Snake-in-the-Box Codes for Rank Modulation

Yonatan Yehezkeally
Electrical and Computer Engineering
Ben-Gurion University of the Negev
Beer Sheva 84105, Israel
yonatany@post.bgu.ac.il

Moshe Schwartz
Electrical and Computer Engineering
Ben-Gurion University of the Negev
Beer Sheva 84105, Israel
schwartz@ee.bgu.ac.il

***Abstract*—Motivated by the rank-modulation scheme with applications to flash memory, we consider Gray codes capable of detecting a single error, also known as snake-in-the-box codes. We study two error metrics: Kendall's τ-metric, which applies to charge-constrained errors, and the ℓ_∞-metric, which is useful in the case of limited-magnitude errors. In both cases we construct snake-in-the-box codes with rate asymptotically tending to 1.**

***Index Terms*—Snake-in-the-box codes, rank modulation, permutations, flash memory**

I. Introduction

Flash memory is a non-volatile storage medium which is electrically programmable and erasable. Its current wide use is motivated by its high storage density and relative low cost. Among the chief disadvantages of flash memories is their inherent asymmetry between cell programming (injecting cells with charge) and cell erasure (removing charge from cells). While single cells can be programmed with relative ease, in the current architecture, the process of erasure can only be performed by completely depleting large blocks of cells of their charge. Moreover, the removal of charge from cells physically damages them over time.

This issue is exacerbated as a result of the ever-present demand for denser memory: smaller cells are more delicate, and are damaged faster during erasure. They also contain less charge and are therefore more prone to error. In addition, flash memories, at present, use multilevel cells, where charge-levels are quantized to simulate a finite alphabet – the more levels, the less safety margins are left, and data integrity is compromised. Thus, over-programming (increasing a cell's charge-level above the designated mark) is a real problem, requiring a costly and damaging erasure cycle. Hence, in a programming cycle, charge-levels are usually made to gradually approach the desirable amount, making for lengthier programming cycles as well (see [3]).

In an effort to counter these effects, a different modulation scheme has recently been suggested for flash memories – rank modulation [11]. This scheme calls for the representation of the data stored in a group of cells in the permutation suggested by their relative charge-levels. That is, if $c_1, c_2, \ldots, c_n \in \mathbb{R}$ represent the charge-levels of $n \in \mathbb{N}$ cells, then that group of cells is said to encode that permutation $\sigma \in S_n$ such that:

$$c_{\sigma(1)} > c_{\sigma(2)} > \ldots > c_{\sigma(n)} > 0.$$

This work was supported in part by ISF grant 134/10.

This scheme eliminates the need for discretization of charge-levels. In addition, rank modulation also improves the memory's robustness against other noise types. Retention, the process of slow charge leakage from cells, tends to affect all cells with a similar trend [3]. Since rank modulation stores information in the differences between charge-levels rather than their absolute values, it offers more resilience against that type of noise. It is also worth noting that the advantages of rank modulation have been experimentally applied to phase-change memory with promising results (see [16]).

The Gray code [8] was first introduced as a sequence of distinct binary vectors of fixed length, where every adjacent pair differs in a single coordinate. It has since been generalized to sequences of states over other spaces, where each state is derived from the former by a transformation from a predetermined set of allowed transformations (see [17] for an excellent survey). Among these, [11] studied Gray odes over permutations, and presented such codes traversing the entire group of permutations. In this fashion, it was suggested that a set of n rank-modulation cells could implement a single logical multilevel cell with $n!$ levels, where increasing the logical cell's level by 1 corresponds to a single transition in the code. This allows for a natural integration of rank modulation with other multilevel approaches such as rewriting schemes [4], [9], [10], [21]. This was done by restricting the allowed transformations to *"push-to-the-top" operations*, namely only programming a group of cells by increasing the charge-level of a single cell above that of all other cells in the group. The use of such Gray codes allows for faster cell programming and eliminates overshoot problems (see [11]). In the context of flash memory, "push-to-the-top" operations have also been used in [6], [7]. We also note that generating permutations using "push-to-the-top" operations is of independent interest, called "nested cycling" in [18] (see also references therein), motivated by a fast "push-to-the-top" operation[1] (cycling) available on some computer architectures.

Other recent works have explored error-correcting codes for rank modulation, where different types of errors are addressed by a careful choice of metric. In [2], [12], [15], Kendall's τ-metric was considered, since a small charge-constrained error

[1]The operations described in [18] are actually mirror images of "push-to-the-top" . Furthermore, the permutation-generation scheme there is not a Gray code since it repeats some of the previously generated permutations, also making it inefficient.

translates into a small distance in the metric. In contrast, the ℓ_∞-metric was used in [14], [19], as small distances in this metric correspond to small limited-magnitude errors.

In this paper, we explore Gray codes for rank modulation which detect a single error, under Kendall's τ-metric (further results in the ℓ_∞-metric are mentioned in the conclusion to this paper). Such codes are known as *snake-in-the-box codes*, and have been studied extensively for binary vectors with the Hamming metric and with single-bit flips as allowable transitions (see [1] and references therein).

The paper is organized as follows: In Section II we present basic notation and definitions. In Section III we review properties of Kendall's τ-metric, then present a recursive construction of snake-in-the-box codes over the alternating groups of odd orders with rate asymptotically tending to 1. We conclude in Section IV with a description of further results given without proof, along with some ad-hoc results, and open questions. Some proofs for stated results are omitted due to the limited space; they can be found in the journal version of this work, to appear in [22].

II. Preliminaries

We shall denote a permutation σ on n elements by $\sigma = [\sigma(1), \sigma(2), \ldots, \sigma(n)]$. This form is called the *vector notation* for permutations. We let S_n be the group of all permutations on $[n]$. For $\sigma, \tau \in S_n$, their composition, denoted $\sigma\tau$, is the permutation for which $\sigma\tau(i) = \sigma(\tau(i))$ for all $i \in [n]$. It is well known that $|S_n| = n!$.

A cycle, denoted $(a_1, a_2, \ldots, a_k)$, is a permutation mapping $a_i \mapsto a_{i+1}$ for all $i \in [k-1]$, as well as $a_k \mapsto a_1$. We shall occasionally use *cycle notation* in which a permutation is described as a composition of cycles. We also recall that any permutation may be represented as a composition of cycles of size 2 (known as *transpositions*), and that the parity of the number of transpositions does not depend on the decomposition. Thus we have *even* and *odd* permutations. We let A_n be the subgroup of all even permutations on $[n]$, called the *alternating group of order n*. Again, it is well known that $|A_n| = \frac{1}{2}|S_n|$.

Definition 1. *Given a set S and a subset of transformations $T \subseteq \{f \mid f : S \to S\}$, a* Gray code *over S, using transitions T, of size M, is a sequence $C = (c_0, c_1, \ldots, c_{M-1})$ of M distinct elements of S, called* codewords, *such that for all $j \in [M-1]$ there exists $t \in T$ such that $c_j = t(c_{j-1})$.*

Alternatively, when the original permutation c_0 is known (or irrelevant), we use a slight abuse of notation in referring to the sequence of transformations $(t_{k_1}, \ldots, t_{k_{M-1}})$ generating the code (i.e., $c_j = t_{k_j}(c_{j-1})$) as the code itself.

In the above definition, when $M = |S|$ the Gray code is called *complete*. If there exists $t \in T$ such that $t(c_{M-1}) = c_0$ the Gray code is *cyclic*, M is called its *period*, and we shall, when listing the code by its sequence of transformations, include $t_{k_M} = t$ at the end of the list. The *rate* of C, denoted $R(C)$, is defined as

$$R(C) = \frac{\log_2 M}{\log_2 |S|}.$$

In the context of rank modulation for flash memories, the set of transformations T comprises of "push-to-the-top" operations, first used in [11], and later also in [7], [20]. We denote by $t_i : S_n \to S_n$ the "push-to-the-top" operation on index i, i.e.,

$$t_i[a_1, a_2, \ldots, a_{i-1}, a_i, a_{i+1}, \ldots, a_n] = \\ = [a_i, a_1, a_2, \ldots, a_{i-1}, a_{i+1}, \ldots, a_n],$$

and throughout the paper we set $T = \{t_2, t_3, \ldots, t_n\}$. We also note that, in cycle notation,

$$t_i\sigma = \sigma\,(i, i-1, \ldots, 1)\,. \tag{1}$$

For ease of presentation only, we also denote by $\underline{t}_i$ the "push-to-the-bottom" operation on index $n+1-i$, i.e.,

$$\underline{t}_i[a_1, a_2, \ldots, a_{n-i}, a_{n+1-i}, a_{n+2-i}, \ldots, a_n] = \\ = [a_1, a_2, \ldots, a_{n-i}, a_{n+2-i}, \ldots, a_n, a_{n+1-i}].$$

Let $d : S \times S \to \mathbb{N} \cup \{0\}$ be a distance function inducing a metric $\mathcal{M}$ over S. Given a transmitted codeword $c \in C$ and its received version $\tilde{c} \in S$, we say a single error occurred if $d(c, \tilde{c}) = 1$. We are interested in Gray codes capable of detecting single errors, which we now define.

Definition 2. *Let $\mathcal{M}$ be a metric over S induced by a distance measure d. A* snake-in-the-box code *over $\mathcal{M}$ and S, using transitions T, is a Gray code C over S and using T, in which for every pair of distinct elements $c, c' \in C$, $c \neq c'$, one has $d(c, c') \geqslant 2$.*

Since throughout this paper our ambient space is S_n, and the transformations we use are the "push-to-the-top" operations T, we shall abbreviate our notation and call a snake-in-the-box code of size M an $(n, M, \mathcal{M})$-*snake*, or an $\mathcal{M}$-*snake*. We will be considering two metrics in the next sections: Kendall's τ-metric, $\mathcal{K}$, and the ℓ_∞-metric, with their respective $\mathcal{K}$-snakes and ℓ_∞-snakes.

III. Kendall's τ-Metric and $\mathcal{K}$-Snakes

Kendall's τ-metric [13], denoted $\mathcal{K}$, is induced by the bubble-sort distance which measures the minimal amount of adjacent transpositions required to transform one permutation into the other. For example, the distance between the permutations $[2,1,4,3]$ and $[2,4,3,1]$ is 2, as

$$[2,1,4,3] \to [2,4,1,3] \to [2,4,3,1]$$

is a shortest sequence of adjacent transpositions between the two. More formally, for $\alpha, \beta \in S_n$, as noted in [12],

$$d_K(\alpha, \beta) = |\{(i,j) \mid \alpha(i) < \alpha(j) \wedge \beta(i) > \beta(j)\}|\,.$$

The metric $\mathcal{K}$ was first introduced by Kendall [13] in the study of ranking in statistics. It was observed in [12] that a bounded distance in Kendall's τ-metric models errors caused

by bounded changes in charge-levels of cells in the flash memory. Error-correcting codes for this metric were studied in [2], [12], [15].

We let Kendall's τ *adjacency graph* of order $n \in \mathbb{N}$ be the graph $G_n = (V_n, E_n)$ whose vertices are permutations on n elements (i.e., $V_n = S_n$), and $\{\alpha, \beta\} \in E_n$ if and only if $d_K(\alpha, \beta) = 1$. It is well known that Kendall's τ-metric is *graphic* [5], i.e., for every $\alpha, \beta \in S_n$, $d_K(\alpha, \beta)$ equals the length of the shortest path between the two in G_n.

A. Construction

We begin the construction process by restricting ourselves to Gray codes using only "push-to-the-top" operations on odd indices. The following lemma provides the motivation for this restriction.

Lemma 3. *A Gray code over S_n using only "push-to-the-top" operations on odd indices is a $\mathcal{K}$-snake.*

Lemma 3 saves us the need to check whether a Gray code is in fact a $\mathcal{K}$-snake, at the cost of restricting the set of allowed transitions (and the size of the resulting code, although Theorems 12,13, presented below, work to mitigate this concern). In particular, if n is even, the last element cannot be moved.

By starting with an even permutation, and using only "push-to-the-top" operations on odd indices, we get a sequence of even permutations. Thus, throughout this part, the context of the alternating group A_{2n+1} is assumed, where $n \in \mathbb{N}$.

The construction we are about to present is recursive in nature. As a base for the recursion, we note that three consecutive "push-to-the-top" operations on the 3rd index of permutations in A_3 constitute a complete cyclic $(3, 3, \mathcal{K})$-snake:

$$C_3 = ([1,2,3], [3,1,2], [2,3,1]).$$

We shall extend C_3 to the next order as a running example alongside the general construction below.

Now, assume that there exists a cyclic $(2n-1, M_{2n-1}, \mathcal{K})$-snake, C_{2n-1}, and let $t_{k_1}, t_{k_2}, \ldots, t_{k_{M_{2n-1}}}$ be the sequence of transformations generating it, where k_j is odd for all $j \in [M_{2n-1}]$. We also assume that $k_1 = 2n-1$ (this requirement, while perhaps appearing arbitrary, is actually quite easily satisfied. Indeed, every sufficiently large cyclic $\mathcal{K}$-snake over S_{2n-1} must, w.l.o.g., satisfy it. We shall make it a point to demonstrate that this holds for our construction).

We fix arbitrary values for $a_0, a_1, \ldots, a_{2n-2}$ such that

$$\{a_0, a_1, \ldots, a_{2n-2}\} = [2n+1] \setminus \{1, 3\}. \tag{2}$$

For all $i \in [2n-1]$ we define

$$\sigma_0^{(i)} = [1, a_i, 3, a_{i+1}, \ldots, a_{i+2n-2}],$$

where the indices are taken modulo $2n-1$, and such that we indeed have $\sigma_0^{(i)} \in A_{2n+1}$, i.e., $\sigma_0^{(i)}$ is an even permutation (one simple way of achieving this is to choose them in ascending order).

Example 4. *We recall that C_3 is generated by the operations (t_3, t_3, t_3), which satisfy our requirement. As suggested above, we order $[5] \setminus \{1,3\}$ in ascending order, i.e., $(a_0, a_1, a_2) = (2, 4, 5)$. We define the following permutations as starting points for our construction*

$$\sigma_0^{(0)} = \sigma_0^{(3)} = [1,2,3,4,5] \quad \sigma_0^{(1)} = [1,4,3,5,2] \quad \sigma_0^{(2)} = [1,5,3,2,4]$$

and readily verify that they are all even. □

We now define for all $i \in [2n-1]$ and $j \in [M_{2n-1}]$ the permutation

$$\sigma_{j(2n+1)}^{(i)} = \underline{t}_{k_j}\left(\sigma_{(j-1)(2n+1)}^{(i)}\right),$$

i.e., we construct cycles corresponding to a mirror view of C_{2n-1} on all but the two first elements of $\sigma_0^{(i)}$ (which, as we recall, are $(1, a_i)$).

Example 5. *In our running example, we define the following permutations:*

$$\begin{array}{ll} \sigma_5^{(0)} = \underline{t}_3\sigma_0^{(0)} = [1,2,4,5,3] & \sigma_5^{(1)} = \underline{t}_3\sigma_0^{(1)} = [1,4,5,2,3] \\ \sigma_{10}^{(0)} = \underline{t}_3\sigma_5^{(0)} = [1,2,5,3,4] & \sigma_{10}^{(1)} = \underline{t}_3\sigma_5^{(1)} = [1,4,2,3,5] \\ \sigma_{15}^{(0)} = \underline{t}_3\sigma_{10}^{(0)} = [1,2,3,4,5] & \sigma_{15}^{(1)} = \underline{t}_3\sigma_{10}^{(1)} = [1,4,3,5,2] \end{array}$$

$$\begin{array}{l} \sigma_5^{(2)} = \underline{t}_3\sigma_0^{(2)} = [1,5,2,4,3] \\ \sigma_{10}^{(2)} = \underline{t}_3\sigma_5^{(2)} = [1,5,4,3,2] \\ \sigma_{15}^{(2)} = \underline{t}_3\sigma_{10}^{(2)} = [1,5,3,2,4] \end{array}$$

and resume our construction. □

We now note the following properties of our construction:

Lemma 6. *Let $i, k \in [2n-1]$ and $j, l \in [M_{2n-1}]$. The following are equivalent:*

1) *The permutations $\sigma_{j(2n+1)}^{(i)}$ and $\sigma_{l(2n+1)}^{(k)}$ are cyclic shifts of each other.*
2) *$\sigma_{j(2n+1)}^{(i)} = \sigma_{l(2n+1)}^{(k)}$.*
3) *$i = k$ and $j = l$.*

Proof: First, if $\sigma_{j(2n+1)}^{(i)}$ is a cyclic shift of $\sigma_{l(2n+1)}^{(k)}$, since

$$\sigma_{j(2n+1)}^{(i)}(1) = 1 = \sigma_{l(2n+1)}^{(k)}(1)$$

then necessarily

$$\sigma_{j(2n+1)}^{(i)} = \sigma_{l(2n+1)}^{(k)}.$$

It then follows that

$$a_i = \sigma_{j(2n+1)}^{(i)}(2) = \sigma_{l(2n+1)}^{(k)}(2) = a_k,$$

hence $i = k$. Moreover, since the two permutations' last $n-1$ elements agree, and $t_{k_1}, t_{k_2}, \ldots, t_{k_{M_{2n-1}}}$ induce a Gray code, we necessarily have $j = l$.

Finally, that the last statement implies the first is trivial. ■

Lemma 7. *For all $i \in [2n-1]$ it holds that*

$$\sigma_{M_{2n-1}(2n+1)}^{(i)} = \sigma_0^{(i)}.$$

Proof: The transformations $t_{k_1}, t_{k_2}, \ldots, t_{k_{M_{2n-1}}}$ induce a cyclic code, and the claim follows directly. ■

Therefore we have constructed $2n-1$ cycles comprised of cyclically non-equivalent permutations (although, at this point they are not generated by "push-to-the-top" operations).

It shall now be noted that

$$\underline{t}_k = t_{2n+1}^{2n} t_{2n+2-k}.$$

Hence, if we define for all $i \in [2n-1]$, $0 \leqslant j < M_{2n-1}$, and $1 < m \leqslant 2n$, the permutations

$$\sigma^{(i)}_{j(2n+1)+1} = t_{2n+2-k_{j+1}} \sigma^{(i)}_{j(2n+1)}$$
$$\sigma^{(i)}_{j(2n+1)+m} = t_{2n+1}^{m-1} \sigma^{(i)}_{j(2n+1)+1},$$

then it holds that

$$\sigma^{(i)}_{(j+1)(2n+1)} = t_{2n+1} \sigma^{(i)}_{j(2n+1)+2n}.$$

Our observation from one paragraph above means that at this point we have $2n-1$ disjoint cycles, which we conveniently denote

$$C^{(i)}_{2n+1} = \left(\sigma^{(i)}_0, \sigma^{(i)}_1, \ldots, \sigma^{(i)}_{M_{2n-1}(2n+1)-1}\right),$$

for all $i \in [2n-1]$ (for ease of notation, we let $C^{(0)}_{2n+1} = C^{(2n-1)}_{2n+1}$).

Example 8. *In our construction, the cycles we produced are:*

$$\begin{array}{lll|l}
\sigma_0^{(0)} = [\mathbf{1,2,3,4,5}] & \sigma_0^{(1)} = [\mathbf{1,4,3,5,2}] & \sigma_0^{(2)} = [\mathbf{1,5,3,2,5}] & \downarrow t_3 \\
\sigma_1^{(0)} = [3,1,2,4,5] & \sigma_1^{(1)} = [3,1,4,5,2] & \sigma_1^{(2)} = [3,1,5,2,4] & \downarrow t_5 \\
\sigma_2^{(0)} = [5,3,1,2,4] & \sigma_2^{(1)} = [2,3,1,4,5] & \sigma_2^{(2)} = [4,3,1,5,2] & \downarrow t_5 \\
\sigma_3^{(0)} = [4,5,3,1,2] & \sigma_3^{(1)} = [5,2,3,1,4] & \sigma_3^{(2)} = [2,4,3,1,5] & \downarrow t_5 \\
\sigma_4^{(0)} = [2,4,5,3,1] & \sigma_4^{(1)} = [4,5,2,3,1] & \sigma_4^{(2)} = [5,2,4,3,1] & \downarrow t_5 \\
\sigma_5^{(0)} = [\mathbf{1,2,4,5,3}] & \sigma_5^{(1)} = [\mathbf{1,4,5,2,3}] & \sigma_5^{(2)} = [\mathbf{1,5,2,4,3}] & \downarrow t_3 \\
\sigma_6^{(0)} = [4,1,2,5,3] & \sigma_6^{(1)} = [5,1,4,2,3] & \sigma_6^{(2)} = [2,1,5,4,3] & \downarrow t_5 \\
\sigma_7^{(0)} = [3,4,1,2,5] & \sigma_7^{(1)} = [3,5,1,4,2] & \sigma_7^{(2)} = [3,2,1,5,4] & \downarrow t_5 \\
\sigma_8^{(0)} = [5,3,4,1,2] & \sigma_8^{(1)} = [2,3,5,1,4] & \sigma_8^{(2)} = [4,3,2,1,5] & \downarrow t_5 \\
\sigma_9^{(0)} = [2,5,3,4,1] & \sigma_9^{(1)} = [4,2,3,5,1] & \sigma_9^{(2)} = [5,4,3,2,1] & \downarrow t_5 \\
\sigma_{10}^{(0)} = [\mathbf{1,2,5,3,4}] & \sigma_{10}^{(1)} = [\mathbf{1,4,2,3,5}] & \sigma_{10}^{(2)} = [\mathbf{1,5,4,3,2}] & \downarrow t_3 \\
\sigma_{11}^{(0)} = [5,1,2,3,4] & \sigma_{11}^{(1)} = [2,1,4,3,5] & \sigma_{11}^{(2)} = [4,1,5,3,2] & \downarrow t_5 \\
\sigma_{12}^{(0)} = [4,5,1,2,3] & \sigma_{12}^{(1)} = [5,2,1,4,3] & \sigma_{12}^{(2)} = [2,4,1,5,3] & \downarrow t_5 \\
\sigma_{13}^{(0)} = [3,4,5,1,2] & \sigma_{13}^{(1)} = [3,5,2,1,4] & \sigma_{13}^{(2)} = [3,2,4,1,5] & \downarrow t_5 \\
\sigma_{14}^{(0)} = [2,3,4,5,1] & \sigma_{14}^{(1)} = [4,3,5,2,1] & \sigma_{14}^{(2)} = [5,3,2,4,1] & \circlearrowleft t_5
\end{array}$$

where the permutations in bold are those from Example 5. □

Each of the cycles of size $(2n+1)M_{2n-1}$, is generated by "push-to-the-top" operations, and contains all cyclic shifts of elements present in our previous version of that cycle. We merge these cycles into a single cycle in the following theorem.

Theorem 9. *Given a cyclic $(2n-1, M_{2n-1}, \mathcal{K})$-snake using only "push-to-the-top" operations on odd indices such that its first transformation is t_{2n-1}, there exists a cyclic $\mathcal{K}$-snake over S_{2n+1} with the same properties, whose size is*

$$M_{2n+1} = (2n-1)(2n+1)M_{2n-1}.$$

Proof: Since $k_1 = 2n-1$, it holds for all $i \in [2n-1]$ that $\sigma_1^{(i)} = t_3 \sigma_0^{(i)}$, and we recall $\sigma_2^{(i)} = t_{2n+1}\sigma_1^{(i)}$. More explicitly,

$$\sigma_1^{(i)} = [3, 1, a_i, a_{i+1}, \ldots, a_{i+2n-2}]$$
$$\sigma_2^{(i)} = [a_{i+2n-2}, 3, 1, a_i, a_{i+1}, \ldots, a_{i+2n-3}],$$

where, again, the indices are taken modulo $2n-1$. Thus for all $i \in [2n-2]$ we have

$$t_3\sigma_1^{(i)} = [a_i, 3, 1, a_{i+1}, \ldots, a_{i+2n-2}] = \sigma_2^{(i+1)}$$

and $t_3\sigma_1^{(2n-1)} = \sigma_2^{(1)}$.

Let E denote the left-shift operator, and so

$$E^2 C^{(i)}_{2n+1} = \left(\sigma_2^{(i)}, \sigma_3^{(i)}, \ldots, \sigma^{(i)}_{M_{2n-1}(2n+1)-1}, \sigma_0^{(i)}, \sigma_1^{(i)}\right).$$

By the observations made above we conclude that

$$C_{2n+1} = E^2 C^{(0)}_{2n+1}, E^2 C^{(1)}_{2n+1}, \ldots, E^2 C^{(2n-2)}_{2n+1}$$

is a cyclic $(2n+1, M_{2n+1}, \mathcal{K})$-snake, consisting of

$$M_{2n+1} = (2n-1)(2n+1)M_{2n-1}$$

permutations. A careful observation readily shows that C_{2n+1} has t_{2n+1} for its first generating transformation. ■

Example 10. *Our running example ends with the full construction of a $(5, 45, \mathcal{K})$-snake, C_5, from Theorem 9. The down arrows stand for an omitted sequence of t_5 transformations. The transition from column to column uses a single t_3 transformation.*

$$\begin{array}{c|c|c|c|c|c|c|c}
[5,3,1,2,4] & \sigma_2^{(0)} & [2,3,1,4,5] & \sigma_2^{(1)} & [4,3,1,5,2] & \sigma_2^{(2)} \\
\downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\
[1,2,4,5,3] & \sigma_5^{(0)} & [1,4,5,2,3] & \sigma_5^{(1)} & [1,5,2,4,3] & \sigma_5^{(2)} \\
[4,1,2,5,3] & \sigma_6^{(0)} & [5,1,4,2,3] & \sigma_6^{(1)} & [2,1,5,4,3] & \sigma_6^{(2)} \\
\downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\
[1,2,5,3,4] & \sigma_{10}^{(0)} & [1,4,2,3,5] & \sigma_{10}^{(1)} & [1,5,4,3,2] & \sigma_{10}^{(2)} \\
[5,1,2,3,4] & \sigma_{11}^{(0)} & [2,1,4,3,5] & \sigma_{11}^{(1)} & [4,1,5,3,2] & \sigma_{11}^{(2)} \\
\downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\
[1,2,3,4,5] & \sigma_0^{(0)} & [1,4,3,5,2] & \sigma_0^{(1)} & [1,5,3,2,4] & \sigma_0^{(2)} \\
[3,1,2,4,5] & \sigma_1^{(0)} & [3,1,4,5,2] & \sigma_1^{(1)} & [3,1,5,2,4] & \sigma_1^{(2)}
\end{array}$$

□

We now turn to consider the size and rate of the constructed codes, and show that their rate asymptotically tends to 1.

Theorem 11. *The size of $\mathcal{K}$-snakes constructed in Theorem 9 behaves asymptotically as*

$$|C_{2n+1}| = M_{2n+1} = \frac{(2n)!(2n+1)!}{n!^2 \cdot 2^{2n}} \sim \frac{1}{\sqrt{\pi n}} |S_{2n+1}|,$$

which leads to an asymptotic rate of 1.

One observes that the codes C_{n+1} utilize the transformation t_{2n+1} overwhelmingly more than any other. This property allows one to calculate the transformation required to advance any given permutation in the code to its successor in $O(1)$ amortized run time. In addition, the recursive nature of Theorem 9 lends itself to the ranking and unranking of permutations in the code (that is, the processes of attaching to a given permutation its position in the code, and vice versa) in $O(n^2)$ run time. Methods of achieving these tasks are presented and analyzed in [22]. Together, they facilitate the use of the codes C_{2n+1} in the implementation of logic memory cells, by allowing one to increase the cell's 'level' as well as directly write data to it (and naturally, to read written data as well).

IV. Conclusion

In this paper we explored rank-modulation snake-in-the-box codes under Kendall's τ-metric, and presented a construction yielding codes with rates asymptotically tending to 1. Some results w.r.t. bounds on the size of such codes were also proven in [22], which can be summarized by the following two theorems:

Theorem 12. *If C is an $(n, M, \mathcal{K})$-snake then*

1) $M \leqslant \frac{1}{2}|S_n|$.
2) $M = \frac{1}{2}|S_n|$ *if and only if for all* $\{\alpha, \beta\} \in E_n$ *it holds that* $\alpha \in C$ *or* $\beta \in C$.

Theorem 13. *If an $(n, M, \mathcal{K})$-snake C contains a "push-to-the-top" operation on an even index then*

$$M \leqslant \frac{1}{2}|S_n| - \frac{1}{n-1}\binom{n-3}{2}.$$

However, it is not presently known whether these bounds are achievable, and therefore we are unable to determine how close the codes generated by our construction come to being optimal with respect to their sizes (rather than their asymptotic rates). A computer search for *cyclic* codes, performed on S_5, yielded $(5, M, \mathcal{K})$-snakes of maximal size $M = 57$ (for comparison, the construction from Theorem 9 yields a $(5, 45, \mathcal{K})$-snake). While an abundance of such codes were found (well over 500 nonequivalent codes), they all were in fact codes over A_5.

It shall be noted that a complete (but not cyclic) $(5, 60, \mathcal{K})$-snake over A_5 can also be constructed by amending the code presented in Example 10. However, we do not currently know whether $(2n+1, \frac{(2n+1)!}{2}, \mathcal{K})$-snakes over A_{2n+1} exist for every length.

These results, along with the bounds we showed in Theorems 12,13 give rise to the following conjecture: For all $n \in \mathbb{N}$ a $\mathcal{K}$-snake exists over A_n whose size is no less than that of every $\mathcal{K}$-snake over S_n.

In addition, [22] explores rank-modulation snake-in-the-box codes using a different metric, the ℓ_∞-metric, which is induced by the embedding of S_n in $\mathbb{Z}^n$. More precisely, for $\alpha, \beta \in S_n$ one defines

$$d_\infty(\alpha, \beta) = \max_{i \in [n]} |\alpha(i) - \beta(i)|.$$

We use the ℓ_∞-metric to model a different kind of noise-mechanism than that modeled by Kendall's τ-metric, namely spike noise. In this model, the rank of each memory cell is assumed to have been changed by a bounded amount (see [19]). Under this metric, the authors were able to present a construction which gives rise to the following theorem:

Theorem 14. *For all $4 \leqslant n \in \mathbb{N}$ there exists an (n, M, ℓ_∞)-snake of size*

$$M = \left\lceil \frac{n}{2} \right\rceil ! \left(\left\lfloor \frac{n}{2} \right\rfloor + \left(\left\lfloor \frac{n}{2} \right\rfloor - 1 \right)! \right).$$

And it was again shown that these codes have rates asymptotically tending to 1.

References

[1] H. L. Abbot and M. Katchalski, "On the construction of snake in the box codes," *Utilitas Math.*, vol. 40, pp. 97–116, 1991.

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

[3] J. Brewer and M. Gill, *Nonvolatile Memory Technologies with Emphasis on Flash*. Wiley-IEEE Press, 2008.

[4] F. Chierichetti, H. Finucane, Z. Liu, and M. Mitzenmacher, "Designing floating codes for expected performance," *IEEE Trans. on Inform. Theory*, vol. 56, no. 3, pp. 968–978, Mar. 2010.

[5] M. Deza and H. Huang, "Metrics on permutations, a survey," *J. Comb. Inf. Sys. Sci.*, vol. 23, pp. 173–185, 1998.

[6] E. En Gad, A. Jiang, and J. Bruck, "Compressed encoding for rank modulation," in *Proceedings of the 2011 IEEE International Symposium on Information Theory (ISIT2011), St. Petersburg, Russia*, Aug. 2011, pp. 884–888.

[7] E. En Gad, M. Langberg, M. Schwartz, and J. Bruck, "Constant-weight Gray codes for local rank modulation," *IEEE Trans. on Inform. Theory*, vol. 57, no. 11, pp. 7431–7442, Nov. 2011.

[8] F. Gray, "Pulse code communication," March 1953, U.S. Patent 2632058.

[9] A. Jiang, V. Bohossian, and J. Bruck, "Rewriting codes for joint information storage in flash memories," *IEEE Trans. on Inform. Theory*, vol. 56, no. 10, pp. 5300–5313, Oct. 2010.

[10] A. Jiang, M. Langberg, M. Schwartz, and J. Bruck, "Universal rewriting in constrained memories," in *Proceedings of the 2009 IEEE International Symposium on Information Theory (ISIT2009), Seoul, Korea*, Jun. 2009, pp. 1219–1223.

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

[12] A. Jiang, M. Schwartz, and J. Bruck, "Correcting charge-constrained errors in the rank-modulation scheme," *IEEE Trans. on Inform. Theory*, vol. 56, no. 5, pp. 2112–2120, May 2010.

[13] M. Kendall and J. D. Gibbons, *Rank Correlation Methods*. Oxford University Press, NY, 1990.

[14] T. Kløve, T.-T. Lin, S.-C. Tsai, and W.-G. Tzeng, "Permutation arrays under the Chebyshev distance," *IEEE Trans. on Inform. Theory*, vol. 56, no. 6, pp. 2611–2617, Jun. 2010.

[15] A. Mazumdar, A. Barg, and G. Zémor, "Constructions of rank modulation codes," in *Proceedings of the 2011 IEEE International Symposium on Information Theory (ISIT2011), St. Petersburg, Russia*, Aug. 2011, pp. 834–838.

[16] N. Papandreou, H. Pozidis, T. Mittelholzer, G. F. Close, M. Breitwisch, C. Lam, and E. Eleftheriou, "Drift-tolerant multilevel phase-change memory," in *Proceedings of the 3rd IEEE International Memory Workshop (IMW), Monterey, CA, U.S.A.*, May 2011, pp. 22–25.

[17] C. D. Savage, "A survey of combinatorial Gray codes," *SIAM Rev.*, vol. 39, no. 4, pp. 605–629, Dec. 1997.

[18] R. Sedgewick, "Permutation generation methods," *Computing Surveys*, vol. 9, no. 2, pp. 137–164, Jun. 1977.

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

[20] Z. Wang and J. Bruck, "Partial rank modulation for flash memories," in *Proceedings of the 2010 IEEE International Symposium on Information Theory (ISIT2010), Austin, TX, U.S.A.*, Jun. 2010, pp. 864–868.

[21] E. Yaakobi, A. Vardy, P. H. Siegel, and J. K. Wolf, "Multidimensional flash codes," in *Proc. of the Annual Allerton Conference*, 2008.

[22] Y. Yehezkeally and M. Schwartz, "Snake-in-the-box codes for rank modulation," *IEEE Trans. on Inform. Theory*, 2012.