

New Linear Codes as Quasi-Twisted Codes from Long Constacyclic Codes

Nuh Aydin

Department of Mathematics and Statistics
Kenyon College
Gambier, OH 43022
Email: aydinn@kenyon.edu

Thomas Guidotti

Kenyon College
Gambier, OH 43022
Email: guidotti1@kenyon.edu

Peihan Liu

Kenyon College
Gambier, OH 43022
Email: liu4@kenyon.edu

Abstract—One of the most important and challenging problems in coding theory is to determine the optimal values of the parameters of a linear code and to explicitly construct codes with optimal parameters, or as close to the optimal values as possible. The class of quasi-twisted (QT) codes has been very promising in this regard. Over the past few decades various search algorithms to construct QT codes with better parameters have been employed. Most of these algorithms (such as ASR [11]) start by joining constacyclic codes of smaller lengths to obtain QT codes of longer lengths. There has been an algorithm that works in the opposite way that constructs shorter QT codes from long constacyclic codes. We modified and generalized this algorithm and obtained new linear codes via its implementation. We also observe that the new algorithm is related to the ASR algorithm.

Index Terms—quasi-twisted codes, best known linear codes, constacyclic codes, search algorithms for linear codes

I. INTRODUCTION

A linear block code C of length n over the finite field \mathbb{F}_q is a vector subspace of \mathbb{F}_q^n . The elements of C are called codewords. If the dimension of C is k , then it is referred to as an $[n, k]_q$ -code. If the minimum (Hamming) distance (weight) is d , then it is an $[n, k, d]_q$ -code. One of the most important and challenging problems in coding theory is to determine the optimal values of the parameters n, k, d , given the alphabet size q and explicitly construct codes whose parameters attain the optimal values. The problem can be formulated in a few different ways. One common version is to fix q, n and k and look for the maximum possible value of d . In general, this optimization problem is very hard and in most cases codes with optimal parameters are not known. The online database [1] contains data about what is known about this problem for codes over the alphabets \mathbb{F}_q for $q \leq 9$ up to certain length for each alphabet. In most cases, there are gaps between the minimum distances of best known linear codes (BKLC) and the best theoretical upper bound on d .

This optimization problem is hard for two main reasons. First, the number $\frac{(q^n - 1)(q^n - q) \cdots (q^n - q^{k-1})}{(q^k - 1)(q^k - q) \cdots (q^k - q^{k-1})}$ of linear codes over \mathbb{F}_q of length n and dimension k is large and grows fast. Hence exhaustive computer searches are not feasible for all but small values of n or k . Secondly, computing the minimum distance (weight) of a linear code is computationally intractable [10]. For most entries in the database [1], optimal

values of the parameters are not attained. Optimal codes are generally known when either k or $n - k$ is small.

Numerous approaches, techniques, and search methods have been employed to improve the parameters of BKLCs to get closer to the optimal values. It is unlikely that a single method will work to solve most instances of this challenging problem. One method that has been quite effective to obtain new codes has been computer searches in the class of quasi-twisted (QT) codes. The algorithm ASR introduced in [11] is one such algorithm. It has been improved in recent years and produced dozens of record breaking codes. The ASR algorithm searches for new linear codes using a special type of 1-generator QT codes that have generators in a particular form. It starts with a short constacyclic code and uses it as a building block to construct longer QT codes. Another method was introduced in [13] that works in the opposite direction, that is by starting with a very long constacyclic code and obtains shorter QT codes from it. In a recent paper [15], new linear codes have been obtained via a modification of the original method. In this work, we modified and generalized this method and obtained new linear codes from its implementation. Finally, we notice a connection between the new method and the ASR algorithm.

II. PRELIMINARIES

Cyclic codes are one of the most important classes of codes in algebraic coding theory for both theoretical and practical purposes. They are extensively studied and generalized in many directions. Some of the well known generalizations of cyclic codes are constacyclic codes, quasi-cyclic (QC) codes, and quasi-twisted (QT) codes.

Definition II.1. Let \mathbb{F}_q be the finite field with q elements and let $a \in \mathbb{F}_q^* = \mathbb{F}_q \setminus \{0\}$. A linear code C of length n over \mathbb{F}_q is called a quasi-twisted (QT) code of index ℓ (or an ℓ -QT code) if it is closed under the constacyclic shift by ℓ positions, i.e., for any codeword $\mathbf{c} = (c_0, c_1, \dots, c_{n-1}) \in C$, we also have $\pi_{\ell, a}(\mathbf{c}) = (ac_{n-\ell}, \dots, ac_{n-1}, c_0, c_1, \dots, c_{n-\ell-1}) \in C$.

The smallest such positive integer ℓ is called the index of C and it must divide the code length n . Hence, $n = m \cdot \ell$ for some $m \in \mathbb{Z}^+$. The scalar $a \in \mathbb{F}_q^*$ is called the shift constant. The following are some of the most important special cases of QT codes:

- $a = 1, \ell = 1$ gives cyclic codes
- $a = -1, \ell = 1$ gives negacyclic codes
- $a = 1$ gives quasi-cyclic (QC) codes
- $\ell = 1$ gives constacyclic codes

One of the reasons why cyclic codes are so prominent in coding theory is they establish a key link between algebra and coding theory through the correspondence between vectors $\mathbf{v} = (v_0, v_1, \dots, v_{n-1})$ and polynomials

$v(x) = v_0 + v_1x + \dots + v_{n-1}x^{n-1}$. This map establishes a vector space isomorphism between \mathbb{F}_q^n and $\mathbb{F}_q[x]_{<n} = \{p(x) \in \mathbb{F}_q[x] : \deg(p(x)) < n\}$, the set of all polynomials of degree $< n$ over \mathbb{F}_q . It is well known that under this identification cyclic codes of length n over \mathbb{F}_q correspond to the ideals of the quotient ring $\frac{\mathbb{F}_q[x]}{\langle x^n - 1 \rangle}$. Under the same identification, the algebraic structure of a QT code of length $n = m \cdot \ell$ is an R -module of R^ℓ , where $R = \frac{\mathbb{F}_q[x]}{\langle x^m - a \rangle}$. If C is generated by r elements of R^ℓ then it is called an r -generator QT code. A generator matrix of an r -generator QT code can be put, by applying a suitable permutation of the columns if necessary, into the form:

$$\begin{bmatrix} G_{11} & G_{12} & \cdots & G_{1\ell} \\ G_{21} & G_{22} & \cdots & G_{2\ell} \\ \cdots & \cdots & \cdots & \cdots \\ G_{r1} & G_{r2} & \cdots & G_{r\ell} \end{bmatrix}$$

where each G_{ij} is an a -circulant (also called a twistulant) matrix of the form

$$\begin{bmatrix} c_0 & c_1 & c_2 & \cdots & c_{m-1} \\ ac_{m-1} & c_0 & c_1 & \cdots & c_{m-2} \\ ac_{m-2} & ac_{m-1} & c_0 & \cdots & c_{m-3} \\ \cdots & \cdots & \cdots & \cdots & \cdots \end{bmatrix}$$

where each row is a constacyclic shift of the previous row.

The ASR search algorithm that is introduced in [11] is based on the following theorem.

Theorem II.2. [11] *Let C be a 1-generator QT code of length $n = m\ell$ over \mathbb{F}_q with a generator of the form*

$$(g(x)f_1(x), g(x)f_2(x), \dots, g(x)f_\ell(x))$$

where $x^m - a = g(x)h(x)$ and $\gcd(h(x), f_i(x)) = 1$ for all $i = 1, \dots, \ell$. Then $\dim(C) = m - \deg(g(x))$, and $d(C) \geq \ell \cdot d$ where d is the minimum distance of the constacyclic code C_g generated by $g(x)$.

This algorithm has been refined and automatized in more recent works such as ([3], [4], [5], [6], [8]) and dozens of record breaking codes have been obtained over every finite field \mathbb{F}_q , for $q = 2, 3, 4, 5, 7, 8, 9$ through its implementation. Moreover, it has been further generalized in [7] and more new codes were discovered that would have been missed by its earlier versions.

In an implementation of the ASR algorithm, we begin by choosing the alphabet \mathbb{F}_q and the shift constant a . We then pick a desired block length m and number of blocks ℓ such that the resulting length of the code is $n = m \cdot \ell$. Next, we find

all divisors of the polynomial $x^m - a$ and end up with a list of possible generator polynomials $g(x)$ (with corresponding check polynomial $h(x)$ such that $g(x) \cdot h(x) = x^m - a$) of varying degrees, each generator giving a constacyclic code C_g of length m with dimension $k = m - \deg(g(x))$. Then the search program generates polynomials $f_i(x)$ where $\gcd(f_i(x), h(x)) = 1$ for all i $1 \leq i \leq \ell$, and for each set $\{f_1(x), \dots, f_\ell(x)\}$ of polynomials constructs the QT code C with a generator of the form given in Theorem II.2. Note that, each block is a constacyclic code generated by $g(x)f_i(x)$. In fact, $\langle g(x)f_i(x) \rangle = \langle g(x)f_j(x) \rangle$ for all i, j under the condition $\gcd(h(x), f_i(x)) = 1$. We simply join all of these ℓ blocks together to obtain a QT code whose minimum distance is guaranteed to be at least $\ell \cdot d$. In many cases, its actual minimum distance is much bigger.

There is another method of constructing QC and QT codes that works in the opposite way (which we may informally refer to as ‘‘top-down method’’). The basic idea of this method goes back to [12] and [13]. It starts with a long cyclic code of composite length N and permutes the coordinates of the code so that each block is a cyclic code of length dividing N . This method was later refined to search algorithms that produced QT 2-weight codes from constacyclic codes in [14]

One improvement of the algorithm is given in [15] which applied the method to the specific case of simplex codes, the duals of Hamming codes. Simplex codes are constacyclic themselves, so the idea of applying a permutation to a long constacyclic code to form a matrix that is a generator for a QT code can be applied here. This version of the algorithm uses the idea of a weight matrix to construct better QT codes. After applying the permutation to the generator of the constacyclic code, a $p \times p$ weight matrix is created to store the weights of the defining polynomials $g_1(x), \dots, g_p(x)$. The first row of this matrix has the weights $wt(g_1(x)), \dots, wt(g_p(x))$ and all subsequent rows are its cyclic shifts, so it suffices to store the first row of the matrix only. Because of this construction, it follows that the minimum distance of a QT $[n = t \cdot m, k]_q$ code is determined by the minimum row sums of the chosen columns from the weight matrix. So to form a QT code with these parameters having the highest possible minimum distance it suffices to maximize the minimum row sums. This gives rise to the iterative algorithm found in [15]. The idea is that one will build a QT code from maximizing the minimum row sums one column at a time. So given a QT $[i \cdot m, k]_q$ code the algorithm tries to construct a QT $[(i + 1) \cdot m, k]_q$ code by using the weight matrix to add another block to the resulting QT code. If the maximum number of blocks desired is t , then the algorithm constructs codes with parameters $[m, k]_q, [2 \cdot m, k]_q \cdots [t \cdot m, k]_q$.

III. OUR CONTRIBUTION

In this work, we generalize and modify the search method described in the previous section. As a result of our implementation of the generalized algorithm, we have found 5 new linear codes that improve the bounds in the database [1].

The previous version of the algorithm is restricted to the class of simplex codes. Our first generalization is to apply it to the broader class of all constacyclic codes. We first choose the alphabet \mathbb{F}_q , a shift constant $a \in \mathbb{F}_q^*$, and a length N with many factors. Next, we find a number of generator polynomials $G(x)$ such that $G(x)|(x^N - a)$. We want the degree of this generator to be relatively high, so our initial program (written in Magma) that outputs the generator polynomials has constraints on their degrees. After reading in all of these long generator polynomials to a C++ program, we want to find all combinations of $m \cdot p = N$. This is obviously determined by the prime factorization of N . We refer to m as the block length and then p is the total number of blocks. For a given combination of (m, p) we first represent our long polynomial $G(x)$ as a vector in the usual way. Then we split up this vector into p vectors of length m each, such that each vector combines columns $i, i + p, i + 2p, \dots, i + (m - 1)p$ for $0 \leq i \leq p - 1$. Next, we convert all of these vectors back to polynomials $g_i(x)$ in the usual way and then compute $\gcd(x^m - a, g_1(x), g_2(x), \dots, g_p(x))$. Computing this gcd is very similar to the ASR search in which we begin with a standard generator polynomial $g(x)$ (a divisor of $x^m - a$) and then multiplying it by polynomials $f_i(x)$ in each block such that $\gcd(h(x), f_i(x)) = 1$. The process here is simply the reverse: we are starting with $g(x) \cdot f_i(x)$ in each block and we want to determine the standard generator $g(x)$ based on this. Obviously then the dimension of the constacyclic code with standard generator $g(x) = \gcd(x^m - a, g_1(x), g_2(x), \dots, g_p(x))$ is $k = m - \deg(g(x))$. If k is equal to the dimension of the original constacyclic code, then we are dealing with the 1-generator QT case, otherwise we are dealing with a multi-generator case. Our last step here is to construct a a -circulant (twistulant) matrix corresponding to each of these defining polynomials $g_i(x)$ for $1 \leq i \leq p$, keeping only those matrices whose rank is equal to k or $k - 1$. We observed that using blocks that have the same rank usually gives the best results in terms of obtaining codes with high minimum distances.

Obviously the dimension k has an upper bound of m , so it only makes sense to consider combinations of (m, p) that will yield reasonable dimensions. In many cases where there are a high number of defining polynomials, their gcds with $x^m - a$ have a very low degree, resulting in a QT code whose dimension is very close to block length m .

For each combination of (m, p) we apply the column permutations as described to put the large circulant matrix into p blocks of length m circulant matrices. We then horizontally join t of these matrices to form the generator matrix for a QT code. An improvement we made to this algorithm deals with the rank of these circulant matrices. In [13], the author notes that in cases where $N - \deg(G(x)) \neq m - \deg(g(x))$ ($G(x)$ is the generator polynomial of the long constacyclic code and $g(x)$ is the gcd of defining polynomials) we are dealing with a multi-generator QT code. Chen suggests that in such a case the smaller dimension generator polynomial can be augmented by adding rows one at a time such that the rows are linearly independent, in this way a code with

Note
THE TABLE BELOW JUST SHOWS A SMALL PART OF ALL RANKS
DISTRIBUTION

$N = m \cdot p$	Rank1: No.	Rank2: No.	Rank3: No.	Rank4: No.
924 = 4 · 231	3 : 49	2 : 140	1 : 21	n/a
924 = 6 · 154	4 : 126	3 : 14	2 : 14	n/a
924 = 21 · 44	7 : 32	6 : 12	n/a	n/a
924 = 22 · 42	11 : 28	10 : 14	n/a	n/a
924 = 28 · 33	14 : 20	13 : 9	13 : 4	n/a
924 = 12 · 77	5 : 50	4 : 18	3 : 6	2 : 3
924 = 4 · 231	2 : 151	1 : 59	n/a	n/a
924 = 12 · 77	4 : 70	2 : 6	n/a	n/a
924 = 11 · 84	11 : 56	10 : 28	n/a	n/a
924 = 14 · 66	13 : 48	12 : 18	n/a	n/a

dimension $N - \deg(G(x))$ may be constructed. After some testing using this idea we had no success in finding codes with parameters close to those of BKLCs, so instead we decided to construct codes with dimension $k = m - \deg(g(x))$ simply taking the blocks of circulant matrices of rank k rather than by adding additional linearly independent rows. Additionally we noticed that amongst all of the p circulant matrices, there are many matrices of rank $k - 1$ so we keep those as well. So once we choose k after finding the defining polynomials, we first determine if k is large enough to find reasonable results. After that we go through all circulant matrices and keep those whose rank is equal to k or $k - 1$. So when we end up selecting t blocks of circulant matrices for a given (m, p) combination, we are selecting t blocks from the set of rank k matrices and an additional t blocks from the rank $k - 1$ matrices. After horizontally joining these matrices we construct two QT codes with lengths and dimensions $[tm, k]$ and $[tm, k - 1]$ and check their minimum distances against the BKLC for those parameters.

In the next table, we give a few data points about the rank distribution for $N = 924$ over $GF(5)$. We give the values of $m \cdot p$ on the left and then the following columns are the rank and the number of matrices of that rank.

If we choose the highest rank matrices, there is an interesting property of the gcd of all possible generator polynomials with $x^m - a$ and the gcd of all generator polynomials that corresponds to the highest rank circulant matrices with $x^m - a$. Given the context above, we have

Theorem III.1. *Let $B = \{g_1(x), \dots, g_p(x)\}$ be a set of generators of constacyclic codes, and let $A = \{f_1(x), \dots, f_t(x)\}$ be the subset of B consisting of those polynomials that correspond to circulant matrices of highest rank. Then we have $\gcd(A, x^m - a) = \gcd(B, x^m - a)$*

Proof. Let $B = \{g_1(x), \dots, g_p(x)\}$ be the set of p defining polynomials as in the algorithm. Let $g(x) = \gcd\{B, x^m - a\}$ and let $k = m - \deg(g(x))$. Thus for every $g_i(x) \in B$, we may rewrite $g_i(x) = g(x) \cdot u_i(x)$ for some $u_i(x) \in \mathbb{F}_q[x]$. Let $A = \{f_1(x), \dots, f_t(x)\}$ be the subset of B such that the constacyclic code generated by each $f_i(x)$ has dimension k .

Thus for every $f_i(x) \in A$, we may rewrite $f_i(x) = g(x) \cdot v_i(x)$ for some $v_i(x) \in \mathbb{F}_q[x]$ such that $\gcd(v_i(x), x^m - a) = 1$, for if this were not true then the constacyclic code generated by $f_i(x)$ would not have dimension k . Thus $\gcd(A, x^m - a) = g(x) = \gcd(B, x^m - a)$

□

Thus, the codes obtained by our algorithm are of the form $(g_1(x), \dots, g_t(x))$ where $g_1(x), \dots, g_t(x)$ correspond to the highest rank matrices. We can find a lower bound on the minimum distance, and an equivalence between the ASR algorithm and this algorithm.

Theorem III.2. *[11] Let $\gcd(g_1(x), \dots, g_t(x), x^m - a) = D(x)$ and $f_i(x) = \frac{g_i(x)}{D(x)}$ for $i = 1, 2, \dots, t$. Then the 1-generator QT code C generated by $(g_1(x), \dots, g_t(x))$ is of length $m \cdot t$, dimension $m - \deg(D(x))$, and $d(C) \geq t \cdot d$, where d is the minimum distance of the constacyclic code generated by $D(x)$.*

In the ASR algorithm, we start with a generator $g(x)$, which corresponds to the $D(x)$ above. Then by finding $q_i(x)$ that is co-prime with $h(x) = \frac{x^m - a}{g(x)}$, which corresponds to the $f_i(x)$ above, we form QT codes with generators of the form $(g(x)q_1(x), g(x)q_2(x), \dots, g(x)q_t(x))$. Hence, a code constructed by the top down method of the form $(D(x)f_1(x), D(x)f_2(x), \dots, D(x)f_t(x))$ is essentially the same as a code constructed by the ASR algorithm.

So for each possible value t for the number of blocks, we want to construct QT codes of length $t \cdot m$ and dimensions k or $k - 1$. These t blocks come from the p circulant matrices we have already constructed, and we only join circulant matrices of the same rank. Clearly the total number of ways we can do this is $\binom{p}{t}$, which in general is very large so we have imposed a limit of 20,000. The next step is to randomly select t circulant matrices of rank k , and t of rank $k - 1$, horizontally join them, and construct two codes with parameters $[t \cdot m, k]_q$ and $[t \cdot m, k - 1]_q$. Finally we compute the minimum distance of each code and compare it against the BKLC in [1].

Take our record breaking code $[84, 19, 41]_5$ as an example. We chose $N = 840$, $q = 5$ and $a = 1$. Firstly, we need to find out the list of possible generator polynomials $g(x)$, which are divisors of $x^{840} - 1$. Using an original generator polynomial of degree 765 (so the dimension of the original constacyclic code is 75), we found all possible m 's and p 's. The values of $m = 21$ and $p = 40$ yielded a record breaker with $t = 4$ blocks. In this case, of the 40 blocks in total, we chose 4 blocks from the 35 blocks with the highest rank, which is 19. Each of the remaining 5 blocks has rank 18. The dimension is determined by computing $k = m - \deg(\gcd(x^m - a, g_1(x), g_2(x), \dots, g_p(x)))$. After we split up the original long generator polynomial using our selected values of m and p , we construct a circulant matrix for each of the $p = 40$ polynomials. Once we have computed k , we go through all of these circulant matrices and only keep those with rank k or $k - 1$. After selecting our $t = 4$ full rank matrices, we horizontal join them, and get a final matrix of $84 = t \cdot m = 4 \cdot 21$ columns with rank 19. For a particular

choice of these 4 blocks, we obtained a new linear code whose minimum distance is 41, improving the minimum distance of the previously BKLC of this length and dimension given in [1]. The defining (generating) polynomials of each block of this code are given in the table below.

IV. NEW CODES

We found two types of new codes from an implementation of the algorithm. We have found 5 QT codes that are new among all linear codes according to the database [1]. They are listed in the table below.

					Note
					RECORD BREAKING QT CODES
	$[n, k, d]_q$	α	N	m	Polynomials
1	$[85, 16, 45]_5$	1	2142	17	$g_1 = [1331311000103332]$ $g_2 = [2030141001241411]$ $g_3 = [3412143022013031]$ $g_4 = [1123200013130012]$ $g_5 = [4233002104312041]$
2	$[84, 19, 41]_5$	1	840	21	$g_1 = [44010311002304111222]$ $g_2 = [141032024233443231]$ $g_3 = [1310443344442044020]$ $g_4 = [4011113222243123010]$
3	$[84, 13, 48]_5$	1	3276	14	$g_1 = [34030034422424]$ $g_2 = [4023414111414]$ $g_3 = [10342023404034]$ $g_4 = [44300024241344]$ $g_5 = [3221030030213]$ $g_6 = [433111043203]$
4	$[65, 12, 39]_7$	1	35100	13	$g_1 = [2322660251501]$ $g_2 = [4415215004556]$ $g_3 = [1551620013551]$ $g_4 = [4030032120616]$ $g_5 = [4626364150311]$
5	$[78, 13, 47]_7$	1	4680	13	$g_1 = [16536106450546]$ $g_2 = [1640410515651]$ $g_3 = [32251003506]$ $g_4 = [524220205542]$ $g_5 = [520330333466]$ $g_6 = [15211116242]$

The second type of codes we have found are new in the class of QC and QT codes as presented in the online database [2]. Often times, these codes have the parameters of the BKLCs given in [1] and at the same time have more simple construction than the BKLCs in the database. For example, consider the code with parameters $[66, 11, 40]_7$ in the table below (number 4). It has the same parameters as the BKLC given in [1] but the construction of the code in the database is indirect and complicated, involving many steps and manipulations of other codes. The code that we have is QC, so it has a more useful and desirable construction. We have found a large number of such codes. The table below contains 40 of them. In this table, \star delineates a new QC code for the online database of QT codes [2], $\star\star$ delineates a new QT code for the online database of QC codes [2], and \circ delineates a code with better construction than what is given in [1]

	$[n, k, d]_q$	α	N	m	Polynomials		$[n, k, d]_q$	α	N	m	Polynomials
1	$[75, 16, 41]_7^*$	1	29700	25	$g_1 = [263433044421333266145152]$ $g_2 = [315432322306666040522014]$ $g_3 = [423226456521644016532614]$	19	$[57, 10, 33]_5^*$	1	2394	19	$g_1 = [4001122202200132343]$ $g_2 = [2001010134034202224]$ $g_3 = [423311200332331241]$
2	$[70, 10, 45]_7^*$	1	1530	10	$g_1 = [6303410364]$ $g_2 = [2050420624]$ $g_3 = [2210321115]$ $g_4 = [45041324]$ $g_5 = [435123455]$ $g_6 = [231612611]$ $g_7 = [651163446]$	20	$[54, 16, 24]_5^*$	1	2520	18	$g_1 = [111112401404020422]$ $g_2 = [210103420211004144]$ $g_3 = [430443201130124111]$ $g_4 = [12023114004324224]$ $g_5 = [31032100304014224]$ $g_6 = [140212320140240333]$
3	$[50, 12, 28]_7^*$	1	1650	25	$g_1 = [540452551353554052546141]$ $g_2 = [641633425035353011331645]$	21	$[57, 18, 24]_5^*$	1	2394	19	$g_1 = [22011333323034]$ $g_2 = [2122232302344]$ $g_3 = [23043242422123]$ $g_4 = [14114212211]$ $g_5 = [3130431234204]$
4	$[66, 11, 40]_7^{**}$	1	1650	11	$g_1 = [2322660251501]$ $g_2 = [4415215004556]$ $g_3 = [1551620013551]$ $g_4 = [4030032120616]$ $g_5 = [4626364150311]$	22	$[70, 12, 39]_5^*$	1	2394	14	$g_1 = [22011333323034]$ $g_2 = [2122232302344]$ $g_3 = [23043242422123]$ $g_4 = [14114212211]$ $g_5 = [3130431234204]$
5	$[77, 11, 49]_7^*$	1	1650	11	$g_1 = [564114442]$ $g_2 = [2461136364]$ $g_3 = [1516121646]$ $g_4 = [6616204266]$ $g_5 = [1645256106]$ $g_6 = [261414153]$ $g_7 = [5230445332]$	23	$[57, 19, 23]_5^*$	1	2394	19	$g_1 = [21201010102211234]$ $g_2 = [24014322314212313]$ $g_3 = [120021024011344434]$
6	$[84, 11, 54]_7^{**}$	1	2340	14	$g_1 = [444322262213]$ $g_2 = [554352030341]$ $g_3 = [10166310604]$ $g_4 = [15346150532]$ $g_5 = [61316462321]$ $g_6 = [32314565553]$ $g_7 = [15302241541]$	24	$[54, 17, 23]_5^*$	1	2394	18	$g_1 = [111142143224042044]$ $g_2 = [13203240213132332]$ $g_3 = [43013432321002133]$
7	$[63, 21, 27]_7^*$	1	3276	21	$g_1 = [124023403431134343212]$ $g_2 = [213132232321244113304]$ $g_3 = [4310344424422211142]$	25	$[54, 15, 25]_5^*$	1	2394	18	$g_1 = [21201010102211234]$ $g_2 = [24014322314212313]$ $g_3 = [120021024011344434]$
8	$[75, 15, 42]_7^*$	1	4680	15	$g_1 = [2322660251501]$ $g_2 = [4415215004556]$ $g_3 = [1551620013551]$ $g_4 = [4030032120616]$ $g_5 = [4626364150311]$	26	$[48, 21, 16]_5^*$	1	840	24	$g_1 = [43220423003100340303233]$ $g_2 = [1420414002032004203044]$
9	$[76, 10, 50]_7^{**}$	1	5586	19	$g_1 = [5031221102660240321]$ $g_2 = [2363456125525046026]$ $g_3 = [261352230616301205]$ $g_4 = [645316064010314463]$	27	$[63, 21, 25]_5^{**}$	1	1638	21	$g_1 = [301304303123033402013]$ $g_2 = [143331010304032042133]$ $g_3 = [22102000202342013201]$
10	$[60, 19, 27]_7^{**}$	1	7020	20	$g_1 = [15443624352620433401]$ $g_2 = [50055310242611250602]$ $g_3 = [2424542451060242561]$	28	$[48, 20, 17]_5^{**}$	1	840	24	$g_1 = [4112013131321102321333]$ $g_2 = [20221032200410113034424]$
11	$[88, 10, 59]_7^*$	1	9900	10	$g_1 = [5203436246]$ $g_2 = [51043451221]$ $g_3 = [12401350525]$ $g_4 = [41243403]$ $g_5 = [53444636644]$ $g_6 = [1031143125]$ $g_7 = [33452534051]$ $g_8 = [22465515122]$	29	$[48, 19, 18]_5^{**}$	1	840	24	$g_1 = [4132313101410322242343]$ $g_2 = [2234233102121102440302]$
12	$[60, 22, 24]_7^{**}$	1	23400	30	$g_1 = [36635456033343033602466243345]$ $g_2 = [656321204662143163155634466551]$	30	$[98, 14, 56]_5^*$	1	840	14	$g_1 = [30434340014012]$ $g_2 = [404230140102]$ $g_3 = [12113334213201]$ $g_4 = [3042242213423]$ $g_5 = [4144330200204]$ $g_6 = [1400023210111]$ $g_7 = [1140211211241]$
13	$[66, 22, 28]_7^{**}$	1	23760	22	$g_1 = [6563251503162345354255]$ $g_2 = [533056566022662664544]$ $g_3 = [54064355203553654613]$	31	$[54, 18, 22]_5^*$	1	2520	18	$g_1 = [104230420440014]$ $g_2 = [1002400240424024]$ $g_3 = [440320311201032]$
14	$[50, 20, 20]_7^*$	1	23400	25	$g_1 = [4443213452310165035153651]$ $g_2 = [1630165556635256134420642]$	32	$[54, 16, 24]_5^*$	1	2394	18	$g_1 = [213013442013241243]$ $g_2 = [343444014210104]$ $g_3 = [1141401024010033]$
15	$[78, 23, 33]_5^{**}$	2	1638	39	$g_1 = [13032111444411404203244031330204130]$ $g_2 = [424210422221110200143114323340030334]$	33	$[63, 20, 26]_5^{**}$	1	840	21	$g_1 = [433230444340230011144]$ $g_2 = [330024240231414133442]$ $g_3 = [30222313400013211232]$
16	$[84, 13, 47]_5^{**}$	1	2394	42	$g_1 = [130321114444114042032440313302041302]$ $g_2 = [424210422221110200143114323340030334]$	34	$[63, 19, 27]_5^{**}$	1	1638	21	$g_1 = [234410410332004300314]$ $g_2 = [14213013022200423341]$ $g_3 = [20003430124131132214]$
17	$[63, 14, 32]_5^*$	1	2394	21	$g_1 = [321343402000310324]$ $g_2 = [34343420432104113431]$ $g_3 = [11021044124132020034]$	35	$[63, 18, 28]_5^{**}$	1	1638	21	$g_1 = [212000340143014102304]$ $g_2 = [10040010022420444214]$ $g_3 = [11432313222320030332]$
18	$[70, 13, 38]_5^{**}$	1	2520	14	$g_1 = [14442024404443]$ $g_2 = [4003124334114]$ $g_3 = [31142014342]$ $g_4 = [4233234211104]$ $g_5 = [4133124411213]$	36	$[63, 15, 31]_5^*$	1	2310	21	$g_1 = [3334101320122112311111]$ $g_2 = [1341022300043442024]$ $g_3 = [300211014411004001]$
						37	$[84, 15, 45]_5^*$	1	2310	21	$g_1 = [342412033421313430222]$ $g_2 = [100211424324222410432]$ $g_3 = [223300414213401242213]$ $g_4 = [11214331334014103314]$
						38	$[72, 22, 30]_5^*$	1	2520	24	$g_1 = [3020423420442333414224]$ $g_2 = [20030300341134112302]$ $g_3 = [1203004014301110344232]$
						39	$[48, 16, 21]_5^*$	1	6552	24	$g_1 = [30131044301202334202204]$ $g_2 = [34041432001111423440011]$
						40	$[52, 14, 25]_5^{**}$	2	9828	26	$g_1 = [422412211222002300041024]$ $g_2 = [4112244244422042111422143]$

REFERENCES

- [1] M. Grassl, Code Tables: Bounds on the parameters of codes, [online server], <http://www.codetables.de/>
- [2] E. Chen, Online Database of Quasi-Twisted Codes [online server], <http://www.tec.hkr.se/~chen/research/codes/>
- [3] N. Aydin, N. Connolly, and J. Murphree, "New binary linear codes from QC codes and an augmentation algorithm, *AAEC*, vol. 28, no. 4, pp. 339-350, 2017.
- [4] N. Aydin, N. Connolly, and M. Grassl, "Some results on the structure of constacyclic codes and new linear codes over $GF(7)$ from quasi-twisted codes", *Advances in Mathematics of Communication*, vol. 11, no. 1, pp. 245-258, 2017.
- [5] N. Aydin and D. Foret, "New Linear Codes over $GF(3)$, $GF(11)$, and $GF(13)$ ", *Journal of Algebra Combinatorics Discrete Structures and Applications*, vol. 6, no. 1, pp. 13-20, 2019
- [6] N. Aydin, G. Bakbouk, and J. Lambrinos, "New Linear Codes over non-Prime Fields", *Cryptography and Communications*, 2018.
- [7] N. Aydin, J. Lambrinos, and Oliver VandenBerg, "On Equivalence of Cyclic Codes, Generalization of a Quasi-Twisted Search Algorithm, and New Linear Codes, *Designs, Codes and Cryptography*, 2019.
- [8] N. Aydin, T. Guidotti, P. Liu, A. Shaikh, O. VandenBerg, "Some Generalizations of the ASR Search Algorithm for Quasi-Twisted Codes", *Involve*, Vol 13, no 1, pp. 137-148, 2020.
- [9] Magma computer algebra system, [online], <http://magma.maths.usyd.edu.au/>
- [10] A. Vardy, "The intractability of computing the minimum distance of a code", *IEEE Trans. Inform. Theory*, vol. 43, no. 6, pp. 1757-1766, 1997.
- [11] N. Aydin, I. Siap, and D. K. Ray-Chaudhuri, "The structure of 1-generator quasi-twisted codes and new linear codes", *Des. Codes Cryptogr.*, vol. 24, no. 3, pp. 313-326, 2001.
- [12] V. Bhargava, G. Seguin and J. Stein.: "Some (mk, k) cyclic codes in quasi-cyclic form", *IEEE Trans. Inform. Theory*, vol. 24, no. 5, pp. 630-632, 1978.
- [13] E. Z. Chen, "Quasi-cyclic codes derived from cyclic codes", *Proceedings of the International Symposium on Information Theory and its Applications*, pp. 162-165, 2004.
- [14] E. Z. Chen, "Computer construction of quasi-twisted two-weight codes", *Sixth International Workshop on Optimal Codes and Related Topics*, pp. 62-68, 2009.
- [15] E. Z. Chen, "A new iterative computer search algorithm for good quasi-twisted codes", *Des. Codes Cryptogr.*, vol. 76, pp. 3073-323, 2015.