Limited-Magnitude Error-Correcting Gray Codes for Rank Modulation

Yonatan Yehezkeally, Student Member, IEEE, Moshe Schwartz, Senior Member, IEEE

Abstract—We construct error-correcting codes over permutations under the infinity-metric, which are also Gray codes in the context of rank modulation, i.e., are generated as simple circuits in the rotator graph. These errors model limited-magnitude or spike errors, for which only single-error-detecting Gray codes are currently known. Surprisingly, the error-correcting codes we construct achieve a better asymptotic rate than that of presently known constructions not having the Gray property, and exceed the Gilbert-Varshamov bound. Additionally, we present efficient ranking and unranking procedures, as well as a decoding procedure that runs in linear time. Finally, we also apply our methods to solve an outstanding issue with error-detecting rank-modulation Gray codes (also known in this context as snake-in-the-box codes) under a different metric, the Kendall $\tau$-metric, in the group of permutations over an even number of elements $S_{2n}$, where we provide asymptotically optimal codes.

Index Terms—Gray codes, error-correcting codes, permutations, spread-$d$ circuit codes, rank modulation

I. INTRODUCTION

Rank modulation is a method for storing information in non-volatile memories [23], which has been researched in recent years. It calls for encoding information in relative values in a group of cells rather than the absolute values of each single cell. More precisely, it stores information in the permutation suggested by sorting a group of cells by their relative values; Such values may be charge levels in flash memory cells or electrical resistance in phase-change memory [30]. Rank modulation allows for increased robustness against certain noise mechanisms (e.g., charge leakage in flash memory cells), as well as alleviating some inherent challenges in flash memories (e.g., programming/eraser-asymmetry and programming-overshoot). Permutation codes in general have also previously been seen usages in source-encoding [3]–[5], [38] and signal detection [7], as well as other fields [6], [9], [11], and more recently been used in power-line communications [41].

Several error models have been studied for rank modulation, including the Kendall $\tau$-metric [2], [24], [28], [47], the $\ell_\infty$-metric [27], [36], [39], [40] and other examples [12], [18]. In this paper we focus on the $\ell_\infty$-metric, which models limited-magnitude or spike noise, i.e., we assume that the rank of any given cell–its position when sorting the group of cells–could not have changed by more than a given amount. [27], [39] have presented constructions for error-correcting codes under this metric, as well as explored some non-constructive lower- and upper-bounds on the parameters of existing codes. [40] has since employed methods of relabeling to optimize the minimal distance of known constructions.

In the context of rank modulation, a generalization of the Gray code has been shown to reduce write-time–by eliminating the risk of programming-overshoot–and allow integration with other multilevel-cells coding schemes [13], [14], [23]. Gray codes were first considered over the space of binary vectors, where they were generally defined as a listing of distinct vectors–sometimes exhaustive–such that each pair of consecutive vectors differed by a single bit-flip [19]; the concept has since been generalized in some contexts to include codes over arbitrary alphabets, requiring only that codewords could be ordered in a sequence, where each codeword is derived from the previous by one of a predefined set of functions. Put differently, Gray codes may be considered as simple paths on the digraph whose nodes are elements of the alphabet, and edges are induced by the aforementioned functions set (e.g., Cayley graphs). Suggested usages of Gray codes in contexts other than rank modulation, surveyed in [33], include permanent-computation [29], circuit-testing [31], image-processing [1], hashing [17], coding [15], [23], [34] and data storing/extraction [8]. Within rank modulation, particular Cayley graphs were used, which were first proposed (for use in multiprocessor networks) in [10], [16] as Faber-Moore- or rotator graphs, and later rediscovered (the authors being apparently unaware) for use in Flash memories in [23] (including one of its constructions). These codes are in fact also an example of greedily constructed Gray codes [43].

Gray codes with error-correction capabilities have sometimes been referred to as spread-$d$ circuit codes (see [21] and references therein). Specifically, in the context of rank modulation, such codes were so far only studied for the case of single-error detection, where they were dubbed (see [20]–[22], [42], [44]–[46]) snake-in-the-box codes (or, more appropriately, coil-in-the-box codes, when they are cyclic); this, again, draws on terminology first used with Gray codes in the hypercube, where snake-in-the-box codes are defined as spread-2 codes using the Hamming distance [37]. [44] studied such rank-modulation codes under both the Kendall $\tau$-metric and the $\ell_\infty$-metric, and more recent papers [20], [22], [45] have categorized and constructed optimally sized coil-in-the-box codes under the former metric for odd orders, although the case of even orders proved more challenging (see [46] in
In this work we focus on the \( \ell_{\infty} \)-metric and present a construction of error-correcting Gray codes capable of correcting an arbitrary number of limited-magnitude errors. The allowed transitions between codewords are the “push-to-the-top” operations, used in most previous works [13], [14], [20], [22], [23], [44], [45] (which are isomorphic to the prefix-rotations of [10]). An example of such a code, generated in the paper in Example 19, is presented in Figure 1. The resulting codes will be shown to have greater size than known constructions in the case of fixed minimal distance, as well as achieving better asymptotic rates than known codes in the case of \( d = \Theta(n) \); both size and rate are also compared against known bounds. In particular, in the case of error-detecting \( \sigma \)-constructions, we let \( \tau \in \mathcal{C} \) have the same parity, that is, \( \text{sign}\, \sigma = \text{sign}\, \tau \) (put differently, either \( C \subseteq A_n \) or \( C \subseteq S_n \setminus A_n \)).

We also use the vector notation for permutations, 
\[
\sigma = [\sigma(1), \sigma(2), \ldots, \sigma(n)].
\]

This allows us to more easily note, for \( 1 \leq i < j \leq n \), the “push-to-the-ith-index” transition \( t_{ij} : S_n \to S_n \) by 
\[
t_{ij}(\{a_1, a_2, \ldots, a_{i-1}, a_i, a_{i+1}, \ldots, a_j, a_{j+1}, \ldots, a_n\}) = [a_1, a_2, \ldots, a_{i-1}, a_j, a_i, a_{i+1}, \ldots, a_{j-1}, a_{j+1}, \ldots, a_n].
\]

We follow recent works [22], [23], [44], [45] (among others) in dubbing “push-to-the-1st-index” transitions as “push-to-the-top” transitions (although these operations—or more precisely their inverse—were originally introduced as prefix-rotations in [10]); we denote \( t_{ij} = t_{ij} \). Finally, we define the “push-to-the-bottom” transition on the \( j \)th index, \( t_{j} : S_n \to S_n \), 
\[
t_{j}(\{a_1, a_2, \ldots, a_{j-1}, a_j, a_{j+1}, \ldots, a_n\}) = [a_1, a_2, \ldots, a_{j-1}, a_j, a_{j+1}, \ldots, a_n].
\]

Given any set \( S \), and a collection of transitions
\[
T \subseteq \{ f \mid f : S \to \overline{S} \},
\]
we define a \( T \)-Gray code over the set \( S \) to be a sequence \( C = (c_r)_{r=1}^M \subseteq S \) such that for all \( 1 \leq r \leq r' \leq M \) we have \( c_r \neq c_{r'} \) and such that for all \( 1 \leq r < M \) there exists \( t_r \in T \) satisfying \( c_{r+1} = t_r(c_r) \). We say that a sequence \( C \) is contained in \( S \), by abuse of notation, if \( c_r \in S \) for all \( r \). That is, we may refer to a Gray code as an unordered set—or simply a code—when desired for simplicity. Conversely, we say that a code has the Gray property, or is a Gray code, if it can be so ordered. We call \( M = |C| \) the size of the code, and \( t_1, t_2, \ldots, t_{M-1} \) the transition sequence generating \( C \). If there exists \( t \in T \) such that \( c_1 = t(c_M) \) we say that \( C \) is cyclic, and include \( t_M = t \) in its generating transition sequence. If \( C = S \), we say that \( C \) is a complete code.

**Example 1** In the classic example of a Gray code we have, e.g., \( S = F_2^3 \), with \( T \) consisting of the group action of \( \{001, 010, 100\} \subseteq S \) on \( S \), defined
\[
v(u) = u + v.
\]

We then have the complete cyclic Gray code given by
\[
\begin{array}{cccccccc}
000 & 001 & 010 & 011 & 010 & 001 & 010 & 000 \\
100 & \uparrow & 001 & 010 & 111 & \downarrow 100 & 001 & 110 \\
100 & \downarrow 010 & 001 & 111 & \uparrow 010 & 001 & 110 & 100
\end{array}
\]

In this paper, we fix \( S = S_n \). Since we intend to work with minutely distinct classes of codes on the symmetric group in this paper, we will introduce notations to distinguish them, which are organized in Table 1 for the readers’ comfort. We say that \( C = (c_1, c_2, \ldots, c_M) \subseteq S_n \) is a \( G_t(n, M) \) if it is a cyclic representation (with no particular importance given to which is which). The set of even permutations forms a subgroup \( A_n \subseteq S_n \) named the alternating group. We will say that \( C \subseteq S_n \) is parity-preserving if every two elements \( \sigma, \tau \in C \) have the same parity, that is, \( \text{sign}\, \sigma = \text{sign}\, \tau \) (put differently, either \( C \subseteq A_n \) or \( C \subseteq S_n \setminus A_n \)).
Gray code with transition set \( T = \{ t_{ij} \mid i < j \leq n \} \). When \( i = 1 \) we refer to \( C \) as a “push-to-the-top” code and denote it \( G_1(n, M) \), and we likewise denote “push-to-the-bottom” codes \( G_i(n, M) \).

**Example 2** We observe (a fact that has been remarked in [23]) that

\[
\begin{bmatrix}
1 & 2 & 3 \\
2 & 2 & 3 \\
3 & 2 & 3 \\
\end{bmatrix}
\]

is a \( G_3(3, 6) \), i.e., a complete cyclic “push-to-the-top” Gray code over \( S_3 \).

\[\Box\]

It is worthwhile to note that when \( S \) is a group, and \( T \) consists of the group action of some subset on \( S \), and \( C \) is a (complete- and/or cyclic-) Gray code generated by \( t_1, t_2, \ldots, t_{m-1}, (\ell, t_m) \), then \( C \) can be viewed as a simple path (or circuit) in the Cayley graph with generators from \( T \). Moreover for all \( \sigma \in S \) we observe that \( (\sigma, t_1(\sigma), t_2(t_1(\sigma)), \ldots) \) is also a (complete- and/or cyclic- respectively) Gray code. In other words, the code is shift invariant as Cayley graphs are vertex-transitive. In these cases we might refer to the transition sequence generating the code as the code itself, when desirable for simplicity. It is of particular interest to observe that \( t_{ij}(\sigma) = \sigma \circ (j, j-1, \ldots, i) \), i.e., “push-to-the-ith-index” transitions are indeed group actions, hence we shall make that simplification in places. We remark that the Cayley graphs generated by \( \{ (j, j-1, \ldots, 1) \mid j \in [n] \} \), i.e., prefix-rotations or “push-to-the-top” operations, were named \( (n-1, n) \)-Faber-Moore graphs in [16], or (the transpose/converse to) \( n \)-rotator graphs in [10]; as mentioned above, we follow the terminology used in more recent works.

When \( S \) is equipped with a metric \( d_M : S \times S \rightarrow \mathbb{R}_+ \), and \( C \subseteq S \) has the property that for all \( \sigma, \tau \in C \) either \( \sigma = \tau \) or \( d_M(\sigma, \tau) \geq d \), for some constant \( d > 0 \), then \( C \) (when considered as an unordered set) is commonly referred to as an error-correcting code with minimal distance \( d \). If \( d_M(\cdot, \cdot) \) models an error mechanism, such that a single error corresponds to distance 1, and \( 2p + q < d \), it is well known that \( C \) can then correct \( p \) errors, and also detect \( q \) additional errors (e.g., see [32, Prop. 1.5]).

Error-correcting Gray codes were (as mentioned above) named spread-\( d \) circuit codes (e.g., in [21]), where they were defined by requiring that for all \( c_r, c_r' \in C \),

\[
(r - r' \mod |C|) \geq d \implies d_M(c_r, c_r') \geq d.
\]

In that way, e.g., spread-1 circuit codes are simply Gray codes. This eased requirement was made necessary since, working with the Hamming distance \( \tilde{d} \) in the \( n \)-cube, one cannot have codewords at distance less than \( d \) in the code sequence attain a distance of at least \( \tilde{d} \). We shall depart from it here to deal with Gray codes which are classic error-correcting codes, but the codes presented in this paper are nevertheless also, in particular, spread-\( d \) circuit-codes. This is naturally true in the special case of \( d = 2 \), which to the authors’ knowledge is the only case of error-correcting codes studied thus far in the context of rank modulation with the Gray property. In an analogue to classic Gray codes in the hypercube mentioned above, using the Hamming distance [37], they were dubbed snake-in-the-box codes regardless of the metric being used on \( S_n \), although only two such metrics were considered [44].

We shall focus on the \( \ell_\infty \)-metric defined on \( S_n \) by

\[
d_{\ell_\infty}(\sigma, \tau) = \max_{j \in [n]} |\sigma(j) - \tau(j)|.
\]

That is, it is the metric induced on \( S_n \) by the embedding into \( \mathbb{Z}^n \) (and, indeed, \( \mathbb{R}^n \)) implied by the vector notation, and the \( \ell_\infty \)-metric in these spaces.

**Example 3** In \( S_4 \), the code

\[
\begin{bmatrix}
1 & 2 & 3 & 4 \\
2 & 1 & 3 & 4 \\
3 & 1 & 2 & 4 \\
4 & 1 & 2 & 3 \\
\end{bmatrix}
\]

has minimal distance 2 (e.g., \( d_{\ell_\infty}([1, 2, 3, 4], [2, 3, 1, 4]) = 2 \), but as 4 is fixed no two codewords have distance 3). In contrast,

\[
\begin{bmatrix}
1 & 2 & 3 & 4 \\
2 & 1 & 4 & 3 \\
3 & 1 & 2 & 4 \\
4 & 1 & 3 & 2 \\
\end{bmatrix}
\]

has minimal distance 3 (which we can verify to be the diameter of the metric space \( (S_4, d_{\ell_\infty}) \)).

\[\Box\]

Error-correcting codes in \( S_n \) with \( d_{\ell_\infty} \) were studied in [39], where they were dubbed limited-magnitude rank-modulation codes. A code \( C \) with minimal distance \( d \) was denoted as an \( (n, |C|, d) \)-LMRM code. In our case, if a \( G_1(n, M) \) is also an \( (n, M, d) \)-LMRM code, we shall denote it a \( G_1(n, M, d) \) (likewise for \( G_1 \) and \( G_1 \)).

III. Auxiliary construction

Before we present the main construction of our paper, we first describe in this section a construction for auxiliary codes which will be a component of the main construction.
TABLE I

<table>
<thead>
<tr>
<th>Notation</th>
<th>Code notations for $C \subseteq S_n$.</th>
</tr>
</thead>
<tbody>
<tr>
<td>$G_{i1}(n, M)$</td>
<td>$C = (c_r)^{M}<em>{i=1} \subseteq S_n$ such that for all $r$: $c</em>{(r \mod M + i - 1)} = f_{i1}(c_r)$.</td>
</tr>
<tr>
<td>$G_{i1}(n, M)$</td>
<td>$C$ is a $G_{i1}(n, M)$.</td>
</tr>
<tr>
<td>$G_{i1}(n, M)$</td>
<td>$C = (c_r)^{M}<em>{i=1} \subseteq S_n$ such that for all $r$: $c</em>{(r \mod M + i - 1)} = f_{i1}(c_r)$.</td>
</tr>
<tr>
<td>$(n, M, d)$-LMRM</td>
<td>$C \subseteq S_n$, $</td>
</tr>
<tr>
<td>$G_{i1}(n, M, d)$</td>
<td>$C$ is a $G_{i1}(n, M)$ and an $(n, M, d)$-LMRM.</td>
</tr>
<tr>
<td>$G_{i1}^{\text{aux}}(k, M)$</td>
<td>$C$ is a $G_{i1}(k, M)$, beginning with $(\text{Id}, t_{12} \text{Id})$, and for all $q \in [k - 1]$: $\sigma \in C \implies (q, k) \sigma \not\in C$. (See Section III.)</td>
</tr>
</tbody>
</table>

We say that $C \subseteq S_k$ is $j$-nontransposing, for some $j \in [k]$, if for all $q \in [k] \setminus \{j\}$ it holds that $\sigma \in C \implies (q, j) \sigma \not\in C$. Unlike some of the codes mentioned thus far, if we shift the first permutation of a $j$-nontransposing $G_{i1}(k, M)$ code $C$, or rotate the generating sequence of transitions, it is no longer assured that the resulting code will be $j$-nontransposing. We therefore make further requirements and define an auxiliary code $G_{i1}^{\text{aux}}(k, M)$ as a $G_{i1}(k, M)$ which is $k$-nontransposing, beginning at Id, and its first transition is $t_{12}$. We will use such codes in our main construction, and we therefore study their existence.

Firstly, note that the only existing $G_{i1}^{\text{aux}}(2, M)$ codes are the singleton sets $\{\text{Id}\}, \{(1, 2)\}$. However, for $k \geq 3$ there do exist $G_{i1}^{\text{aux}}(k, M)$ codes with $M \geq 3$, as one such example is $\left(\text{Id}, t_{13} \text{Id}, t_{12}^2 \text{Id}\right)$. We also note the following:

**Lemma 4** If $C \subseteq S_k$ is $k$-nontransposing, then $M \leq \frac{|S_k|}{2}$. **Proof:** Take $q \in [k - 1]$, and observe that $\sigma \mapsto (q, k) \sigma$ is an $S_k$-automorphism, under which $C$ and its image are disjoint. Hence $2M \leq |S_k|$. This motivates us to examine another family of codes, namely, parity-preserving codes, due to the following observations.

**Lemma 5** If $C \subseteq S_k$ is parity-preserving then $|C| \leq \frac{|S_k|}{2}$. **Proof:** Either $C \subseteq A_k$ or $C \subseteq S_k \setminus A_k$. It is well-known that $|A_k| = \frac{|S_k|}{2}$. **Lemma 6** If $C \subseteq S_k$ is a parity-preserving $G_{i1}(k, M)$, then $C$ is $k$-nontransposing. **Proof:** Take $\sigma \in C$ and observe that sign $\sigma \neq \text{sign}(q, k) \sigma$ for all $q \in [k - 1]$, hence $(q, k) \sigma \not\in C$, since $C$ is parity-preserving.

Parity-preserving $G_{i1}^{\text{aux}}(2m + 1, M)$ codes are known to exist, achieving the aforementioned bound.

**Lemma 7** [20] For all $m \neq 2$, there exist parity-preserving $G_{i1}(2m + 1, \frac{(2m + 1)!}{2})$ codes. The largest parity-preserving $G_{i1}(5, M)$ codes have $M = 57$.

Although not declared, it is shown in [20] that such codes can be assumed to have $t_{2(m+1)}$ as the first transition in their generating transition sequence, and furthermore, that they also employ at least one $t_{2m-1}$ transition.

In comparison, as noted in [44], a parity-preserving $G_{i1}(2m, M)$ must satisfy $M \leq \frac{|S_{2m}|}{2m}$, as it must never employ a $t_{2m}$ transition. This evidently yields much smaller codes than the case of odd orders, and we therefore examine more general $G_{i1}^{\text{aux}}$ codes, which are not parity-preserving. We begin by noting the following lemma.

**Lemma 8** [10, Thm. 4] For all $n \geq 1$ there exist $G_{i1}(n, n!)$ codes, that is, complete and cyclic “push-to-the-top” Gray codes over the symmetric group $S_n$.

Relying on these codes, we construct auxiliary codes in the following theorem. The method we apply here of using “push-to-the-bottom” transitions was also used in [10], [23] as a building block for their proposed constructions of a complete and cyclic Gray code in $S_n$ (which are equivalent), then later in [44] for parity-preserving codes in $S_{2m+1}$.

**Theorem 9** For all $m \geq 2$ there exists a $G_{i1}^{\text{aux}}(2m, \frac{|S_{2m}|}{2m-1})$.

**Proof:** Take a $G_{i1}(2m - 2, (2m - 2)!)$ code $C'$, provided by Lemma 8. We follow the concept of [23, Thm. 7] in extending $C'$ to $S_{2m}$. Let us define $\sigma_0 = t_{12m} \text{Id} = [2m, 1, \ldots, 2m - 1]$. If we take $t_{1t_{11}}, t_{1t_{12}}, \ldots, t_{1t_{1(2m-2)}}$ to be the transition sequence generating $C'$, then the transition sequence $t_{1(2m+1-1)}, t_{1(2m+1-2)}, \ldots, t_{1(2m+1-2m)}$ of “push-to-the-bottom” operations, applied in succession to $\sigma_0$, generates $C'' \subseteq S_{2m}$, a $G_{i1}(2m, (2m - 2)!)$, all of whose elements’ vector notations begin with $[2m, 1]$.

We now note that $t_{1(2m+1-j)} = t_{1(2m-1-j)} \cdot t_{1j}$. Thus, by replacing each $t_{1(2m+1-j)}$ with $t_{1j}$ followed by a sequence of $2m - 1$ occurrences of $t_{12m}$, we get $C \subseteq S_{2m}$, a $G_{i1}(2m, (2m - 2)!2m)$, where every block of $2m$ elements is comprised of cyclic shifts of some $\sigma \in C''$.

The code $C$ is known to be a Gray code [23, Thm. 7]. Moreover, if $\sigma \in C$ satisfies $\tau = (q, 2m) \sigma \in C$, note that both have a vector notation with 1 immediately (cyclically) following 2m, but since $\tau = (q, 2m) \sigma$ its vector notation has 1 following $q$. It follows (by abuse of notation) that $q = 2m$. Hence, $C$ is $2m$-nontransposing.

Finally, note that $C$ is generated by a transition sequence ending with $2m - 1$ instances of $t_{12m}$, and begins with $\sigma_0 = t_{12m} \text{Id}$. Therefore, it includes Id followed by a $t_{12m}$ transition. A cyclic shift of $C$ therefore satisfies the theorem.

**Example 10** To construct a $G_{i1}^{\text{aux}}(4, 8)$ we utilize the complete $G_{i1}(2, 2)$ code $\{\text{Id}, t_{12} \text{Id}\}$, generated by $t_{12}, t_{12}$, to arrive by the $G_{i1}(4, 2)$ code starting with $\sigma_0 = t_{14} \text{Id} = [4, 1, 2, 3]$: $C'' = \{[4, 1, 2, 3], [4, 1, 3, 2]\}$.
which is generated by $t_{13}, t_{13}$. We recall that $t_{13} = t_{13} \circ t_{13}$, allowing us to expand $\mathcal{C}_r$ in the following manner:

\[
\begin{array}{|c|c|c|}
\hline
\gamma & \delta & \epsilon \\
\hline
2 & 4 & 3 \\
4 & 3 & 2 \\
3 & 1 & 4 \\
\hline
\end{array}
\]

Finally, we observe that as seen in the proof to Theorem 9, the code we constructed has $\text{Id}$ as its last codeword, followed by a $t_{13}$ operation. A cyclic shift of the code now begins with $\text{Id}$ and the required operation, which satisfies Theorem 9. \( \square \)

We remark that, while Theorem 9 does not produce codes much larger than the parity-preserving code of size $\frac{|S_{2m}|}{2m}$, it does at least allow us to permute the last element $2m$ while preserving the property of being $2m$-nontransposing, and thus construct auxiliary codes.

Next, we present another construction which yields larger codes, for even $k \geq 6$ (but not $k = 4$). From now on, we fix $m \geq 2$ and let $k = 2m + 2$. We also define $\varphi : S_{2m+2} \to S_{2m+2}$ by

\[
\varphi = t_{12m+2} \circ t_{12m-1}^{-1}.
\]

We note that

\[
\varphi(\pi) = \pi \circ (1, 2m+1)(2m+2, 2m, 2m-1, \ldots, 2),
\]

Hence, informally, in $\pi$'s vector notation, $\varphi$ transposes the elements in indices $1, 2m+1$, and cyclically shifts all other elements once to the bottom (i.e., as if applying a “push-to-the-top” operation on the last index — acting only on these indices). We can also observe that $\varphi(2m) = \text{Id}$.

We conveniently define, for all $r \geq 0$, the permutations

\[
\tilde{\pi}_r = \varphi'(\text{Id})
\]

\[
= (1, 2m+1, 2m+2, 2m, 2m-1, \ldots, 2) \in S_{2m+2}.
\]

In particular, we note that when $r \equiv r'( \mod 2m)$, and only then, we have $\tilde{\pi}_r = \tilde{\pi}_{r'}$.

**Lemma 11** For all $r \geq 0$ a parity-preserving $G_1(2m + 2, M_{2m+2})$ code $P_r$ exists which begins with $\tilde{\pi}_r$ and ends with $t_{12m-1}^{-1}\tilde{\pi}_r$, where

\[
M_{2m+2} = \begin{cases} 
57 & m = 2, \\
\frac{(2m+1)!}{2} & m > 2.
\end{cases}
\]

**Proof:** The claim follows trivially from the codes provided by Lemma 7, if we shift the generating transition sequence such that it ends with $t_{12m-1}$ and apply it to $\tilde{\pi}_r$. \( \square \)

We note in particular that for all $r$, $\tilde{\pi}_r$ is even, and thus $P_r \subseteq A_{2m+2}$. Moreover, since the parity-preserving code $P_r$ does not employ $t_{12m+2}$, for all $\pi \in P_r$ it holds that

\[
\pi(2m + 2) = \tilde{\pi}_r(2m + 2)
\]

\[
= \begin{cases} 
2m + 2 & r \equiv 0 \pmod{2m}, \\
2m + 1 - (r \pmod{2m}) & r \not\equiv 0 \pmod{2m}.
\end{cases}
\]

Thus, when considered as sets,

\[
P_r \cap P_{r'} = \emptyset,
\]

for all $0 \leq r < r' < 2m$.

We shall construct a $G_2^{\text{aux}}(2m + 2, M)$ code by stitching together $P_1, P_2, \ldots, P_{2m-1}$. We will need to amend $P_0$ before incorporating it into our code, for reasons we shall discuss below. First, we describe the stitching method in the following lemma.

**Lemma 12** For all $r \geq 0$ (including, in particular, $r = 2m - 1$), we may concatenate $P_r, P_{r+1}$ into a (non-cyclic) “push-to-the-top” code by applying the transitions $t_{12m+2}, t_{12m+2}$ to the last permutation of $P_r$, which is $t_{12m-1}^{-1}\tilde{\pi}_r$. Additionally, the only odd permutation in the resulting code is

\[
\beta_{r+1} = t_{12m+2}^{-1}(\tilde{\pi}_{r+1}).
\]

We shall refer to it as the $(r+1)$-bridge.

**Proof:** The claim follows trivially from the definition

\[
\tilde{\pi}_{r+1} = \varphi(\tilde{\pi}_r) = t_{12m+2} \circ \left( t_{12m+2} \circ t_{12m-1}^{-1}(\tilde{\pi}_r) \right).
\]

since $P_r, P_{r+1}$ are parity-preserving, and $t_{12m+2}$ flips parity. \( \square \)

Lemma 12 can be used iteratively to concatenate $P_1, P_2, \ldots, P_{2m-1}$, with a single odd permutation—the $r$-bridge—between each pair of $P_{r-1}, P_r$. Thus, we obtain the sequence

\[
P_1, P_2, P_3, \ldots, P_{2m-1}, P_{2m-1}.
\]

Note that if any two permutations $\pi_1, \pi_2$ in the resulting sequence satisfy $\pi_1 = (q, 2m+2) \circ \pi_2$ for some $q \in [2m+1]$, then w.l.o.g $\pi_2$ is odd and hence an $r$-bridge for some $r$, and $\pi_1$ is even and thus not a bridge. Since in every bridge the last element is

\[
\beta_r(2m + 2) = \tilde{\pi}_r(1) \in \{1, 2m + 1\},
\]

and in every non-bridge it is not, it must follow, then, that

\[
q = \beta_r(2m + 2),
\]

and in particular

\[
\pi_1(2m + 2) = (\beta_r(2m + 2), 2m + 2) \circ \beta_r(2m + 2)
\]

\[
= 2m + 2,
\]

thus $\pi_1 \in P_0$.

We witness, therefore, that no such pair of permutations exist, since we have not yet incorporated $P_0$ into our code. Hence, so far we have a $(2m+2)$-nontransposing code. It also becomes apparent that $P_0$ must necessarily be amended prior to its inclusion, so it does not include any permutations of the form

\[
(\beta_r(2m + 2), 2m + 2) \circ \beta_r, \quad 0 < r < 2m.
\]

In order to do so, we note that for all $r \geq 0$

\[
\beta_r(2m + 2) = \tilde{\pi}_r(2m + 1) \in \{1, 2m + 1\},
\]

and in particular $\beta_r(2m) \neq 2m + 2$, hence

\[
(\beta_r(2m + 2), 2m + 2) \circ \beta_r(2m) = \beta_r(m) \in \{1, 2m + 1\}.
\]

It follows that if we let $P'_0$ be generated by the transition sequence $t_{12m-1}^{-1}\tilde{\pi}_0$ applied to $\tilde{\pi}_0$, then it is parity-preserving, its last permutation is $t_{12m-1}^{-1}\tilde{\pi}_0$, and for all $\pi \in P'_0$ we have $\pi(2m) = 2m \not\in \{1, 2m + 1\}$, thus

\[
P'_0 \cap \{(\beta_r(2m + 2), 2m + 2) \circ \beta_r\}_{r=1}^{2m} = \emptyset.
\]
Lemma 13 The following sequence $P$,
\[ P = P'_0, \beta_1, P_1, \beta_2, P_2, \beta_3, \ldots, \beta_{2m-1}, P_{2m-1}, \beta_{2m}, \]
is a cyclic and $k$-nontransposing $G_{1}(k, M)$.

Proof: By Lemma 12, and since when considered as sets,
\[ P_r \cap P'_{r'} = \emptyset \]
for all $0 < r < r' < 2m$, and similarly $P'_0$ is disjoint from $P_1, P_2, \ldots, P_{2m-1}$, we know that $P$ is a $G_{1}(2m, 2, M)$.

As seen above, if for any two permutations $\pi_1, \pi_2 \in P$ and $q \in [2m+1]$ we have $\pi_1 = (q, 2m+2) \circ \pi_2$, then w.l.o.g. $\pi_2 = \beta_r$ for some $0 < r \leq 2m$ and $q = \beta_r(2m+2)$. In particular, $\pi_1(2m+2) = 2m+2$, thus
\[ \pi_1 \notin \{ \beta_r \}_{r=1}^{2m} \cup \bigcup_{r=1}^{2m-1} P_r, \]
thus $\pi_1 \in P'_0$. But
\[ P'_0 \cap \{ (\beta_r(2m+2), 2m+2) \circ \beta_r \}_{r=1}^{2m} = \emptyset, \]
in contradiction.

The code from Lemma 13 is almost what we need. The only property lacking is the fact that $Id$ is not followed in $t$ transition $t_{2m+2}$. We fix this in the following theorem.

Theorem 14 Let $k \geq 6$ be even. Then there exists a $G_{1}^{\text{aux}}(k, M)$, with
\[ M = \begin{cases} 178 & k = 6, \\ (k-3) \frac{(k-1)!}{2} + 1 & k > 6. \end{cases} \]
In particular, for all $k > 6$,
\[ M > k - 3 \cdot k! 2. \]

Proof: Denote $k = 2m+2$ for $m \geq 2$, and let $P = (c_j)_{j=1}^{M}$ be the code from Lemma 13. Since $Id \subseteq S_k$ is not followed with a $t_{2m+2}$ transition in $P$, we denote the last permutation of $P'_0$ by $\tilde{\pi}$, and replace $P$ with
\[ \tilde{P} = \tilde{\pi}^{-1}P = \left( \tilde{\pi}^{-1} \circ c_j \right)_{j=1}^{M}. \]
We observe that $\tilde{P}$ is still a “push-to-the-top” code since “push-to-the-top” transitions are group actions by right-multiplications. Moreover, since $\tilde{\pi}(k) = k$, if for some $\pi_1, \pi_2 \in P$ we have $\tilde{\pi}^{-1} \circ \pi_1 = (q, k) \circ (\tilde{\pi}^{-1} \circ \pi_2)$, where $q \in [k-1]$, then
\[ \pi_1 = \tilde{\pi} \circ (q, k) \circ (\tilde{\pi}^{-1} \circ \pi_2) \]
\[ = [\tilde{\pi} \circ (q, k) \circ \tilde{\pi})^{-1} \circ \pi_2 = (\tilde{\pi}(q), k) \circ \pi_2, \]
and $\tilde{\pi}(q) \in [k-1]$, in contradiction.

As for the size of the code, note that $|P'_0| = 2m-1 = k-3$ and
\[ |P_1| = |P_2| = \ldots = |P_{2m-1}| = \begin{cases} 57 & k = 6, \\ \frac{(k-1)!}{2} & k > 6. \end{cases} \]
Counting $\beta_1, \ldots, \beta_{2m}$, the claim is thus substantiated, up to a rotation to make $Id$ the first permutation.

To conclude this section, we combine Lemma 7, Theorem 9 and Theorem 14 into the following corollary.

Corollary 15 For all $k \geq 3$ there exists a $G_{1}^{\text{aux}}(k, M_k)$, where
\[ M_k = \begin{cases} 8 & k = 4, \\ 57 & k = 5, \\ 178 & k = 6, \\ \frac{k!}{2} & 5 \neq k \equiv 1 \pmod{2}, \\ \rho_k \frac{k!}{2} & 6 \leq k \equiv 0 \pmod{2}, \end{cases} \]
and $\rho_k > k^{-3} k$.  

IV. CODE CONSTRUCTION

In this section we present the main construction of our paper, and discuss the size and asymptotic rate of the resulting codes. We will show, surprisingly, that our method generates codes which are larger than formerly known families of codes, even though we require the additional structure of a Gray code.

A. Main code construction

We now present a construction of $G_{1}(n, M, d)$ codes, for $d < n$, which we base on Corollary 15 and Lemma 8.

It will simplify the presentation to assume $n = kd$ for some positive $k \geq 2$, since in that case every congruence class modulo $d$ of $[n]$ has size $k$. Nonetheless, the construction is applicable to any $n > d$ with natural amendments. We discuss these changes, focusing on special cases, after presenting the simple construction first.

Construction A Let $n, k, d \in \mathbb{N}$, with $n = kd$ and $k \geq 2$. We recursively construct a sequence of codes, $C_0, C_{d-1}, \ldots, C_1$. An explicit construction is given for $C_d$ and a recursion step constructs $C_m$ from $C_{m+1}$.

Recursion base: We construct the code $C_d$ by starting at the permutation $\sigma_0 \in S_n$ defined by
\[ \sigma_0(j) = d \cdot (j \mod k) + \left\lfloor \frac{j}{k} \right\rfloor. \]
We obtain a transition sequence $t_{t_{1}}, t_{t_{2}}, \ldots, t_{t_{k}}$, which generates the $G_{1}(k, k!)$ provided by Lemma 8. The code $C_d$ starts with $\sigma_0$, and uses the transition sequence
\[ t_{k(d-1)+1}, t_{k(d-1)+1}, t_{k(d-1)+1}, \ldots, t_{k(d-1)+1}. \]

Recursion step: Assume $C_{m+1}$ has already been constructed, starting with permutation $\sigma_0$. Additionally, let
\[ t_{t_{k+1}}, t_{t_{k+1}}, \ldots, t_{t_{M_{k+1}}} \]
be a transition sequence generating a $G_{1}^{\text{aux}}(k+1, M_{k+1})$ code provided by Corollary 15.

We construct the code $C_m$ as follows: replace each $t_{km+1}$ transition of $C_{m+1}$ with $t_{k(m-1)+1}$, followed
by \( l_{k(m-1)+1} l_{k(m-1)+i_2}, \ l_{k(m-1)+1} l_{k(m-1)+i_3}, \) and so on until \( l_{k(m-1)+1} l_{k(m-1)+i_{k+1}}. \)

\[ \text{Lemma 16} \] For all \( n = kd, \ k \geq 2, \) the code \( C_d \) from Construction A is a \( G_{k(d-1)+1} l_k l^t \).

**Proof:** The parameters of the code are obvious, except perhaps the minimal distance \( d. \) The fact that the codewords of \( C_d \) are distinct follows from Lemma 8.

To prove the minimal distance \( d, \) note that for all \( 0 \leq u < d \) and \( ku + 1 \leq i \leq j \leq k(u+1) \) it holds that \( \sigma_0(i) = \sigma_0(j) \) (mod \( d \)). Thus, for every distinct \( \sigma, \tau \in C_d, \) there exists \( j, k(d-1) < j < kd = n, \) such that \( \sigma(j) = \tau(j). \) Since by construction \( \sigma(j) = \tau(j) = 0 \) (mod \( d \)), we observe

\[ d_{\infty}(\sigma, \tau) \geq |\sigma(j) - \tau(j)| \geq d, \]

implying that \( C_d \) is a \( G_{k(d-1)+1} l_k \), \( l^t \).

\[ \text{Example 17} \] We let \( d = 3, \ k = 2, \) and \( n = kd = 6. \) We construct the code \( C_3 \) starting at

\[ \sigma_0 = [4, 1, 5, 2, 6, 3] \in S_6. \]

We use the complete \( G_1(2, 2) \) shown in Example 10, which is generated by the sequence \( t_{12}, t_{14} \). We arrive at a generating sequence \( t_{156}, t_{157} \) for \( C_3. \) Hence, in our example

\[ C_3 = \{(4, 1, 5, 2, 6, 3), (4, 1, 5, 2, 3, 6)\}, \]

which is readily seen to be a \( G_5(2, 3) \) code.

We shall follow Construction A to develop this example into a \( G_1(6, 18, 3) \) in Example 19. First, we prove the validity of the construction.

\[ \text{Theorem 18} \] For all \( n = kd, \ k \geq 2, \) the code \( C_1 \) from Construction A is a \( G_1 l_k l^t \).

**Proof:** To prove the claim we will prove by induction that \( C_m \) from Construction A, for all \( m \in \{d\}, \) is a \( G_{k(m-1)+1} l_k l^t \). The base case of \( C_d \) was proved in Lemma 16. Assume the claim holds for \( C_{m+1} \) and we now prove it for \( C_m. \)

Recall (1) gives the sequence of transitions for a \( G_{k+1} l_k l^t. \) Then

\[ t_{15} l_{k+1} t_{15} t_{k+1} \ldots t_{15} t_{15} = t_{15} l_{k+1}^{-1}. \]

Thus,

\[ l_{k+m+1} = \left( \prod_{r=2}^{k+1} l_{k+1} l_{k+1}^{-1} \right) l_{k+1}^{-1}, \]

(whose product is expanded right-to-left). Therefore, \( C_m \) expands each “push-to-the-\((k+1)\)st-index” transition of \( C_{m+1} \) into \( l_{k+m+1} \) “push-to-the-\([k(m+1)+1]\)st-index” transitions.

It follows that \( C_m \) contains the codewords of \( C_{m+1} \) in the same order, with \( l_{k+m+1} \) new words inserted between any two words originally from \( C_{m+1}. \) We say that each codeword of \( C_{m+1} \) (now appearing in \( C_m) \) is the \( C_{m+1} \)-parent of each of of the \( \tilde{M}_{k+1} \) preceding codewords in \( C_m \) (including itself), since their vector notations agree on the order of the elements

\[ \sigma_0(km+1), \sigma_0(km+2), \ldots, \sigma_0(n). \]

We note here (and will further examine later) that when \( x = \sigma^*(km+1) \) and \( \sigma \) is a \( C_{m+1} \)-parent of \( \tilde{M}_{k+1} \) codewords in \( C_m \) (inclusive), then that subsequence of \( C_m \) is an \( x \)-nontransposing \( G_{k(m-1)+1} l_k l^t \) code. This follows since we used an auxiliary code to construct that subsequence, of which the parent takes the role of first permutation, and in \( \sigma^*, x \) is the last element among the indices being permuted.

Now, suppose that \( \sigma, \tau \in C_m \) satisfy \( d_{\infty}(\sigma, \tau) < d. \) Let \( \sigma, \tau \) be their \( C_{m+1} \)-parents, respectively. To complete the proof we will show that \( \sigma = \tau. \)

**Case 1:** \( \sigma = \tau. \) Denote

\[ x = \sigma^*(km+1) = \tau^*(km+1) \]

and \( s = \sigma^{-1}(x), \ a = \tau(s). \)

If \( a = x \) then for all \( j \neq s, k(m-1) < j \leq km + 1, \) we have

\[ \sigma(j) \equiv \tau(j) \pmod{d} \]

\[ |\sigma(j) - \tau(j)| \leq d_{\infty}(\sigma, \tau) < d, \]

hence \( \sigma(j) = \tau(j), \) and \( \sigma = \tau. \)

Otherwise, \( a \neq x, \) and denote \( t = \tau^{-1}(x) \neq s. \) It similarly holds for all \( j \notin \{s, t\}, k(m-1) < j \leq km + 1, \) that \( \sigma(j) = \tau(j). \) We therefore observe \( \tau = \sigma o \{s, t\}. \) This implies that, if we let \( \tilde{\sigma}, \tilde{\tau} \in S_{k+1} \) be the permutations in the \( G^\text{aux} \) we obtained, generated similarly to \( \sigma, \tau, \) respectively (i.e., by their corresponding transition sequences), then

\[ \tilde{\tau} = \tilde{\sigma} o \{s - k(m-1), t - k(m-1)\} = (q, k+1)\tilde{\sigma} \]

for some \( q \notin \{k\}, \) in contradiction to the fact it was a \( G^\text{aux}(k+1, \tilde{M}_{k+1}) \).

**Case 2:** \( \sigma \neq \tau. \) Since \( \sigma \neq \tau \in C_{m+1} \) we have by assumption \( d_{\infty}(\sigma^*, \tau^*) < d. \) and note that for all \( j \) satisfying \( j \leq k(m-1) \) or \( j > km + 1, \) it holds that \( \sigma(j) = \sigma^*(j) \) and \( \tau(j) = \tau^*(j). \) Hence there exists \( j, k(m-1) < j \leq km + 1, \) such that

\[ |\sigma(j) - \tau(j)| < d \]

but \( |\sigma^*(j) - \tau^*(j)| \geq d. \)

Note particularly, since for all \( j \leq km + 1 \) it holds that \( \sigma^*(j) = \sigma_0(j) = \tau^*(j), \) that we have

\[ |\sigma^*(km+1) - \tau^*(km+1)| \geq d. \]

Denote \( x = \sigma^*(km+1), \ y = \tau^*(km+1), \) and note that

\[ \{\sigma(j)\}^{km+1}_{j=km+1} = \{a \}_{i=\alpha} \cup \{x\}; \]

\[ \{\tau(j)\}^{km+1}_{j=km+1} = \{a \}_{i=\alpha} \cup \{y\}, \]

where \( \{a \}_{i=\alpha} \) is a congruence class modulo \( d\) \( of\) \( \{n\}, \) of which \( x, y \) are not members.

Let \( s = \sigma^{-1}(x) \) and denote \( a = \tau(s). \) Since

\[ |x - a| = |\sigma(s) - \tau(s)| \leq d_{\infty}(\sigma, \tau) < d \]

we have \( a \neq y. \) Let \( t = \sigma^{-1}(a). \) Since \( a \in \{a \}_{i=\alpha} \) is a congruence class modulo \( d, \) for all \( b \in \{a \}_{i=\alpha} \setminus \{a\} \) we observe

\[ |a - b| > d, \]

but

\[ |a - \tau(t)| = |\sigma(t) - \tau(t)| \leq d_{\infty}(\sigma, \tau) < d \]
and therefore $\tau(t) = y$. For all $j \notin \{s, t\}$ satisfying $k(m - 1) < j \leq km + 1$ we then have $\sigma(j) \equiv \tau(j) \pmod d$ and $|\sigma(j) - \tau(j)| \leq d_{\sigma}(\sigma, \tau) < d$, hence $\sigma(j) = \tau(j)$.

This implies that, if we again let $\hat{\sigma}, \hat{\tau} \in S_{k+1}$ be the permutations in the $G_{\hat{\sigma}}^{\hat{\tau}}$ generated similarly to $\sigma, \tau$ respectively, then

$$\hat{\tau} = \hat{\sigma} \circ (s - k(m - 1), t - k(m - 1)) = (q, k + 1)\hat{\sigma}$$

where $q$ is given by $a = a_q \in \{a_1, \ldots, a_{k+1}\}$ again contradicting the properties of a $G_{\hat{\sigma}}^{\hat{\tau}}(k+1, \hat{M}_{k+1})$. Hence $C_m$ has minimal $\ell_{\sigma}$-distance of at least $d$, as required.

Example 19 We complete Example 17 into a $G_\tau(6, 3^2 \cdot 2, 3)$ code by applying the recursion step twice. In each step, since $k = 2$, we utilize the trivial parity-preserving $G_{\tau}^{\tau}(3, 3)$ code generated by the sequence $t_1, t_3, t_5, t_7$.

Firstly, recall that we used

$$\sigma_0 = [4, 1, 5, 2, 6, 3] \in S_6,$$

and the sequence $t_5, t_6, t_7$ generates

$$C_3 = \{[4, 1, 5, 2, 6, 3], [4, 1, 5, 2, 3, 6]\}.$$

We build $C_2$ by exchanging each $t_{5}t_{6}$ transition by $t_{7}$ followed by 2 instances of $t_{2}t_{3}t_{4}$; the middle level of Figure 2 shows the resulting code.

Secondly, as seen in the same figure, each $t_{7}$ transition of $C_2$, $j \in \{5, 6\}$, can be replaced by $t_{1}t_{j}$, followed by 2 instances of $t_{0}t_{1}t_{2}$, to generate $C_1$. We complete Example 19 into a $G_\tau(6, 3^2 \cdot 2, 3)$ code (the general version of) Construction A, we refer to its most general version

$$\rho_{k+1} = \begin{cases} \frac{(k+1)!}{2} & 4 \neq k \equiv 0 \pmod 2; \\ \frac{\rho_{k+1}}{k+1} & 5 \leq k \equiv 1 \pmod 2, \end{cases}$$

where, again, $\rho_{k+1} > \frac{k^2}{d}$. It is therefore important to note that when $\frac{n}{d} \equiv 0 \pmod 2$, $[\frac{n}{d}]$ has $(n \text{ mod } d)$ congruence classes modulo $d$ of odd size $\frac{n}{d}$, and $d - (n \text{ mod } d)$ classes of even size $\frac{n}{d}$. Thus, if additionally $\frac{n}{d} > 4$, the constructed code $C_1$ is of size

$$|C_1| = \left(\frac{(\frac{n}{d} + 1)!}{d} \right)^{\frac{n \text{ mod } d}{2}} \cdot \frac{\frac{n}{d}}{d}!,$$

and simple rearranging gives us the first case of (2). Similar considerations give us the next five cases of (2).

Finally, we consider the case of $n < 2d$, which implies $\frac{n}{d} = 1$. In this special case we only permute $(n \text{ mod } d)$ congruence classes of $[n]$, and each such class has $2 = \frac{n}{d} + 1$ elements. As mentioned, we therefore construct a code of size $|C_1| = 3^n \text{ mod } d$.

We comment that it is also possible to achieve a slight gain in code size by reordering $\sigma_0$ so that the last block consists of a congruence class of odd size, rather than even, where the added complexity of index calculation is inconsequential. The gains are negligible for large enough $n$. 

Finally, we discuss the special case of $n < 2d$, in which all but $(n \text{ mod } d)$ congruence classes are singletons. We will amend our construction by replacing the recursion base with

$$C_m = \{\sigma_0, t_{2m-1}t_{2m+1}, \sigma_0, t_{2m-1}t_{2m+1} \sigma_0\},$$

where $m = n \text{ mod } d$, and continuing the recursion step as discussed above. Thus, we are effectively only using the first member of $R_{m+1}$ together with the previous congruence classes, fixing $\sigma_0(j)$ for $j > 2m + 1$. In this case, we obtain $C_1$ which is a $G_\tau(n, 3^m \text{ mod } d, d)$.

Thus, in what follows, whenever we mention Construction A, we refer to its most general version applying to all $n$ and $d$.

B. Code-size analysis and comparison

We would like to give an explicit expression for the size of the codes constructed by Construction A. This would enable a comparison with previously known results.

Lemma 20 Let $C_1$ be the code from (the general version of) Construction A. Then its size, $|C_1|$, is given by (2).

Proof: Let us first assume $n \geq 2d$. We note the asymmetry in Construction A between congruence classes $R_d$ of odd and even sizes. Indeed, a class of size $|R_d| = k \geq 2$ (for all classes other than $R_d$, which is used in the recursion base and whose contribution is based on the $G_\tau(k, d)$ code) contributes to the code size, according to Corollary 15, a multiplicative factor of

$$\frac{n}{d}! \cdot \frac{(\frac{n}{d} + 1)!}{d} \cdot \frac{\frac{n}{d}}{d}!.$$
We now turn to comparing the size of the resulting code with that of previously constructed codes, as well as known bounds on the cardinality of such codes.

The first comparison we make is with codes that have the Gray property. Such codes were only studied for \(d = 2\), i.e., snake-in-the-box codes or \(G_1(n, M, 2)\) codes in our notation. These codes were studied in [44, Thm. 24], where it was shown that such codes can be constructed with sizes

\[
\left\lceil \frac{n}{d} \right\rceil + 1 \mod d = \left\lfloor \frac{n}{d} \right\rfloor + 1 \mod d
\]

\[
C_1 = \left\lceil \frac{n}{d} \right\rceil + 1 \mod d \cdot \frac{n \mod d}{2^{d-1}} \cdot 3^{d-1} \cdot 2
\]

\[
C_2 = \left\lfloor \frac{n}{d} \right\rfloor \mod d \cdot 57^{d-1} \cdot 24
\]

\[
C_3 = \left\lfloor \frac{n}{d} \right\rfloor \mod d \cdot 8d \cdot \frac{n \mod d}{3}
\]

Construction A improves this size by a factor of \(\frac{1}{2} \left\lfloor \frac{n}{d} \right\rfloor + 1 \left\lfloor \frac{n}{2} \right\rceil \cdot \left(\left\lfloor \frac{n}{d} \right\rceil - 1\right)\), times a factor of \(\rho_{\left\lfloor n/2 \right\rceil}+1\) when \(n \equiv 2 \mod 4\) (in the case of \(n \equiv 1 \mod 4\), the factor \(\rho_{\left\lfloor n/2 \right\rceil}+1\) is eliminated by changing the order of congruence classes modulo \(\sigma_0\)). We note that a similar improvement was made concurrently by [42] in a preprint devoted solely to the case of \(d = 2\), i.e., snake-in-the-box codes.

We now also compare our results to error-correcting codes with the \(\ell_\infty\)-metric which are not necessarily Gray codes (LMRM-codes). We observe that the best known general LMRM-code construction to date, [39, Cst. 1, Thm. 2] and [27, Sec. III-A], presented \((n, M, d)\)-LMRM codes with sizes

\[
M = \left\lceil \frac{n}{d} \right\rceil \mod d \cdot \left(\left\lfloor \frac{n}{d} \right\rceil - 1\right) \cdot \frac{n \mod d}{d}.
\]

which our construction improves upon, more pronouncedly the more \(n\) has even-sized congruence classes modulo \(d\) (cf. (2)).

Finally, we also note the following lemma:

**Lemma 21** [39, Thm. 16] If \(C \subseteq S_n\) is a code with minimal \(\ell_\infty\) distance \(d\), then

\[
|C| \leq \frac{n!}{(d)^{\left\lfloor n/d \right\rceil} \cdot (n \mod d)!}
\]

We remark that in the case of \(d = 2\), [44] confirmed that the optimal size of error-correcting codes for \(n = 4, 5, 6\) to be 6, 30, 90 respectively, meeting the bound of Lemma 21. It also presented Gray codes achieving these sizes by computer search. Searches in higher dimension were reported unfeasible. For higher values of \(d\), the optimal size is unknown, as well as whether Gray codes can achieve it. While the reader can appreciate that the bound of Lemma 21 is exponentially greater than the size provided by Lemma 20, we note anecdotally that the \(G_1(6, 18, 3)\) code presented in Example 19 almost meets the bound \((M < 20)\).

In the asymptotic regime, we go on to examine the case of \(d = \Theta(n)\). For an \((n, M, d)\)-LMRM code (and in particular a \(G_1(n, M, d)\)), we follow the convention (e.g., [39]) of defining the rate of the code

\[
R = \frac{1}{n} \log_2 M,
\]

and its normalized distance

\[
\delta = \frac{d}{n}.
\]

The following were proven in [39].
Lemma 22 [39, Thm. 23] For any \((n, M, n\delta)\)-LMRM code it holds that
\[
R \leq 2 - 2\delta \left(\frac{1}{\delta} \log_2 \left(\frac{1}{\delta} \right) - \phi(\frac{1}{\delta})\right) - \left(\frac{1}{\delta} \log_2 \left(\frac{1}{\delta} \right) - \phi(\frac{1}{\delta})\right) + o(1).
\]

Lemma 23 [39, Thm. 27] For any \(0 < \delta < 1\) the construction of [39, Cst. 1, Thm. 2] and [27, Sec. III-A] yields codes with
\[
R = \left(1 - \delta \left(1 - \frac{1}{\delta} \right) \right) \log_2 \left(\frac{1}{\delta} \right) + \left(\delta + \delta \left(1 - \frac{1}{\delta} \right) \right) \log_2 \left(\frac{1}{\delta} \right) + o(1).
\]

Previous works have also established the following lower bound on achievable rates of error-correcting codes:

Lemma 24 (Gilbert-Varshamov) For any \(0 < \delta < 1\) there exist \((n, M, d)\)-LMRM codes satisfying \(\frac{d}{n} \geq \delta\) with rate
\[
R = f_{GV}(\delta) - o(1),\quad \text{where } \Phi(\delta) \leq f_{GV}(\delta) \leq \varphi(\delta).
\]

\[
\Phi(\delta) = \left\{ \begin{array}{ll} 
\log_2 \left(\frac{1}{\delta} \right)^2 + 2\delta \log_2(e - 1) - 1 & 0 < \delta < \frac{1}{2} \\
-2\delta \log_2 \left(\frac{1}{\delta} \right)^2 + 2(1 - \delta) \log_2(e) & \frac{1}{2} < \delta < 1,
\end{array} \right.
\]
\[
\varphi(\delta) = \left\{ \begin{array}{ll}
\log_2 \left(\frac{1}{\delta} \right) - 1 & 0 < \delta < \rho \\
\log_2 \left(\frac{1}{\delta} \right) + 2\delta \log_2(e - 1) & \rho < \delta < \frac{1}{2} \\
\log_2 \left(\frac{1}{\delta} \right) - \ell(\delta) \cdot (2\delta - 1) + \log_2 \left(\frac{1}{\delta} \right) & \frac{1}{2} < \delta < 1,
\end{array} \right.
\]
\[
\rho = \frac{2 - \log_2(e - 1)}{3 - 2\log_2(e)}, \quad W(t) \text{ is the Lambert function, and}
\]
\[
\ell(\delta) = \log_2(e - 1) - W\left(\frac{1}{2} - \frac{(1 - \delta) \cdot \exp\left(\frac{2(1-\delta)}{2\delta - 1}\right)}{2\delta - 1}\right).
\]

Proof: We derive \(f_{GV}\) from the Gilbert-Varshamov bound:
\[
f_{GV}(\delta) = \frac{1}{n} \log_2 \left(\frac{n! }{|B_{\delta n,n}|}\right)
\]
\[
= \log_2(n) - \frac{1}{n} \log_2 |B_{\delta n,n}| + o(1),
\]
where \(|B_{\delta n,n}|\) is the size of ball of radius \(\delta n\) in \(S_n\), and is independent of the center of the ball since the \(\ell_\infty\) metric is right-invariant, i.e., for all \(\sigma, \pi, \tau \in S_n\),
\[
d_{\infty}(\sigma \pi, \tau \pi) = d_{\infty}(\sigma, \tau).
\]

Unfortunately, the asymptotic size of \(|B_{\delta n,n}|\) isn’t precisely known as \(n \to \infty\). Recently, however, [35] proved new lower bounds on \(|B_{\delta n,n}|\), namely:
\[
n \log_2(n) - \log_2 |B_{\delta n,n}| \leq \left\{ \begin{array}{ll} 
\frac{1}{n} \left[\log_2 \left(\frac{1}{\delta} \right)^2 + \delta - 1 \right] - \sigma(n) & 0 < \delta \leq \rho \\
\frac{1}{n} \left[\log_2 \left(\frac{1}{\delta} \right)^2 + 2\delta \log_2(e - 1)\right] - \sigma(n) & \rho < \delta < \frac{1}{2} \\
\frac{1}{n} \left[\log_2 \left(\frac{1}{\delta} \right)^2 + \ell(\delta) \cdot (2\delta - 1) + \log_2 \left(\frac{1}{\delta} \right)\right] - \sigma(n) & \frac{1}{2} < \delta \leq 1.
\end{array} \right.
\]

An upper bound for \(|B_{\delta n,n}|\) was established in [26, Eq. (4)] and [39, Lem. 25]:
\[
n \log_2(n) - \log_2 |B_{\delta n,n}| \geq \left\{ \begin{array}{ll} 
\frac{1}{n} \left[\log_2 \left(\frac{1}{\delta} \right)^2 + (2\delta + 1) \log_2(e - 1)\right] - \sigma(n) & 0 < \delta \leq \frac{1}{2} \\
\frac{1}{n} \left[3 - 2\delta \log_2(e)\right] - \sigma(n) & \frac{1}{2} < \delta \leq 1.
\end{array} \right.
\]

Deriving the lemma is now straightforward.

The works cited in Lemma 24 establish a narrow rate-range for the Gilbert-Varshamov bound, as can be seen in Figure 3, i.e., the true Gilbert-Varshamov bound passes somewhere within the gray-shaded area in Figure 3.

Next, we aim to show that our construction can bridge some of the gap between the given bounds and known constructions.

Lemma 25 Let \(C_1\) be the code from (the general version of) Construction A. Then an estimate from below of its rate \(R\) as a function of its normalized distance \(\delta\) is given by (3):

Proof: The proof follows by a simple substitution of \((n \mod d) = n - d \left[\frac{d}{n}\right]\) and \(d = n\delta\) into (2). We also recall that \(\rho_k > \frac{k-1}{k}\).

In conclusion, these asymptotic rates and bounds are shown in Figure 3. We note in particular that the rate of codes produced by Construction A is strictly higher than that of previously known constructions (as in Lemma 23). Furthermore, it produces codes with rates higher than those guaranteed by the Gilbert-Varshamov bound shown in Lemma 24 for all \(\delta\) greater than \(\approx 0.1\) except in a small neighborhood of \(\frac{1}{\pi}\), whereas known constructions only bypassed these rates for \(\delta\) greater than \(\approx 0.349\).

V. Decoding Algorithm

This section is devoted to devising a decoding algorithm capable of correcting a noisy received version of a transmitted codeword.

Known constructions of \((n, M, d)\)-LMRM codes, presented in [39, Cst. 1, Thm. 2] and [27, Sec. III-A], lend themselves to straightforward decoding algorithms, efficiently done in \(O(n)\) operations, since for any given codeword \(\sigma\) and index \(i \in [n]\),
\[
r = [\sigma(i) \mod d] \text{ is known. Hence, if a retrieved permutation } \tau \text{ satisfies } d_{\infty}(\sigma, \tau) \leq \left[(d - 1)/2\right], \text{ then } \sigma(i) \text{ is known to be the unique element } k \in r + d\Z \text{ satisfying } |k - \tau(i)| \leq \left[(d - 1)/2\right].
\]

Our proposed construction diverges from that rigid partition. However, we can still efficiently decode noisy information, provided errors of magnitude no more than \(t\) have occurred, where \(2r + 1 \leq d\). More precisely, we assume that for every stored permutation \(\sigma\) and retrieved permutation \(\tau\) it holds that
\[
d_{\infty}(\sigma, \tau) \leq t \leq \left[(d - 1)/2\right]
\]

To simplify our presentation we assume \(n = kd\), since then our construction only makes (repeated) use of a single auxiliary \(G_{\tau}^\text{aux}(k + 1, \tilde{M}_{k+1})\) code. Extensions to the general version of Construction A are easily obtainable.

We first require a function \(\text{ValidAux}\) capable of detecting whether a given permutation \(\pi \in S_{k+1}\) belongs to the auxiliary \(G_{\tau}^\text{aux}(k + 1, \tilde{M}_{k+1})\) code.
Figure 3. (a) The range of uncertainty for the Gilbert-Varshamov lower bound seen in Lemma 24. (b) The rate of codes from Lemma 23 constructed in [39]. (c) A lower bound for the rate of codes $C_1$ from Construction A. (d) The upper bound of Lemma 22.

\[
R \geq \begin{cases}
(1 - \delta \left\lfloor \frac{1}{\delta} \right\rfloor) \log_2 \left(\left\lfloor \frac{1}{\delta} \right\rfloor + 1\right) + \delta \log_2 \left(\left(\left\lfloor \frac{1}{\delta} \right\rfloor + 1\right)!\right) \\
(1 - \delta \left\lfloor \frac{1}{\delta} \right\rfloor) \log_2 \left(\left\lfloor \frac{1}{\delta} \right\rfloor + 1\right) - \delta \\
(1 - 4\delta) \log_2(178) + (5\delta - 1) \log_2(57) \\
(1 - 2\delta) (3 - \log_2(3)) + \delta \log_2(3) \\
(1 - 5\delta) \log_2(315) + (6\delta - 1) \log_2(89) + 2 - 9\delta \\
(1 - 3\delta) \log_2(57) - 4 + 1 \\
(1 - \delta) \log_2(3)
\end{cases}
\]  
(3)

Lemma 26 For an auxiliary $G_{aux}(k + 1, M_{k+1})$ code provided by Lemma 7, Theorem 9 or Theorem 14, a function $\text{ValidAux}$ can be implemented to operate in $O(k)$ steps.

**Proof:** If we use Lemma 7, then the auxiliary code consists of all even permutations, and it is well known that we can determine the signature of a permutation $\pi \in S_{k+1}$ in $O(k)$ operations, e.g., by finding a cycle decomposition of $\pi$. The case of $k + 1 = 5$ requires special attention, as $M_5 = 57$. In fact, in that case [44] showed that a parity-preserving code of size 57 exists consisting of

\[A_5 \setminus \{\sigma, t_{13}\sigma, t_{31}^2\sigma\},\]

for every choice of $\sigma \in A_5$. The user may arbitrarily decide on $\sigma$, and check for the missing codewords in $O(k)$.

If we instead use Theorem 9, then we know that the vector notation of every codeword in the auxiliary code has 1 following $k + 1$ (cyclically). Since there are

\[\frac{|S_{k+1}|}{k} = k! + (k - 1)! = k! \left(1 \cdot \frac{k - 1}{k} + 2 \cdot \frac{1}{k}\right)\]
such codewords, we observe that the auxiliary code consists of precisely all permutations so characterized (when counting valid permutations, we partition permutations on $[k]$ by whether the first index in their vector notation equals 1. If so, we may insert $k + 1$ either at the beginning or the end of their vector notation; otherwise, its position is uniquely determined), i.e.,

\[\pi \left(\pi^{-1}((k + 1) \mod (k + 1)) + 1\right) = 1,\]

which again requires $O(k)$ steps to verify.

Finally, for Theorem 14 we note that the problem can equivalently be solved for the codes of Lemma 13, as composition with $\bar{\pi}$ can be done in $O(1)$ steps with a simply implemented rule, or naively in at most $O(k)$ steps. If we divide into cases according to $\pi(k + 1) = j$ we may identify $r$ such that $\pi$ must belong to $P_r$, or not belong to our code. For values $j = 1, k$, we know $\pi$ can only be a bridge: For $j = k + 1$, it must belong to $P_0$. In these cases, only cycle shifts (on a subset of indices, by case) of a known permutation are valid, which we can easily verify in linear time. For all other elements $P_r$ consists of all even permutations satisfying $\pi(k + 1) = j$, hence the problem again reduces to determining sign $\pi$, as in the case based on Lemma 7 (or managed as discussed above for $k = 5$).

An important notion of a window will be useful. Let $\sigma \in S_n$ be a permutation, $n = kd$. For all $j \in [d]$ we define the $j$th
window as the set of indices
\[ W_j = \{k(j-1)+2, k(j-1)+3, \ldots, k(j-1)+3, k(j-1)+k\} \cap [n]. \]
The windows partition \([n] \setminus \{1\}\), and are all of size \(k\) except \(W_d\) which is of size \(k-1\).

Given a set \(I \subseteq [n]\), we conveniently denote
\[ \sigma(I) = \{\sigma(i) \mid i \in I\}. \]
We prove a simple lemma concerning properties of windows of codewords from Construction A.

**Lemma 27** Let \(\sigma\) be a codeword of \(C_1\) from Construction A, with \(n = kd\). Then for all \(j \in [d]\),
\[ k-1 \leq |\sigma(W_j) \cap R_j| \leq k, \]
i.e., at most one element of \(\sigma(W_j)\) does not leave a residue of \(j\) modulo \(d\). In particular, \(\sigma(W_d) \subseteq R_d\).

Additionally, if \(|\sigma(W_j) \cap R_j| = k-1, j \in [d-1]\), and we denote \(\{x\} = \sigma(W_j) \setminus R_j\), then there exists some \(j’ > j\) such that \(x \in R_j\).

**Proof:** Take any \(1 < j \in [d]\), and let \(\sigma_j\) be the \(C_j\)-parent of \(\sigma\). Then, in \(C_1\), no transition between \(\sigma\) and \(\sigma_j\) is induced by \(C_j\), and hence \(\sigma_j\) is derived from \(\sigma\) by a (perhaps empty) sequence of \(t^i\) transitions, for \(i’ \in W_1 \cup \cdots \cup W_{j-1}\).

Therefore, for all \(i \in W_j \cup \cdots \cup W_d\) we have \(\sigma_j(i) = \sigma(i)\), and the same also holds for \(j = 1\) (since \(\sigma_1 = \sigma\)). In particular, \(\sigma(W_j) = \sigma_j(W_j)\).

Now, since \(C_j\) only applies “push-to-the-\((k-1) + \text{1st-index}” transitions, and
\[ \sigma_0(\{(k(j-1)+1) \cup W_j \cup \cdots \cup W_d\} = R_j \cup \cdots \cup R_d, \]
if for any \(i \in W_j\) we have \(\sigma(i) = \sigma_j(i) \in R_j\), then by necessity \(\sigma(i) \in R_j\) for some \(j’ > j\). In particular, \(\sigma(W_d) \subseteq R_d\).

For all \(j \in [d-1]\), we also consider \(\sigma_{j+1}\), the \(C_j+1\)-parent of both \(\sigma\) and \(\sigma_j\). Since \(C_j+1\) only applies “push-to-the-\(k+1\) \text{1st-index}” transitions,
\[ \sigma_{j+1}(\{(k(j-1)+1) \cup W_j\} \setminus \{k(j+1)\}) = \sigma_0(\{(k(j-1)+1) \cup W_j\} \setminus \{k(j+1)\}) = R_j. \]
Finally, since \(\sigma_{j+1}\) is derived from \(\sigma_j\) by a sequence of \(t_{(k(j-1)+1)}\) transitions for \(i’ \in W_j\), it follows that
\[ \sigma_{j+1}(\{(k(j-1)+1) \cup W_j\} = \sigma_j(\{(k(j-1)+1) \cup W_j\} \]
thus
\[ \sigma_j(W_j) \subseteq \sigma_{j+1}(\{(k(j-1)+1) \cup W_j\} = R_j \cup \sigma_{j+1}(k(j+1)\}. \]
Noting that \(|\sigma_j(W_j)| = |R_j| = k\) and recalling that \(\sigma(W_j) = \sigma_j(W_j)\), we are done.

**Corollary 28** Let \(\sigma\) be a codeword of \(C_1\) from Construction A, with \(n = kd\). Then for each \(j \in [d]\), there is a unique element \(\sigma_j^\circ \in R_j \cup \cdots \cup R_d\) satisfying
\[ \sigma(W_j \cup \cdots \cup W_d) = R_j \cup \cdots \cup R_d \setminus \{\sigma_j^\circ\}. \]

**Proof:** The proposition follows from Lemma 27 for \(j = d\) since \(|\sigma(W_d)| = |W_d| = k-1 < |R_d \cap \sigma(W_d)|\). Now suppose the proposition holds for \(j + 1\), and we prove that it holds for \(j\).

We again observe by Lemma 27 that \(|R_j \cap \sigma(W_j)| \in \{k-1, k\}\). If \(|R_j \cap \sigma(W_j)| = k\), since \(|\sigma(W_j)| = |W_j| = k\), then \(R_j = \sigma(W_j)\) and \(x_j^\circ = x_{j+1}^\circ\) satisfies the claim.

Otherwise \(\sigma(W_j) \setminus R_j = \{y\}\) for some \(y \in [n]\); it would suffice to show \(y = x_j^\circ\), since then \(R_j \cap \sigma(W_j) = \{x_j^\circ\}\) would satisfy the claim.

Consider then \(j\), the \(C_j\)-parent of \(\sigma\). Note that \(\sigma_j(W_j) = \sigma(W_j)\), and since \(C_j\) employs “push-to-the-\((k(j-1) + 1)\) \text{1st-index}” transitions only, and
\[ \sigma_0(W_j \cup \cdots \cup W_d) \subseteq R_j \cup \cdots \cup R_d, \]
we know that \(\sigma(W_j) \subseteq R_j \cup \cdots \cup R_d\). We now use the induction hypothesis
\[ \sigma(W_{j+1} \cup \cdots \cup W_d) = (R_{j+1} \cup \cdots \cup R_d) \setminus \{x_j^\circ\}, \]
and it follows that \(\sigma(W_j) \subseteq R_j \cup \{x_j^\circ\}\), hence \(y = x_j^\circ\).

From now on, we denote \(i_j^\circ = \sigma^{-1}(x_j^\circ)\). Another useful notation we shall employ is a function that quantizes any integer to the nearest integer leaving a residue of \(j\) modulo \(d\). We denote this function by \(q^\circ_{d}(z) : \mathbb{Z} \to (d-1)/2\), defined by
\[ q^\circ_{d}(a) = \arg\min\{|a-b|, \]
where we assume \(\arg\min\) returns a single value, and ties are broken arbitrarily.

For the decoding procedure description, let us fix the parameters \(n = kd\), and the code \(C_1\) from Construction A. Additionally, we denote by \(\sigma \in C_1\) the transmitted permutation, by \(\tau \in S_n\) the received permutation, and by \(\hat{\sigma} \in S_n\) the decoded permutation. We denote the decoding radius by \(t = \lfloor(d-1)/2\rfloor\), and assume \(d = \sigma(\tau, t) < t\).

We will decode \(\tau\) iteratively by window, from \(W_1\) to \(W_d\). We shall make sure—inductively—that when we begin the process of decoding \(W_j\), for some \(j \in [d]\), we know \(i_j^\circ\). Initially, as mentioned, we set \(j = 1\). Trivially, \(i_1^\circ = 1\).

**Step 1** We set the decoding window
\[ \hat{W}_j = W_j \cup \{i_j^\circ\}, \]
and naively decode \(W_j\) by setting for all \(i \in \hat{W}_j\),
\[ \hat{\sigma}(i) = q^\circ_{d}(\tau(i)). \]

**Lemma 29** After Step 1, for all \(i \in \hat{W}_j\) such that \(\sigma(i) \in R_j\) it holds that \(\hat{\sigma}(i) = \sigma(i)\).

**Proof:** For all such \(i\) we have \(\hat{\sigma}(i) \equiv \sigma(i) \pmod{d}\) and
\[ |\hat{\sigma}(i) - \sigma(i)| = |\sigma(i) - \sigma(i)| + |\tau(i) - \sigma(i)| = q^\circ_{d}(\tau(i)) + |\tau(i) - \sigma(i)| \leq |\sigma(i)| + t < d. \]


Corollary 30 After Step I, 
\[ \sigma(\hat{W}_j) = \sigma \left( \hat{W}_j \setminus \{i_{j+1}^\sigma\} \right) = \mathcal{R}_j. \]

Proof: By Corollary 28 we know that \( \mathcal{R}_j \subseteq \sigma(\hat{W}_j) \). We further recall that \( \sigma(i_{j+1}^\sigma) \notin \mathcal{R}_j \), hence
\[ \mathcal{R}_j \subseteq \sigma \left( \hat{W}_j \setminus \{i_{j+1}^\sigma\} \right), \]
and since
\[ \left| \sigma \left( \hat{W}_j \setminus \{i_{j+1}^\sigma\} \right) \right| = k = |\mathcal{R}_j| \]
we have equality. The claim now follows from Lemma 29.

Corollary 30 implies that after Step I, \( \sigma(\hat{W}_j) \) contains a unique element of \( \mathcal{R}_j \) which appears twice, and every other element appears exactly once; by Lemma 29 these other elements have been decoded correctly. Before we can continue inductively to decode \( W_{j+1} \), it only remains to find \( i_{j+1}^\sigma \); the other instance in \( \hat{W}_j \) of \( \sigma(i_{j+1}^\sigma) \) we therefore also know to have been decoded correctly.

We shall identify \( i_{j+1}^\sigma \) using \( \mathcal{C}_{\text{aux}} \), the auxiliary \( \mathcal{C}^\text{aux}(k + 1, \mathcal{M}_{k+1}) \) code used in Construction A. By construction, if we examine \( \sigma_j \), the \( C_j \)-parent of \( \sigma \), then for all \( i \in W_j \) we observe \( \sigma(i) = \sigma_j(i) \), and \( \sigma(i_j^\sigma) = \sigma_j(k(j - 1) + 1) \). The ordering of the \( k + 1 \) elements of \( \sigma(\hat{W}_j) \) is \( \sigma_j \left( \{ k(j - 1) + 1 \} \cup W_j \right) \) is then induced by a permutation of \( \mathcal{C}_{\text{aux}} \). We construct this induced permutation from the auxiliary code \( \mathcal{C}_{\text{aux}} \), which we denote \( \hat{\pi} \in \mathcal{S}_{k+1} \). We first define a simple bijection \( \alpha_j : \mathcal{R}_j \to [k] \), which is the inverse of the enumeration of \( \mathcal{R}_j \) given by the arbitrary initial order of elements in \( \sigma_0 \) used in Construction A, e.g., in the simple case \( n = kd \),
\[ \alpha_j(m) = \begin{cases} \frac{m}{k} & j < m \in \mathcal{R}_j, \\ k & m = j. \end{cases} \]

With \( \alpha_j \) we define \( \hat{\pi} \) as,
\[ \hat{\pi}(i) = \begin{cases} \alpha_j(\sigma_j(i)) & i = 1; \\ \alpha_j(\sigma_j(k(j - 1) + i)) & i \in \{2, 3, \ldots, k + 1\}, \end{cases} \]
and note that—as it currently stands—\( \hat{\pi} \) is not a permutation of \( [k + 1] \) because its range is \( [k] \) and some unique \( a \in [k] \) has two distinct pre-images.

Theorem 31 Let \( s, t \in [k + 1] \) be the unique pair of indices such that \( \hat{\pi}(s) = \hat{\pi}(t) = a \in [k] \). There is a unique way to re-define \( \hat{\pi} \mid_{\{s, t\}} \) (the restriction of \( \hat{\pi} \) to \( \{s, t\} \)) as a bijection onto \( \{a, k + 1\} \) that yields \( \hat{\pi} \in \mathcal{C}_{\text{aux}} \). Furthermore, if we define \( I_j : [k + 1] \times [n] \to [n] \) by
\[ I_j(q, r) = \begin{cases} \hat{r} & q = 1; \\ (k(j - 1) + q) & \text{otherwise}, \end{cases} \]
then after performing that correction
\[ i_{j+1}^\sigma = I_j(\hat{\pi}^{-1}(k + 1), i_j^\sigma). \]

Proof: First, arbitrarily set \( \hat{\pi}(t) = k + 1 \), where \( t > s \). Once corrected, \( \hat{\pi} \in \mathcal{S}_{k+1} \) by Corollary 30 and because \( \alpha_j : \mathcal{R}_j \to [k] \) is a bijection.

Now, we take \( \pi \in \mathcal{C}_{\text{aux}} \) which generates \( \sigma_j \) in the recursion step of Construction A—while constructing \( C_j \)—from its \( C_{j+1} \)-parent. Hence
\[ \pi(i) = \begin{cases} \alpha_j(\pi_j^\sigma) & i = 1; \\ \alpha_j(\pi(k(j - 1) + i)) & i \in \{2, 3, \ldots, k + 1\}, \end{cases} \]
and therefore either \( \hat{\pi} = \pi \) or \( \hat{\pi} = (k + 1, a) \circ \pi \). Crucially, we observe that in the latter case \( \hat{\pi} \notin \mathcal{C}_{\text{aux}} \) since \( \mathcal{C}_{\text{aux}} \) is a \( \mathcal{G}^\text{aux}(k + 1, \mathcal{M}_{k+1}) \) code and \( \pi \in \mathcal{C}_{\text{aux}} \); we utilize \text{ValidAux} to discover whether our original arbitrary correction should be reversed.

To complete the proof, we note by the recursion step of Construction A that, indeed, \( i_{j+1}^\sigma = I_j(\pi^{-1}(k + 1), i_j^\sigma) \).

We can therefore complete our iterative decoding round with the following step.

Step II We construct \( \hat{\pi} \) as described, identify \( s, t, s < t \), and arbitrarily correct \( \hat{\pi}(t) = k + 1 \). We test \text{ValidAux}(\hat{\pi}) \); if true, we have \( i_{j+1}^\sigma = I_j(t, i_j^\sigma) \); otherwise, it holds that \( i_{j+1}^\sigma = I_j(s, i_j^\sigma) \).

Finally, observe that when decoding \( W_2 \) it’s known that \( \sigma(\hat{W}_d) = \mathcal{R}_d \), hence by Lemma 29 \( \hat{W}_d \) is decoded correctly, and we need not (and—indeed—cannot) perform Step II.

Example 32 We shall demonstrate the decoding process assuming once again \( n = kd \) for simplicity, and using the parameters \( d = 3 \) (hence \( t = 1 \), \( k = 2 \) and code constructed in Example 19. Recall that the \( \mathcal{G}^\text{aux}(3, 3) \) code used in that example is
\[ \mathcal{C}_{\text{aux}} = \{[1, 2, 3], [3, 1, 2], [2, 3, 1]\}. \]

We choose the transmitted codeword \( \sigma = [1, 2, 4, 6, 5, 3] \), and a noisy received permutation \( \tau = [1, 3, 4, 5, 6, 2] \).

We start by defining \( i_1 = 1 \) and observing (by abuse of the vector notation) \( \tau \mid_{\hat{W}_1} = [1; 3, 4] \) (the first element is differentiated because—generally although never when \( j = 1 \)—it does not immediately precede the rest in \( \tau \)’s vector notation).

Since \( j = 1 \), we define \( \sigma \mid_{\hat{W}_1} = [1; 4, 2] \). This leads us to construct \( \hat{\pi} = [2, 1, 3] \notin \mathcal{C}_{\text{aux}} \), so we instead correct \( \hat{\pi} = [2, 3, 1] \) and define \( i_2 = 2 \). (So far we have \( \hat{\pi} = [1, 4, 2, 3, 5, \ldots] \), where an underline marks \( i_{j+1}^\sigma \).

Next, we have \( \tau \mid_{\hat{W}_2} = [3; 5, 6] \), which (\( j = 2 \)) we decode \( \hat{\pi} \mid_{\hat{W}_2} = [2, 5, 5] \). This again generates \( \hat{\pi} = [2, 1, 3] \notin \mathcal{C}_{\text{aux}} \), and we correct in similar fashion to \( \hat{\pi} = [2, 3, 1] \) and define \( i_2 = 4 \). (Up to this point, we have \( \hat{\pi} = [1, 4, 2, 5, 4, \ldots] \).

Finally, we have \( \tau \mid_{\hat{W}_3} = [5; 2] \) and since \( j = 3 \) we decode \( \hat{\pi} \mid_{\hat{W}_1} = [6, 3] \), and overall \( \hat{\pi} = [1, 2, 4, 6, 5, 3] = \sigma \). \( \square \)

Example 33 We present another example, intended to demonstrate the process in more detail, for which we depart from the parameters used in Example 19 by setting \( d = 5 \) (allowing for \( t = 2 \leq \lfloor (d - 1)/2 \rfloor \)), \( k = 3 \). In each recursion step of Construction A the \( \mathcal{G}^\text{aux}(4, 8) \) code used is \( \mathcal{C}_{\text{aux}} \) presented in Example 10.

The codeword
\[ \sigma = [11, 1, 8, 6, 7, 2, 12, 13, 3, 5, 9, 14, 4, 10, 15] \]
to be the noisy version of the transmitted codeword $\sigma$, and verify that $d_{\infty}(\tau, \sigma) = 2 = t$.

Beginning with $j = 1$, we have $\tau \upharpoonright_{W_1} = [12; 3, 9, 7]$, which we decode $\hat{\sigma} \upharpoonright_{W_1} = [11; 1, 11, 6]$; generating $\hat{x} = [2, 3, 2, 1]$ which is corrected to $\hat{r} = [2, 3, 4, 1] \in C_{\text{aux}}$. We identify $i_2 = 3$, and keep

$$\hat{\sigma} = [11, 1, 11, 6, \ldots, \ldots, \ldots, \ldots, \ldots] .$$

Next, for $j = 2$, observe that $\tau \upharpoonright_{W_2} = [9; 5, 2, 11]$, and we decode $\hat{\sigma} \upharpoonright_{W_2} = [7; 7, 2, 12]$. This generates $\hat{x} = [1, 1, 3, 2]$, which we initially correct to $\hat{r} = [1, 4, 3, 2] \notin C_{\text{aux}}$, so (skip correcting $\hat{x}$, as it has no further consequence) $i_3 = i_2 = 3$ instead of $i_3 = 4$. We summarize

$$\hat{\sigma} = [11, 1, 11, 6, 7, 2, 12, \ldots, \ldots, \ldots, \ldots, \ldots] .$$

We turn to $W_3$ and see that $\tau \upharpoonright_{W_3} = [9; 15, 1, 6]$, decoded to $\hat{\sigma} \upharpoonright_{W_3} = [8; 13, 3, 8]$. We generate $\hat{x} = [1, 2, 3, 1]$ and correct it to $\hat{r} = [1, 2, 3, 4] \in C_{\text{aux}}$, indicating that $i_4 = 10$. We now have

$$\hat{\sigma} = [11, 1, 11, 6, 7, 2, 12, 13, 3, 8, \ldots, \ldots, \ldots, \ldots, \ldots] .$$

Moving on to $j = 4$, while decoding $W_4$ we note $\tau \upharpoonright_{W_4} = [6; 8, 13, 4]$, which we decode as $\hat{\sigma} \upharpoonright_{W_4} = [4; 9, 14, 4]$. This generates $\hat{x} = [3, 1, 2, 3]$ which is corrected to $\hat{r} = [3, 1, 2, 4] \notin C_{\text{aux}}$. We therefore define $i_5 = i_4 = 10$ instead of $i_5 = 13$. Up to now,

$$\hat{\sigma} = [11, 1, 8, 6, 7, 2, 12, 13, 3, 8, 9, 14, 4, \ldots] .$$

Finally, $j = 5$, and we get $\tau \upharpoonright_{W_5} = [4; 10, 14]$ which we decode to $\hat{\sigma} \upharpoonright_{W_5} = [5; 10, 15]$, and overall

$$\hat{\sigma} = [11, 1, 8, 6, 7, 2, 12, 13, 3, 8, 9, 14, 4, 10, 15] = \sigma .$$

The decoding algorithm is formalized in Decode($\tau$). With appropriate simple data structures, the algorithm requires $O(kd) = O(n)$ steps. We assume simple arithmetic operations to take constant-time.

VI. RANKING AND UNRANKING

In this section we discuss the process of encoding data $m \in \{0, 1, \ldots, |C_1| - 1\}$ to a codeword $\sigma \in C_1$, which is also known as unranking $m$, and the inverse process of ranking $\sigma \in C_1$, i.e., obtaining its rank in the code. Throughout this section, $C_1$ stands for the code obtained via Construction A.

Due to the nature of our construction, performing these tasks with the codes generated by Theorem 18 is reliant on our ability to do the same with the codes provided by Lemma 8 and Corollary 15. We therefore recall the following known result.

**Lemma 34** [23] The complete $G_1(n, n!)$ codes provided by Lemma 8 has a ranking algorithm operating in $O(n)$ steps, and an unranking scheme operating in $O(n^2)$ steps.

This gives rise to the following corollary.

**Corollary 35** The $G_{\text{aux}}(2m, \lfloor S_{2m} \rfloor)$ codes generated by Theorem 9 can be ranked in $O(m)$ operations and unranked in $O(m^2)$ operations.

**Proof:** Ranking a permutation $\sigma$ in the code may proceed by finding the cyclic shift required for $[2m, 1]$ to be the first two elements. After removing these two first elements, and then reversing the permutation we may use a ranking algorithm from Lemma 34. A simple combination of the results gives the required ranking of $\sigma$. By Lemma 34, the entire procedure takes $O(m)$ operations. A symmetric argument gives an $O(m^2)$ algorithm for unranking.

Unfortunately, no ranking and unranking schemes are known for parity-preserving $G_1(2m + 1, \mathcal{M}_{2m+1})$ codes provided by Lemma 7 (developed in [20]), or previous constructions presented in [22], [45]. Consequentially, we rely on Theorem 9 instead of Theorem 14 for even sized congruence classes. In the case of odd sized classes, we can leverage the following codes.

**Lemma 36** [44] For all $m \geq 1$ there exist parity-preserving $G_1(2m + 1, \mathcal{M}_{2m+1})$ codes with sizes

$$\mathcal{M}_{2m+1} = \left( \frac{2m!}{m!} \right)^2 \frac{(2m + 1)!}{2m^2!} = \frac{(2m)!}{m!^2 2^m} \lfloor S_{2m+1} \rfloor .$$
These codes can be ranked and unranked in $O(m^2)$ operations.

We summarize those observations in the following corollary.

**Corollary 37** For all $k \geq 3$ there exist a $G^{\text{aux}}(k, \hat{M}_k)$ code, which have ranking and unranking schemes operating in $O(k^2)$ steps, where

$$
\hat{M}_k = \left\{ \begin{array}{ll}
\left( \frac{k-1}{k} \right)! & k \equiv 1 \pmod{2}; \\
\left( \frac{k}{k-1} \right)! & k \equiv 0 \pmod{2}.
\end{array} \right.
$$

Note that we can now replace Corollary 15 by Corollary 37 in Construction A to obtain codes which we shall denote $\hat{C}_1$, and each auxiliary code on a congruence class of size $k > 1$ contributes to $|\hat{C}_1|$ a multiplicative factor of

$$
\hat{M}_{k+1} = \left\{ \begin{array}{ll}
\left( \frac{k}{k+1} \right)! & k \equiv 0 \pmod{2}; \\
\left( \frac{k}{k+1} \right)! & k \equiv 1 \pmod{2}.
\end{array} \right.
$$

We also note, using Stirling’s approximation

$$
e^{-\frac{n\log n}{n} - \frac{\log n}{2n \log n}} < e^{-\frac{n}{n}},
$$

that

$$
\frac{k!}{(k/2)!^2 2^k} > \sqrt{\frac{2}{\pi}} \left( e^{\frac{1}{12\pi}} \right)^{-\frac{1}{4}} > \sqrt{\frac{2}{\pi}} e^{-1/(4k)}.
$$

We can then recalculate code size, in the case of $\left\lfloor \frac{n}{4} \right\rfloor \equiv 0 \pmod{2}$:

$$
|\hat{C}_1| = \left( \frac{[n/d]}{[n/d]} \right)! n \mod d \cdot \left( \frac{n}{d} \right)! \cdot \left( \frac{n/d}{2} \right)! \cdot \left( \frac{n}{d} + 1 \right) \cdot \left( \frac{n}{d} \right)! \cdot \left( \frac{n}{d} \right)! = \left( \frac{[n/d]}{[n/d]} \right)! n \mod d \cdot \left( \frac{n}{d} \right)! \cdot \left( \frac{n}{d} \right)! \cdot \left( \frac{n}{d} + 1 \right) \cdot \left( \frac{n}{d} \right)! 
$$

and when $\left\lfloor \frac{n}{4} \right\rfloor \equiv 1 \pmod{2}$:

$$
|\hat{C}_1| = \left( \frac{[n/d]}{[n/d]} \right)! n \mod d \cdot \left( \frac{n}{d} \right)! \cdot \left( \frac{n}{d} \right)! \cdot \left( \frac{n}{d} + 1 \right) \cdot \left( \frac{n}{d} \right)! 
$$

and we note that in the special case $\left\lfloor \frac{n}{4} \right\rfloor = 1$ we have $|\hat{C}_1| = |C_1|$.  

We likewise observe the rates of codes based on Corollary 37, and find for $[1/\delta] = 0 \pmod{2}$

$$
\hat{R} \geq \left( 1 - \delta \left\lfloor \frac{1}{\delta} \right\rfloor \right) \log_2 \left( \left\lceil \frac{1}{\delta} \right\rceil \left( 1 + \frac{1}{\left\lceil \frac{1}{\delta} \right\rceil} \right) \right) + \left( \delta + \delta \left\lceil \frac{1}{\delta} \right\rceil - 1 \right) \log_2 \left( \left\lceil \frac{1}{\delta} \right\rceil \left( 1 + \frac{1}{\left\lceil \frac{1}{\delta} \right\rceil} \right) \right) - \frac{1}{2} \left( \delta + \delta \left\lceil \frac{1}{\delta} \right\rceil - 1 \right) \log_2 \left( \left\lfloor \frac{1}{\delta} \right\rfloor + \log_2(e) \frac{2}{\left\lceil \frac{1}{\delta} \right\rceil} + \log_2(\pi) - 1 \right) - o(1),
$$

and for $[1/\delta] \equiv 1 \pmod{2}$

$$
\hat{R} \geq \left( 1 - \delta \left\lfloor \frac{1}{\delta} \right\rfloor \right) \log_2 \left( \left\lceil \frac{1}{\delta} \right\rceil \frac{1}{\delta} \left( 1 + \frac{1}{\left\lceil \frac{1}{\delta} \right\rceil} \right) \right) + \left( \delta + \delta \left\lceil \frac{1}{\delta} \right\rceil - 1 \right) \log_2 \left( \left\lceil \frac{1}{\delta} \right\rceil \frac{1}{\delta} \left( 1 + \frac{1}{\left\lceil \frac{1}{\delta} \right\rceil} \right) \right) - \frac{1}{2} \left( 1 - \delta \left\lceil \frac{1}{\delta} \right\rceil \right) \log_2 \left( \left\lfloor \frac{1}{\delta} \right\rfloor + \log_2(e) \frac{2}{\left\lceil \frac{1}{\delta} \right\rceil} + \log_2(\pi) - 1 \right) - o(1).
$$

The losses in asymptotic rate are shown in Figure 4. We observe in particular that we still manage to achieve better rates than previously known error-correcting codes (without the Gray property), even with the significantly smaller $G^{\text{aux}}(k, \hat{M}_k)$ of Corollary 37.

Let us denote by $\text{RankComplete}(\pi)$, $\text{UnrankComplete}(m)$ the ranking and unranking procedures for the complete codes from Lemma 34. Additionally, let $\text{RankAux}(\pi)$ and $\text{UnrankAux}(m)$ denote the ranking and unranking procedures for the auxiliary codes of Corollary 37. We can readily take advantage of $\hat{C}_1$’s tiered structure to use these functions in order to perform the same tasks for our construction. We include pseudo-code for these algorithms, which we call $\text{Rank}(\sigma)$ and $\text{Unrank}(m)$, for completeness. As before, we assume $n = kd$ to simplify the presentation.

**Theorem 38** For the code $\hat{C}_1$ of length $n = kd$, the algorithms $\text{Rank}(\sigma)$, $\text{Unrank}(m)$ operate in $O(k^2 d)$ steps.

**Proof:** Both algorithms perform a single loop over all indices of $\sigma$, making simple integer operations, which requires

<table>
<thead>
<tr>
<th><strong>Function Rank</strong> $(\sigma)$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>input</strong> $: \sigma \in \hat{C}_1.$</td>
</tr>
<tr>
<td><strong>output</strong> $: m \in {0, 1, \ldots,</td>
</tr>
<tr>
<td>/* Build a permutation $\pi_{d} \in S_{k}$ */</td>
</tr>
<tr>
<td>for $i \in W_{d}$ do</td>
</tr>
<tr>
<td>$\pi_{d} \left[ \left( k(d - 1) - 1 \right) \right] \leftarrow \alpha_{d}(\pi_{d}(i))$</td>
</tr>
<tr>
<td>$\pi_{d}[1] \leftarrow [k] \setminus \pi_{d}[2, \ldots, k] = k!$</td>
</tr>
<tr>
<td>$m \leftarrow \left( \text{RankComplete}(\pi_{d}) - 1 \right) \mod k!$</td>
</tr>
<tr>
<td>for $j = d - 1, \ldots, 1$ do</td>
</tr>
<tr>
<td>/* Build a permutation $\pi_{j} \in S_{k+1}$ */</td>
</tr>
<tr>
<td>for $i \in W_{j}$ do</td>
</tr>
<tr>
<td>$\pi_{j}(i - k(j - 1)) \leftarrow \alpha_{j}(\pi_{j}(i))$</td>
</tr>
<tr>
<td>$\pi_{j}[1] \leftarrow [k + 1] \setminus \pi_{j}[2, \ldots, k + 1]$</td>
</tr>
<tr>
<td>$m \leftarrow m \cdot M_{k+1} + \left( \text{RankAux}(\pi_{j}) - 1 \right) \mod M_{k+1}$</td>
</tr>
<tr>
<td>return $(m + 1) \mod</td>
</tr>
</tbody>
</table>

```
Figure 4. (a) The rate of codes from Lemma 23 constructed in [39]. (b) The rate of codes $C_1$ from Construction A. (c) The rate of codes $\hat{C}_1$ constructed using auxiliary codes from Corollary 37.

Function Unrank$(m)$

\[
\begin{align*}
\text{input} & : m \in [0, 1, \ldots, |C_1| - 1]. \\
\text{output} & : \sigma \in \mathcal{C}_1 \text{ with rank } m \text{ in } \mathcal{C}_1. \\
& \text{/* Convert } m \text{ to local ranks } R[1, 2, \ldots, d]. \\
m & \leftarrow (m - 1) \bmod |C_1| \\
R[i] & \leftarrow (m + 1) \bmod \mathcal{M}_{k+1} \\
m & \leftarrow m/M_{k+1} \\
\sigma_d & \leftarrow \text{UnrankComplete}(R[d]) \\
& \text{/* Construct } \sigma. \\
\tau & \leftarrow \pi_1[i] - k(d-1) \cdot d \\
x & \leftarrow \pi_d[i] \cdot d \\
& \text{for } j = 1, \ldots, d \text{ do} \\
\sigma[j] & \leftarrow \pi_d[i - k(j-1) + d] \\
\pi_j & \leftarrow \text{UnrankAux}(R[j]) \\
& \text{for } i \in W_d \text{ do} \\
\pi_i & \leftarrow \text{UnrankComplete}(R[i]) \\
& \text{if } \pi_i[1] = k + 1 \text{ then} \\
\sigma[i] & \leftarrow x \\
& \text{else} \\
\sigma[i] & \leftarrow \pi_i[1] \cdot d + j \\
\text{if } \pi_i[1] \neq k + 1 \text{ then} \\
x & \leftarrow \pi_i[1] \cdot d + j \\
\text{return } \sigma
\end{align*}
\]

Informally, as noted in [24], it measures the minimal number of adjacent transpositions required to transform one permutation into the other, that is, the minimal $r$ such that

\[
\sigma = \tau \circ (i_1, i_1 + 1) \circ (i_2, i_2 + 1) \circ \ldots \circ (i_r, i_r + 1)
\]

for some $i_1, i_2, \ldots, i_r \in [n - 1]$. An $(n, M, \mathcal{K})$-snake, or $\mathcal{K}$-snake for short, is a single-error-detecting rank-modulation Gray code of size $M$, or more formally, a $G_1(n, M)$ code $C$ such that for all $\sigma, \tau \in C$, $\sigma \neq \tau$, it holds that $d_{\mathcal{K}}(\sigma, \tau) \geq 2$. Put differently, for no $i \in [n - 1]$ does it hold that $\sigma = \tau \circ (i, i + 1)$.

The authors have shown in [44][Thm. 17] that any $\mathcal{K}$-snake $C \subseteq S_n$ which employs a “push-to-the-top” transition on an even index $\tau_{2m}^{\text{odd}}$—for any $m \in \left[\frac{n}{2}\right]$—must satisfy $|C| \leq \frac{n}{2} - \Theta(n)$. Horovitz and Etzion posited in [22] that $\mathcal{K}$-snakes in $S_{2m+2}$ do not exceed the size of those in $S_{2m+1}$, a conjecture refuted when Zhang and Ge demonstrated in [46] the existence of $\mathcal{K}$-snakes in $S_{2m+2}$ of size $\frac{2m+2}{2}$. Concurrently and independently, Holroyd conjectured in [20] that $\mathcal{K}$-snakes can be found in $S_{2m+2}$ with size greater than $\frac{2m+2}{2} - O(m^2)$.

A resemblance is evident in the definitions of $(n, M, \mathcal{K})$-snakes and $G_{\text{aux}}(n, M)$ codes, which is reinforced by the observations that, similarly to properties seen in Section III, any parity-preserving $G_1(n, M)$ code is an $(n, M, \mathcal{K})$-snake (see [44][Lem. 5]), and any $(n, M, \mathcal{K})$-snake satisfies $M \leq \frac{n}{2}$ (see [44][Thm. 15]).

We wish to demonstrate how the principles behind Theorem 14 can be applied to the construction of a $\mathcal{K}$-snake in $S_{2m+2}$ of size $M \geq \frac{2m+2}{2}$.

Lemma 39 [22, Thm. 18] [45] For $m \geq 2$, there exist parity-preserving $G_{\text{aux}}(2m + 1, M_{2m+1})$ codes with

\[
M_{2m+1} = |A_{2m+1}| - (2m - 1) = \left(\frac{2m+1}{2}\right) - (2m - 1).
\]

In particular, such a code $C$ was constructed such that, as a group,

\[
C = A_{2m+1} \setminus \left\{ t_{2m-1,q}^\sigma \right\}_{q=0}^{2m-2}
\]
for some $\sigma \in A_{2m+1}$. Finally, $C$ only employed $t_{2m-1}, t_{2m+1}$.

As before, we fix $m \geq 2$. We also reuse

$$\varphi(\pi) = t_{2m+2}^{2} \circ t_{2m+1}^{-1}(\pi)$$

$$= \pi \circ (1, 2m+1)(2m+2, 2m, 2m-1, \ldots, 2)$$

and the permutations $\hat{\pi}_r = \varphi'(1d)$.

**Theorem 40** For all $r \geq 0$ a parity-preserving $G_1 \left( 2m + 2, \frac{(2m+1)q}{2} - (2m-1) \right)$ code $\hat{P}_r$ exists which satisfy:

1. The first permutation in $\hat{P}_r$ is $\hat{\pi}_r$.
2. The last permutation in $\hat{P}_r$ is $t_{2m-1}^{-1} \hat{\pi}_r$.
3. For all $\pi \in \hat{P}_r$ it holds that

$$\pi(2m+2) = \hat{\pi}_r(2m+2)$$

$$\equiv \begin{cases} 2m + 2 & r \equiv 0 \ (\text{mod } 2m), \\ 2m + 1 - (r \mod 2m) & r \not\equiv 0 \ (\text{mod } 2m). \end{cases}$$

4. $\sigma_r \notin \hat{P}_r$, where we denote

$$\sigma_r = \left( t_{2m+2}^{-1} \hat{\pi}_r \right) \circ (2m + 1, 2m + 2)$$

(and observe $\sigma_r = t_{2m+1}^{-1}(\hat{\pi}_r)$, hence in particular $\sigma_r(2m+2) = \hat{\pi}_r(2m+2)$).

**Proof:** By Lemma 39 we know that there exist a parity-preserving $G_1(2m+1, M_{2m+1})$ code $P$ such that, as a set,

$$P = S_{2m+1} \setminus \left\{ t_{2m-1}^{-q} \sigma_r^{2m-2} \right\}_{q=0}^{2m-2}$$

for some $\sigma \in A_{2m+1}$. We also know that $P$ only employs $t_{2m-1}, t_{2m+1}$ transitions.

We apply its generating sequence to $\hat{\pi}_r$ to generate the $G_1(2m+2, M_{2m+1})$ code $\hat{P}$, which employs only $t_{2m-1}, t_{2m+1}$ transitions (in particular, it never employs $t_{2m+2}$, hence point 3 is established), and note that as a set

$$\hat{P} = \{ \tau \in A_{2m+2} \mid \tau(2m+2) = \hat{\pi}_r(2m+2) \}$$

$$\setminus \left\{ t_{2m-1}^{-q} \sigma_r^{2m-2} \right\}_{q=0}^{2m-2}.$$

for some $\sigma \in A_{2m+1}$, satisfying $\sigma(2m+2) = \hat{\pi}_r(2m+2)$.

Denote $\hat{P} = (c_j)_{j=1}^{M_{2m+1}}$. We modify our code by defining

$$\hat{P}_r = \left( c_j \right)_{j=1}^{M_{2m+1}} = \left( \sigma_r^{-1} c_j \right)_{j=1}^{M_{2m+1}},$$

which is still a $G_1(2m+2, M_{2m+1})$ since “push-to-the-top” transitions are group-actions by right-multiplication. Moreover, since $\sigma_r(2m+2) = \hat{\pi}(2m+2) = \hat{\pi}(2m+2)$, as a set we have

$$\hat{P}_r = \{ \tau \in A_{2m+2} \mid \tau(2m+2) = \hat{\pi}_r(2m+2) \}$$

$$\setminus \left\{ t_{2m-1}^{-q} \sigma_r^{2m-2} \right\}_{q=0}^{2m-2}.$$

Note in particular that

$$\sigma_r(2m+1) = \hat{\pi}_r(1) \neq \hat{\pi}_r(2m+1),$$

hence $\hat{\pi}_r \in \hat{P}_r$. In addition, point 4 is thus substantiated.

Finally, $t_{2m+1}^{-1}(\hat{\pi}_r) = \sigma_r \notin \hat{P}_r$ implies that $\hat{\pi}_r$ must necessarily be preceded in $\hat{P}_r$ by $t_{2m-1}$, which substantiates point 2 (after a proper cyclic shift of $\hat{P}_r$).

As in Section III, $\hat{P}_r \subseteq A_{2m+2}$ for all $r$. We construct a $(2m+2, M, K)$-snake by stitching together $\hat{P}_0, \hat{P}_1, \ldots, \hat{P}_{2m-1}$ in the following lemma.

**Lemma 41** For all $r \geq 0$, we may concatenate $\hat{P}_r, \hat{P}_{r+1}$ into a (non-cyclic) “push-to-the-top” code by applying the transitions $t_{2m+2}, t_{2m+2}$ to the last permutation of $\hat{P}_r$, which is $t_{2m+1}^{-1} \hat{\pi}_r$.

The only odd permutation in the resulting code is then

$$\beta_{r+1} = t_{2m+2}^{-1}(\hat{\pi}_r),$$

which we again call the $(r+1)$-bridge.

**Proof:** Exactly as in the proof of Lemma 12, given that $P_r, \hat{P}_r$ are parity-preserving, and have the same first and last permutations.

Again, similarly to Section III, Lemma 41 can be used iteratively to cyclically concatenate $\hat{P}_0, \hat{P}_1, \ldots, \hat{P}_{2m-1}$, with a single odd permutation—the $r$-bridge—between $P_{(r-1) \mod 2m}, \hat{P}_r$. Let us prove that fact in the following theorem.

**Theorem 42** There exists a $(2m+2, M_{2m+2}, K)$-snake for all $m \geq 2$, with

$$M_{2m+2} = \frac{2m}{2m+2} \frac{(2m+2)!}{2} - (2m-2)m$$

$$= \frac{2m}{2m+2} \frac{(2m+2)!}{2} - (2m-2)m.$$

**Proof:** We define $P$, similarly to Section III, as the cyclic concatenation

$$\hat{P}_0, \hat{\pi}_1, \hat{P}_r, \hat{\pi}_2, \ldots, \hat{\pi}_{2m-1}, \hat{P}_{2m-1}, \beta_0.$$

Suppose $\pi_1, \pi_2 \in C$ satisfy

$$\pi_1 = \pi_2 \circ (i, i+1)$$

for some $i \in [2m+1]$, then w.l.o.g $\pi_2$ is odd and hence $\pi_2 = \pi_r$ for some $0 \leq r < 2m$, and $\pi_1$ is even and thus not a bridge; it must follow, then, that

$$\pi_2(2m+2) \in \{1, 2m+1\} \neq \pi_1(2m+2),$$

hence $i = 2m+1$ and

$$\pi_1 = \pi_2 \circ (2m+1, 2m+2)$$

$$= \left( t_{2m+2}^{-1}(\hat{\pi}_r) \right) \circ (2m+1, 2m+2)$$

$$= t_{2m+1}^{-1}(\hat{\pi}_r) = \sigma_r.$$

This is in contradiction to Theorem 40, since $\pi_1(2m+2) = \hat{\pi}_r(2m+2)$ and thus $\pi_1 \in \hat{P}_r$. Hence $\hat{P}$ is a $K$-snake. Now, that

$$|\hat{P}| = 2m \left( \frac{(2m+1)!}{2} - (2m-1) \right) + 2m$$

$$= \frac{2m}{2m+2} \frac{(2m+2)!}{2} - (2m-2)m$$

is trivial.

To conclude this section, we note that $\frac{M_{2m+2}}{|S_{2m+2}|} \approx \frac{1}{2}$, which is optimal. The authors are unaware of any current result achieving this. We add that, in particular, in the context
of \( \mathcal{K} \)-snakes it is common to define the rate of codes as 
\[ R = \lim_{m \to \infty} \frac{\log|\mathcal{M}_{\mathcal{K}}|}{\log|\mathcal{S}_{2m}|} \] (see [44]), and we naturally observe that in our case \( R = 1 \) (which, again, is optimal, although \( R = 1 \) is also achieved by existing constructions, e.g., that of [46]).

VIII. Conclusion

In this paper we proposed a new class of codes, which we dubbed \( j \)-nontransposing, leveraging codes designed for the rank-modulation scheme under the Kendall \( \tau \)-metric, which we show can be used in the construction of error-correcting codes for the \( \ell_\infty \)-metric. By doing so, we were able to construct codes that achieve better asymptotic rates than previously known constructions, while also incorporating the property of being Gray codes. As with previously known constructions, we have shown that these codes allow for linear-time encoding and decoding of noisy data.

However, there remains a gap between the best known upper-bound for code sizes (either in the general case or in the specific case of Gray codes), based on the code-antidrome approach presented in [39], and achievable sizes (both known constructions and proven lower-bounds). We therefore propose that more research into upper and lower bounds on achievable code sizes is warranted.

Furthermore, much as in the case of codes designed for the Kendall \( \tau \)-metric, our auxiliary construction of \( k \)-nontransposing codes in \( S_k \) has some asymmetry between the cases of even- and odd-sized congruence classes. Although mostly alleviated by Theorem 14—in particular for large \( k \)—this creates an irregularity in the slope of the graph of asymptotic rate; for rankable codes, certain regions of \( \delta \) even admit a positive slope, whereby a code with a higher normalized distance also has a higher rate. We posit that, as Holroyd conjectured in [20] for \( \mathcal{K} \)-snakes, \( 2n \)-nontransposing codes in \( S_{2n} \) exist with size \( M > (2n)!/2 - O(n^2) \). This irregularity is especially pronounced when \( 2n = 6 \), where we have constructed an auxiliary code of size \( 178 \ll 360 = \frac{8}{3} \). We may note, however, that in the case of \( 2n = 4 \), the constructed auxiliary code of size 8 can be confirmed to be optimal by a manual search.

Finally, we have presented an adaptation of the solutions discussed above to the problem of \( (2n, M, \mathcal{K}) \)-snakes, which although not yet validating Holroyd’s conjecture above, is asymptotically tight.

Acknowledgments

The authors gratefully acknowledge the three anonymous reviewers and associate editor, whose insight and meticulous suggestions helped shape this paper.

References


Yonatan Yehezkeally (S’12) is a graduate student at the Department of Electrical and Computer Engineering, Ben-Gurion University of the Negev, Beer-Sheva, Israel. His research interests include algebraic coding, combinatorial structures, and finite group theory.

Moshe Schwartz (M’03–SM’10) is an associate professor at the Department of Electrical and Computer Engineering, Ben-Gurion University of the Negev, Israel. His research interests include algebraic coding, combinatorial structures, and digital sequences.