What is Coding Theory?

Coding Theory is the study of error transmission. It is not Cryptography, as it has to do with encoding and reliability as opposed to encryption and security. Thus, in a perfect world where data could be transferred without any error, this field of study would not be needed.

The Structure of a Linear Code

A code is a subset of $n$-tuples of a finite field. A linear code is a vector space over a finite field, so any element, or codeword, in this set is a linear combination of other codewords. A linear code $C$ is defined by three parameters:

1. **Length $n$**
   The total number of “letters” in each codeword of $C$. (The total number of bits in a codeword.)

2. **Dimension $k$ ($\leq n$)**
   The number of basis elements for $C$. (The number of “information” bits in a codeword.)

3. **Minimum Distance $d$**
   The smallest number of differences between the positions of any two codewords in $C$. For Linear Codes, this is equivalent to the Minimum Weight, or smallest number of non-zero bits in a codeword that isn’t the 0 vector.

We refer to these codes as $[n, k, d]_q$ codes, where $q$ is the size of the finite field $\mathbb{F}_q$ over which $C$ is a vector space.

Best Known Linear Codes

The distance between any two codewords is the number of positions by which they differ. The distance between the following two codewords is one:

$$\begin{bmatrix} 101100 \end{bmatrix}$$

The minimum distance $d$ of a linear code is what determines that code’s capacity for error detection and error correction. Detectable Errors: $d-1$ Correctable Errors: $\left\lfloor \frac{d}{2} \right\rfloor$

For a given value of length $n$ and dimension $k$, a known $[n, k, d]$ code with the largest known minimum distance is said to be a best known linear code (BKLC). There exist databases with both lower and upper bounds for these codes; thus, our goal is to find codes with the largest known minimum distance.

Types of Codes

**Quasi-Twisted Codes**

Thus, we can use constacyclic codes as building blocks for them. To do this, we start with the generating polynomial of a constacyclic code, and add on that generator multiplied with random polynomials of a certain condition, and use this as a new generator array. This ensures that the new minimum distance $d' \geq d + \ell$.

$$G = \begin{bmatrix} g(x) \end{bmatrix} \Rightarrow \begin{bmatrix} g(x), g(x) \cdot f_1(x), g(x) \cdot f_2(x), \ldots \end{bmatrix}$$

**Constatyclic Generation**

Algebraically, it is easier to work with polynomials instead of vectors. Thus, we convert the codewords we deal with into polynomial notation as such:

$$101100 \rightarrow (x^3 + 0x^2 + 1x^1 + 0x^0)(x^3 + 0x^2 + 1x^1 + 0x^0) = 1 + x^2 + x^3$$

Thus, we can describe a constacyclic shift as multiplication modulo $x^n - a$. We can also use algebra to find all possible generators.

1. **Constacyclic codes are ideals in** $\mathbb{F}_q[x]$.
2. **$\mathbb{F}_q[x]$ is a principal ideal ring.**
3. **Each $C \subseteq \mathbb{F}_q[x]$ is generated by some $g(x) \in \mathbb{F}_q[x]$.**
4. **If $(g(x)) = C$, then $g(x)$ is a divisor of $x^n - a$.**
5. **There is a one-to-one correspondence between the divisors of $x^n - a$ and $[n, k, d]$ constacyclic codes.**

Thus, we can find every single possible constacyclic code by factoring $x^n - a$ and using all possible factor combinations as a generator. While this gives us an exhaustive search in theory, it is not possible to compute every single minimum distance of these codes due to computational complexity. Thus, we use two techniques to find as many as possible:

1. For polynomials with a large number of irreducible factors, we try to find a cap (such as $10^6$) where looking at more factor combinations does not seem to find better codes.
2. We would discard codes with large dimensions which made computations too complex.

**Boundaries of Our Search**

Each divisor of $x^n - a$ will generate a constacyclic code with shift constant $a \in \mathbb{F}_q$. However, for a given code length $n$, we need not consider each $a \in \mathbb{F}_q$. For some values of $a$ and $n$, the generated code will be equivalent to a cyclic code. Thus, we only need to consider certain combinations of $a$ and $n$ to exhaustively generate all constacyclic codes over a certain finite field, as long as we have looked over all cyclic codes.

<table>
<thead>
<tr>
<th>$q$</th>
<th>$a \neq 0, 1$</th>
<th>$n$</th>
<th>Maximum $n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>all $n = 2m$</td>
<td>243</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>all $n = 2m$ or $n = 5m$</td>
<td>255</td>
<td></td>
</tr>
<tr>
<td></td>
<td>all $n = 5m$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>all $n = 2m$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>all $n = 2m$ or $n = 3m$</td>
<td>255</td>
<td></td>
</tr>
<tr>
<td></td>
<td>all $n = 3m$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>all $n = 3m$ or $n = 4m$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>all $n = 2m$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>all $n = 4m$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

For $GF(3)$, we used the bounds found at [3], while for $GF(11)$ and $GF(13)$, we used the bounds found at [7] and [8]. We consider all values of $k < n$ for $GF(3)$, $3 \leq k < 8$ for $GF(11)$, and $3 \leq k < 7$ for $GF(13)$ as those are the bounds that are in the databases.

Our Results

We were able to find 42 record-breaking codes using the methods previously outlined:

- $GF(3)$ Quasi-twisted: 29
- $GF(11)$ Constacyclic: 2
- $GF(11)$ Quasi-twisted: 6
- $GF(13)$ Constacyclic: 5

An Example: [22, 7, 14] $GF(11)$ QT-Code ($\ell = 2$)

Factoring $x^{12} - 1$

$$g(x) = x^7 + 7x^6 + x^5 + 7x + 1,
\quad f_1(x) = 10x^6 + 3x^5 + x^3 + 4x^2 + 7x + 2$$

The reason we found many more $GF(3)$ QT codes than in $GF(11)$ or $GF(13)$ is simply because we have a much large database for $GF(3)$ than the other two alphabets.

Moving forward, we look to expand this method over the other finite fields we have databases for. Furthermore, we are working on a new, top-down method, and have been able to find new codes using it. However, we are still working on both the theory and implementation of this new method.

Top-Down Method

The new method starts by taking a specific kind of constacyclic code, called a simplex code. A simplex code has parameters $[q^d-1]/(q-1), k, q^d-1$ where $q$ must be a prime power (which is true for any finite field). We then take the (redundant) $n \times n$ generator matrix where each row is a constacyclic shift of the previous row. This is also known as a twistulant matrix. We then choose two integers $m$ and $p$ such that $n = mp$. By grouping the $t$-th, $(p+1)-t$-th, …, $(m-1)p+t$-th rows and columns, this matrix becomes one made up of smaller, cyclic matrices. We then take the defining polynomial of these cyclic matrices, find their weight, and replace each cyclic matrix with that weight to create the weightmatrix. We can find new quasi-twisted codes by taking $r$ columns from this matrix. The defining polynomials of the top row cyclic matrices of these columns will form a generating matrix of an $[r \times n, k]_q$ quasi-twisted code with minimum distance equal to the minimum row sum of these columns.

As it is computationally complex to find the $r$ columns that form the largest minimum row sum, we use the heuristic method as described in [9] to find new codes. This method starts with any one column and then continues to add on the column that produces the largest minimum row sum the least amount of times. In the case of ties, the first one found is the one chosen.

Quasi-Twisted Extension

**Quasi-twisted codes** are a generalization of constacyclic codes. Thus, we can use constacyclic codes as building blocks for them. To do this, we start with the generating polynomial of a constacyclic code, and add on that generator multiplied with random polynomials of a certain condition, and use this as a new generator array. This ensures that the new minimum distance $d' \geq d + \ell$.

**References**


