On Rank Modulation Codes

Prastudy Fauzi
University of Tartu, Estonia
prastudy.fauzi@gmail.com

Abstract. We survey rank modulation codes based on a recent result by Mazumdar, Barg and Zemor [MBZ13]. The main results are explicit constructions of rank modulation codes that approach the bounds given by Barg and Mazumdar [BM10].

Keywords: Rank modulation codes, Kendall tau, permutation, inversion vector

1 Introduction

Flash memory consists of cells, and data is represented by the level of electric charge in the cells. In the write operation of flash memory, we can efficiently increase the value of individual cells, but decreasing a value of an individual cell is not efficient since we need to rewrite the whole block of cells which contain that individual cell. The most common issue with flash memory is leakage of electric charge. Leakage can happen to multiple cells simultaneously, creating errors in multiple cells.

One solution to this problem is to store information by keeping note of the relative values between data cells. In this way, for each cell we store not a specific value, but the rank of the cell’s value relative to the other \(n - 1\) cells in the block of \(n\) cells. Here, one block can be viewed as an ordered set of \(n\) distinct natural numbers, each from 1, 2, \(\cdots\), \(n\). One block of cells does not contain \(n\) values in \(\mathbb{F}\), but rather one value in the permutation group \(S_n\). Hence if electric charge leaks simultaneously and at the same rate from all cells, then the relative ordering of the levels of electric charge are preserved, meaning no errors occur. Moreover, rewriting a block requires only increasing the values in specific cells, such that we move from one permutation to another. This scheme is known as the rank modulation scheme, and can be seen in Fig. 1.

However, even if we use rank modulation, errors in flash memory can still arise. The most common occurrence of such an error is when electric charge leaks faster from a certain (typically damaged) cell than the other cells. Then the charge level at that particular cell, say the cell with \(i\)-th highest charge, goes down sufficiently such that it becomes the \(i + 1\)-th highest charge. Here the permutation which states the relative values for the cells will be unchanged, save for position \(i\) and \(i + 1\), which will be swapped. This can be seen in Fig. 2.

The main question regarding rank modulation schemes is how to construct error-correcting codes over sets of permutations such that recovering from errors is as efficient as possible. We will try to formulate this problem in mathematical terms by defining an error in the form of distance between two permutations, and analyse some constructions from [MBZ13].

2 Preliminaries

We will give an overview of the mathematical concepts used in the later sections.
2.1 Basic Definitions

Sets and Permutations. A set is a collection of elements, where each element in the collection is distinct from another. One common example is the set of natural numbers from 1 to \( n \), denoted by \( [n] = \{1, 2, \cdots, n\} \). The number of elements of a set \( S \) is called the cardinality of the set, denoted by \( |S| \).

Given a set \( S \), another set \( T \) is a subset of \( S \) (denoted \( T \subseteq S \)) if all elements in \( T \) are also elements in \( S \). For every set \( S \), we can define its powerset \( 2^S \) as the set of all subsets of \( S \), including \( \emptyset \) and \( S \). The notation comes from the fact that a set \( S \) has \( 2^{|S|} \) elements in its powerset.

A permutation is a rearranging of elements in a set into a particular order. A permutation of a set of \( n \) elements can be written as an \( n \)-tuple \((a_1, a_2, \cdots, a_n)\) where each \( a_i \in [n] \) are distinct, and hence it is easy to see that there are \( n! \) such permutations of \( n \) elements. These \( n! \) permutations create a set \( S_n \), known as the set of permutations.

For a permutation \( \sigma \) and an integer \( i \in [n] \), \( \sigma(i) \) denotes the \( i \)-th element in \( \sigma \). For example, if \( \sigma = (3, 1, 2) \) then \( \sigma(1) = 3, \sigma(2) = 1, \sigma(3) = 2 \). The permutation \( \sigma \) can thus be seen as a function from \([n]\) to \([n]\) and, by its definition, codomain also \([n]\). This means that every permutation \( \sigma \) is a bijection from \([n]\) to \([n]\), and hence has an inverse permutation \( \sigma^{-1} \).
Inversions. Given a permutation $\sigma \in S_n$, an inversion is a pair of positions $(i, j)$ such that $i > j$ but $\sigma^{-1}(i) < \sigma^{-1}(j)$. The inversion vector $x_\sigma \in G_n = [0, 1] \times [0, 2] \times \cdots \times [0, n-1]$ is a vector of $n-1$ elements where $x_\sigma(i)$ denotes the number of inversions of the form $(i + 1, j)$, i.e. where $i + 1$ is the first element in the inversion. It is easy to obtain $x_\sigma$ from $\sigma$. For example, if $\sigma = (1, 4, 2, 8, 5, 7, 6, 3)$ then the set of inversions is $\{(4, 2), (4, 3), (8, 3), (8, 5), (8, 6), (8, 7), (5, 3), (7, 3), (7, 6), (6, 3)\}$. The inversion vector of this permutation is $x_\sigma = (0, 0, 2, 1, 2, 4)$.

Gray Maps. A Gray code is a representation of the non-negative integers in binary, such that consecutive integers have only one bit different in their representation.

Let $s$ be a positive integer. Gray map $\phi_s$ is a mapping from the integers in $[0, 2^s - 1]$ to bit string $\{0, 1\}^s$ such that the mapping results in the properties of a Gray code. One such mapping is as follows.

Let $u = (b_s b_{s-1} b_{s-2} \cdots b_1 b_0)_2$ with $b_s = 0$ be the standard binary representation of $u \in [0, 2^s - 1]$. Then $\phi_s(u) = (g_{s-1}, g_{s-2}, \cdots, g_1, g_0)$, where

\[ g_j = (b_j + b_{j-1}) \pmod{2}, j = 1, 2, \cdots, s - 1 \]
It is easy to see that this gives the properties of the Gray code. Moreover, the mapping is injective and surjective. Hence we can define an inverse Gray map \( \psi : \{0,1\}^{m_i} \rightarrow [0, i - i] \) where \( m_i = \lceil \log_2 i \rceil, i = 2, \cdots, 2^s \). This mapping is injective, but not surjective unless \( i \) is a power of 2.

**Linear Transformations.** Let \( A, B \) be vector spaces over the same field \( F \). \( f : A \rightarrow B \) is a *linear transformation* if it satisfies the following two properties:

- \( f(x + y) = f(x) + f(y) \) for all \( x, y \in A \).
- \( f(cx) = cf(x) \) for all \( x \in A, c \in F \).

For a linear transformation \( f \) define the *kernel* of \( f \) as the set of values in the domain \( A \) that is mapped to 0 by the transformation. That is, \( \text{ker}(f) = \{ x \in A : f(x) = 0 \} \). Due to the properties of linear transformation above, this will be a vector space.

One interesting property of a linear transformation is that the kernel space is a subspace of \( A \). Hence if \( A \) is finite, \( |\text{ker}(f)| = \frac{|A|}{|A|} \), where \( |A| \) denotes the number of vectors in the vector space \( A \). Also, \( f \) is injective if and only if \( \text{ker}(f) \) only contains the trivial solution 0.

**Linearized Polynomials.** Let \( q = p^k \) for some prime \( p \), and consider polynomials over \( \mathbb{F}_q \). A polynomial with coefficients in \( \mathbb{F}_q \) is called *linearized of degree* \( v \) if it has the form \( L(x) = \sum_{i=0}^{v} a_i x^{p^i} \).

Consider the transformation \( L : \mathbb{F}_q \rightarrow \mathbb{F}_q^v \). For a scalar \( c \neq 0 \) and \( x, y \in \mathbb{F}_q \) we have that

\[
L(x + y) = \sum_{i=0}^{v} a_i (x + y)^{p^i} \\
= \sum_{i=0}^{v} a_i (x^{p^i} + y^{p^i}) \text{(since the characteristic is } p, \text{ so } (x + y)^p = x^p + y^p + pf(x, y, p) = x^p + y^p) \\
= \sum_{i=0}^{v} a_i x^{p^i} + \sum_{i=0}^{v} a_i y^{p^i} \\
= L(x) + L(y)
\]

and

\[
L(cx) = \sum_{i=0}^{v} a_i (cx)^{p^i} \\
= \sum_{i=0}^{v} a_i (c^{p^i} x^{p^i}) \\
= \sum_{i=0}^{v} a_i (cx^{p^i}) \text{(since } c^p \equiv c \pmod{p}) \\
= \sum_{i=0}^{v} a_i x^{p^i} + \sum_{i=0}^{v} a_i y^{p^i} \\
= cL(x)
\]
between any two valid codewords is at least \( d \).

An \((c, d)\)-tuple such that the distance \(d\) between these codes is the far apart these codes are from each other, known as the distance between the two codes. An \((n, d)\) code is such that every codeword can be written as an \(n\)-tuple such that the distance between any two valid codewords is at least \(d\), also known as the minimum distance.

**Codes** A codeword \(c\) is a representation of some information \(x\) with the goal that the receiver can obtain the correct \(x\) even if there were errors in the transmission of \(c\). More formally, for some positive integers \(n, k\) we define an \((n, M)\) code \(C\) over an alphabet \(F\) to be a set of \(M = |F|^k\) vectors, each of length \(n\) over \(F\). Then a codeword \(c\) is just an element of \(C\). Here, \(k\) is the dimension of \(C\) and \(M\) is the cardinality of \(C\). Some families of codes widely used in practice are the Reed-Solomon codes and BCH codes. These codes have several nice properties, including encoding and decoding algorithms that run in time polynomial in the codeword length \(n\).

**Distance** In coding theory, an error occurs when \(c_1 \in C\) was encoded and transmitted, but the receiver decodes and receives a different word \(c_2\), not necessarily in \(C\). The magnitude of the error is how far apart these codes \(c_1, c_2\) are from each other, known as the distance between the two codes. An \((n, d)\) code is such that every codeword can be written as an \(n\)-tuple such that the distance between any two valid codewords is at least \(d\), also known as the minimum distance.

**Code rate** Code rate is a quantity which measures how many bits of information are contained in each bit of code. Let \(X\) be the set of valid \(n\)-length codewords. Then the rate of a code \(C\) can be defined as \(R(C) = \frac{\log_2 |C|}{\log_2 |X|}\).

### 2.2 Rank Modulation Codes

In this section we will use the basic definitions in the context of rank modulation codes.

**Kendall tau.** Let \(c_1, c_2 \in S_n\). We define a Kendall tau distance between two permutations. The Kendall tau distance \(d_r\) is the minimum number of adjacent transpositions, or element swaps, needed to transform a permutation \(c_1 \in S_n\) to another permutation \(c_2 \in S_n\). Note that as each transposition is invertible, we always have that \(d_r(c_1, c_2) = d_r(c_2, c_1)\). Also, by definition \(d_r(c_1, c_2) \geq 0\) and equality holds if and only if \(c_1 = c_2\). Finally, \(d_r(c_1, c_2) \leq d_r(c_1, c_3) + d_r(c_1, c_3)\) Hence \(d_r\) is a metric.

We define an \((n, d)\) rank modulation code as a set of permutations of length \(n\) (i.e. a subset of \(2^{S_n}\)) which has Kendall tau distance at least \(d\). The code rate is defined as \(R(C) = \frac{\log_2 |C|}{\log_2 n^k}\).

**Bounds on Code Rates** Define \(\chi_n = (S_n, d_r)\) to be the metric space of permutations in \(S_n\) with the metric \(d_r\). Define the maximal code rate \(R(n, d) = \max_{C \subseteq \chi_n} R(C)\) and asymptotic code rate \(\mathcal{C}(d) = \lim_{n \to \infty} R(n, d)\). The following theorem gives precise values of \(\mathcal{C}(d)\) for an optimal family of codes.

**Theorem 1.** \([BM10]\) \(\mathcal{C}(d) = \begin{cases} 1 & \text{if } d = O(n) \\ 1 - \epsilon & \text{if } d = \Theta(n^{1+\epsilon}), 0 < \epsilon < 1 \\ 0 & \text{if } d = \Theta(n^2) \end{cases}\)
Proof. This is a proof sketch based on the proof in [BM10]. First we define $A(n,d)$ as the maximum size of the code in $X_n = 2^{S_n}$. Then $C(d) = \lim_{n \to \infty} \frac{\log_2 A(n,d)}{\log_2 n}$. Let $B_r$ denote the sphere of radius $r$ around a codeword, or formally $B_r(c) = \{ \sigma \in S_n | d_r(c,\sigma) \leq r \}$. Obviously, spheres of radius $\frac{d-1}{2}$ around each codeword do not intersect due to the triangle inequality, which gives us the bounds $\frac{n!}{(d)!} \leq R(n,d) \leq \frac{n!}{d^{2n}}$. Hence if we can find bounds on $B_n$, we can get a result. However, we also note that $B_r = \sum_{i=1}^{n} K_n(i)$, where $K_n(i)$ is the number of permutations of $n$ elements with exactly $i$ inversions. So we only need bounds for $K_n$. A result by Louchard and Prodinger [LP03] and Margolius [Mar01] showed that there exist constants $c_1,c_2$ such that

$$K_n(k) \leq 2^{\epsilon n}, \text{if } k = O(n)$$

$$K_n(k) = \frac{n!}{2^{\epsilon n}}, \text{if } k = \Theta(n^2)$$

This leads to the proof that $C(d) = 1$ for $d = O(n)$ and $C(d) = 0$ for $d = \Theta(n^2)$.

The condition $d = \Theta(n^{1+\epsilon}), 0 < \epsilon < 1$ requires some very advanced steps to show that

$$\frac{n^n}{(12e)^n n^{ne}} \leq A(n,d) \leq \frac{2n^n(n-1)^{n-1}}{(n+d-1)^{n-1}}$$

which leads to $1 - \epsilon \leq C(d) \leq 1 - \epsilon + o(1)$, meaning $C(d) = 1 - \epsilon$.

3 Construction I: Linearized Polynomials

The first and most simple construction of rank modulation codes is based on permutation polynomials. Let $q = p^n$ be a prime power for some prime $p$. Then there exists a finite field $\mathbb{F}_q$ with $q$ elements. Let $\{\alpha_0, \alpha_1, \ldots, \alpha_{q-1}\}$ be a set of elements of $\mathbb{F}_q$.

A permutation polynomial is a polynomial $f(x) \in \mathbb{F}_q[x]$ such that $\{f(\alpha_0), \ldots, f(\alpha_{q-1})\}$ is a permutation of $\{\alpha_0, \ldots, \alpha_{q-1}\}$. It is very difficult to create codewords directly from permutation polynomials. Consider what happens if we transform these permutation polynomials into linearized polynomials. Let $\mathcal{L}_v$ be the set of all linearized polynomials of degree $v$. We can get a good lower bound to the number of permutations in the linearized case, based on the following theorem.

Lemma 1. Let $\mathcal{L}_v$ be the set of all linearized polynomials of degree $v$. The number of polynomials in $\mathcal{L}_v$, which are permutation polynomials in $\mathbb{F}_q$, is at least $q^v$.

Proof. Since $\mathcal{L}(x) = \sum_{i=0}^{v} a_i x^i$ has coefficients in $\mathbb{F}_q$, there are exactly $q^{v+1}$ of them in $\mathcal{L}_v$. As we have seen, the linearized polynomial $\mathcal{L}(x)$ is a linear transformation over $\mathbb{F}_q$. Hence we have that for every $x, y \in \mathbb{F}_q$, we have $\mathcal{L}(x) = \mathcal{L}(y) \iff \mathcal{L}(x - y) = 0$. This means that a linearized polynomial is a permutation polynomial if and only if its only root is the trivial root $0$.

We will now try to give an upper bound on the number of polynomials $\mathcal{L} \in \mathcal{L}_v$ with non-trivial roots $\alpha \in \mathbb{F}_q$. Let the coefficients $a_0, a_1, \ldots, a_v$ of $\mathcal{L}$ be chosen uniformly random from $\mathbb{F}_q$. For each such $\mathcal{L}$ and fixed $\alpha \in \mathbb{F}_q$, we know by the Schwartz-Zippel lemma [Sch80] that $Pr[\mathcal{L}(\alpha) = 0] \leq 1/q$. Moreover, we know from MacWilliams and Sloane [MS91] that the set of roots of a linearized
polynomial form a vector space over $\mathbb{F}_p$, so if $\alpha \in \mathbb{F}_p^m$ is a root, so is $c\alpha, c \in \mathbb{F}_p$. This implies that there are a multiple of $p - 1$ such non-zero roots. But then we only need to sum over the possible 1-dimensional subspaces of $\mathbb{F}_q$ over $\mathbb{F}_p$, and there are $\frac{q-1}{p-1}$ of these.

Therefore, the union bound gives for randomly chosen $L$,

$$\Pr[\exists \alpha \in \mathbb{F}_q^* | L(\alpha) = 0] \leq \sum_{i=1}^{\frac{q-1}{p-1}} \frac{1}{q} = \frac{q-1}{p-1} \cdot \frac{1}{q} \leq \frac{q-1}{q}.$$ 

Hence the number of permutation polynomials is at least $(1 - \frac{q-1}{q})q^{n-1} = \frac{1}{q} \cdot q^{n} = q^n$.

Let $t$ be a positive integer and $v = n - 2t - 1$. Let $P_v$ be the set of linearized polynomials of degree $v = n - 2t - 1$ which are permutation polynomials. Let $A = \{(L(a), a \in \mathbb{F}_q^*) | L \in P_v\}$ be a set of distinct $q-1=n$-dimensional vectors achieved by evaluating each linearized permutation polynomial on every non-zero polynomial. We then fix a bijection between $\mathbb{F}_q^*$ and $\{1, 2, ..., n\}$, for example $\alpha_i \leftrightarrow i$, such that each vector $(\alpha_{a_1}, \alpha_{a_2}, \cdots, \alpha_{a_n})$ becomes a permutation $(a_1, a_2, \cdots, a_n)$. By this bijection we can get from $A$ to a rank modulation code $C_r$.

**Theorem 2.** The code $C_r$ has length $n = q - 1$ and at least $q^v = q^{\lceil \log_p n - 2t - 1 \rceil}$ elements. It can correct up to $t$ Kendall errors in the rank modulation scheme, and has an efficient (polynomial time in $n$) decoding.

**Proof.** For first part of the theorem, since there is a bijection, we have $|C_r| = |A|$. But from Lemma 2 we have that $|A| \geq q^n$. The proof of the second part of the theorem can be seen in the following construction. The decoding algorithm is dominated by one matrix multiplication and one run of the Reed-Solomon decoding algorithm for $(n, m = n - 2t, d = 2t + 1)$ Reed-Solomon codes, both of which are polynomial in $n$.

Note that since $n - 2t - 1 > 0$, we can correct up to $t = O(n)$ errors using this construction.

### 3.1 Encoding

We start by defining the $n \times n$ matrix $P$ which has ones on and under the main diagonal, and zeroes everywhere else. That is,

$$P = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ 1 & 1 & 0 & \cdots & 0 \\ 1 & 1 & 1 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 1 & 1 & 1 & \cdots & 0 \end{pmatrix}$$

This is known as the accumulator matrix, since for $x = (x_1, x_2, \cdots, x_n)$ we have that

$$P x^T = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ 1 & 1 & 0 & \cdots & 0 \\ 1 & 1 & 1 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 1 & 1 & 1 & \cdots & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \cdots \\ x_n \end{pmatrix} = \begin{pmatrix} x_1 \\ x_1 + x_2 \\ x_1 + x_2 + x_3 \\ \cdots \\ x_1 + x_2 + x_3 + \cdots + x_n \end{pmatrix}$$
The idea is that if we have an error in which \( \sigma = (a_1, \ldots, a_i, a_{i+1}, \ldots, a_n) \) becomes \( \sigma' = (a_1, \ldots, a_{i+1}, a_i, \ldots, a_n) \). But since \( \sigma - \sigma' = (0, 0, \cdots, a_i - a_{i+1}, a_{i+1} - a_i, \cdots, 0) \), we have that \( P(\sigma - \sigma')^T = (0, 0, \cdots, a_i - a_{i+1}, 0, \cdots, 0)^T \). This means that \( P \) converts an error in the form of adjacent transposition in a permutation to a Hamming error in a vector.

The encoding algorithm is as follows:

1. Given a permutation \( \pi \) set \( \sigma \) as the corresponding linearized polynomial according to the bijection.
2. Send the codeword \( \sigma \).

### 3.2 Decoding

The decoding algorithm is as follows:

1. Given \( \hat{\sigma} \), evaluate \( z = P\hat{\sigma}^T \).
2. Use Reed-Solomon decoding to correct up to \( t \) Hamming errors of the Reed-Solomon "code" \( z \), giving us \( y \).
3. Compute \( \sigma = P^{-1}y^T \).

Decoding works since \( P\hat{\sigma}^T \) is a permutation of the elements of \( \mathbb{F}_q^n \) (since \( P \) is invertible, \( P\sigma^T = P\sigma'^T \) \( \implies \sigma^T = \sigma'^T \implies \sigma_1 = \sigma_2 \)), but it is also a vector which can be seen as an evaluation of a polynomial of degree \( p' \leq p \log_q n - 2t - 1 = n - 2t - 1 \). Hence the set of vectors of the form \( P\hat{\sigma}^T \) is a subset of an \( (n, n - 2t, 2t + 1) \) Reed-Solomon code. So we can use standard Reed-Solomon decoding algorithms to efficiently get the correct \( y = P\sigma^T \) from the "received" \( z = P\hat{\sigma}^T \), as long as the number of errors \( d_r(\sigma, \sigma_1) \leq t \). From here it is easy to obtain the submitted codeword as \( \sigma = P^{-1}y^T \).

### 4 Construction II: Codes with Optimal Scaling Rate

We start by noting that there is a one-to-one correspondence between the permutations \( \sigma \in S_n \) and inversion vectors \( x_\sigma \in G_n \). We can thus define the inverse map \( J : G_n \rightarrow S_n \) which maps an inversion vector \( x_\sigma \) back to its permutation \( \sigma \). The \( \ell_1 \) distance function \( d_1 \) on \( G_n \) is defined as

\[
d_1(x, y) = \sum_{i=1}^{n-1} |x(i) - y(i)|
\]

with \( x, y \in G_n \). The inverse map \( J \) has a useful property that it increases the distance between two inversion vectors. Formally we have the following lemma:

**Lemma 2.** For every \( \sigma_1, \sigma_2 \in S_n \), we have

\[
d_r(J(x_{\sigma_1}), J(x_{\sigma_2})) = d_r(\sigma_1, \sigma_2) \geq d_1(x_{\sigma_1}, x_{\sigma_2}).
\]

Next, let \( m_i = \lfloor \log i \rfloor, i = 2, 3, \cdots n \). Then the inverse map \( \psi_i : \{0,1\}^{m_i} \rightarrow \{0, 1, \cdots, i - 1\} \) also increases distance, more formally:

**Lemma 3.** For every \( x, y \in \{0, 1\}^{m_i} \), we have

\[
|\psi_i(x) - \psi_i(y)| \geq d_H(x, y).
\]
The above lemma can be generalized if we define $m = m_2 + \cdots + m_n$ and a mapping $\Psi : \{0,1\}^m \to \mathcal{G}_n$ as

$$\Psi(x) = \Psi(x_2|x_3|\cdots|x_n) = (\psi(x_2), \psi(x_3), \cdots, \psi(x_n)).$$

**Lemma 4.** For every $x, y \in \{0,1\}^m$, we have

$$d_1(\Psi(x), \Psi(y)) \geq d_H(x, y).$$

Thus we have a general idea for the second construction: start with a binary codeword in $\{0,1\}^m$ and apply $\Psi$ to get an inversion vector in $\mathcal{G}_n$, then apply the inverse Gray map $J$ to transform this inversion vector to a permutation in $S_n$. By the previous lemmas, the transformations will not decrease the distance between codewords, and hence decoding works as long as the binary code decoding is successful.

### 4.1 Encoding

We start by fixing a binary $(m, M, d)$ code $A$ which has length $n$, cardinality $M$ and Hamming distance $d$. The encoding algorithm is as follows:

1. Given a binary message $m$, encode it using the code $A$ to obtain a binary vector $x$ of length $m$.
2. Write $m = m_2 + \cdots + m_n$, and hence split $x$ such that $x = (x_2|x_3|\cdots|x_n)$, where the vector $x_i$ has length $m_i$.
3. Send the permutation $\sigma = J(\Psi(x))$.

### 4.2 Decoding

The decoding algorithm is as follows:

1. Given $\hat{\sigma}$, create the inversion vector $x_{\hat{\sigma}}$. Fix it into a vector $y_{\hat{\sigma}}$ as follows:
2. Create a vector $y = (y_2|y_3|\cdots|y_n)$ where $y_i$ has length $m_i$.
3. Apply the decoding algorithm of the binary code $A$. It will either return a decoding $x$, or detect that there were more than $t$ errors.
4. Output $\sigma = J(\Psi(x))$.

### 4.3 Code Rates for Construction II

**Theorem 3.** Let $A$ be the binary $(m, M, d)$ code of length $m = (n+1)\lfloor \log_2 n \rfloor - 2\lfloor \log_2 n \rfloor + 2$, and $C_\tau$ be the rank modulation code defined in Construction II. Then $C_\tau$ also has distance at least $d$ in the Kendall space. If $d = cm$ for some $0 < \epsilon < 1/2$ then the minimum Kendall distance of $C_\tau$ is $\Omega(n^{1+\epsilon})$, and the code rate converges to $R = 1 - \epsilon$.

Construction II will thus be an optimal code for $0 < \epsilon < 1/2$. Using ideas from the above theorem, we can also prove the existence of optimal codes with $d = \Omega(n^{1+\epsilon})$ and code rate $R = 1 - \epsilon$ for $0 < \epsilon < 1$.

**Theorem 4.** Let $m = m_2 + \cdots + m_n$ be defined as in Construction II. For any $0 < \epsilon < 1$ there exists a family of binary $(m, M)$ codes $C$ as in Construction II such that the code rate $R$ converges to $1 - \epsilon > 0$ and can correct codes with up to $d$ errors, where the minimum Kendall distance $d = \Omega(n^{1+\epsilon})$. 
5 Construction III: Dealing with Many Errors

In this section, we will deal with codes that can correct up to $\Theta(n^2)$ errors. Note that as the number of Kendall errors is at most $\frac{n(n-1)}{2}$, then this is the most number of errors that can occur. Also, as we saw in Theorem 1, the rate converges to 0 for large $n$. However, there still exist reasonable code constructions for smaller $n$.

5.1 Encoding

We start by fixing a binary code $\mathcal{A}$ that maps $k$ bits to $n-1$ bits. Let $\mathcal{V} : \{0, 1\}^{n-1} \rightarrow \mathcal{G}_n$ be defined as $\mathcal{V}(b_1, \cdots, b_{n-1}) = (x_1, x_2, \cdots, x_{n-1})$ where

$$x_i = \begin{cases} 0 & \text{if } b_i = 0 \\ i & \text{if } b_i = 1 \end{cases}$$

Obviously, the range of this function is $\{0, 1\}^n$ with cardinality $2^n$, and since $\frac{2^n}{m} \rightarrow 0$, the code rate converges to 0.

The encoding algorithm is as follows:

1. Given a binary message $m$, encode it using the code $\mathcal{A}$ to obtain a binary vector $b$ of length $n - 1$.
2. Compute $x = \mathcal{V}(b)$.
3. Output the encoding $\sigma = J(x)$

Note that this construction can be generalized for any $q$-ary codes.

5.2 Decoding

The decoding algorithm is as follows:

1. Given $\hat{\sigma}$, create the inversion vector $x_{\hat{\sigma}}$.
2. Create a vector $y = (y_1, y_2, \cdots, y_{n-1})$ where $y_i$ has length $m_i$.
3. Decode $y$ in code $\mathcal{A}$ to get $c$
4. Compute $\sigma = J(\mathcal{V}(c))$

We saw that the code rate converges to 0, but we will now show that it indeed corrects $\Theta(n^2)$ errors.

**Theorem 5.** If $A$ is a binary $(n-1, M, d)$ code such that $d \geq 2k + 1$, then the constructed code $C_\tau$ above can correct up to $\left\lfloor \frac{k^2}{4} \right\rfloor$ errors.

Hence letting $k = \Theta(n)$ we can correct $\Theta(n^2)$ errors. Here, the maximum number of errors we can correct will increase for a specific $n$, if we also use a $q$-ary code with large enough $q$. 

6 Conclusion

In this report, we have discussed rank modulation codes of length $n$. The main motivation of rank modulation codes was flash memory, where electric charge leakage can decrease charge values in many cells but typically changes much less of the relative values, hence the idea of using relative values and not absolute values.

We have seen constructions of rank modulation codes that are optimal with Kendall tau distance $d$, for the cases $d = O(n)$, $d = \Theta(n^{1+\epsilon})$, $0 < \epsilon < 1/2$ and $d = \Theta(n^2)$. An existence theorem showing that there exist optimal rank modulation codes in the leftover case $d = \Theta(n^{1+\epsilon})$, $1/2 < \epsilon < 1$ was also proven. The optimality of the constructions are due to Theorem 1.

Although the constructions are asymptotically optimal, there might be better constructions for specific values $n, d$ that would be more suitable for practical use. One idea for improvement is to use a reasonable metric as an alternative to Kendall tau. The code construction can possibly be improved is the case $d = \Theta(n^{1+\epsilon})$, $1/2 < \epsilon < 1$. We only have an existence result, but it would be interesting to see if there can be a construction that reaches optimality in that case.

References


