and the l-DTEC code C in Construction 3 are used, τ-decoding recovers the original codeword.*

Proof: Using dynamic thresholding, errors will be part of cycles. Let the cells involved be $x_{p_1}, \ldots, x_{p_k}$ where $p_1 < \ldots < p_k$. By definition of C, the cell values also satisfy $a_1 < a_2 < \ldots < a_k$. The error cycle permutes the a_i's, so that $x_{p_1}, \ldots, x_{p_k}$ take on the values $\pi(a_1), \ldots, \pi(a_k)$, where π is the permutation induced by the error cycle. Let $\mathbf{y}_\alpha$ be the current output after α iterations of the decoding process. We will evaluate the distance $d_\tau(\mathbf{y}_\alpha, \mathbf{x})$ after each iteration. If $d_\tau(\mathbf{y}_\alpha, \mathbf{x}) = 0$, we have recovered the original codeword.

The decoding rule at each iteration finds a particular inversion (a_i, a_j) with $i < j$ and $\pi(a_i) > \pi(a_j)$ and exchanges their values. We compute the distance before and after one decoding iteration, i.e., $d_\tau(\mathbf{y}_\alpha, \mathbf{x})$ and $d_\tau(\mathbf{y}_{\alpha+1}, \mathbf{x})$. The only possible pairs that might change (become an inversion or stop being an inversion) after application of the decoding rule are those which have $\pi(a_i)$ or $\pi(a_j)$ as members. We count the number of such pairs. Consider the following (disjoint) sets:

$$\begin{aligned} Y_1 &= \{k \mid k < i, \pi(a_k) > \pi(a_i) > \pi(a_j)\}, \\ Y_2 &= \{k \mid k < i, \pi(a_i) > \pi(a_k) > \pi(a_j)\}, \\ Z_1 &= \{k \mid i < k < j, \pi(a_k) > \pi(a_i) > \pi(a_j)\}, \\ Z_2 &= \{k \mid i < k < j, \pi(a_i) > \pi(a_k) > \pi(a_j)\}, \\ Z_3 &= \{k \mid i < k < j, \pi(a_i) > \pi(a_j) > \pi(a_k)\}, \\ W_1 &= \{k \mid j < k, \pi(a_i) > \pi(a_j) > \pi(a_k)\}, \\ W_2 &= \{k \mid j < k, \pi(a_i) > \pi(a_k) > \pi(a_j)\}. \end{aligned}$$

The entries in the permutation where these sets are located:

$$\overbrace{\pi(a_1), \ldots,}^{Y_1, Y_2} \pi(a_i), \underbrace{\pi(a_{i+1}), \ldots, \pi(a_{j-1})}_{Z_1, Z_2, Z_3}, \pi(a_j), \overbrace{\ldots, \pi(a_k)}^{W_1, W_2}.$$

It is easy to see how the members of these sets give rise to inversions. For example, if $k \in W_1$, then, $i < j < k$ and $\pi(a_i) > \pi(a_j) > \pi(a_k)$, so $(\pi(a_i), \pi(a_k))$ and $(\pi(a_j), \pi(a_k))$ are inversions. It is clear that each element of each set contributes to either one or two inversions.

Say that there are M pairs that do not involve $\pi(a_i)$ or $\pi(a_j)$. Then, counting the contribution of the pair $(\pi(a_i), \pi(a_j))$, we have the following value for $d_\tau(\mathbf{y}_\alpha, \mathbf{x})$:

$$\begin{aligned} d_\tau(\mathbf{y}_\alpha, \mathbf{x}) = M + 1 + 2|Y_1| + |Y_2| + |Z_1| + 2|Z_2| \\ + |Z_3| + 2|W_1| + |W_2|. \end{aligned}$$

Now, after application of the decoding rule, the pair $(\pi(a_i), \pi(a_j))$ becomes $(\pi(a_j), \pi(a_i))$, and is no longer an inversion. Moreover, Z_2 does not contribute to any inversions, since $i < j < k$ and $\pi(a_j) < \pi(a_k) < \pi(a_i)$. Then,

$$\begin{aligned} d_\tau(\mathbf{y}_{\alpha+1}, \mathbf{x}) = M + 2|Y_1| + |Y_2| + |Z_1| + |Z_3| + \\ 2|W_1| + |W_2|. \end{aligned}$$

Thus, we have that $d_\tau(\mathbf{y}_\alpha, \mathbf{x}) > d_\tau(\mathbf{y}_{\alpha+1}, \mathbf{x})$. Each decoding step strictly reduces the distance, so there exists a finite s such that $d_\tau(\mathbf{y}_s, \mathbf{x}) = 0$ and the original codeword will have been recovered. ■

In the case $l = 1$, this decoder has a simple implementation. The received sequence is first sorted by value. All errors will then be adjacent, and can be corrected in a single pass through the list. The original order is then restored. This algorithm has complexity $O(n \log n)$.

We note that we can build an l-DTEC code from Construction 3 by including every codeword from an existing l-asymmetric error-correction (l-AEC) code, which has minimum distance $l + 1$, as shown in [9]. Note that although limited-magnitude errors under dynamic thresholding are not asymmetric, the permutation property of dynamic thresholding errors ensures that l-AEC constructions are sufficient to correct l-DT errors. For this reason, l-DTEC constructions most closely resemble l-AEC constructions. However, several additional codewords can also be included in addition to the l-AEC codewords, allowing us to maximize the rate under the dynamic thresholding conditions. We illustrate this idea with an example of a code based on Construction 3. We take $q = 3$, limited magnitude $l = 1$, and $n = 3$. An 1-AEC code of length 3 has minimum distance of 2, requiring codewords to exclusively use symbols from $\{0, 2\}$. In our case, we may add several codewords that are "ordered" the correct way. Our codewords are then:

$$\begin{aligned} (0,0,0), (0,0,2), (0,2,0), (2,0,0), (0,2,2), \\ (2,0,2), (2,2,0), (2,2,2), (0,0,1), (0,1,1), \\ (1,1,1), (0,1,2), (1,1,2), (1,2,2). \end{aligned}$$

We see that the first 8 codewords are from the 1-AEC code with minimum distance 2, but we may add 6 additional codewords that maintain the correct order. For $q = 3$, $l = 1$

and general n, we have that the number of codewords is $2^n + \binom{n+1}{2}$. This is easy to see: If a codeword does not contain any 1s, each digit may be 0 or 2, for a total of 2^n such codewords. If a codeword does contain a 1, it is clear that all its 0s must be to the left of all its 1s. Similarly, all 2s must be to the right of all 1s. Thus, there is a left and a right boundary (not in the same position, since there are a positive number of 1s.) There are $n+1$ positions to place the boundaries, so that we have a total of $\binom{n+1}{2}$ codewords containing 1s.

Not including the metadata digits, the rate of such a code is $\log_3(2^n + \binom{n+1}{2})/n$. When our codes are in the finite-length regime, we get the benefit of an additional $\binom{n+1}{2}$ codewords versus directly combining dynamic thresholding and an existing 1-AEC codewith minimum distance 2.

We note that the construction above is only suitable for higher field sizes q. Furthermore, asymptotically, the size of the code is $\lceil \frac{q}{l+1} \rceil^n$, which is the size of the largest l-AEC code [9]. Therefore, our construction is superior to l-AEC constructions for finite-length codes with $q \geq 4$. These conditions match the requirements of MLC non-volatile memories.

VII. Conclusion

Dynamic thresholds have many advantages when applied to non-volatile memories. In this paper, we showed how to apply dynamic threshold schemes to multi-level cell memories. We demonstrated that dynamic thresholds perform better than fixed thresholds. We showed that our proposed scheme has performance close to that of the optimal scheme. Two implementations for our scheme were introduced, based on metadata and balanced codes.

We studied several flavors of error-correction codes tailored specifically to the dynamic thresholding scheme, and we showed that if the total number of errors is limited, the problem of constructing such codes is closely related to problems such as location-correction and rank modulation. For the case where the number of errors is not limited, we introduced a new code construction and an associated decoder.

There are several directions available for future work. It remains to be determined in the general case whether the code constructions based on the metadata implementation are superior to those based on the balanced-codes approach. Another question is whether there is a benefit to mixing these two implementations: that is, requiring that codewords are balanced within a certain degree, so that for codewords $\mathbf{x}_1$ and $\mathbf{x}_2$, $\mathbf{k}(\mathbf{x}_1)$ and $\mathbf{k}(\mathbf{x}_2)$ differ in each component only by a limited amount.

Acknowledgement

The authors would like to thank Hongchao Zhou for his valuable comments and suggestions regarding our work.

References

[1] P. Cappelletti *et al.*, Eds., *Flash Memories*. Kluwer, 1999.

[2] A. Pirovano and A. Redaelli, "Reliability study of phase-change non-volatile memories," *IEEE Trans. Dev. Mat. Rel.*, vol. 4, no. 3, pp. 422–427, Sept. 2004.

[3] R. Bez, E. Camerlanghi, A. Modelli, and A. Visconti, "Introduction to flash memory," *Proc. IEEE*, vol. 91, no. 4, pp. 489–502, Apr. 2003.

[4] A. Jiang, R. Mateescu, and J. Bruck, "Rank modulation for flash memories," *IEEE Trans. Inf. Theory*, vol. 55, no. 6, pp. 2659–2673, Jun. 2009.

[5] F. Farnoud, V. Skachek, and O. Milenkovic, "Rank modulation for translocation error correction," in *Proc. 2012 IEEE Int. Symp. Information Theory*, pp. 2988–2992.

[6] A. Jiang, M. Schwartz, and J. Bruck, "Correcting charge-constrained errors in the rank-modulation scheme," *IEEE Trans. Inf. Theory*, vol. 56, no. 5, pp. 2112–2121, May 2010.

[7] A. Barg and A. Mazumdar, "Codes in permutations and error correction for rank modulation," *IEEE Trans. Inf. Theory*, vol. 56, no. 7, pp. 3158–3165, Jul. 2010.

[8] Y. Cassuto, M. Schwartz, V. Bohossian, and J. Bruck, "Codes for asymmetric limited-magnitude errors with application to multi-level flash memories," *IEEE Trans. Inf. Theory*, vol. 56, no. 4, pp. 1582–1596, Apr. 2010.

[9] N. Elarief and B. Bose, "Optimal, systematic, q-ary codes correcting all asymmetric and symmetric errors of limited magnitude," *IEEE Trans. Inf. Theory*, vol. 56, no. 3, pp. 979–984, Mar. 2010.

[10] T. Kløve, B. Bose, and N. Elarief, "Systematic, single limited magnitude error correcting codes for flash memories," *IEEE Trans. Inf. Theory*, vol. 57, no. 7, pp. 4477–4487, Jul. 2011.

[11] L. Tallini and B. Bose, "On symmetric L_1 distance error control codes and elementary symmetric functions," in *Proc. 2012 IEEE Int. Symp. Information Theory*, pp. 741–745.

[12] L. Tallini and B. Bose, "On symmetric/asymmetric Lee distance error control codes and elementary symmetric functions," in *Proc. 2012 IEEE Int. Symp. Information Theory*, pp. 746–750.

[13] M. Schwartz, "Quasi-cross lattice tilings with applications to flash memory," *IEEE Trans. Inf. Theory*, vol. 58, no. 4, pp. 2397–2405, Apr. 2012.

[14] A. Berman and Y. Birk, "Constrained flash memory programming," in *Proc. 2011 IEEE Int. Symp. Information Theory*, pp. 2128–2132.

[15] H. Mahdavifar, P. Siegel, A. Vardy, J. Wolf, and E. Yaakobi, "A nearly optimal construction of flash codes," in *Proc. 2009 IEEE Int. Symp. Information Theory*, pp. 1239–1243.

[16] A. Jiang, V. Bohossian, and J. Bruck, "Floating codes for joint information storage in write asymmetric memories," in *Proc. 2007 IEEE Int. Symp. Information Theory*, pp. 1166–1170.

[17] A. Jiang, Y. Li, and J. Bruck, "Bit-fixing codes for multi-level cells," in *Proc. 2012 IEEE Information Theory Workshop*.

[18] J. Wang, T. Courtade, H. Shankar, and R. Wesel, "Soft information for LDPC decoding in flash: mutual-information optimized quantization," in *Proc. 2011 IEEE Global Telecommunications Conf.*, pp. 1–6.

[19] H. Zhou, A. Jiang, and J. Bruck, 2012. "Balanced modulation for nonvolatile memories." Available: http://arxiv.org/pdf/1209.0744.pdf

[20] A. Jiang, H. Zhou, Z. Wang, and J. Bruck, "Patterned cells for phase change memories," in *Proc. 2011 IEEE Int. Symp. Information Theory*, pp. 2333–2337.

[21] H. Zhou, A. Jiang, and J. Bruck, "Error-correcting schemes with dynamic thresholds in non-volatile memories," in *Proc. 2011 IEEE Int. Symp. Information Theory*, pp. 2143–2147.

[22] R. Stanley, *Enumerative Combinatorics, Vol. 1.* Cambridge University Press, 2000.

[23] M. Blum, F. Floyd, V. Pratt, R. Rivest, and R. Tarjan, "Time bounds for selection," *J. Computer and System Sciences*, vol. 7, no. 4, pp. 448–461, Aug. 1973.

[24] D. Knuth, "Efficient balanced codes," *IEEE Trans. Inf. Theory*, vol. 32, no. 1, pp. 51–53, Jan. 1986.

[25] J. Weber, K. Immink, P. Siegel, and T. Swart, "Polarity-balanced codes," in *Proc. 2013 IEEE Information Theory Applications Workshop*.

[26] Y. Chee and S. Ling, "Constructions for q-ary constant-weight codes," *IEEE Trans. Inf. Theory*, vol. 53, no. 1, pp. 135–146, Jan. 2007.

[27] R. Roth and G. Seroussi, "Location-correcting codes," *IEEE Trans. Inf. Theory*, vol. 42, no. 2, pp. 554–565, Mar. 1996.

[28] L. Babai and V. Sos, "Sidon sets in groups and induced subgraphs of Cayley graphs," *European J. Combinatorics*, vol. 6, no. 2, pp. 101–114, 1985.

[29] R. Bose and S. Chowla, "Theorems in the additive theory of numbers," *Commentarii Mathematici Helvetici*, vol. 37, no. 1, pp. 141–147, Dec. 1962.

[30] R. Varshamov and G. Tenengolts, "Codes which correct single asymmetric errors," *Automatika i Telemekhanika*, vol. 26, no. 2, pp. 288–292, 1965.

[31] I. Tamo and M. Schwartz, “Correcting limited-magnitude errors in the rank-modulation scheme,” *IEEE Trans. Inf. Theory*, vol. 56, no. 6, pp. 2551–2560, Jun. 2010.

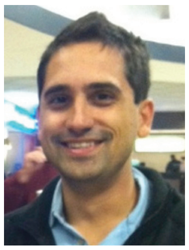
[32] A. Mazumdar, A. Barg, and G. Zemor, “Constructions of rank modulation codes,” *IEEE Trans. Inf. Theory*, vol. 59, no. 2, pp. 1018-1029, Feb. 2013.

[33] R. Ahlswede, H. Aydinian, L. Khachatrian, and L. Tolhuizen, “On q-ary codes correcting all unidirectional errors of a limited magnitude,” in *Proc. 2004 Int. Workshop Algebraic and Combinatorial Coding Theory*, pp. 20–26.

[34] M. Kendall, “A new measure of rank correlation,” *Biometrika*, vol. 30, no. 1/2, pp. 81–93, Jun. 1938.

Frederic Sala received the B.S.E. degree in Electrical Engineering from the University of Michigan, Ann Arbor, in 2010. He is currently pursuing the Ph.D. degree in Electrical Engineering at the University of California, Los Angeles (UCLA), where he is associated with the LORIS lab.

His research interests include information theory and coding with a focus on error-correction codes, including applications to synchronization and data storage in non-volatile memories. He is a recipient of the NSF Graduate Research Fellowship.



Ryan Gabrys received the B.S. degree in mathematics and computer science from the University of Illinois at Champaign-Urbana in 2005. In 2010 he received the Master of Engineering degree from the University of California at San Diego. In 2010, he was awarded the SMART scholarship through the Department of Defense, and since that time he has been pursuing a Ph.D. in electrical engineering at the University of California at Los Angeles. His research interests include coding theory and its applications to storage.

Lara Dolecek is an Assistant Professor with the Electrical Engineering Department at the University of California, Los Angeles (UCLA). She holds a B.S. (with honors), M.S. and Ph.D. degrees in Electrical Engineering and Computer Sciences, as well as an M.A. degree in Statistics, all from the University of California, Berkeley. She received the 2007 David J. Sakrison Memorial Prize for the most outstanding doctoral research in the Department of Electrical Engineering and Computer Sciences at UC Berkeley. Prior to joining UCLA, she was a postdoctoral researcher with the Laboratory for Information and Decision Systems at the Massachusetts Institute of Technology. She received Okawa Research Grant in 2013, NSF CAREER Award in 2012, and Hellman Fellowship Award in 2011. She is an Associate Editor for IEEE TRANSACTIONS ON COMMUNICATIONS and for IEEE COMMUNICATION LETTERS and is the lead guest editor for the IEEE JSAC special issue on emerging data storage. Her research interests span coding and information theory, graphical models, statistical algorithms, and computational methods, with applications to emerging systems for data storage, processing, and communication.